การพัฒนาเว็บแอปพลิเคชัน

แนวคิดและหลักการ...สู่การปฏิบัติสำหรับผู้เริ่มต้น



เอกรัฐ ปัญญาเทพ

การพัฒนาเว็บแอปพลิเคชัน:
แนวคิดและหลักการสู่การปฏิบัติสำหรับผู้เริ่มต้น

เอกรัฐ ปัญญาเทพ

มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด

พ.ศ. 2569

การพัฒนาเว็บแอปพลิเคชัน:
แนวคิดและหลักการสู่การปฏิบัติสำหรับผู้เริ่มต้น

พิมพ์ครั้งแรก: มิถุนายน 2569

ราคา: 300 บาท

ข้อมูลทางบรรณานุกรมของหอสมุดแห่งชาติ

เอกรัฐ ปัญญาเทพ.
การพัฒนาเว็บแอปพลิเคชัน: แนวคิดและหลักการสู่การปฏิบัติสำหรับผู้เริ่มต้น.-- ตาก : สำนักวิทยบริการและเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด, 2569.
488 หน้า.

1. โปรแกรมประยุกต์บนเว็บ. I. ชื่อเรื่อง.

005.355
ISBN 978-974-388-407-8

ผู้เขียน: เอกรัฐ ปัญญาเทพ (ปร.ด. เทคโนโลยีสารสนเทศเพื่อธุรกิจและการศึกษา)
พิสูจน์อักษร: มัลลิกา ปัญญาเทพ
จัดทำโดย:
สำนักวิทยบริการและเทคโนโลยีสารสนเทศ มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด
เลขที่ 222 หมู่ 7 ตำบลแม่ปะ อำเภอแม่สอด
จังหวัดตาก 63110

ลิขสิทธิ์เป็นของผู้เขียน ปีพ.ศ. 2569

คำนำ

ทักษะการพัฒนาเว็บแอปพลิเคชันในทุกวันนี้ไม่ได้เป็นเรื่องของนักพัฒนาอาชีพเท่านั้นอีกต่อไป เว็บแอปพลิเคชันได้กลายเป็นส่วนหนึ่งของชีวิตประจำวัน ทั้งในเชิงธุรกิจ การศึกษา และการบริการสาธารณะ ผู้ที่เข้าใจกลไกเบื้องหลังการทำงานของเว็บแอปพลิเคชันจึงมีความได้เปรียบอย่างมีนัยสำคัญ ไม่ว่าจะอยู่ในสาขาวิชาใดก็ตาม หนังสือ "การพัฒนาเว็บแอปพลิเคชัน: แนวคิดและหลักการสู่การปฏิบัติสำหรับผู้เริ่มต้น" เล่มนี้ เขียนและเรียบเรียงขึ้นจากการค้นคว้า การปฏิบัติงานจริง และประสบการณ์ในการสอนที่สั่งสมมา โดยมุ่งเน้นการเชื่อมโยงแนวคิดของเนื้อหาที่เกี่ยวข้องให้เป็นองค์รวมที่ต่อเนื่อง เพื่อสร้างฐานความรู้ที่จำเป็นให้ผู้อ่านสามารถพัฒนาเว็บแอปพลิเคชันได้ด้วยตนเองอย่างมั่นใจ พร้อมเข้าร่วมสังคมนักพัฒนาเว็บแอปพลิเคชันแม้จะไม่มีพื้นฐานด้านการเขียนเว็บมาก่อน และมีความรู้เพียงพอที่จะใช้ปัญญาประดิษฐ์เป็นเครื่องมือช่วยในการพัฒนาเว็บแอปพลิเคชันได้อย่างมีวิจารณญาณ สามารถตรวจสอบความถูกต้องของผลลัพธ์และควบคุมคุณภาพได้อย่างเหมาะสม

เนื้อหาในหนังสือจัดลำดับจากพื้นฐานสู่การนำไปปฏิบัติ แบ่งออกเป็น 4 ส่วน ส่วนที่ 1 ส่วนนำ (บทที่ 1-2) นำเสนอพื้นฐานความเข้าใจเกี่ยวกับกลไกการทำงานของเว็บแอปพลิเคชันและความสัมพันธ์กับระบบสารสนเทศ ส่วนที่ 2 สภาพแวดล้อมการพัฒนา (บทที่ 3-4) ครอบคลุมการเตรียมสภาพแวดล้อมการพัฒนาโดยใช้โปรแกรมพื้นฐานและเทคโนโลยี Docker ส่วนที่ 3 Frontend (บทที่ 5-10) นำเสนอภาษาและเครื่องมือสำหรับการพัฒนาส่วนติดต่อผู้ใช้ และส่วนที่ 4 Backend (บทที่ 11-12) นำเสนอภาษาและกลไกสำหรับจัดการข้อมูลฝั่งผู้ให้บริการ เนื้อหาทั้งสี่ส่วนนี้มุ่งสร้างความพร้อมให้ผู้อ่านก่อนจะก้าวไปสู่การใช้เฟรมเวิร์กสมัยใหม่ที่ตอบโจทย์การพัฒนาเว็บแอปพลิเคชันในโลกการทำงานจริง

หนังสือเล่มนี้เหมาะสำหรับนักศึกษาระดับปริญญาตรีทั้งในสาขาที่เกี่ยวข้อง และสาขาที่ต้องการเรียนรู้ข้ามศาสตร์ รวมถึงผู้สนใจทั่วไปที่ต้องการสร้างฐานความรู้ด้านการพัฒนาเว็บแอปพลิเคชันในยุคที่เทคโนโลยีเปิดกว้างให้ทุกคนเข้าถึงได้ นอกจากนี้ยังสามารถใช้เป็นหนังสืออ่านประกอบในการจัดการเรียนการสอนสำหรับรายวิชาที่เกี่ยวข้องกับการพัฒนาเว็บแอปพลิเคชันในระดับอุดมศึกษา

ผู้เขียนหวังเป็นอย่างยิ่งว่าหนังสือเล่มนี้จะช่วยให้ผู้อ่านเกิดความรู้ ความเข้าใจ และสามารถพัฒนาเว็บแอปพลิเคชันได้อย่างมั่นใจ หากมีข้อบกพร่องประการใด ผู้เขียนต้องขออภัยและพร้อมที่จะปรับปรุงให้ดียิ่งขึ้นต่อไป และขอโอกาสนี้ขอบคุณรองศาสตราจารย์ ดร.อรสา เตติวัฒน์ ผู้ช่วยศาสตราจารย์รุ่ง หมูล้อม และผู้ช่วยศาสตราจารย์ ดร.ไก้รุ่ง เฮงพระพรหม ตลอดจนครูอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาพร้อมให้ข้อเสนอแนะกับผู้เขียนในการพัฒนาตนเองมาโดยตลอด

เอกรัฐ ปัญญาเทพ

ดาวน์โหลดตัวอย่างประกอบหนังสือได้ที่

https://github.com/ajnok/book-web-application



สารบัญ

หน้า

คำนำ i

สารบัญ iii

สารบัญภาพ vii

สารบัญตาราง xi

ส่วนที่ 1 ส่วนนำ 1

บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน 3

1. ความรู้เบื้องต้นเกี่ยวกับเว็บ 4
2. ความหมายและคุณลักษณะของเว็บแอปพลิเคชัน 9
3. กลไกการทำงานของเว็บแอปพลิเคชัน 11
4. ภาษาในการพัฒนาเว็บแอปพลิเคชัน 21
5. สรุปท้ายบท 23
6. กิจกรรมท้ายบท 24

บทที่ 2 ระบบสารสนเทศกับการพัฒนาเว็บแอปพลิเคชัน 25

1. ระบบสารสนเทศ 26
2. ความสัมพันธ์ของระบบสารสนเทศและเว็บแอปพลิเคชัน 38
3. วงจรการพัฒนาระบบ 44
4. การพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ 54
5. สรุปท้ายบท 59
6. กิจกรรมท้ายบท 60

ส่วนที่ 2 สภาพแวดล้อมการพัฒนา 61

บทที่ 3 สภาพแวดล้อมเว็บแอปพลิเคชันและการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูป 63

1. สภาพแวดล้อมเว็บแอปพลิเคชัน 64
2. องค์ประกอบหลักของสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชัน 67
3. ชื่อเรียกชุดโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บ 71
4. การสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูปในสภาพแวดล้อมการพัฒนา 72

หน้า

5. สรุปท้ายบท 79
6. กิจกรรมท้ายบท 80

บทที่ 4 การสร้างกลไกผู้ให้บริการเว็บด้วยแนวคิดการจำลองเสมือนในสภาพแวดล้อมการพัฒนา 81

1. แนวคิดการจำลองเสมือน 82
2. หลักการของ Docker Desktop 97
3. กรณีศึกษา: การสร้าง LAMP Stack ด้วย Docker Desktop 104
4. สรุปท้ายบท 116
5. กิจกรรมท้ายบท 117

ส่วนที่ 3 Frontend: ภาษาสำหรับสร้างส่วนติดต่อผู้ใช้ 119

บทที่ 5 ภาษา HTML 121

1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML 122
2. การกำกับในภาษา HTML 123
3. ค่าเริ่มต้นการแสดงผลของอิลิเมนต์ 130
4. โครงสร้างของภาษา HTML 131
5. การเขียนบรรทัดหมายเหตุ 145
6. สรุปท้ายบท 145
7. กิจกรรมท้ายบท 146

บทที่ 6 แท็ก HTML ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน 147

1. แท็กสำหรับการแสดงรูปภาพ 148
2. แท็กสำหรับการสร้างลิงก์ 149
3. แท็กสำหรับการกำหนดรายการ 152
4. แท็กสำหรับการกำหนดหัวเรื่องและข้อความ 154
5. แท็กสำหรับการสร้างตาราง 161
6. แท็กสำหรับการสร้างฟอร์มรับข้อมูล 167
7. แท็กสำหรับการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา 173
8. แท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมาย 176
9. สรุปท้ายบท 183
10. กิจกรรมท้ายบท 184

หน้า

บทที่ 7 ภาษา CSS 187

1. ความรู้เบื้องต้นเกี่ยวกับ CSS 188
2. รูปแบบการกำหนดคุณสมบัติการแสดงผล 190
3. วิธีใช้ CSS ร่วมกับเอกสารเว็บ 191
4. สีและหน่วยวัด 212
5. สรุปท้ายบท 218
6. กิจกรรมท้ายบท 218

บทที่ 8 การกำหนดลักษณะการแสดงผลและการสร้างองค์ประกอบของเว็บแอปพลิเคชันด้วย CSS 221

1. คุณสมบัติ CSS พื้นฐานในการจัดรูปแบบอิลิเมนต์ 222
2. หลักการพื้นฐานของการจัดวางโครงร่างด้วย CSS Flexbox และ CSS Grid 238
3. CSS Framework 249
4. การใช้งาน Bootstrap สำหรับสร้างองค์ประกอบในเว็บแอปพลิเคชัน 250
5. สรุปท้ายบท 266
6. กิจกรรมท้ายบท 266

บทที่ 9 ภาษา JavaScript 269

1. ความรู้เบื้องต้นเกี่ยวกับภาษา JavaScript 270
2. องค์ประกอบของภาษา JavaScript 272
3. ชนิดของข้อมูล 297
4. โครงสร้างการควบคุม 301
5. สรุปท้ายบท 314
6. กิจกรรมท้ายบท 316

บทที่ 10 การจัดการข้อมูลและอิลิเมนต์ของหน้าเว็บด้วย JavaScript 317

1. ฟังก์ชัน 317
2. ออบเจกต์ 324
3. อาร์เรย์ 339
4. การจัดการเอกสารเว็บด้วย JavaScript 349
5. สรุปท้ายบท 373
6. กิจกรรมท้ายบท 374

หน้า

ส่วนที่ 4 Backend: ภาษาจัดการข้อมูลฝั่งผู้ให้บริการ 375

บทที่ 11 ภาษา PHP 377

1. ความรู้เบื้องต้นเกี่ยวกับภาษา PHP 378
2. การสร้างไฟล์และไวยากรณ์ของภาษา PHP 381
3. ตัวแปรและชนิดข้อมูล 388
4. การแสดงผลด้วยฟังก์ชัน echo และ print 392
5. โครงสร้างการควบคุม 399
6. ฟังก์ชันและอาร์เรย์ 405
7. สรุปท้ายบท 418
8. กิจกรรมท้ายบท 419

บทที่ 12 การจัดการข้อมูลจากผู้ใช้ด้วย PHP 421

1. การรับค่าจากฟอร์ม 422
2. การทำงานร่วมกับฐานข้อมูล 432
3. ระบบ CRUD 443
4. สรุปท้ายบท 474
5. กิจกรรมท้ายบท 475

บรรณานุกรม 477

ดัชนี 485

สารบัญภาพ

หน้า

ภาพที่ 1.1	รูปแบบมาตรฐานของ URL และตัวอย่างของ URL	6
ภาพที่ 1.2	ตัวอย่างการกำหนดตำแหน่งที่เก็บทรัพยากรในเครื่องผู้ให้บริการเว็บ	7
ภาพที่ 1.3	ภาพรวมส่วนประกอบและการทำงานของเว็บแอปพลิเคชัน	11
ภาพที่ 1.4	ส่วนประกอบพื้นฐานของเว็บเบราว์เซอร์	12
ภาพที่ 1.5	การแยกวิเคราะห์ HTML เป็นโครงสร้างต้นไม้ DOM	13
ภาพที่ 1.6	การแยกวิเคราะห์ CSS เป็นโครงสร้างต้นไม้ CSSOM	14
ภาพที่ 1.7	โครงสร้างต้นไม้การแสดงผล	15
ภาพที่ 1.8	องค์ประกอบที่แสดงในเว็บเบราว์เซอร์และแบบจำลองกล่อง	16
ภาพที่ 1.9	การคำนวณขนาดขององค์ประกอบที่แสดงในเว็บเบราว์เซอร์ตามแบบจำลองกล่อง	17
ภาพที่ 1.10	ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บ	18
ภาพที่ 1.11	ขั้นตอนการทำงานของเว็บแอปพลิเคชัน	20
ภาพที่ 1.12	องค์ประกอบการพัฒนาเว็บแอปพลิเคชัน	21
ภาพที่ 2.1	องค์ประกอบของระบบ	26
ภาพที่ 2.2	ตัวอย่างของการได้มาซึ่งสารสนเทศ	28
ภาพที่ 2.3	องค์ประกอบของสารสนเทศ	29
ภาพที่ 2.4	องค์ประกอบของระบบสารสนเทศ	34
ภาพที่ 2.5	แบบจำลองความสัมพันธ์ขององค์ประกอบของระบบสารสนเทศ	38
ภาพที่ 2.6	แบบจำลองการบูรณาการเว็บแอปพลิเคชันในระบบสารสนเทศ	39
ภาพที่ 2.7	ระยะต่าง ๆ ของวงจรการพัฒนาระบบ	46
ภาพที่ 2.8	รายละเอียดการดำเนินการในแต่ละระยะของการวงจรการพัฒนาระบบ	46
ภาพที่ 2.9	ขั้นตอนการพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ	55
ภาพที่ 3.1	การสร้างกลไกการทำงานของผู้ให้บริการเว็บภายใต้สภาพแวดล้อมการพัฒนา	68
ภาพที่ 3.2	เว็บไซต์ทางการของผู้พัฒนาโปรแกรม XAMPP	73
ภาพที่ 3.3	การเลือกองค์ประกอบที่ต้องการติดตั้ง	74
ภาพที่ 3.4	การกำหนดตำแหน่งที่จัดเก็บไฟล์และโฟลเดอร์ที่เกี่ยวข้องกับการทำงานของ XAMPP	75
ภาพที่ 3.5	การแสดงสถานะการติดตั้ง XAMPP เสร็จสมบูรณ์	75
ภาพที่ 3.6	หน้าต่างควบคุมการทำงานของ XAMPP	76
ภาพที่ 3.7	หน้าต่างควบคุมการทำงานของ XAMPP เมื่อมีการเปิดการทำงานโปรแกรม Apache และ MySQL	76

หน้า

ภาพที่ 3.8	หน้าเว็บทดสอบการทำงานของ Apache	77
ภาพที่ 3.9	หน้าเว็บแอปพลิเคชัน phpMyAdmin	78
ภาพที่ 4.1	สถาปัตยกรรมเทคโนโลยีเครื่องเสมือน	82
ภาพที่ 4.2	สถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์	86
ภาพที่ 4.3	การเปรียบเทียบสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์และเทคโนโลยีเครื่องเสมือน	87
ภาพที่ 4.4	หน้าจอสอบถามการติดตั้ง WSL 2	99
ภาพที่ 4.5	หน้าต่างโปรแกรม Docker Desktop	99
ภาพที่ 4.6	หน้าจอผลลัพธ์การทดสอบคอนเทนเนอร์ที่ได้จากการติดตั้ง Docker Desktop	100
ภาพที่ 4.7	กลไกการทำงานของ Docker Desktop บนเครื่องผู้พัฒนา	100
ภาพที่ 4.8	การออกแบบโครงสร้าง LAMP Stack	105
ภาพที่ 4.9	ตัวอย่างการออกแบบโครงสร้างโฟลเดอร์สำหรับโครงการพัฒนาเว็บแอปพลิเคชัน	106
ภาพที่ 4.10	การตรวจสอบการทำงานของ Docker Engine	107
ภาพที่ 4.11	การสร้างโฟลเดอร์ด้วยโปรแกรม VSCode	108
ภาพที่ 4.12	การสั่งให้ Docker Compose ทำการสร้างคอนเทนเนอร์และองค์ประกอบต่าง ๆ	111
ภาพที่ 4.13	ผลลัพธ์ของคำสั่ง docker ps	112
ภาพที่ 4.14	การแสดงสถานะของคอนเทนเนอร์จากหน้าต่างโปรแกรม Docker Desktop	113
ภาพที่ 4.15	ผลลัพธ์การทำงานของไฟล์ index.php จากคอนเทนเนอร์ webserver	114
ภาพที่ 4.16	หน้าจอเข้าสู่ระบบของเว็บแอปพลิเคชัน phpMyAdmin	115
ภาพที่ 4.17	หน้าเว็บแอปพลิเคชัน phpMyAdmin ที่อยู่ในคอนเทนเนอร์ dbmanager	115
ภาพที่ 4.18	โครงสร้างโฟลเดอร์สำหรับกิจกรรมท้ายบท	117
ภาพที่ 5.1	ตัวอย่างบทบาทของเนื้อหาในเอกสารเว็บ	124
ภาพที่ 5.2	อิลิเมนต์ที่มีเนื้อหา	127
ภาพที่ 5.3	อิลิเมนต์ที่ไม่มีเนื้อหา	128
ภาพที่ 5.4	อิลิเมนต์ที่ซ้อนกัน	129
ภาพที่ 5.5	โครงสร้างภาษา HTML	132
ภาพที่ 5.6	องค์ประกอบสำคัญที่ควรกำหนดในส่วนหัวของเอกสาร	135
ภาพที่ 5.7	ตัวอย่างการอ้างอิงไฟล์ CSS ที่จัดเก็บในโฟลเดอร์	141
ภาพที่ 6.1	องค์ประกอบของโครงสร้างตาราง	161
ภาพที่ 6.2	ตัวอย่างการจัดกลุ่มเนื้อหาเป็นส่วนต่าง ๆ	177
ภาพที่ 6.3	ตัวอย่างการจัดกลุ่มเนื้อหาและบทบาทของเนื้อหาในเอกสารเว็บ	185
ภาพที่ 7.1	ชุดกฎของ CSS	195

หน้า

ภาพที่ 7.2	ตัวอย่างของ DOM Tree ที่นำมาใช้พิจารณาความสัมพันธ์เชิงลำดับชั้นของอิลิเมนต์	203
ภาพที่ 7.3	ตัวอย่างผลลัพธ์ของการใช้ CSS ภายนอก	219
ภาพที่ 8.1	องค์ประกอบของ CSS Flexbox	239
ภาพที่ 8.2	ทิศทางการจัดเรียง Flex Items แบบ row	241
ภาพที่ 8.3	ทิศทางการจัดเรียง Flex Items แบบ row-reverse	241
ภาพที่ 8.4	ทิศทางการจัดเรียง Flex Items แบบ column	242
ภาพที่ 8.5	ทิศทางการจัดเรียง Flex Items แบบ column-reverse	242
ภาพที่ 8.6	การจัดตำแหน่งในแนวแกนข้ามเพื่อแกนหลักมีทิศทางในแนวนอน	243
ภาพที่ 8.7	การจัดตำแหน่งในแนวแกนข้ามเพื่อแกนหลักมีทิศทางในแนวตั้ง	243
ภาพที่ 8.8	โครงสร้างของ CSS Grid ที่มีการจัดวาง Grid Items แบบ 3 แถว 3 คอลัมน์	244
ภาพที่ 8.9	ผลลัพธ์ที่คาดหวังในการใช้ Bootstrap CSS Framework	267
ภาพที่ 9.1	ตัวอย่างลำดับการประเมินค่าของนิพจน์เชิงตรรกะ	280
ภาพที่ 9.2	ตำแหน่งการประกาศตัวแปรแบบโกลบอลและขอบเขตการเรียกใช้งาน	289
ภาพที่ 9.3	ตำแหน่งการประกาศตัวแปรในฟังก์ชันและขอบเขตการเรียกใช้งาน	290
ภาพที่ 9.4	ตำแหน่งการประกาศตัวแปรในกลุ่มประโยคคำสั่งและขอบเขตการเรียกใช้งาน	292
ภาพที่ 9.5	โครงสร้างการตัดสินใจแบบทางเลือกเดียว	302
ภาพที่ 9.6	โครงสร้างการตัดสินใจแบบสองทางเลือก	304
ภาพที่ 9.7	โครงสร้างการตัดสินใจแบบหลายทางเลือก	306
ภาพที่ 9.8	โครงสร้างการตัดสินใจแบบหลายกรณี	307
ภาพที่ 9.9	โครงสร้างการวนซ้ำแบบ for	310
ภาพที่ 9.10	โครงสร้างการวนซ้ำแบบ while	311
ภาพที่ 9.11	โครงสร้างการวนซ้ำแบบ do-while	313
ภาพที่ 10.1	โครงสร้างอาร์เรย์	340
ภาพที่ 10.2	ตัวอย่างอาร์เรย์ที่มีข้อมูลจำนวน 3 รายการ	343
ภาพที่ 10.3	ตัวอย่างผลลัพธ์ของการเพิ่มข้อมูลในอาร์เรย์	343
ภาพที่ 10.4	ตัวอย่างของผลลัพธ์การเพิ่มข้อมูลใหม่ด้วยการใช้ค่าคุณสมบัติ length เป็นดัชนี	344
ภาพที่ 10.5	ตัวอย่างผลการเพิ่มข้อมูลด้วยการระบุดัชนีให้มีค่าเกินกว่าดัชนีสูงสุดเดิมของอาร์เรย์	344
ภาพที่ 11.1	กลไกการทำงานร่วมกันระหว่างซอฟต์แวร์ผู้ให้บริการเว็บและตัวแปลภาษา PHP	379
ภาพที่ 11.2	ตัวอย่างฟอร์มที่สร้างด้วยภาษา HTML และผลลัพธ์จากการประมวลผลของภาษา PHP	419
ภาพที่ 11.3	ผลลัพธ์ที่คาดหวังในการแสดงข้อมูลจากอาร์เรย์สองมิติในภาษา PHP	420
ภาพที่ 12.1	กลไกการรับค่าจากฟอร์ม	422

หน้า

ภาพที่ 12.2 การตรวจสอบสถานะของการติดตั้ง PDO Extension สำหรับฐานข้อมูล MySQL 435
ภาพที่ 12.3 ตัวอย่างแผนภาพ ER-Diagram ของฐานข้อมูล blog_system 436
ภาพที่ 12.4 การแสดงสถานการณ์นำเข้าไฟล์ blogsystem.sql และสร้างฐานข้อมูลได้สำเร็จ 437
ภาพที่ 12.5 ความสัมพันธ์ระหว่างไฟล์และการไหลของข้อมูลในระบบ CRUD 446
ภาพที่ 12.6 ตัวอย่างผลลัพธ์ของหน้า index.php เมื่อเชื่อมต่อฐานข้อมูลสำเร็จกรณีที่มีและไม่มีข้อมูลในระบบ 454
ภาพที่ 12.7 ตัวอย่างผลลัพธ์ของไฟล์ add.php สำหรับกรอกข้อมูลผู้ใช้ใหม่ 457
ภาพที่ 12.8 ตัวอย่างผลลัพธ์ของไฟล์ add_process.php เมื่อเพิ่มข้อมูลสำเร็จและกรณีที่มีอีเมลซ้ำ 460
ภาพที่ 12.9 ตัวอย่างผลลัพธ์ของไฟล์ edit.php กรณีพบข้อมูลและไม่พบข้อมูลผู้ใช้ 465
ภาพที่ 12.10 ตัวอย่างผลลัพธ์ของไฟล์ edit_process.php กรณีแก้ไขข้อมูลสำเร็จและไม่มีการเปลี่ยนแปลงข้อมูล 470
ภาพที่ 12.11 ตัวอย่างผลลัพธ์ของไฟล์ delete_process.php ในกรณีที่ลบข้อมูลสำเร็จและไม่สำเร็จ 473

สารบัญตาราง

หน้า

ตารางที่ 2-1 ตัวอย่างงานวิจัยที่สนับสนุนความสำคัญของระบบสารสนเทศที่มีผลเชิงบวกต่อองค์กรหรือการดำเนินงาน 33
ตารางที่ 2-2 บุคลากรที่เกี่ยวข้องกับระบบสารสนเทศ 35
ตารางที่ 3-1 การเปรียบเทียบสภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริง 66
ตารางที่ 3-2 ตัวอย่างโปรแกรมสำเร็จรูปสำหรับการติดตั้งชุดโปรแกรมกลไกผู้ให้บริการเว็บ 72
ตารางที่ 4-1 ตารางเปรียบเทียบการสร้างกลไกผู้ให้บริการเว็บด้วยวิธีต่าง ๆ 92
ตารางที่ 4-2 คุณสมบัติขั้นพื้นฐานที่สำคัญของเครื่องคอมพิวเตอร์ที่จะติดตั้ง Docker Desktop 97
ตารางที่ 5-1 ชื่อคุณลักษณะที่ใช้ในการกำหนดข้อมูลสนับสนุนเครื่องมือค้นหา 138
ตารางที่ 7-1 เปรียบเทียบลักษณะการใช้ตัวเลือกของ CSS 196
ตารางที่ 7-2 สถานะจำเพาะที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน 209
ตารางที่ 7-3 องค์ประกอบเสมือนที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน 211
ตารางที่ 7-4 ตัวอย่างหน่วยวัดสัมบูรณ์ใน CSS 216
ตารางที่ 7-5 ตัวอย่างหน่วยวัดสัมพัทธ์ใน CSS 217
ตารางที่ 8-1 รายการคุณสมบัติในการควบคุมขนาดของอิลิเมนต์ 222
ตารางที่ 8-2 รายการคุณสมบัติที่เกี่ยวข้องกับการจัดการเส้นขอบของอิลิเมนต์ 224
ตารางที่ 8-3 รายการคุณสมบัติที่ใช้ในการจัดการระยะห่างภายในของอิลิเมนต์ 225
ตารางที่ 8-4 รายการคุณสมบัติในการจัดการระยะห่างภายนอกของอิลิเมนต์ 227
ตารางที่ 8-5 รายการคุณสมบัติในการจัดตำแหน่งข้อความและอิลิเมนต์ 228
ตารางที่ 8-6 คุณสมบัติที่เกี่ยวข้องกับการกำหนดแบบตัวอักษร 230
ตารางที่ 8-7 รายการคุณสมบัติในการจัดการพื้นหลังของอิลิเมนต์ 231
ตารางที่ 8-8 รายการคุณสมบัติในการกำหนดลักษณะของลิงก์ 233
ตารางที่ 8-9 รายการคุณสมบัติในการจัดรูปแบบตาราง 235
ตารางที่ 8-10 คลาสที่ใช้ในการกำหนดสีเชิงบริบทใน Bootstrap 255
ตารางที่ 8-11 ชื่อตัวย่อของ Breakpoint 261
ตารางที่ 9-1 เครื่องหมายดำเนินการทางคณิตศาสตร์ 275
ตารางที่ 9-2 เครื่องหมายเปรียบเทียบ 276
ตารางที่ 9-3 เครื่องหมายกำหนดค่า 278
ตารางที่ 9-4 เครื่องหมายตรรกะ 279
ตารางที่ 9-5 ตัวอย่างการตั้งชื่อตัวแปร 294

หน้า

ตารางที่ 10-1 แอตทริบิวต์เหตุการณ์ของภาษา HTML 350
ตารางที่ 10-2 ชื่อเหตุการณ์ที่พบบ่อย 366
ตารางที่ 11-1 ตัวดำเนินการทางคณิตศาสตร์ในภาษา PHP 385
ตารางที่ 11-2 ตัวดำเนินการเปรียบเทียบในภาษา PHP 386
ตารางที่ 11-3 ตัวดำเนินการทางตรรกะในภาษา PHP 386
ตารางที่ 11-4 ตัวดำเนินการกำหนดค่าในภาษา PHP 387
ตารางที่ 11-5 ตัวดำเนินการเพิ่ม/ลดค่าในภาษา PHP 387
ตารางที่ 11-6 รายการฟังก์ชันสำหรับการตรวจสอบข้อมูล 392



ส่วนนำ



ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน



ระบบสารสนเทศกับการพัฒนาเว็บแอปพลิเคชัน



ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน

บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน มุ่งเน้นให้ผู้อ่านมีความรู้พื้นฐานเกี่ยวกับเว็บแอปพลิเคชัน สร้างความเข้าใจถึงภาพรวมของเทคโนโลยีและองค์ประกอบที่เกี่ยวข้องในระบบนิเวศของเว็บ เพื่อให้ผู้อ่านสามารถเชื่อมโยงความรู้เบื้องต้นไปสู่การพัฒนาเว็บแอปพลิเคชันอย่างเป็นระบบซึ่งเป็นพื้นฐานสำคัญสำหรับการศึกษาในบทต่อไป เนื้อหาในบทนี้มีดังนี้

1. ความรู้เบื้องต้นเกี่ยวกับเว็บ
2. ความหมายและคุณลักษณะของเว็บแอปพลิเคชัน
3. กลไกการทำงานของเว็บแอปพลิเคชัน
4. ภาษาในการพัฒนาเว็บแอปพลิเคชัน
5. สรุปท้ายบท
6. กิจกรรมท้ายบท

1. ความรู้เบื้องต้นเกี่ยวกับเว็บ

เว็บเป็นหนึ่งในบริการที่มีอยู่ในเครือข่ายอินเทอร์เน็ต ได้รับการคิดค้นมาตั้งแต่ปีพ.ศ. 2532 โดยเซอร์ทิม เบอร์เนิร์ส-ลี (Sir Tim Berners-Lee) นักวิทยาศาสตร์คอมพิวเตอร์ชาวอังกฤษ (แทปส์ค็อต, 2567) มีความหมายและพัฒนาการของเว็บดังต่อไปนี้

1.1 ความหมายของเว็บ

เว็บ (Web) เป็นชื่อเรียกโดยย่อของคำว่าเวิลด์ไวด์เว็บ (World Wide Web: WWW) หมายถึง ระบบข้อมูลหรือสารสนเทศที่เชื่อมโยงกันผ่านเครือข่ายคอมพิวเตอร์ ผู้ใช้สามารถเข้าถึงข้อมูลหรือสารสนเทศรวมทั้งทรัพยากรต่าง ๆ โดยใช้โปรแกรมเว็บเบราว์เซอร์

1.2 องค์ประกอบของเว็บ

องค์ประกอบของเว็บมีดังนี้ (สุพรรณิการ์ ย่องซื่อ, 2563; Mozilla Contributors, n.d.)

1.2.1 หน้าเว็บ

หน้าเว็บ (Web page) หรือเอกสารเว็บ (Web Document) เป็นเอกสารที่สร้างขึ้นเพื่อใช้แสดงผลในโปรแกรมเว็บเบราว์เซอร์ หน้าเว็บแต่ละหน้าสามารถเชื่อมโยงถึงกันได้โดยใช้จุดเชื่อมโยง (Hyperlink) ที่อยู่ในหน้าเว็บ โดยพื้นฐานแล้วหน้าเว็บจะใช้ภาษาหลักในการสร้างเว็บดังต่อไปนี้

1) ภาษา HTML

ภาษา HTML (HyperText Markup Language) เป็นภาษาหลักในการกำหนดโครงสร้างเอกสาร มีการใช้แท็ก (Tag) เพื่อกำหนดองค์ประกอบต่าง ๆ ในหน้าเว็บ รวมทั้งกำหนดบทบาทให้กับเนื้อหาที่จะแสดงผล เช่น หัวเรื่อง ย่อหน้า รูปภาพ หัวข้อแบบมีลำดับและไม่มีลำดับ เป็นต้น

2) ภาษา CSS

ภาษา CSS (Cascading Style Sheets) เป็นภาษากำหนดรูปแบบและลักษณะการแสดงผลรวมทั้งการจัดวางองค์ประกอบหรือการวางเค้าโครงของการแสดงผล (Layout)

3) ภาษาประมวลผลฝั่งผู้ใช้บริการ

ภาษาประมวลผลฝั่งผู้ใช้บริการ (Client-Side Programming Language) เป็นภาษาคอมพิวเตอร์ที่ช่วยเพิ่มความสามารถให้กับหน้าเว็บในการโต้ตอบกับผู้ใช้และการประมวลผลข้อมูลในเว็บเบราว์เซอร์ โดยภาษา JavaScript เป็นภาษาประมวลผลฝั่งผู้ใช้บริการที่เป็นที่นิยมนำมาใช้ในหน้าเว็บเป็นอย่างมาก รหัสคำสั่งของภาษาประมวลผลที่ประมวลผลโดยเว็บเบราว์เซอร์จะเรียกว่าภาษาประมวลผลฝั่งผู้ใช้บริการ มีหน้าที่หลักเกี่ยวกับการปรับปรุงการแสดงผล การตรวจสอบแบบฟอร์มและการโต้ตอบกับการกระทำใด ๆ ของผู้ใช้ต่อหน้าเว็บ

4) ภาษาประมวลผลฝั่งผู้ให้บริการ

ภาษาประมวลผลฝั่งผู้ให้บริการ (Server-Side Programming Language) เป็นภาษาคอมพิวเตอร์ที่ทำหน้าที่ในการประมวลผลข้อมูลต่าง ๆ หรือเลือกเนื้อหาในเครื่องผู้ให้บริการเว็บก่อนที่จะส่ง

ผลลัพธ์กลับไปยังเว็บเบราว์เซอร์ ภาษาประมวลผลฝั่งผู้ให้บริการนี้ช่วยให้หน้าเว็บมีความเป็นพลวัต (Dynamic) กล่าวคือ ข้อมูลสามารถเปลี่ยนแปลงได้จากผลลัพธ์ของการประมวลผล เช่น การเรียกข้อมูลจากฐานข้อมูลเพื่อการแสดงผลในหน้าเว็บ การจัดเก็บข้อมูลจากผู้ใช้ลงในฐานข้อมูล เป็นต้น ภาษาประมวลผลที่สามารถประมวลผลในเครื่องผู้ให้บริการเว็บ เช่น ภาษา PHP และภาษา Python เป็นต้น

1.2.2 เว็บไซต์

เว็บไซต์ (Website) เป็นชุดของหน้าเว็บที่เชื่อมโยงกันและมักถูกจัดเก็บภายใต้ชื่อที่อยู่เว็บเดียวกัน (Domain Name) เปรียบเสมือนเป็นหนังสือหนึ่งเล่มที่ประกอบด้วยเนื้อหาหลาย ๆ หน้า และเนื้อหาแต่ละหน้ามีความสอดคล้องหรือมีวัตถุประสงค์แนวเดียวกัน เช่น เว็บไซต์ของหน่วยงาน ประกอบด้วยเว็บเพจหลายๆ หน้าโดยแบ่งตามหมวดหมู่ของเนื้อหาของหน่วยงานที่ต้องการเผยแพร่ เป็นต้น

1.2.3 โฮมเพจ

โฮมเพจ (Homepage) หรือหน้าเว็บหลัก หรือหน้าเว็บเริ่มต้น เป็นหน้าเว็บที่เครื่องผู้ให้บริการเว็บกำหนดไว้ให้แสดงเป็นหน้าแรกหรือเป็นหน้าหลักเมื่อมีการร้องขอหน้าเว็บโดยไม่มีการระบุทรัพยากรที่ต้องการเป็นการเฉพาะ

1.2.4 เว็บเบราว์เซอร์

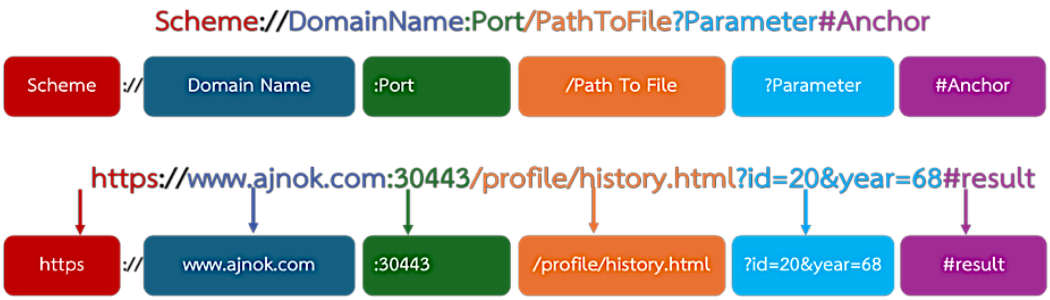
เว็บเบราว์เซอร์ (Web Browser) เป็นโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่ในการเรียกดูและแสดงผลหน้าเว็บ มีความสามารถพื้นฐานในการแปลและแสดงผลลัพธ์ที่ได้จากภาษา HTML ภาษา CSS และภาษา JavaScript ตัวอย่างของเว็บเบราว์เซอร์เช่น Google Chrome, Mozilla Firefox, Microsoft Edge และ Safari เป็นต้น

1.2.5 เครื่องผู้ให้บริการเว็บ

เครื่องผู้ให้บริการเว็บ (Web Server) เป็นเครื่องคอมพิวเตอร์ที่ทำหน้าที่ในการให้บริการหน้าเว็บหรือทรัพยากรอื่นๆ แก่เว็บเบราว์เซอร์ที่ร้องขอ มีหน้าที่สำคัญในการรอคอยหรือรับฟังการร้องขอหน้าเว็บ (Listen) การประมวลผลตามคำร้องขอ เช่น การค้นหาไฟล์หรือทรัพยากรที่ระบุ การติดต่อกับตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ และการส่งผลลัพธ์กลับไปยังเว็บเบราว์เซอร์ผู้ร้องขอ (Response) เป็นต้น เครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นเครื่องผู้ให้บริการเว็บนี้จึงควรมีสมรรถนะที่รองรับการร้องขอเป็นจำนวนมาก สามารถทำงานได้ต่อเนื่องตลอดเวลา อีกทั้งติดตั้งโปรแกรมที่ทำหน้าที่ในการให้บริการเว็บ เช่น Nginx, Apache HTTP Server หรือ Microsoft IIS เป็นต้น

1.2.6 URL

URL (Uniform Resource Locator) เป็นข้อความที่มีรูปแบบเป็นมาตรฐานที่ใช้ระบุตำแหน่งของทรัพยากรในเครือข่ายอินเทอร์เน็ตเพื่อนำมาแสดงผลในเว็บเบราว์เซอร์ เช่น หน้าเว็บ รูปภาพ หรือวิดีโอ เป็นต้น รูปแบบมาตรฐานของ URL และตัวอย่างของ URL แสดงดังภาพที่ 1.1



ภาพที่ 1.1 รูปแบบมาตรฐานของ URL และตัวอย่างของ URL

ที่มา: ปรับปรุงจาก (Mozilla Developer Network, n.d.c)

1) Scheme

Scheme เป็นส่วนที่ใช้บอกวิธีการสื่อสารหรือโปรโตคอลระหว่างเว็บเบราว์เซอร์และเครื่องผู้ให้บริการเว็บ โดยปกติแล้วโปรโตคอลสำหรับการเรียกดูหน้าเว็บจะใช้โปรโตคอล HTTP (HyperText Transfer Protocol) หรือ HTTPS (HyperText Transfer Protocol Secure) ทั้งนี้ โปรโตคอล HTTPS เป็นโปรโตคอลที่มีความปลอดภัยกว่า HTTP อันเนื่องมาจากโปรโตคอล HTTPS จะมีการตรวจสอบความถูกต้องของเครื่องผู้ให้บริการเว็บด้วยใบรับรองดิจิทัล มีการเข้ารหัสข้อมูลในการรับส่งข้อมูลระหว่างเว็บเบราว์เซอร์และเครื่องผู้ให้บริการเว็บ จากตัวอย่างของ URL ในภาพที่ 1.1 พบว่ามีการกำหนดโปรโตคอลที่ใช้ในการสื่อสารระหว่างเว็บเบราว์เซอร์และเครื่องผู้ให้บริการเว็บเป็น HTTPS

2) Domain Name

Domain Name หรือชื่อโดเมน เป็นระบบการใช้ชื่อแทนหมายเลขที่อยู่ทางอินเทอร์เน็ต (IP Address) ของเครื่องคอมพิวเตอร์ผู้ให้บริการเว็บ โดยในการสื่อสารของระบบเครือข่ายอินเทอร์เน็ตจะใช้หมายเลขที่อยู่ทางอินเทอร์เน็ตในการระบุต้นทางและปลายทาง อย่างไรก็ตาม เพื่อเป็นการเพิ่มความสะดวกและการจดจำชื่อของเครื่องคอมพิวเตอร์จึงมีการใช้ระบบชื่อโดเมน (Domain Name System: DNS) เพื่อเป็นตัวกลางในการแปลงชื่อโดเมนและหมายเลขที่อยู่ทางอินเทอร์เน็ต จากตัวอย่างของ URL ในภาพที่ 1.1 ชื่อโดเมนของเครื่องผู้ให้บริการเว็บคือ www.ajnok.com

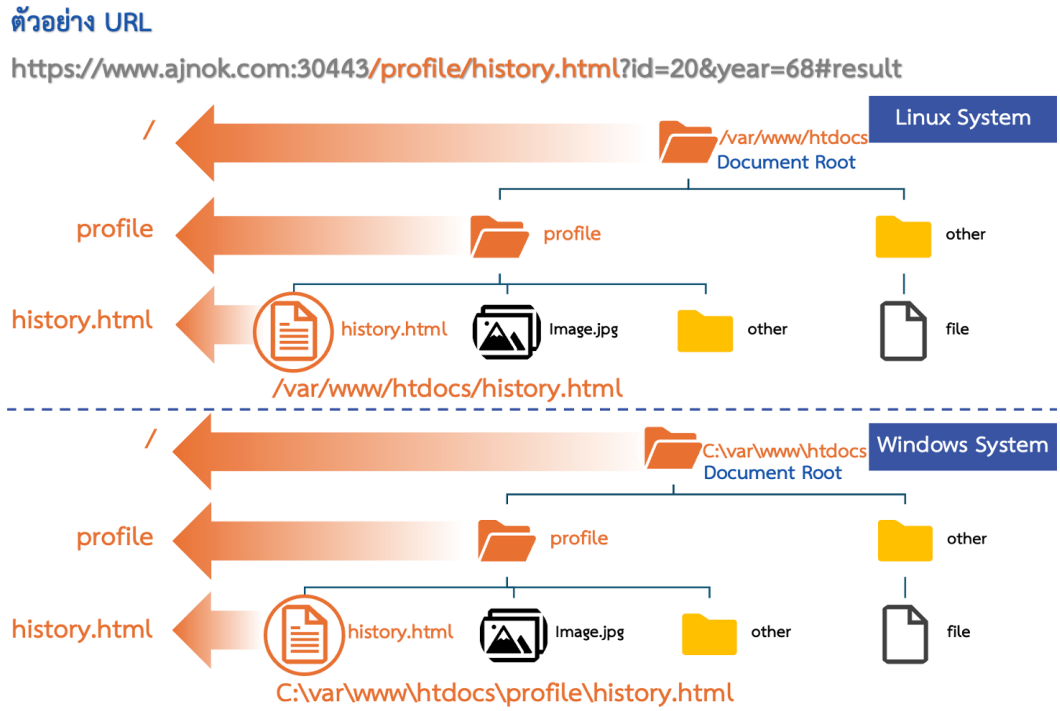
3) Port

Port เป็นหมายเลขกำกับบริการที่เปิดให้บริการในเครื่องคอมพิวเตอร์ผู้ให้บริการเว็บ เพื่อให้สามารถส่งข้อมูลไปยังโปรแกรมได้ถูกต้อง หมายเลขพอร์ตจะมีค่าในช่วง 0-65535 โดยปกติแล้วหากไม่มีการระบุ Port ใน URL เว็บเบราว์เซอร์จะใช้ค่าเริ่มต้นของโปรโตคอล ซึ่งโปรโตคอล HTTP มีค่าเริ่มต้นของ Port เป็น 80 และ HTTPS มีค่าเริ่มต้นของ Port เป็น 443 หากเครื่องผู้ให้บริการเว็บมีการกำหนดค่า Port ของแต่ละโปรโตคอลที่แตกต่างกันออกไป การเรียกดูหน้าเว็บจะต้องมีการกำหนด Port ให้ตรงกับค่าที่เครื่องผู้ให้บริการเว็บกำหนดด้วย

จากตัวอย่าง URL ในภาพที่ 1.1 มีการกำหนด Port เป็น 30443 หมายความว่า ผู้เรียกดูหน้าเว็บต้องการติดต่อไปยังเครื่องผู้ให้บริการเว็บที่ Port หมายเลข 30443 หากเครื่องผู้ให้บริการเว็บไม่ได้มีการกำหนดค่า Port ของการให้บริการเว็บที่ใช้โปรโตคอล HTTPS จากเดิม 443 เป็น 30443 ก็จะทำให้เกิดข้อผิดพลาดในการรับส่งข้อมูลระหว่างเว็บเบราว์เซอร์และเครื่องผู้ให้บริการเว็บ

4) Path to File

Path to File เป็นเส้นทางการเข้าถึงทรัพยากรที่ถูกจัดเก็บไว้ในเครื่องผู้ให้บริการเว็บและชื่อไฟล์ ในเครื่องคอมพิวเตอร์ผู้ให้บริการเว็บจะมีการกำหนดที่เก็บทรัพยากรในการให้บริการเว็บ (Document Root) ซึ่งเป็นจุดเริ่มต้นในการค้นหาไฟล์ของเครื่องผู้ให้บริการเว็บเมื่อมีการร้องขอ ภาพที่ 1.2 แสดงตัวอย่างของ URL ที่มีการระบุเส้นทางการเข้าถึงทรัพยากรที่ต้องการและการค้นหาในตำแหน่งที่เก็บทรัพยากรของระบบปฏิบัติการ Linux และ Microsoft Windows



ภาพที่ 1.2 ตัวอย่างการกำหนดตำแหน่งที่เก็บทรัพยากรในเครื่องผู้ให้บริการเว็บ

จากภาพที่ 1.2 เป็นตัวอย่างที่ระบบปฏิบัติการ Linux กำหนดที่เก็บทรัพยากรในการให้บริการเว็บเป็น /var/www/htdocs และในระบบปฏิบัติการ Microsoft Windows มีตำแหน่งเป็น C:\var\www\htdocs เมื่อมีการร้องขอทรัพยากรตาม ตัวอย่าง URL ในภาพ ลำดับการเข้าถึงจะเริ่มต้นจาก Document Root จากนั้นจะเข้าถึงไดเรกทอรีหรือโฟลเดอร์ชื่อ profile และนำไฟล์ history.html ที่อยู่ในไดเรกทอรีหรือโฟลเดอร์ดังกล่าวส่งกลับไปยังเว็บเบราว์เซอร์ ดังนั้นที่อยู่จริงของไฟล์ในเครื่องคอมพิวเตอร์

ผู้ให้บริการเว็บที่ใช้ระบบปฏิบัติการ Linux จึงเป็น /var/www/htdocs/profile/history.html ส่วนที่อยู่จริงของไฟล์ในเครื่องคอมพิวเตอร์ผู้ให้บริการเว็บที่ใช้ระบบปฏิบัติการ Microsoft Windows จะเป็น C:\var\www\htdocs\profile\history.html

ในกรณีที่เครื่องผู้ให้บริการเว็บค้นหาทรัพยากรตามเส้นทางการเข้าถึงทรัพยากรที่ระบุใน URL ไม่พบ เครื่องผู้ให้บริการเว็บก็จะส่งรหัสข้อผิดพลาดหมายเลข 404 กลับไปยังเว็บเบราว์เซอร์ แต่ถ้าไม่ได้มีการระบุเส้นทางการเข้าถึงทรัพยากรใน URL เช่น https://www.ajnok.com เครื่องผู้ให้บริการเว็บก็จะจัดส่งโฮมเพจกลับไปยังเว็บเบราว์เซอร์

5) Parameter

Parameters เป็นข้อมูลเพิ่มเติมที่ส่งให้กับเครื่องผู้ให้บริการเว็บที่อยู่ในรูปแบบของตัวแปรและค่าของตัวแปร โดยตัวแปรและค่าของตัวแปรจะเขียนตามหลังเครื่องหมาย ? และใช้เครื่องหมาย = ในการกำหนดระบุค่าของตัวแปร หากมีคู่ของตัวแปรและค่าของตัวแปรมากกว่า 1 คู่ จะใช้เครื่องหมาย & ในการคั่นแต่ละคู่ตัวแปร

จากตัวอย่างของ URL ในภาพที่ 1.2 ตัวอย่างการกำหนดตำแหน่งที่เก็บทรัพยากรในเครื่องผู้ให้บริการเว็บมีการส่งค่าข้อมูลเพิ่มเติมเป็น ?id=20&year=68 ให้กับเครื่องผู้ให้บริการเว็บ จะเห็นว่ามีการส่งค่าข้อมูลเพิ่มเติมจำนวน 2 ค่า ค่าแรกคือ id มีค่าเป็น 20 และค่าที่สองคือ year มีค่าเป็น 68 ข้อมูลเพิ่มเติมที่ส่งให้กับเครื่องผู้ให้บริการเว็บเหล่านี้อาจใช้เพื่อการประมวลผลในเครื่องผู้ให้บริการเว็บ เช่น การกรองข้อมูล การค้นหาข้อมูล การส่งค่าจากหน้าเว็บที่เป็นแบบฟอร์ม เป็นต้น หากเครื่องผู้ให้บริการเว็บไม่ได้มีการเขียนคำสั่งโดยใช้ภาษาประมวลผลเพื่อจัดการข้อมูลเพิ่มเติมเหล่านี้ ก็จะไม่ส่งผลต่อการดำเนินการของเครื่องผู้ให้บริการเว็บ แต่จะทำให้เกิดปริมาณข้อมูลในการรับส่งระหว่างเว็บเบราว์เซอร์และเครื่องผู้ให้บริการเว็บที่สูงขึ้นโดยไม่จำเป็น

6) Anchor

Anchor หรือจุดยึด เป็นส่วนที่ใช้ระบุตำแหน่งการแสดงผลของหน้าเว็บหรือส่วนอื่นของทรัพยากร จุดยึดเปรียบเสมือนกับที่คั่นหนังสือที่ช่วยให้เว็บเบราว์เซอร์เลื่อนการแสดงผลไปยังตำแหน่งที่คั่นนั้นได้ การระบุตำแหน่งจุดยึดที่จะแสดงผล จะเขียนตัวระบุส่วน (Fragment Identifier) ไว้หลังเครื่องหมาย # โดยตัวระบุส่วนนี้จะไม่ถูกส่งไปยังเครื่องผู้ให้บริการเว็บ แต่เว็บเบราว์เซอร์จะนำไปใช้ประกอบการประมวลผลหลังจากที่ได้รับทรัพยากรจากเครื่องผู้ให้บริการเว็บแล้ว

จากตัวอย่างของ URL ในภาพที่ 1.1 ส่วนของจุดยึดใน URL เป็น #sum หมายความว่าให้เลื่อนหน้าจอไปยังองค์ประกอบของเว็บที่มี id เป็น sum ในกรณีที่ใช้จุดยึดกับทรัพยากรประเภทเสียงหรือวีดีโอจะเป็นการเลื่อนไปยังตำแหน่งเวลาของเสียงหรือวีดีโอนั้นๆ แนวทางการใช้จุดยึดในหน้าเว็บ เช่น การนำทางในหน้าเว็บที่มีเนื้อหาที่ยาว การสร้างเมนูนำทางหรือสารบัญเพื่อนำทางไปยังเนื้อหาภายในหน้าเว็บเดียวกัน การอ้างอิงถึงส่วนต่าง ๆ ในหน้าเว็บ และการระบุตำแหน่งเริ่มต้นในไฟล์เสียงหรือวีดีโอ เป็นต้น

2. ความหมายและคุณลักษณะของเว็บแอปพลิเคชัน

ความหมายและคุณลักษณะของเว็บแอปพลิเคชัน มีดังนี้

2.1 ความหมายของเว็บแอปพลิเคชัน

เว็บแอปพลิเคชัน (Web Application) หมายถึง โปรแกรมประยุกต์ที่ทำงานบนเว็บเบราว์เซอร์โดยอาศัยเว็บเทคโนโลยีในการสร้างส่วนติดต่อและโต้ตอบกับผู้ใช้ มีการส่งข้อมูลระหว่างผู้ใช้และเครื่องผู้ให้บริการเว็บผ่านโปรโตคอล HTTP หรือ HTTPS โดยการประมวลผลและการจัดการข้อมูลที่ได้จากผู้ใช้จะเกิดขึ้นในเครื่องผู้ให้บริการเป็นหลัก ตัวอย่างของเว็บแอปพลิเคชัน เช่น อีเมล ปฏิทินออนไลน์ ระบบการจัดการเนื้อหา และระบบซื้อขายออนไลน์ เป็นต้น (World Wide Web Consortium, 2009; Sturm, Pollard, & Craig, 2017; อมีนา ฉายสุวรรณ และ ชุมพล จันทร์ฉลอง, 2566; Amazon Web Services, Inc., 2023; สุพรรณิการ์ ย่องซื่อ, 2563)

2.2 ความหมายของเว็บแอปพลิเคชันแบบตอบสนอง

เว็บแอปพลิเคชันแบบตอบสนอง (Responsive Web Application) หมายถึง เว็บแอปพลิเคชันที่สามารถปรับเปลี่ยนรูปแบบการแสดงผลให้เหมาะสมกับขนาดหน้าจอของอุปกรณ์ที่ใช้งาน เช่น คอมพิวเตอร์ แท็บเล็ต หรือสมาร์ตโฟน เป็นต้น โดยไม่จำเป็นต้องพัฒนาเว็บแอปพลิเคชันแยกสำหรับแต่ละอุปกรณ์ ระบบจะปรับขนาด รูปแบบ และตำแหน่งขององค์ประกอบในหน้าจอโดยอัตโนมัติเพื่อให้เกิดประสบการณ์ใช้งานที่ดีแก่ผู้ใช้ (User Experience) อย่างสม่ำเสมอในทุกอุปกรณ์ หลักการพัฒนาเว็บแอปพลิเคชันแบบตอบสนองจะอาศัยแนวคิดการออกแบบที่เรียกว่า Responsive Web Design (RWD) ซึ่งประกอบด้วยการใช้หน่วยวัดแบบยืดหยุ่น เช่น % หรือ em เป็นต้น รวมทั้งการจัดโครงร่างขององค์ประกอบด้วยระบบ Grid หรือ Flexbox และการใช้คำสั่งในภาษา CSS เพื่อกำหนดเงื่อนไขการแสดงผลในแต่ละช่วงความกว้างของหน้าจอ

2.3 ความแตกต่างระหว่างเว็บไซต์และเว็บแอปพลิเคชัน

ความแตกต่างที่สำคัญระหว่างเว็บไซต์กับเว็บแอปพลิเคชัน คือ เว็บไซต์มุ่งเน้นการนำเสนอข้อมูล เช่น ข่าวสาร บทความ หรือสื่อ เป็นต้น โดยผู้ใช้จะโต้ตอบกับระบบในระดับพื้นฐานในลักษณะการนำทางไปยังเนื้อหาส่วนต่าง ๆ ของเว็บไซต์ ภาษาหลักที่ใช้ในการสร้างเว็บไซต์จะเป็นภาษาสำหรับการสร้างส่วนติดต่อกับผู้ใช้ ได้แก่ ภาษา HTML ภาษา CSS รวมทั้งภาษาประมวลผลฝั่งผู้ใช้บริการเพื่อแสดงผลผ่านเว็บเบราว์เซอร์ ในขณะที่เว็บแอปพลิเคชันจะมุ่งเน้นการมีปฏิสัมพันธ์ในการนำข้อมูลจากผู้ใช้ไปประมวลผลและจัดเก็บในเครื่องผู้ให้บริการเว็บและแสดงผลลัพธ์ในเว็บเบราว์เซอร์ตามคำสั่งที่ผู้ใช้สั่งการ ภาษาหลักที่ใช้ในการสร้างเว็บแอปพลิเคชันนอกจากจะใช้ภาษาสำหรับการสร้างส่วนติดต่อกับผู้ใช้เช่นเดียวกับเว็บไซต์แล้ว ยังมีการใช้ภาษาประมวลผลฝั่งผู้ให้บริการและภาษาจัดการฐานข้อมูลเพื่อรองรับการประมวลผลและจัดเก็บข้อมูล (Faradilla, 2025)

2.4 คุณลักษณะสำคัญของเว็บแอปพลิเคชัน

เว็บแอปพลิเคชันมีความสามารถในการโต้ตอบหรือมีปฏิสัมพันธ์ (Interactive) กับผู้ใช้เช่นเดียวกับแอปพลิเคชันที่ติดตั้งบนคอมพิวเตอร์หรืออุปกรณ์เคลื่อนที่ เป็นวิวัฒนาการของเว็บที่แต่เดิมเป็นการแบ่งปันเนื้อหา

ที่คงที่ (Static Web) สู่การแสดงเนื้อหาลักษณะพลวัต (Dynamic Web) โดยสามารถแสดงเนื้อหาที่เปลี่ยนแปลงไปตามบริบทการดำเนินการและการประมวลผลได้ เว็บแอปพลิเคชันมีคุณลักษณะสำคัญดังนี้ (Ramotion, 2024a; Amazon Web Services, Inc., 2023)

2.4.1 การเข้าถึงจากทุกที่ทุกเวลา

เว็บแอปพลิเคชันสามารถเข้าถึงได้จากทุกอุปกรณ์ที่มีการเชื่อมต่ออินเทอร์เน็ตและเว็บเบราว์เซอร์ ทำให้มีความสะดวกและยืดหยุ่นสูง

2.4.2 ความเข้ากันได้กับหลากหลายแพลตฟอร์ม

เว็บแอปพลิเคชันสามารถทำงานได้บนระบบปฏิบัติการและอุปกรณ์ที่แตกต่างกัน โดยไม่จำเป็นต้องพัฒนาแอปพลิเคชันแยกสำหรับแต่ละแพลตฟอร์มเนื่องจากเป็นการทำงานผ่านเว็บเบราว์เซอร์

2.4.3 ความสามารถในการปรับขนาด

เว็บแอปพลิเคชันรองรับการขยายตัวของผู้ใช้งานได้อย่างมีประสิทธิภาพ โดยอาศัยระบบคลาวด์ (Cloud System) ในการปรับเปลี่ยนทรัพยากรตามความต้องการของผู้พัฒนาได้

2.4.4 การปรับรุ่นให้เป็นปัจจุบันอัตโนมัติ

ผู้ใช้งานได้รับการปรับรุ่นของเว็บแอปพลิเคชันให้เป็นรุ่นล่าสุดโดยอัตโนมัติโดยไม่ต้องดำเนินการด้วยตนเอง เนื่องจากเว็บแอปพลิเคชันจะถูกติดตั้งในเครื่องคอมพิวเตอร์ผู้ให้บริการ

2.4.5 การทำงานร่วมกัน

เว็บแอปพลิเคชันสามารถพัฒนาให้เอื้ออำนวยต่อการทำงานร่วมกันระหว่างผู้ใช้หลายคน โดยสามารถแชร์และแก้ไขข้อมูลร่วมกันได้แบบเรียลไทม์

2.4.6 การจัดเก็บข้อมูลแบบรวมศูนย์

ข้อมูลจากการใช้เว็บแอปพลิเคชันจะถูกจัดเก็บบนเครื่องผู้ให้บริการส่วนกลางหรือในระบบคลาวด์ ช่วยให้การจัดการและการเข้าถึงข้อมูลมีประสิทธิภาพมากขึ้น

2.4.7 ความต้องการการเชื่อมต่ออินเทอร์เน็ต

จากการที่เว็บแอปพลิเคชันถูกติดตั้งหรือประมวลผลในเครื่องผู้ให้บริการ ผู้ใช้งานเว็บแอปพลิเคชันจึงจำเป็นต้องมีการเชื่อมต่อเครือข่ายคอมพิวเตอร์เพื่อเข้าถึงเครื่องผู้ให้บริการนั้น ซึ่งคุณลักษณะข้อนี้อาจเป็นข้อจำกัดในบางสถานการณ์

2.4.8 ประสิทธิภาพการทำงาน

เว็บแอปพลิเคชันอาจมีความเร็วในการตอบสนองช้ากว่าแอปพลิเคชันที่ติดตั้งในคอมพิวเตอร์ เนื่องจากต้องส่งข้อมูลผ่านเครือข่ายคอมพิวเตอร์

2.4.9 ความปลอดภัย

เว็บแอปพลิเคชันมีความเสี่ยงด้านความปลอดภัยมากกว่าแอปพลิเคชันที่ติดตั้งในคอมพิวเตอร์ เนื่องจากมีการเข้าถึงผ่านเครือข่ายคอมพิวเตอร์

2.4.10 รูปแบบการคิดค่าบริการ

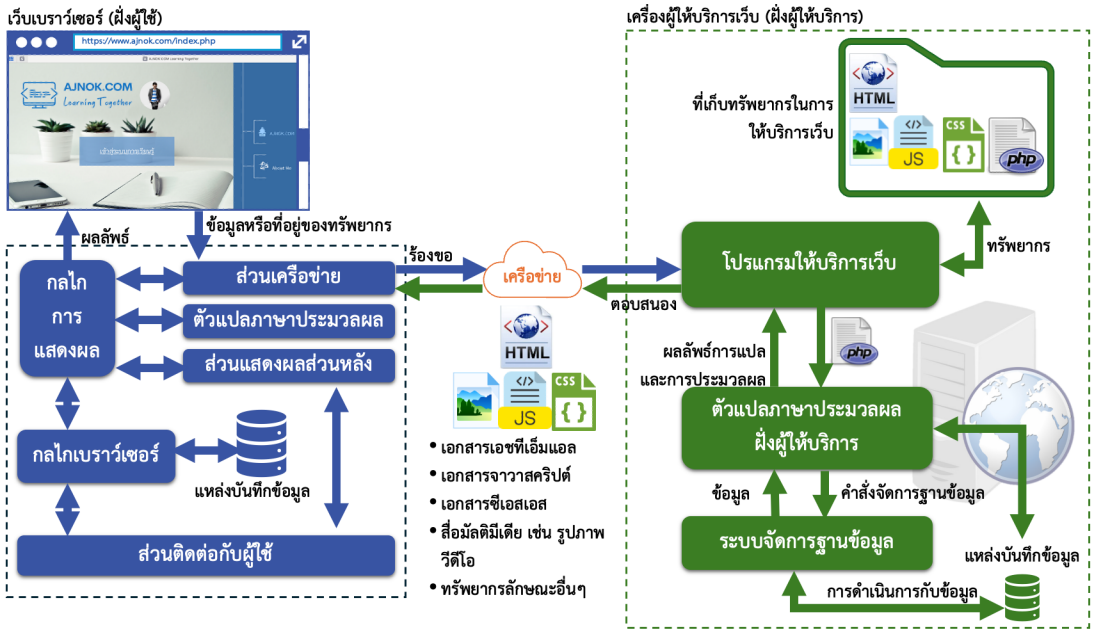
ในการใช้เว็บแอปพลิเคชัน ผู้ให้บริการมีรูปแบบการคิดค่าบริการทั้งการให้บริการโดยไม่คิดค่าใช้จ่าย และรูปแบบการสมัครสมาชิกรายเดือนหรือรายปี

2.4.11 ความสามารถในการปรับแต่ง

เว็บแอปพลิเคชันมีข้อจำกัดในการปรับแต่งมากกว่าแอปพลิเคชันสำหรับติดตั้งในคอมพิวเตอร์ โดยผู้ใช้สามารถปรับแต่งได้เฉพาะส่วนที่ผู้พัฒนากำหนดไว้เท่านั้น

3. กลไกการทำงานของเว็บแอปพลิเคชัน

การทำงานของเว็บแอปพลิเคชันมีส่วนประกอบที่ทำงานร่วมกันทั้งในฝั่งผู้ใช้และฝั่งผู้ให้บริการเว็บ โดยในฝั่งผู้ใช้มีส่วนประกอบสำคัญที่เกี่ยวข้องกับการทำงานของเว็บแอปพลิเคชันคือเว็บเบราว์เซอร์ และในฝั่งผู้ให้บริการเว็บก็มีส่วนประกอบที่เกี่ยวข้องกับการทำงานของเว็บแอปพลิเคชันเช่นกัน ในหัวข้อนี้จะกล่าวถึง 1) ส่วนประกอบพื้นฐานของเว็บเบราว์เซอร์ 2) ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บ และ 3) ขั้นตอนการทำงานของเว็บแอปพลิเคชัน เพื่อให้เข้าใจกลไกการทำงานของเว็บแอปพลิเคชันในภาพรวมตั้งแต่การที่ผู้ใช้เปิดเว็บเบราว์เซอร์จนกระทั่งผู้ใช้ได้ผลลัพธ์จากเครื่องผู้ให้บริการ โดยภาพรวมส่วนประกอบและการทำงานของเว็บแอปพลิเคชันแสดงดังภาพที่ 1.3



ภาพที่ 1.3 ภาพรวมส่วนประกอบและการทำงานของเว็บแอปพลิเคชัน

ที่มา: ปรับปรุงจาก Mozilla Developer Network (2025) Ramotion (2024b) และ Kosaka (2020)

3.1 ส่วนประกอบพื้นฐานของเว็บเบราว์เซอร์

ส่วนประกอบพื้นฐานของเว็บเบราว์เซอร์แสดงดังภาพที่ 1.4 ประกอบด้วย ส่วนติดต่อกับผู้ใช้ กลไกเบราว์เซอร์ แหล่งบันทึกข้อมูล กลไกการแสดงผล ส่วนเครือข่าย ตัวแปลภาษาประมวลผล และส่วนแสดงผลส่วนหลัง



ภาพที่ 1.4 ส่วนประกอบพื้นฐานของเว็บเบราว์เซอร์

3.1.1 ส่วนติดต่อกับผู้ใช้

ส่วนติดต่อกับผู้ใช้ (User Interface) เป็นส่วนที่ผู้ใช้สามารถมองเห็นและโต้ตอบได้โดยตรง รวมถึงส่วนประกอบทั้งหมดของโปรแกรมเว็บเบราว์เซอร์ที่ไม่เกี่ยวข้องกับการแสดงผลหน้าเว็บเพจโดยตรง เช่น แถบที่อยู่ (Address Bar) ปุ่มนำทาง (Navigation Buttons) ปุ่มย้อนกลับ/ไปข้างหน้า ปุ่มรีเฟรช แท็บ (Tabs) เมนู (Menus) พื้นที่แสดงผลเนื้อหาเว็บ และแถบแสดงสถานะ (Status Bar) เป็นต้น

3.1.2 กลไกเบราว์เซอร์

กลไกเบราว์เซอร์ (Browser Engine) ทำหน้าที่เป็นตัวกลางระหว่างส่วนติดต่อกับผู้ใช้และกลไกการแสดงผล กลไกเบราว์เซอร์นี้ตอบสนองต่อการกระทำของผู้ใช้ที่มีต่อโปรแกรมเว็บเบราว์เซอร์ เช่น การโหลดหน้าเว็บใหม่ การเลื่อนหน้าเว็บ เป็นต้น

3.1.3 แหล่งบันทึกข้อมูล

แหล่งบันทึกข้อมูล (Data Persistence/Storage) เป็นส่วนที่ช่วยให้โปรแกรมเว็บเบราว์เซอร์สามารถจัดเก็บข้อมูลไว้ในเครื่องของผู้ใช้ ประกอบด้วยกลไกการจัดเก็บข้อมูลหลายรูปแบบ เช่น กลไกคุกกี้ (Cookies) ใช้จัดเก็บข้อมูลขนาดเล็กที่เว็บไซต์ส่งไปยังโปรแกรมเว็บเบราว์เซอร์ ใช้สำหรับการติดตามสถานะผู้ใช้และการตั้งค่าต่าง ๆ กลไก Local Storage ใช้จัดเก็บข้อมูลแบบคู่กุญแจและค่า (Key-Value) แบบไม่มีวัน

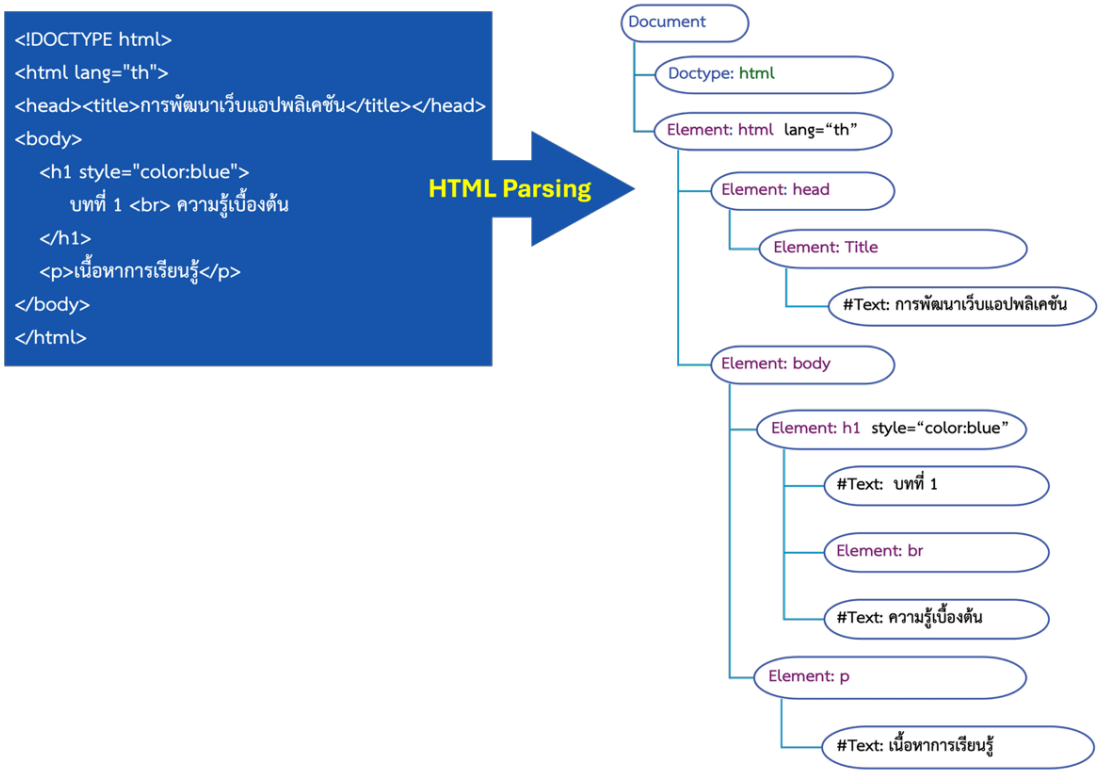
หมดอายุในโปรแกรมเว็บเบราว์เซอร์ กลไก Session Storage จัดเก็บข้อมูลคล้ายกับ Local Storage แต่ข้อมูลจะถูกลบเมื่อถึงกำหนดเวลารวมทั้งเมื่อปิดแท็บหรือเบราว์เซอร์ และกลไก Cache Storage ใช้สำหรับจัดเก็บการตอบสนองจากเครื่องผู้ให้บริการ เพื่อให้สามารถเข้าถึงได้แม้ไม่มีการเชื่อมต่ออินเทอร์เน็ต เป็นต้น

3.1.4 กลไกการแสดงผล

กลไกการแสดงผล (Rendering Engine) มีหน้าที่ในการแสดงผลลัพธ์จากภาษา HTML ภาษา CSS และภาษา JavaScript ให้เป็นหน้าเว็บที่ผู้ใช้สามารถมองเห็น เว็บเบราว์เซอร์ของผู้ผลิตเว็บเบราว์เซอร์แต่ละรายจะมีการใช้กลไกการแสดงผลที่ต่างกัน เช่น กลไกการแสดงผล WebKit เป็นกลไกการแสดงผลที่พัฒนาโดย Apple นำใช้ในเว็บเบราว์เซอร์ Safari กลไกการแสดงผล Blink เป็นกลไกการแสดงผลที่พัฒนาโดย Google นำใช้ในเว็บเบราว์เซอร์ Google Chrome และ Microsoft Edge และกลไกการแสดงผล Gecko ใช้ในเว็บเบราว์เซอร์ Mozilla Firefox พัฒนาโดย Mozilla เป็นต้น กลไกการแสดงผลมีส่วนประกอบพื้นฐานดังนี้

1) ส่วนการแยกวิเคราะห์ HTML

ส่วนการแยกวิเคราะห์ HTML (HTML Parsing) เป็นการแปลงรหัสภาษา HTML ให้เป็นโครงสร้างข้อมูลที่มีลักษณะคล้ายต้นไม้ที่เรียกว่าโครงสร้างต้นไม้ DOM (Document Object Model Tree) แสดงดังภาพที่ 1.5



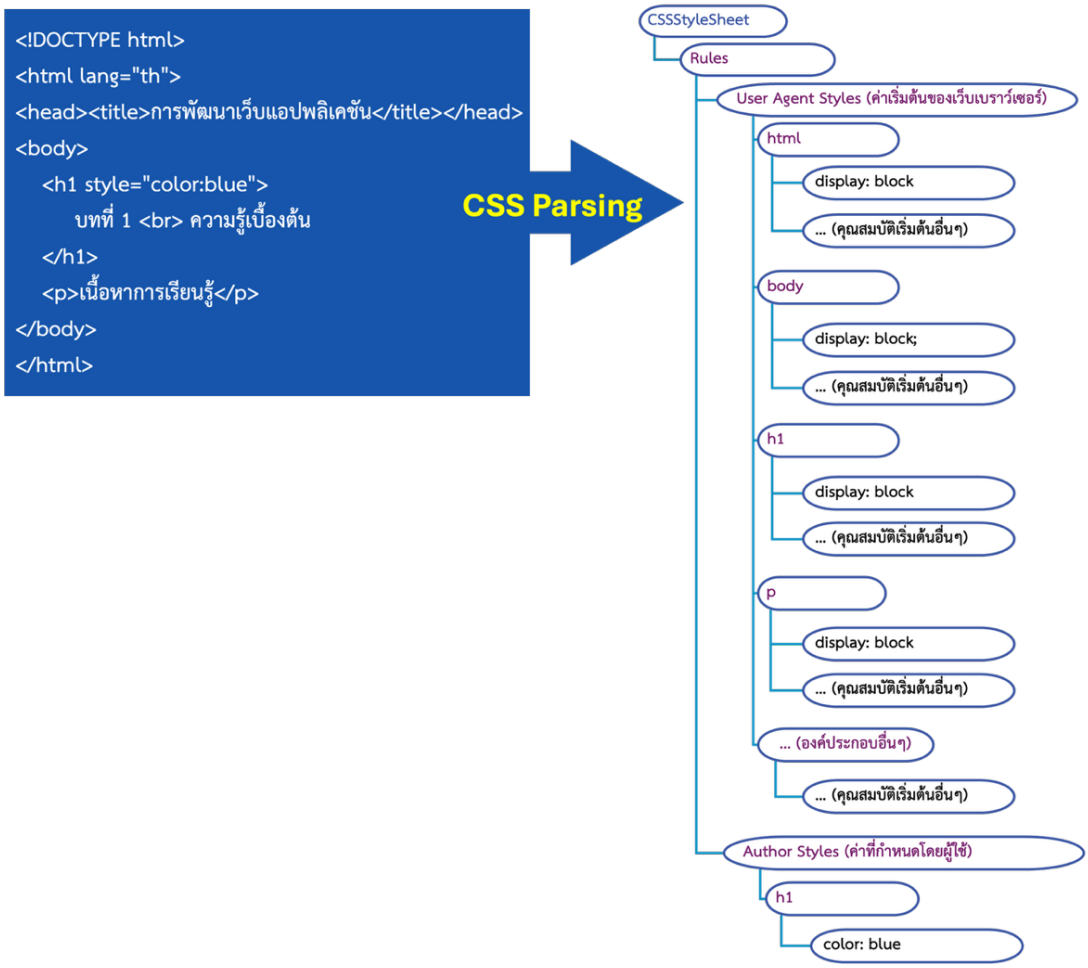
ภาพที่ 1.5 การแยกวิเคราะห์ HTML เป็นโครงสร้างต้นไม้ DOM

ที่มา: ปรับปรุงจาก WHATWG (DOM Standard – Introduction to the DOM, n.d.)

2) ส่วนการแยกวิเคราะห์ CSS

ส่วนการแยกวิเคราะห์ CSS (CSS Parsing) เป็นส่วนแปลงรหัส CSS (Cascading Style Sheet) ในเอกสาร HTML ให้เป็นโครงสร้างต้นไม้ที่เรียกว่า CSSOM (CSS Object Model) โดยส่วนการแยกวิเคราะห์ CSS จะทำการพิจารณาว่าส่วนใดของเอกสารเว็บที่มีการกำหนดกฎการแสดงผลด้วย CSS บ้าง ก็จะมีการนำมาสร้างเป็นโครงสร้างต้นไม้ในส่วนค่าที่กำหนดโดยผู้ใช้ (Author Styles) ลักษณะดังภาพที่ 1.6

ทั้งนี้ หากองค์ประกอบใดของ HTML ไม่มีการกำหนดกฎการแสดงผล เว็บเบราว์เซอร์แต่ละรายจะมีค่าเริ่มต้นของกฎการแสดงผลที่แตกต่างกันออกไปโดยจะถูกกำหนดในส่วน User Agent Styles ทำให้การแสดงผล เว็บที่เขียนโดยรหัสคำสั่งแบบเดียวกัน เมื่อนำไปเปิดด้วยเว็บเบราว์เซอร์ที่ต่างกันก็อาจทำให้มีการแสดงผลแตกต่างกัน

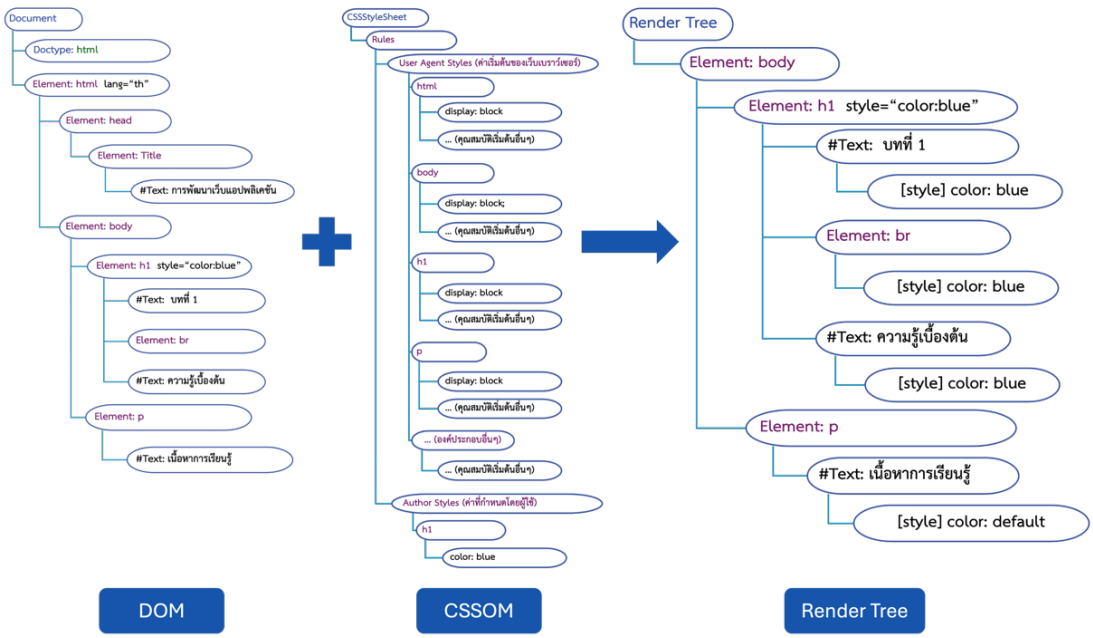


ภาพที่ 1.6 การแยกวิเคราะห์ CSS เป็นโครงสร้างต้นไม้ CSSOM

ที่มา: ปรับปรุงจาก Ilya, G. (2014a)

3) ส่วนการสร้างโครงสร้างต้นไม้การแสดงผล

ส่วนการสร้างโครงสร้างต้นไม้การแสดงผล (Render Tree Construction) เป็นขั้นตอนที่เชื่อมโยง DOM และ CSSOM เข้าด้วยกันเพื่อสร้างเป็นโครงสร้างต้นไม้การแสดงผล มีวัตถุประสงค์เพื่อใช้ในการแสดงผลเนื้อหาพร้อมรูปแบบที่มองเห็นได้จริงบนส่วนแสดงผลของเว็บเบราว์เซอร์ เมื่อมีการรวมแล้วจะทำให้เว็บเบราว์เซอร์ทราบว่าแต่ละองค์ประกอบของเว็บองค์ประกอบใดบ้างที่จะมีการแสดงผล และแต่ละองค์ประกอบจะมีลักษณะการแสดงผลอย่างไร แสดงดังภาพที่ 1.7



ภาพที่ 1.7 โครงสร้างต้นไม้การแสดงผล

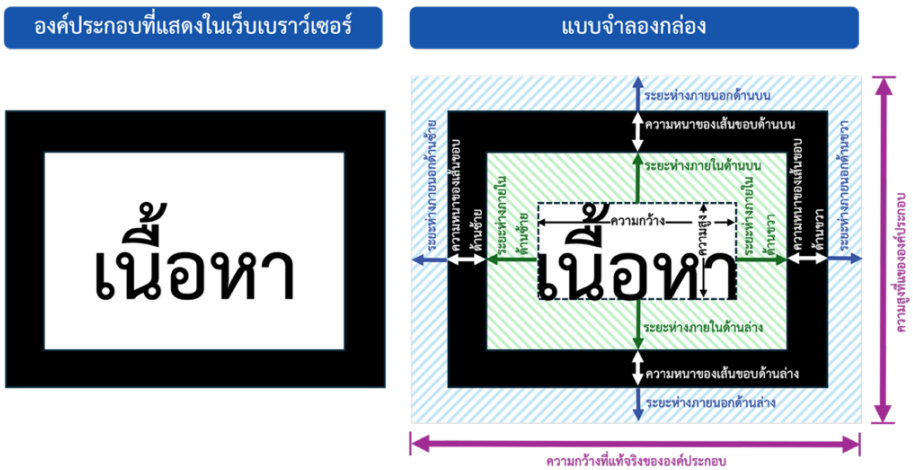
ที่มา: ปรับปรุงจาก Ilya G. (2014b)

ลักษณะที่สำคัญของโครงสร้างต้นไม้การแสดงผล (Mozilla Developer Network , 2025) มีดังนี้

- มีเฉพาะเนื้อหาที่สามารถมองเห็นได้ (Visible Content) เท่านั้นที่จะถูกนำมารวมไว้ในโครงสร้างต้นไม้การแสดงผล
- เนื้อหาในองค์ประกอบ head ตามโครงสร้างของภาษา HTML จะไม่ถูกรวมอยู่ในโครงสร้างต้นไม้การแสดงผล เนื่องจากไม่มีองค์ประกอบใดๆ ที่จะปรากฏให้ผู้ใช้เห็นได้ในพื้นที่การแสดงผลของเว็บเบราว์เซอร์
- ถ้ามีองค์ประกอบใดที่กำหนดคุณสมบัติการแสดงผลเป็น display: none ด้วย CSS องค์ประกอบนั้นและองค์ประกอบภายในทั้งหมดจะไม่นำมาไว้ในโครงสร้างต้นไม้การแสดงผลนี้

4) ส่วนการจัดวาง

ส่วนการจัดวาง (Layout) เป็นส่วนที่มีหน้าที่ในการกำหนดตำแหน่งและขอบเขตขององค์ประกอบที่จะปรากฏบนหน้าเว็บ เช่น ความกว้าง ความสูง และความสัมพันธ์ทางด้านตำแหน่งระหว่างองค์ประกอบต่าง ๆ หลังจากที่มีการสร้างโครงสร้างต้นไม้การแสดงผลเสร็จสมบูรณ์แล้ว ส่วนการจัดวางจะทำการคำนวณตำแหน่งและขนาดของแต่ละองค์ประกอบที่มีอยู่ในโครงสร้างต้นไม้การแสดงผลโดยอาศัยหลักการของแบบจำลองกล่อง (Box Model) แบบจำลองกล่องนี้เป็นแนวคิดพื้นฐานของ CSS ที่มองว่าทุกองค์ประกอบบนหน้าเว็บมีลักษณะเป็นกล่องสี่เหลี่ยม (Box) ซึ่งประกอบด้วย เนื้อหา (Content) ระยะห่างภายใน (Padding) เส้นขอบ (Border) และระยะห่างภายนอก (Margin) ดังภาพที่ 1.8

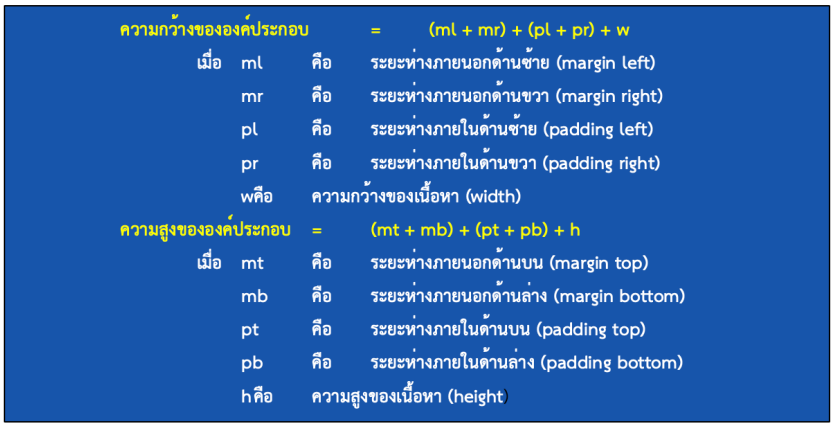


ภาพที่ 1.8 องค์ประกอบที่แสดงในเว็บเบราว์เซอร์และแบบจำลองกล่อง

ที่มา: ปรับปรุงจาก W3C (2024)

จากภาพที่ 1.8 แสดงให้เห็นถึงการแสดงผลขององค์ประกอบในเว็บเบราว์เซอร์ โดยส่วนที่เป็นพื้นที่ระยะห่างภายในและระยะห่างภายนอกจะไม่มีการแสดงผลให้ผู้ใช้เห็น แต่จะนำมาใช้ในการคำนวณหาขนาดที่แท้จริงขององค์ประกอบ ตามปกติแล้วคุณสมบัติการคำนวณขนาดขององค์ประกอบ (box-sizing) จะเป็นแบบ content-box ที่การคำนวณความกว้างที่แท้จริงขององค์ประกอบคำนวณได้มาจากผลรวมของระยะห่างภายนอกด้านซ้ายและด้านขวาของเนื้อหา ความหนาของเส้นขอบด้านซ้ายและด้านขวาของเนื้อหา ระยะห่างภายในด้านซ้ายและด้านขวาของเนื้อหา และความกว้างของเนื้อหา

สำหรับความสูงที่แท้จริงขององค์ประกอบคำนวณได้จากผลรวมของระยะห่างภายนอกด้านบนและด้านล่างของเนื้อหา ความหนาของเส้นขอบด้านบนและด้านล่างของเนื้อหา และความสูงของเนื้อหาแสดงดังภาพที่ 1.9



ภาพที่ 1.9 การคำนวณขนาดขององค์ประกอบที่แสดงในเว็บเบราว์เซอร์ตามแบบจำลองกล่อง

ค่าเริ่มต้นในการพิจารณาตำแหน่งขององค์ประกอบที่จะแสดงในเว็บเบราว์เซอร์ ส่วนการจัดวางจะทำการคำนวณตำแหน่งการแสดงผลตามลำดับขององค์ประกอบในโครงสร้างต้นไม้การแสดงผล นำขนาดพื้นที่แสดงผลของเว็บเบราว์เซอร์ (Viewport) มาใช้ร่วมกันในการคำนวณตำแหน่งที่จะแสดงผล จำนวนองค์ประกอบในโครงสร้างต้นไม้ DOM มีผลโดยตรงต่อประสิทธิภาพของส่วนการจัดวาง ยิ่งมีจำนวนองค์ประกอบมาก การคำนวณตำแหน่งจะใช้เวลานานขึ้น ซึ่งอาจส่งผลให้การแสดงหน้าเว็บไม่ราบรื่น นอกจากนี้ ทุกครั้งที่มีการเปลี่ยนแปลงในโครงสร้างต้นไม้การแสดงผล เช่น การเพิ่มองค์ประกอบ การเปลี่ยนเนื้อหา เว็บเบราว์เซอร์จะต้องจัดทำการคำนวณการจัดวางใหม่อีกครั้ง (Mozilla Developer Network , 2025)

5) ส่วนการวาด

ส่วนการวาด (Painting) เป็นส่วนที่นำองค์ประกอบในโครงสร้างต้นไม้การแสดงผลร่วมกับผลการคำนวณตำแหน่งและขนาดขององค์ประกอบต่าง ๆ ที่ได้จากส่วนการจัดวาง มาทำการแสดงหรือวาดบนจอภาพตามลำดับที่ปรากฏในโครงสร้างต้นไม้การแสดงผล และคุณสมบัติอื่น ๆ เพื่อให้ลำดับการแสดงผลมีความถูกต้อง เว็บเบราว์เซอร์สมัยใหม่ได้พัฒนากลไกเพิ่มประสิทธิภาพในการวาดด้วยการสร้างโครงสร้างข้อมูลสำหรับแยกชั้นการวาด (Layer Tree) การแยกชั้นการวาดยังช่วยให้การประมวลผลแบบขนาน (Parallel Processing) ทำได้ดีขึ้นเนื่องจากแต่ละชั้นของการวาดสามารถประมวลผลแยกกันได้ การเปลี่ยนแปลงคุณสมบัติบางอย่างขององค์ประกอบ เช่น สีพื้นหลัง ข้อความ หรือเส้นขอบ จะกระตุ้นให้เกิดการวาดใหม่ (Repaint) ซึ่งอาจทำให้ประสิทธิภาพการทำงานของเว็บเบราว์เซอร์ลดลงได้ (Kosaka, 2020; Mozilla Developer Network , 2025)

3.1.5 ส่วนเครือข่าย

ส่วนเครือข่าย (Network Component) เป็นองค์ประกอบพื้นฐานของเว็บเบราว์เซอร์ที่ทำหน้าที่ในการจัดการการสื่อสารระหว่างเว็บเบราว์เซอร์กับเครื่องผู้ให้บริการเว็บผ่านโปรโตคอล HTTP หรือ HTTPS เมื่อผู้ใช้ป้อน URL หรือคลิกลิงก์ในหน้าเว็บ ส่วนเครือข่ายจะแปลงคำขอให้อยู่ในรูปแบบข้อมูลที่กำหนดในโปรโตคอล HTTP หรือ HTTPS จากนั้นสร้างการเชื่อมต่อไปยังเครื่องผู้ให้บริการเว็บปลายทาง ส่งคำขอ และรอรับข้อมูลตอบกลับ เมื่อส่วนเครือข่ายได้รับข้อมูลตอบกลับแล้วก็จะส่งต่อข้อมูลที่ได้รับไปยังส่วนการทำงานอื่นๆ ของ

เว็บเบราว์เซอร์ ส่วนเครือข่ายของเว็บเบราว์เซอร์นี้ยังทำหน้าที่ด้านความปลอดภัยโดยตรวจสอบใบรับรองดิจิทัลเมื่อมีการเชื่อมต่อระหว่างเว็บเบราว์เซอร์กับเครื่องผู้ให้บริการเว็บโดยใช้โปรโตคอล HTTPS

3.1.6 ตัวแปลภาษาจาวาสคริปต์

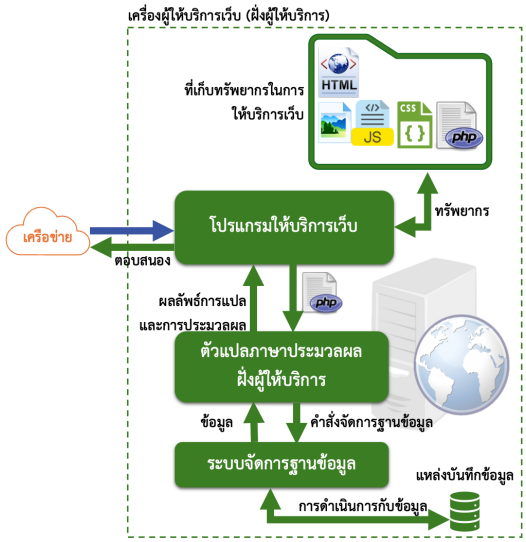
ตัวแปลภาษาจาวาสคริปต์ (JavaScript Translator) มีหน้าที่ในการแปลและดำเนินการตามชุดคำสั่งภาษาจาวาสคริปต์ ทำให้เว็บมีความสามารถในการโต้ตอบกับผู้ใช้ มีความสามารถในการจัดการองค์ประกอบในโครงสร้าง DOM และสามารถประมวลผลข้อมูลได้ เว็บเบราว์เซอร์ของผู้พัฒนาแต่ละรายใช้ตัวแปลภาษาจาวาสคริปต์ที่แตกต่างกันไป เช่น เว็บเบราว์เซอร์ Google Chrome และ Microsoft Edge ใช้ V8 เป็นตัวแปลภาษา เว็บเบราว์เซอร์ Mozilla Firefox ใช้ SpiderMonkey เป็นต้น (Mozilla Developer Network, 2025) ตัวแปลภาษาจาวาสคริปต์มีการพัฒนาความสามารถในการแปลภาษา โดยเริ่มแรกทำงานในลักษณะแปลหรือตีความทีละบรรทัดคำสั่ง (Interpreter) สู่การทำงานในลักษณะ Just-In-Time Compiler ทำให้มีประสิทธิภาพในการทำงานมากยิ่งขึ้น

3.1.7 ส่วนแสดงผลส่วนหลัง

ส่วนแสดงผลส่วนหลัง (UI Backend) เป็นส่วนที่ทำหน้าที่ในการจัดการการแสดงผลขั้นพื้นฐาน เช่น การวาดกรอบ ปุ่ม หรือหน้าต่างโต้ตอบ โดยใช้ฟังก์ชันของระบบปฏิบัติการ ทำงานร่วมกับกลไกการแสดงผลของเว็บเบราว์เซอร์ในการแปลงข้อมูลโครงสร้างต้นไม้ DOM โครงสร้างต้นไม้ CSSOM และโครงสร้างต้นไม้การแสดงผลให้กลายเป็นภาพที่ผู้ใช้เห็นบนหน้าจอ

3.2 ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บ

ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บแสดงดังภาพที่ 1.10 ประกอบด้วย โปรแกรมให้บริการเว็บ ที่เก็บทรัพยากรในการให้บริการเว็บ ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ ระบบจัดการฐานข้อมูล และแหล่งบันทึกข้อมูล



ภาพที่ 1.10 ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บ

3.2.1 โปรแกรมให้บริการเว็บ

โปรแกรมให้บริการเว็บ (Web Server Program) เป็นโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่เฝ้าคอยรับการร้องขอทรัพยากร (Request Listening) ตามหมายเลขกำกับบริการที่กำหนด มีหน้าที่ในการกำหนดที่เก็บทรัพยากรในการให้บริการเว็บ ทำการค้นหาทรัพยากรและตอบสนองทรัพยากรไปยังผู้ร้องขอ ตลอดจนทำงานร่วมกับตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการในการส่งชุดคำสั่งภาษาประมวลผลและจัดการผลลัพธ์เพื่อนำส่งไปยังผู้ร้องขอ ตัวอย่างโปรแกรมให้บริการเว็บ เช่น Nginx, Apache Web Server และ Microsoft IIS เป็นต้น

3.2.2 ที่เก็บทรัพยากรในการให้บริการเว็บ

ที่เก็บทรัพยากรในการให้บริการเว็บ (Document Root) เป็นตำแหน่งที่เก็บไฟล์ทรัพยากรต่าง ๆ ในเครื่องคอมพิวเตอร์ของผู้ให้บริการ โดยที่เก็บทรัพยากรในการให้บริการเว็บนี้จะถูกกำหนดโดยโปรแกรมให้บริการเว็บ

3.2.3 ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ

ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ (Server-Side Processing Language Translator) เป็นโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่ในการแปลรหัสคำสั่งภาษาประมวลผลฝั่งผู้ให้บริการที่ทำงานร่วมกับเอกสาร HTML และส่งผลลัพธ์จากการประมวลผลกลับไปยังโปรแกรมให้บริการเว็บ นอกจากนี้ ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการยังทำหน้าที่ในการติดต่อกับระบบจัดการฐานข้อมูล ตัวอย่างของตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ เช่น ตัวแปลภาษา PHP ตัวแปลภาษา Python และ ตัวแปลภาษา Ruby เป็นต้น

3.2.4 ระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูล (Database Management System: DBMS) เป็นโปรแกรมคอมพิวเตอร์ที่ทำหน้าที่ในการบริหารและจัดการฐานข้อมูล ตัวอย่างของระบบจัดการฐานข้อมูล เช่น MySQL, MariaDB, Microsoft SQL Server, MongoDB และ Redis เป็นต้น

3.2.5 แหล่งบันทึกข้อมูล

แหล่งบันทึกข้อมูล เป็นหน่วยเก็บข้อมูลในเครื่องคอมพิวเตอร์ผู้ให้บริการเว็บ โดยจะทำงานร่วมกับระบบจัดการฐานข้อมูลและตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการในกรณีที่มีการเขียนและอ่านไฟล์ในแหล่งบันทึกข้อมูล เช่น การอัปโหลดไฟล์จากผู้ใช้สู่เครื่องผู้ให้บริการ การอ่านไฟล์ในเครื่องผู้ให้บริการ เป็นต้น

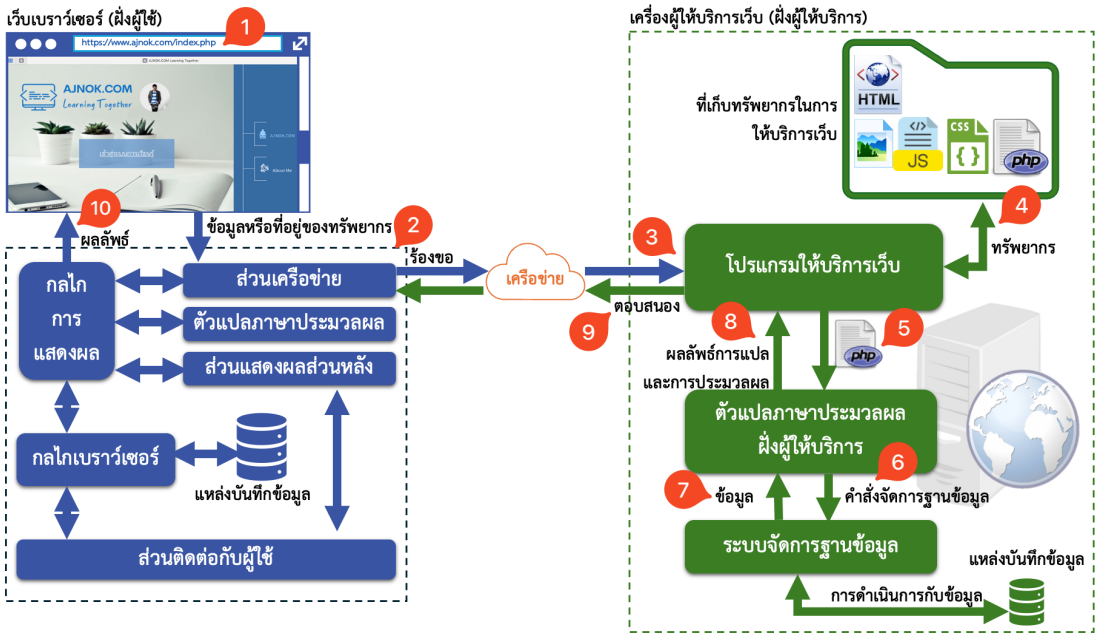
3.3 ขั้นตอนการทำงานของเว็บแอปพลิเคชัน

ขั้นตอนการทำงานของเว็บแอปพลิเคชันโดยทั่วไปแล้วจะเริ่มต้นตั้งแต่ผู้ใช้ทำการพิมพ์ URL ของเว็บแอปพลิเคชันหรือทรัพยากรอื่นๆ ที่ต้องการจากเครื่องผู้ให้บริการเว็บ จนกระทั่งมีการแสดงผลในส่วนแสดงผลของเว็บเบราว์เซอร์ มีขั้นตอนแสดงดังภาพที่ 1.11

ขั้นตอนที่ 1) ผู้ใช้พิมพ์ที่อยู่ของเว็บแอปพลิเคชันหรือทรัพยากรที่ต้องการอื่นๆ ในรูปแบบของ URL ในช่องที่อยู่ของเว็บเบราว์เซอร์

ขั้นตอนที่ 2) ส่วนเครือข่ายของเว็บเบราว์เซอร์จะมีการดำเนินการด้านระบบเครือข่ายในการหาที่อยู่ทางอินเทอร์เน็ตของเครื่องคอมพิวเตอร์ผู้ให้บริการเว็บตามที่ระบุใน URL จากนั้นจะทำการส่งคำร้องขอ HTTP หรือ HTTPS (HTTP/HTTPS Request) ไปยังเครื่องผู้ให้บริการเว็บที่มีที่อยู่ทางอินเทอร์เน็ตที่ได้มาจากการดำเนินการด้านระบบเครือข่าย

ขั้นตอนที่ 3) โปรแกรมให้บริการเว็บได้รับการร้องขอและทำการตรวจสอบคำร้องขอ หากคำร้องขออยู่ในรูปแบบที่ไม่ถูกต้อง โปรแกรมให้บริการเว็บก็จะมีการส่งรหัสสถานะผิดพลาดกลับไปยังเว็บเบราว์เซอร์



ภาพที่ 1.11 ขั้นตอนการทำงานของเว็บแอปพลิเคชัน

ขั้นตอนที่ 4) โปรแกรมให้บริการเว็บดำเนินการค้นหาทรัพยากรตามที่ระบุใน URL โดยทำการค้นหาจากที่เก็บทรัพยากรในการให้บริการเว็บ หากทรัพยากรตามคำร้องขอเป็นไฟล์เอกสาร HTML รูปภาพ วีดีโอ หรือทรัพยากรลักษณะอื่นที่ไม่ใช่ไฟล์รหัสคำสั่งของภาษาประมวลผลฝั่งผู้ให้บริการ โปรแกรมให้บริการเว็บก็จะจัดส่งกลับไปยังเว็บเบราว์เซอร์โดยไม่มีการเปลี่ยนแปลงข้อมูลแต่อย่างใด กรณีที่ตรวจไม่พบทรัพยากรตามที่ระบุหรือทรัพยากรดังกล่าวไม่อนุญาตให้คำร้องขอนั้นเข้าถึงก็จะมีการส่งรหัสข้อผิดพลาดกลับไปยังเว็บเบราว์เซอร์ แต่หากทรัพยากรที่ต้องการเป็นไฟล์รหัสคำสั่งของภาษาประมวลผลฝั่งผู้ให้บริการ เช่น ไฟล์ที่มีส่วนขยายเป็น .php ในภาษา PHP และ .py ในภาษา Python เป็นต้น ไฟล์เหล่านี้จะถูกส่งต่อไปยังตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการเพื่อดำเนินการต่อในขั้นตอนถัดไป

ขั้นตอนที่ 5) ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการได้รับรหัสคำสั่งภาษาประมวลผล ตัวแปลภาษาก็จะทำการแปลและประมวลผลตามรหัสคำสั่งเหล่านั้น ถ้ารหัสคำสั่งมีการเชื่อมต่อกับแหล่งบันทึกข้อมูล เช่น การสร้างไฟล์ การอ่านไฟล์ เป็นต้น ตัวแปลภาษาก็จะทำงานร่วมกับระบบปฏิบัติการของเครื่องคอมพิวเตอร์ผู้

ให้บริการเว็บเพื่อดำเนินการกับแหล่งบันทึกข้อมูล จากนั้นจะรวมผลลัพธ์เข้ากับโครงสร้างเอกสาร HTML และส่งกลับไปยังโปรแกรมให้บริการเว็บต่อไป ในกรณีที่รหัสคำสั่งภาษาประมวลผลมีการใช้คำสั่งเพื่อติดต่อกับระบบจัดการฐานข้อมูล ตัวแปลภาษาก็จะจัดส่งคำสั่งเหล่านั้นไปยังระบบจัดการฐานข้อมูลเพื่อดำเนินการในขั้นตอนถัดไป

ขั้นตอนที่ 6) ระบบจัดการฐานข้อมูลได้รับคำสั่งในการจัดการฐานข้อมูล โดยจะทำการตรวจสอบความถูกต้องของคำสั่งและดำเนินการตามคำสั่ง

ขั้นตอนที่ 7) ระบบจัดการฐานข้อมูลส่งผลลัพธ์จากการดำเนินการกลับไปยังตัวแปลภาษา

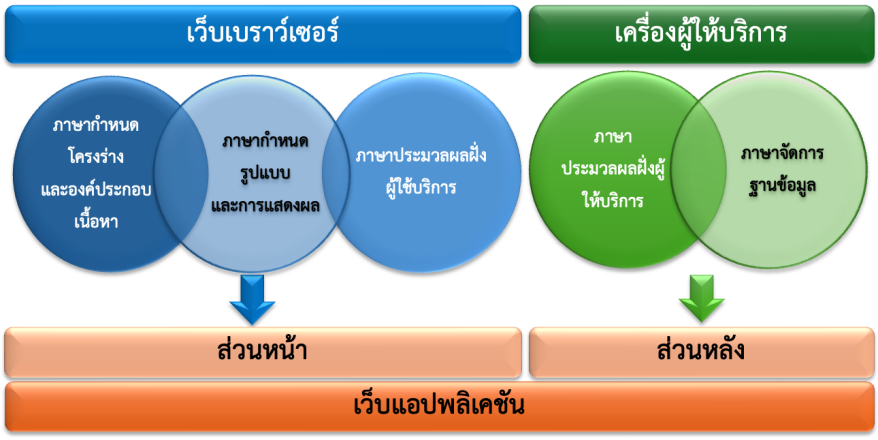
ขั้นตอนที่ 8) ตัวแปลภาษาส่งผลลัพธ์จากการดำเนินการทั้งหมดกลับไปยังโปรแกรมให้บริการเว็บเพื่อจัดส่งหน้าเว็บกลับไปยังเว็บเบราว์เซอร์

ขั้นตอนที่ 9) โปรแกรมให้บริการเว็บเรียบเรียงผลลัพธ์ในรูปแบบที่เว็บเบราว์เซอร์สามารถแสดงผลได้และส่งทรัพยากรกลับไปในรูปแบบการตอบกลับ HTTP (HTTP Response) ซึ่งรูปแบบการตอบกลับนี้จะมีการระบุรหัสสถานะของการดำเนินการไปด้วย เช่น สถานะผิดพลาดจากการไม่พบทรัพยากรที่ระบุ (404) สถานะผิดพลาดจากการที่ไม่ได้รับอนุญาตให้เข้าถึงทรัพยากรที่ระบุ (403) สถานะข้อผิดพลาดจากปัญหาที่เกิดขึ้นจากการทำงานผิดพลาดของเครื่องผู้ให้บริการเว็บ (503) และสถานะตอบกลับตามการร้องขอสำเร็จ (200) เป็นต้น

ขั้นตอนที่ 10) เว็บเบราว์เซอร์ได้รับข้อมูลพร้อมด้วยรหัสสถานะของการดำเนินการตามคำร้องขอ HTTP ส่วนประกอบต่าง ๆ ของเว็บเบราว์เซอร์ก็จะนำทรัพยากรที่ได้ไปดำเนินการตามหน้าที่ของแต่ละส่วนจนกระทั่งแสดงผลในส่วนแสดงผลของเว็บเบราว์เซอร์

4. ภาษาในการพัฒนาเว็บแอปพลิเคชัน

จากความหมายและกลไกการทำงานของเว็บแอปพลิเคชัน เมื่อสังเคราะห์ความสัมพันธ์ในมุมมองของภาษาที่ใช้ในการพัฒนาเว็บแอปพลิเคชันแล้วพบว่าจำแนกได้ 2 ส่วน ประกอบด้วยการพัฒนาเว็บแอปพลิเคชันส่วนหน้าและการพัฒนาเว็บแอปพลิเคชันส่วนหลัง (Sadriddinovich & Muhiddinovna, 2024) แสดงดังภาพที่ 1.12



ภาพที่ 1.12 องค์ประกอบการพัฒนาเว็บแอปพลิเคชัน

4.1.1 การพัฒนาเว็บแอปพลิเคชันส่วนหน้า

การพัฒนาเว็บแอปพลิเคชันส่วนหน้า (Front-end Development) คือ การพัฒนาเว็บแอปพลิเคชันในส่วนที่ผู้ใช้มองเห็นและมีปฏิสัมพันธ์ได้ในเว็บเบราว์เซอร์ เป็นการสร้างส่วนเชื่อมต่อกับผู้ใช้ (User Interface: UI) ที่มุ่งเน้นการรับคำสั่งจากผู้ใช้ นำเข้าและนำเสนอข้อมูล ตลอดจนเชื่อมโยงการทำงานส่วนต่าง ๆ ของเว็บแอปพลิเคชันเข้าด้วยกันอย่างมีประสิทธิภาพ เช่น เมนูนำทาง ปุ่ม แบบฟอร์ม หรือรายการข้อมูล เป็นต้น ภาษาหลักที่ใช้ในการพัฒนาเว็บแอปพลิเคชันส่วนหน้ามี 3 ภาษา ดังนี้

1) ภาษากำหนดโครงร่างและองค์ประกอบเนื้อหา

ภาษากำหนดโครงร่างและองค์ประกอบเนื้อหา เป็นภาษามาตรฐานที่ทำให้เว็บเบราว์เซอร์สามารถแสดงผลได้อย่างถูกต้อง โดยภาษาหลักในกลุ่มนี้คือภาษา HTML มีหน้าที่ในการกำกับว่าเนื้อหาที่จะแสดงในหน้าเว็บแต่ละเนื้อหามีบทบาทอย่างไร เช่น เป็นส่วนหัวเรื่องที่สำคัญที่สุดหรือสำคัญน้อยที่สุดในหน้าเว็บนั้นๆ หรือเป็นส่วนแสดงเนื้อหาแบบลำดับรายการแบบมีเลขลำดับ ตาราง รูปภาพ หรือแบบฟอร์ม เป็นต้น

2) ภาษากำหนดรูปแบบและการแสดงผล

ภาษากำหนดรูปแบบและการแสดงผล เป็นภาษาที่ใช้ร่วมกับภาษา HTML เพื่อกำหนดว่าเนื้อหาแต่ละส่วนจะแสดงผลอย่างไร มีรูปแบบอย่างไร เช่น สีสัน ความกว้าง ความยาว เส้นขอบ ระยะห่าง เป็นต้น ภาษาหลักในกลุ่มนี้คือภาษา CSS

3) ภาษาประมวลผลฝั่งผู้ใช้บริการ

ภาษาประมวลผลฝั่งเครื่องผู้ใช้บริการ เป็นภาษาประมวลผลที่สามารถแปลและดำเนินการได้ในเว็บเบราว์เซอร์ มีหน้าที่หลักในการโต้ตอบกับการกระทำของผู้ใช้ที่มีต่อหน้าเว็บและมุ่งเน้นการปรับเปลี่ยนการแสดงผลในหน้าเว็บ เช่น การเปลี่ยนสีตัวอักษรจากการคลิกปุ่ม การตรวจสอบข้อมูลก่อนส่งไปยังเครื่องผู้ให้บริการเว็บ เป็นต้น ภาษาหลักในกลุ่มนี้คือภาษา JavaScript ซึ่งเป็นภาษาประมวลผลที่เว็บเบราว์เซอร์ให้การรองรับโดยไม่ต้องติดตั้งส่วนเสริมของเว็บเบราว์เซอร์เพิ่ม

4.1.2 การพัฒนาเว็บแอปพลิเคชันส่วนหลัง

การพัฒนาเว็บแอปพลิเคชันส่วนหลัง (Back-end Development) คือ การพัฒนาเว็บแอปพลิเคชันในส่วนที่ทำงานในเครื่องผู้ให้บริการเว็บโดยที่ผู้ใช้ไม่ได้มองเห็นการทำงานโดยตรง เช่น การประมวลผลข้อมูล การจัดการกับฐานข้อมูล และการดำเนินการตามตรรกะทางธุรกิจ เป็นต้น การพัฒนาในส่วนนี้ไม่ได้มุ่งเน้นในการสร้างส่วนติดต่อกับผู้ใช้ แต่มุ่งเน้นในการกระบวนการได้มาซึ่งข้อมูลที่จะนำไปแสดงในเว็บแอปพลิเคชันส่วนหน้าและการจัดการกับข้อมูล ภาษาหลักที่ใช้ในการพัฒนาเว็บแอปพลิเคชันส่วนหลังมี 2 ภาษา ดังนี้

1) ภาษาประมวลผลฝั่งผู้ให้บริการ

ภาษาประมวลผลฝั่งผู้ให้บริการ เป็นภาษาประมวลผลที่แปลและดำเนินการในเครื่องผู้ให้บริการเว็บ มีหน้าที่หลักในการดำเนินการตามตรรกะธุรกิจ การประมวลผลข้อมูล การเชื่อมต่อกับระบบจัดการฐานข้อมูล ตลอดจนการจัดการกับไฟล์ในเครื่องคอมพิวเตอร์ผู้ให้บริการ ภาษาประมวลผลฝั่งผู้ให้บริการ เช่น ภาษา

PHP ภาษา Python และภาษา Java เป็นต้น นอกจากนี้ ภาษา JavaScript ยังมีการพัฒนาให้สามารถประมวลผลในเครื่องผู้ให้บริการเว็บได้โดยใช้งานร่วมกับสภาพแวดล้อมการทำงานเช่น Node.js อีกด้วย

2) ภาษาจัดการฐานข้อมูล

ภาษาจัดการฐานข้อมูล เป็นภาษาที่ใช้ในการโต้ตอบกับระบบจัดการฐานข้อมูลเพื่อดำเนินการเกี่ยวกับข้อมูล เช่น การสร้างโครงสร้างฐานข้อมูล การจัดเก็บ การปรับปรุง การลบ และการดึงข้อมูล เป็นต้น ภาษาจัดการฐานข้อมูลมีหลายภาษาขึ้นอยู่กับประเภทของฐานข้อมูล เช่น ภาษา Structured Query Language (SQL) ใช้กับฐานข้อมูลเชิงสัมพันธ์ และภาษา MongoDB Query Language (MQL) ใช้กับฐานข้อมูล MongoDB เป็นต้น

5. สรุปท้ายบท

เว็บเป็นบริการสำคัญในเครือข่ายอินเทอร์เน็ตที่ถูกคิดค้นขึ้นโดยเซอร์ทิม เบอร์เนิร์ส-ลี เมื่อปี พ.ศ. 2532 โดยมีบทบาทในการเป็นระบบข้อมูลที่เชื่อมโยงกันผ่านเครือข่ายคอมพิวเตอร์ ผู้ใช้งานสามารถเข้าถึงข้อมูลและทรัพยากรต่าง ๆ ได้โดยใช้โปรแกรมเว็บเบราว์เซอร์ซึ่งทำหน้าที่แสดงผลเอกสารเว็บที่สร้างขึ้นจากภาษา HTML สำหรับกำหนดโครงสร้างหน้าเว็บ ภาษา CSS สำหรับจัดรูปแบบการแสดงผล และภาษาประมวลผลฝั่งผู้ใช้ เช่น JavaScript ที่ช่วยให้หน้าเว็บมีความสามารถในการโต้ตอบกับผู้ใช้ ในขณะที่ภาษาประมวลผลฝั่งผู้ให้บริการ เช่น PHP หรือ Python ใช้สำหรับการจัดการข้อมูลและสร้างเนื้อหาพลวัตบนหน้าเว็บ

องค์ประกอบสำคัญของเว็บยังรวมถึงเว็บไซต์ซึ่งเป็นชุดของหน้าเว็บที่เชื่อมโยงกันภายใต้ชื่อโดเมนเดียวกัน โฮมเพจที่ทำหน้าที่เป็นหน้าเริ่มต้น เว็บเบราว์เซอร์ที่ใช้ในการแปลภาษาและแสดงผล เครื่องผู้ให้บริการเว็บที่รับและตอบสนองต่อคำร้องขอ รวมถึง URL ที่ใช้ระบุตำแหน่งทรัพยากรในเครือข่ายอินเทอร์เน็ต โดยประกอบด้วย Scheme, Domain Name, Port, Path, Parameters และ Anchor

เว็บแอปพลิเคชัน คือโปรแกรมที่ทำงานบนเว็บเบราว์เซอร์ โดยอาศัยเว็บเทคโนโลยีในการสร้างส่วนติดต่อผู้ใช้และการประมวลผลข้อมูลบนเครื่องผู้ให้บริการเว็บผ่านโปรโตคอล HTTP หรือ HTTPS เว็บแอปพลิเคชันมีคุณลักษณะที่สำคัญ เช่น การเข้าถึงได้จากทุกอุปกรณ์และทุกที่ที่มีอินเทอร์เน็ต ความเข้ากันได้กับหลายแพลตฟอร์ม ความสามารถในการปรับขยายการใช้งาน การอัปเดตให้เป็นรุ่นล่าสุดโดยอัตโนมัติ รวมทั้งการทำงานร่วมกันและจัดเก็บข้อมูลแบบรวมศูนย์บนเครื่องผู้ให้บริการ อย่างไรก็ตาม เว็บแอปพลิเคชันยังมีข้อจำกัด เช่น ความจำเป็นต้องเชื่อมต่ออินเทอร์เน็ต ความเร็วในการตอบสนอง และประเด็นด้านความปลอดภัยเมื่อมีการสื่อสารผ่านเครือข่าย เป็นต้น

กลไกการทำงานของเว็บแอปพลิเคชันประกอบด้วยการทำงานร่วมกันระหว่างฝั่งผู้ใช้และฝั่งผู้ให้บริการเว็บ เว็บแอปพลิเคชันจะแสดงผลในฝั่งผู้ใช้ผ่านเว็บเบราว์เซอร์ซึ่งมีส่วนประกอบสำคัญ เช่น ส่วนติดต่อผู้ใช้ กลไกเบราว์เซอร์ แหล่งบันทึกข้อมูล กลไกการแสดงผล ที่ทำหน้าที่แปลง HTML และ CSS ให้เป็นหน้าเว็บ และตัวแปลภาษาจาวาสคริปต์ที่ทำให้เว็บมีความสามารถโต้ตอบได้ ในขณะที่ฝั่งผู้ให้บริการเว็บประกอบด้วยโปรแกรม

ให้บริการเว็บ ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ ระบบจัดการฐานข้อมูล และแหล่งบันทึกข้อมูล โดยทั้งสองฝั่งทำงานประสานกันตั้งแต่ผู้ใช้พิมพ์ URL จนกระทั่งผลลัพธ์ถูกแสดงในเว็บเบราว์เซอร์

ภาษาที่ใช้ในการพัฒนาเว็บแอปพลิเคชันแบ่งออกเป็นสองส่วนหลัก ได้แก่ การพัฒนาเว็บแอปพลิเคชันส่วนหน้าและส่วนหลัง การพัฒนาเว็บแอปพลิเคชันส่วนหน้าจะใช้ภาษา HTML สำหรับสร้างโครงสร้างเนื้อหา ภาษา CSS สำหรับจัดรูปแบบการแสดงผล และภาษา JavaScript สำหรับจัดการการโต้ตอบของผู้ใช้ ในขณะที่การพัฒนาเว็บแอปพลิเคชันส่วนหลังใช้ภาษาประมวลผลฝั่งผู้ให้บริการ เช่น PHP, Python, Node.js และภาษาจัดการฐานข้อมูล เช่น SQL เพื่อการประมวลผลและจัดการข้อมูล เป็นต้น

6. กิจกรรมท้ายบท

6.1 จงอธิบายความหมายของเว็บแอปพลิเคชัน และบอกความแตกต่างระหว่างเว็บไซต์และเว็บแอปพลิเคชันพอสังเขป

6.2 คุณลักษณะสำคัญของเว็บแอปพลิเคชันมีอะไรบ้าง? พร้อมทั้งยกตัวอย่างเว็บแอปพลิเคชันที่พบในชีวิตประจำวัน

6.3 ส่วนประกอบพื้นฐานของเว็บเบราว์เซอร์มีอะไรบ้าง? และแต่ละส่วนมีหน้าที่อะไร? จงอธิบายพอสังเขป

6.4 ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บมีอะไรบ้าง? และแต่ละส่วนมีหน้าที่อะไร? จงอธิบายพอสังเขป

6.5 จงเขียนขั้นตอนการทำงานของเว็บแอปพลิเคชันโดยเริ่มตั้งแต่ผู้ใช้พิมพ์ URL จนกระทั่งปรากฏผลลัพธ์บนส่วนแสดงผลของเว็บเบราว์เซอร์ โดยระบุบทบาทของเว็บเบราว์เซอร์ เครื่องผู้ให้บริการเว็บ โปรโตคอล HTTP และโดเมนเนมในแต่ละขั้นตอน

6.6 ภาษาคอมพิวเตอร์ใดบ้างที่เกี่ยวข้องกับการพัฒนาเว็บแอปพลิเคชันทั้งฝั่งผู้ใช้และฝั่งผู้ให้บริการ? พร้อมอธิบายหน้าที่ของแต่ละภาษามาพอสังเขป



ระบบสารสนเทศกับการพัฒนาเว็บแอปพลิเคชัน

บทที่ 2 ระบบสารสนเทศกับการพัฒนาเว็บแอปพลิเคชัน มุ่งเน้นให้ผู้อ่านทราบถึงความสัมพันธ์ระหว่างระบบสารสนเทศและเว็บแอปพลิเคชัน รวมถึงเข้าใจกรอบแนวคิดของวงจรการพัฒนาระบบและขั้นตอนการพัฒนาเว็บแอปพลิเคชันอย่างเป็นระบบ ทั้งนี้เพื่อให้ผู้อ่านตระหนักว่าการพัฒนาเว็บแอปพลิเคชันนั้นไม่ได้จำกัดเพียงแค่การเขียนโปรแกรม แต่จำเป็นต้องวิเคราะห์ความต้องการของผู้ใช้งาน และพิจารณาองค์ประกอบต่าง ๆ ที่เกี่ยวข้อง เพื่อให้ได้ระบบที่มีคุณภาพ ตรงตามวัตถุประสงค์และสามารถตอบสนองความต้องการได้อย่างแท้จริง เนื้อหาในบทนี้มีดังนี้

1. ระบบสารสนเทศ
2. ความสัมพันธ์ของระบบสารสนเทศและเว็บแอปพลิเคชัน
3. วงจรการพัฒนาระบบ
4. การพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ
5. สรุปท้ายบท
6. กิจกรรมท้ายบท

1. ระบบสารสนเทศ

ระบบสารสนเทศ เป็นคำประสมที่ประกอบด้วยคำว่า ระบบ และ สารสนเทศ ซึ่งมีความหมายและองค์ประกอบดังนี้

1.1 ความหมายของระบบและองค์ประกอบของระบบ

ความหมายของระบบและองค์ประกอบของระบบ มีดังนี้

1.1.1 ความหมายของระบบ

ระบบ (System) หมายถึง กลุ่มขององค์ประกอบที่มีความสัมพันธ์กัน แต่ละองค์ประกอบมีการทำงานร่วมกันเพื่อบรรลุเป้าประสงค์เดียวกัน ระบบอาจประกอบด้วยระบบย่อย (Subsystem) ที่ประสานทำงานร่วมกัน ระบบในทางคอมพิวเตอร์จึงหมายถึงระบบที่มีคอมพิวเตอร์เป็นองค์ประกอบพื้นฐานที่ทำงานร่วมกันเพื่อบรรลุเป้าประสงค์ ประกอบด้วย บุคลากร ซอฟต์แวร์ ฮาร์ดแวร์ ข้อมูล และกระบวนการในการดำเนินการ (โอภาส เอี่ยมสิริวงศ์, 2566; อรสา เตติวัฒน์, 2567)

1.1.2 องค์ประกอบของระบบ

อรสา เตติวัฒน์ (2567) ได้นำเสนอองค์ประกอบของระบบโดยจำแนกได้ 2 ลักษณะ คือ องค์ประกอบที่สามารถสัมผัสได้ทั้งที่มีชีวิตและไม่มีชีวิต และองค์ประกอบที่เป็นแนวคิดนามธรรม โดยองค์ประกอบมี 9 องค์ประกอบแสดงดังภาพที่ 2.1



ภาพที่ 2.1 องค์ประกอบของระบบ

1) วัตถุประสงค์และเป้าหมาย

วัตถุประสงค์และเป้าหมาย เป็นสิ่งที่กำหนดขึ้นเพื่อให้องค์ประกอบต่าง ๆ ของระบบทำงานไปในทิศทางเดียวกัน และได้ผลลัพธ์เป็นไปตามเป้าหมาย

2) ขอบเขตของระบบ

ขอบเขตของระบบ เป็นกรอบหรือขอบเขตของการทำงานที่ระบบควรมีการกำหนดขอบเขตเพื่อให้เกิดการทำงานที่สะดวก เป็นไปตามขั้นตอน ลดระยะเวลาในการบรรลุเป้าหมาย และทำให้เห็นภาพรวมของระบบ

3) ข้อมูลนำเข้า

ข้อมูลนำเข้า เป็นข้อเท็จจริงของสิ่งต่าง ๆ ที่เกี่ยวข้องกับวัตถุประสงค์ของระบบที่นำเข้าสู่ระบบเพื่อประกอบการดำเนินการให้ได้ผลลัพธ์หรือบรรลุตามเป้าประสงค์

4) กระบวนการทำงาน

กระบวนการทำงาน เป็นขั้นตอนการทำงานหรือวิธีการดำเนินการที่ระบบควรมีการกำหนดให้เป็นแบบแผนอย่างชัดเจน ซึ่งการกำหนดกระบวนการที่ชัดเจนยังช่วยให้การตรวจสอบข้อผิดพลาดของการดำเนินการและการปรับปรุงทำได้ง่ายขึ้น

5) ข้อมูลนำออก

ข้อมูลนำออก เป็นผลจากกระบวนการทำงานหรือเป็นผลลัพธ์หรือผลสำเร็จที่ได้จากการทำงานของระบบ

6) ความสัมพันธ์

ความสัมพันธ์ เป็นความเชื่อมโยงของสิ่งแวดล้อมทั้งภายในและภายนอกที่ทำงานร่วมกันเพื่อการบรรลุวัตถุประสงค์และเป้าหมายที่กำหนด

7) ส่วนประกอบ

ส่วนประกอบ เป็นสิ่งต่าง ๆ ทั้งที่เป็นวัตถุ สิ่งของ หรือสิ่งมีชีวิตที่ก่อให้เกิดเป็นระบบหรือมีส่วนร่วมในระบบ

8) การควบคุม

การควบคุม เป็นการตรวจสอบการทำงานให้เป็นไปตามกระบวนการทำงานที่กำหนดไว้

9) ปฏิกิริยาย้อนกลับ

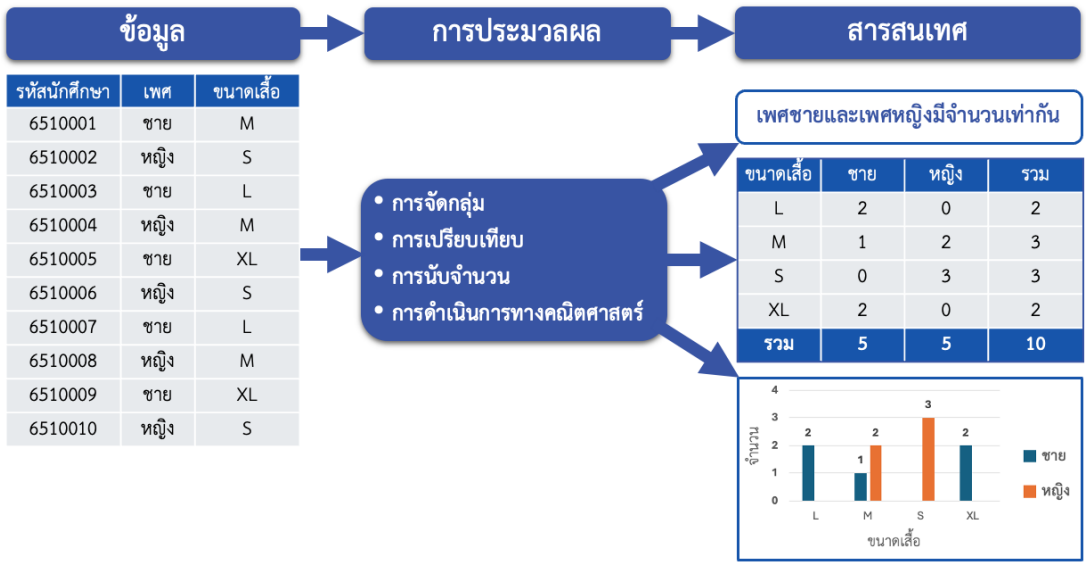
ปฏิกิริยาย้อนกลับ เป็นผลสะท้อนของระบบที่นำมาใช้ในการตรวจสอบ ปรับปรุงและพัฒนาการดำเนินงานของระบบ

1.2 ความหมายของสารสนเทศและองค์ประกอบของสารสนเทศ

ความหมายของสารสนเทศและองค์ประกอบของสารสนเทศ มีดังนี้

1.2.1 ความหมายของสารสนเทศ

สารสนเทศ (Information) หมายถึง ผลลัพธ์ที่ได้จากการประมวลผล จัดการ หรือการกระทำกับข้อมูล (Data) ให้อยู่ในรูปแบบที่มีความหมายและสามารถนำไปใช้ประโยชน์ได้ เช่น ใช้ในการวางแผน การตัดสินใจ หรือการควบคุมงานต่าง ๆ เป็นต้น สารสนเทศจึงเป็นข้อมูลที่ผ่านการดำเนินการหรือการประมวลผล เช่น การจัดกลุ่ม การนับจำนวน การเปรียบเทียบ และการดำเนินการทางคณิตศาสตร์ เป็นต้น ตัวอย่างของการได้มาซึ่งสารสนเทศแสดงดังภาพที่ 2.2

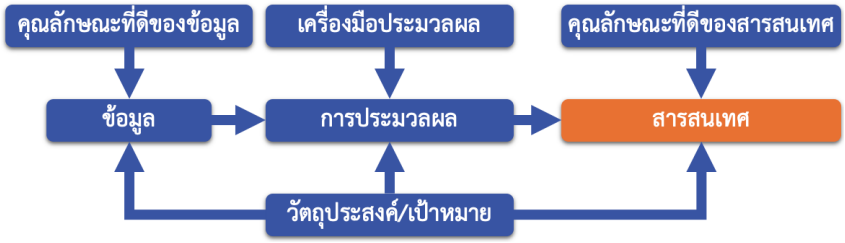


ภาพที่ 2.2 ตัวอย่างของการได้มาซึ่งสารสนเทศ

จากภาพที่ 2.2 เป็นตัวอย่างของการได้มาซึ่งสารสนเทศ โดยมีการรวบรวมข้อมูลของนักศึกษาว่ามีเพศใดบ้าง และแต่ละคนต้องการเสื้อกีฬาขนาดใด ภายใต้วัตถุประสงค์เพื่อการนำไปสั่งซื้อเสื้อกีฬาและการแบ่งกลุ่มการแข่งขันกีฬาให้มีเพศชายหญิงเท่า ๆ เมื่อมีการรวบรวมข้อมูลโดยคำนึงถึงวัตถุประสงค์ที่ต้องการแล้วก็จะนำข้อมูลไปทำการประมวลผลโดยใช้วิธีการที่จะทำให้ได้ผลลัพธ์ตามวัตถุประสงค์ เช่น การจัดกลุ่ม การเปรียบเทียบ การนับจำนวน และการหาผลรวมซึ่งเป็นการดำเนินการทางคณิตศาสตร์ เป็นต้น เมื่อทำการประมวลผลแล้วจะเป็นการนำผลที่ได้นำเสนอโดยใช้วิธีการที่เหมาะสมต่อการนำไปใช้งาน ผลลัพธ์จากส่วนนี้เองที่กล่าวได้ว่าเป็นสารสนเทศที่สามารถนำไปสู่การสั่งซื้อเสื้อและการแบ่งทีม

1.2.2 องค์ประกอบของสารสนเทศ

จากความหมายของสารสนเทศจะพบว่าองค์ประกอบที่ทำให้ได้มาซึ่งสารสนเทศมี 6 องค์ประกอบ แสดงดังภาพที่ 2.3



ภาพที่ 2.3 องค์ประกอบของสารสนเทศ

1) วัตถุประสงค์/เป้าหมาย

วัตถุประสงค์/เป้าหมาย เป็นสิ่งผู้ใช้สารสนเทศกำหนดขึ้นเพื่อกำหนดทิศทางของสิ่งต่าง ๆ ที่เกี่ยวข้องกับการได้มาซึ่งสารสนเทศ

2) ข้อมูล

ข้อมูลเป็นข้อเท็จจริงของสิ่งต่าง ๆ ที่รวบรวมขึ้นเพื่อนำไปใช้ในการสร้างสารสนเทศตามกรอบของวัตถุประสงค์/เป้าหมายที่กำหนด โดยข้อมูลที่ยังไม่ผ่านการประมวลผลเรียกว่าข้อมูลดิบ (Raw Data) ข้อมูลเป็นปัจจัยนำเข้าที่ส่งผลต่อคุณภาพของสารสนเทศ สอดคล้องกับแนวคิด “ขยะเข้า ขยะออก (Garbage In, Garbage Out)” ที่กล่าวว่าหากระบบได้รับข้อมูลที่บกพร่องหรือไม่สมบูรณ์ ผลลัพธ์ที่สร้างขึ้นจะมีข้อบกพร่องเช่นกัน (Sheposh, 2024)

3) กรอบคุณภาพของข้อมูล

กรอบคุณภาพของข้อมูลเป็นองค์ประกอบที่ใช้ในการคัดกรอง ตรวจสอบและเตรียมข้อมูลเพื่อให้ได้ข้อมูลนำเข้าที่มีคุณภาพที่จะนำไปสู่สารสนเทศที่มีคุณภาพเช่นกัน โดยกรอบคุณภาพของข้อมูลมีดังต่อไปนี้ (ปภังกร อิ่นแก้ว, 2566; Jones, 2023)

- **ความถูกต้อง (Accuracy)**

ข้อมูลนำเข้าต้องตรงกับความเป็นจริงปราศจากข้อผิดพลาด โดยความไม่ถูกต้องของข้อมูลอาจเกิดจากความผิดพลาดในการบันทึกข้อมูล เช่น การสะกดคำผิด การเขียนตัวเลขผิดโดยไม่ได้ตั้งใจ เป็นต้น นอกจากนี้อาจเกิดจากอคติหรือการบิดเบือนของผู้บันทึกข้อมูล เช่น ผู้ป่วยให้ข้อมูลประวัติและอาการป่วยแก่แพทย์ที่ไม่ตรงกับความจริงเพื่อหลีกเลี่ยงการถูกตำหนิ เป็นต้น

- **ความสมบูรณ์ (Completeness)**

ข้อมูลต้องมีความครบถ้วนไม่ขาดส่วนที่จำเป็นต่อการใช้งานหรือการประมวลผล เช่น จากตัวอย่างข้อมูลขนาดเสื้อของนักศึกษาแต่ละคนในภาพที่ 2.2 หากนักศึกษาบางคนไม่ได้กรอกขนาดของเสื้อกีฬาหรือไม่ได้กรอกเพศ ก็จะทำให้การสั่งซื้อเสื้อได้ไม่ครบจำนวนหรืออาจได้ขนาดเสื้อที่ไม่ถูกต้องกับเพศก็เป็นได้

- **ความทันเวลา (Timeliness)**

ข้อมูลต้องเป็นปัจจุบัน มีความพร้อมใช้งานในเวลาที่ต้องการ ข้อมูลที่ไม่เป็นปัจจุบันอาจไม่เหมาะสมกับการนำไปประมวลผลเพื่อประกอบการตัดสินใจในสถานการณ์นั้น ๆ เช่น หากข้อมูลขนาดเสื้อ

กีฬาของนักศึกษาเป็นข้อมูลที่ผ่านมา 1 ปี หากไม่มีการปรับปรุงให้เป็นปัจจุบันก่อนก็จะทำให้ได้เสื้อกีฬาที่ไม่เหมาะสมกับร่างกายที่อาจมีการเปลี่ยนแปลงภายหลัง 1 ปี เป็นต้น

- **ความสอดคล้อง (Consistency)**

ข้อมูลต้องมีความสอดคล้องเข้ากันได้ในแต่ละชุดข้อมูล การแทนค่าข้อมูลใช้รูปแบบมาตรวัดหรือมีมาตรฐานเดียวกัน เช่น การบันทึกข้อมูลเพศของนักศึกษาควรมีมาตรฐานการแทนข้อมูลเป็นคำว่า ชาย และ หญิง หากมีการแยกบันทึกข้อมูลของนักศึกษาที่อยู่ต่างห้องกันก็ไม่ควรแทนค่าด้วยค่าอื่น นอกจากนี้รูปแบบของการบันทึกวันที่ที่ต่างกันก็เป็นประเด็นของความสอดคล้องของข้อมูลเช่นกัน เช่น 01/01/2568 และ 01/01/2025 ถึงแม้จะหมายถึงวันเดียวกันแต่อาจทำให้เกิดความผิดพลาดในการประมวลผลได้

- **ความเป็นเอกลักษณ์ (Uniqueness)**

ข้อมูลต้องมีความเป็นเอกลักษณ์ กล่าวคือไม่มีรายการเดียวกันบันทึกซ้ำซ้อนในชุดข้อมูล ข้อมูลหนึ่งๆ มีการบันทึกเพียงแห่งเดียว การที่ข้อมูลเดียวกันปรากฏซ้ำซ้อนในหลายระเบียนในชุดข้อมูลจะทำให้การวิเคราะห์มีความโน้มเอียง (Bias) สูง เนื่องจากถือว่าเป็นข้อมูลคนละตัวกัน เช่น ในการบันทึกขนาดเสื้อกีฬาของนักศึกษาหากมีนักศึกษาหนึ่งคนมีการบันทึกข้อมูลมากกว่า 1 ครั้ง อาจเนื่องมาจากความพลั้งเผลอหรือเข้าใจผิดว่ายังไม่ได้บันทึกข้อมูลก็จะทำให้การประมวลผลผิดพลาดได้

- **ความเกี่ยวข้องกับวัตถุประสงค์ (Relevance)**

ข้อมูลควรมีความเกี่ยวข้องกับวัตถุประสงค์/เป้าหมายที่ได้กำหนดไว้ หากข้อมูลมีความถูกต้องแต่ไม่เกี่ยวข้องกับวัตถุประสงค์/เป้าหมาย ก็จะทำให้เสียทรัพยากรในการจัดเก็บข้อมูลโดยไม่จำเป็น เช่น ข้อมูลขนาดเสื้อกีฬาของนักศึกษาไม่จำเป็นที่จะต้องใช้วันเดือนปีเกิดของนักศึกษาเนื่องจากเป้าประสงค์ของการใช้ข้อมูลเพื่อหาจำนวนสั่งซื้อเสื้อกีฬาแต่ละขนาดเท่านั้น เป็นต้น

- **ความสมเหตุสมผล (Validity)**

ข้อมูลต้องเป็นไปตามกฎเกณฑ์ เงื่อนไขหรือธรรมชาติของข้อมูล เช่น ระยะทางต้องไม่มีค่าเป็นค่าติดลบ การใช้ลำดับที่ของเดือนเป็นข้อมูลจะต้องเป็น 1 ถึง 12 และอำเภอภูมิลำเนาของนักศึกษาจะต้องเป็นอำเภอที่อยู่ภายใต้จังหวัดที่นักศึกษาระบุมาก่อน เป็นต้น ซึ่งความสมเหตุผลนี้จะสอดคล้องกับความถูกต้องของข้อมูล

4) การประมวลผล

การประมวลผลเป็นขั้นตอนวิธีในการดำเนินการกับข้อมูล ซึ่งการประมวลผลนี้ไม่ได้หมายความเฉพาะการประมวลผลทางคณิตศาสตร์เท่านั้น แต่ยังรวมถึงการจัดกลุ่ม การเรียงลำดับ การเปรียบเทียบและการกระทำกับข้อมูลลักษณะอื่นๆ ด้วยเช่นกัน

5) เครื่องมือประมวลผล

เครื่องมือประมวลผล เป็นเครื่องมือหรือระบบที่ใช้ในการดำเนินการกับข้อมูล ซึ่งครอบคลุมถึงการป้อนข้อมูล การประมวลผล และการแสดงผล เครื่องมือเหล่านี้อาจอยู่ในรูปของอุปกรณ์ ระบบ

หรือซอฟต์แวร์ เมื่อมีการนำเทคโนโลยีคอมพิวเตอร์และการสื่อสารโทรคมนาคมมาใช้ในการจัดการข้อมูลจนได้มาซึ่งสารสนเทศที่ตรงตามวัตถุประสงค์/เป้าหมายของการสร้างสารสนเทศ

6) กรอบคุณภาพของสารสนเทศ

กรอบคุณภาพของสารสนเทศเป็นองค์ประกอบที่ใช้ในการประเมินสารสนเทศที่ได้จากการประมวลผลเพื่อพิจารณาระดับความเหมาะสมของการนำสารสนเทศไปใช้ประโยชน์ ซึ่งคุณภาพของสารสนเทศมีผลต่อการนำสารสนเทศไปใช้ อีกทั้งยังส่งผลต่อความพึงพอใจของผู้ใช้ ประสิทธิภาพของระบบ และผลลัพธ์ขององค์กรโดยรวม (Aziz, Haizan Nor, Jusoh, Wan Ab. Rahman, & Mohd Ali, 2024; Solida & Ardiyansyah, 2024; Pata , Manurung, Wardhana, Budiman, & Nataliawati, 2024; ศุภนิดา โคตรชาดา และ ดารณี เอื้อชนะจิต, 2566; สาวิตรี จูเจี่ย และ สุรศักดิ์ มังสิงห์, 2016) โดยกรอบคุณภาพของสารสนเทศมีความคล้ายกับกรอบคุณภาพของข้อมูลแต่ต่างกันที่บริบท กรอบคุณภาพของสารสนเทศมีดังต่อไปนี้

- **ความถูกต้อง (Accuracy)**

สารสนเทศที่ดีต้องมีความถูกต้อง โดยความถูกต้องของสารสนเทศมีผลมาจากข้อมูลและการประมวลผลข้อมูล หากข้อมูลที่นำเข้าผิดพลาด หรือกระบวนการประมวลผลไม่ถูกต้อง สารสนเทศที่ได้จะไม่สะท้อนข้อเท็จจริงและอาจให้ไม่สามารถบรรลุตามวัตถุประสงค์/เป้าหมายของการใช้สารสนเทศที่กำหนดไว้

- **ความสมบูรณ์ (Completeness)**

สารสนเทศที่ดีควรมีความสมบูรณ์ครบถ้วน ไม่ขาดส่วนสำคัญที่จำเป็นต่อการใช้งาน

- **ความเกี่ยวข้อง (Relevance)**

สารสนเทศต้องมีความเกี่ยวข้องกับความต้องการของผู้ใช้หรือวัตถุประสงค์/เป้าหมายที่ได้กำหนดไว้ สารสนเทศที่แม้จะถูกต้องและครบถ้วน แต่หากไม่สอดคล้องกับวัตถุประสงค์/เป้าหมายก็ไม่จัดว่าเป็นสารสนเทศที่มีคุณภาพ

- **ความเชื่อถือได้ (Reliability)**

สารสนเทศต้องสามารถตรวจสอบย้อนกลับได้และมีแหล่งที่มาเชื่อถือได้ ผู้ใช้สามารถไว้วางใจในสารสนเทศนั้นได้ว่าจะไม่ถูกบิดเบือนหรือผิดเพี้ยนจากข้อเท็จจริง

- **ความทันเวลา (Timeliness)**

สารสนเทศที่มีคุณภาพต้องสามารถนำมาใช้ได้ทันต่อความต้องการในช่วงเวลาที่เหมาะสม สารสนเทศที่ไม่เป็นปัจจุบันอาจส่งผลให้การตัดสินใจไม่สอดคล้องกับสถานการณ์จริง

- **ความสามารถในการใช้งาน (Usability)**

สารสนเทศควรอยู่ในรูปแบบที่เข้าใจง่าย ไม่คลุมเครือ ใช้งานสะดวกและเหมาะสมกับผู้นำสารสนเทศใช้ มีการออกแบบให้รองรับการตีความที่ถูกต้องได้ง่าย

- **ความคุ้มค่า (Economical)**

สารสนเทศที่ได้ควรจะต้องมีเหมาะสมในด้านความคุ้มค่ากับทรัพยากรที่ใช้ในการผลิตสารสนเทศ

- **ความปลอดภัย (Security)**

สารสนเทศจัดเป็นสินทรัพย์ที่มีค่าขององค์กร จึงควรได้รับการปกป้องจากการเข้าถึงโดยไม่ได้รับอนุญาต

- **ความสามารถในการเข้าถึง (Accessibility)**

สารสนเทศที่ดีควรสามารถเข้าถึงได้อย่างสะดวกโดยผู้ที่ได้รับอนุญาตให้ใช้งาน ทั้งในมิติของเทคโนโลยี เวลา และสถานที่ หากข้อมูลเข้าถึงได้ยากจะเป็นการลดทอนคุณค่าของสารสนเทศนั้น

1.3 ความหมายของระบบสารสนเทศและองค์ประกอบของระบบสารสนเทศ

ความหมายของและองค์ประกอบของระบบสารสนเทศ มีดังนี้

1.3.1 ความหมายของระบบสารสนเทศ

ตามที่ได้กล่าวว่า ระบบสารสนเทศ เป็นคำประสมที่เกิดจากคำว่า ระบบ และสารสนเทศ โดยได้กล่าวถึงความหมายและองค์ประกอบของระบบและสารสนเทศในหัวข้อก่อนหน้านี้ สามารถสรุปความหมายของระบบสารสนเทศได้ความว่า ระบบสารสนเทศ (Information System) หมายถึง ระบบที่ประกอบด้วยองค์ประกอบต่าง ๆ ได้แก่ ฮาร์ดแวร์ ซอฟต์แวร์ ฐานข้อมูล ระบบสื่อสาร และบุคลากร ที่ทำงานร่วมกันอย่างบูรณาการ เพื่อจัดการข้อมูลให้กลายเป็นสารสนเทศที่มีคุณค่า ระบบนี้ทำหน้าที่รวบรวม จัดเก็บ ประมวลผล นำเสนอผลลัพธ์ รวมทั้งเผยแพร่สารสนเทศในรูปแบบที่เหมาะสมต่อการใช้งานและเป็นไปตามวัตถุประสงค์ของการสร้างสารสนเทศ

การทำงานของระบบสารสนเทศประกอบด้วยกิจกรรม 3 กิจกรรม คือ การนำเข้าข้อมูล (Input) การประมวลผล (Processing) และการนำเสนอผลลัพธ์ (Output) ระบบสารสนเทศอาจมีการสะท้อนกลับ (Feed back) สำหรับใช้ในการประเมินและปรับปรุงข้อมูลนำเข้าเพื่อให้ระบบสารสนเทศมีประสิทธิภาพมากขึ้น (อรสา เตติวัฒน์, 2567; พรรณี สวนเพลง, 2566; โอภาส เอี่ยมสิริวงศ์, 2566)

ระบบสารสนเทศอาจอยู่ในรูปแบบของการดำเนินงานแบบดั้งเดิมหรือดำเนินงานด้วยมือ แต่ภายหลังการแพร่ระบาดของโรคโควิด-19 ซึ่งเป็นปัจจัยสำคัญที่เร่งให้เกิดการปรับตัวขององค์กรและภาคส่วนต่าง ๆ เทคโนโลยีสารสนเทศซึ่งถือเป็นองค์ประกอบหนึ่งของเทคโนโลยีดิจิทัล จึงถูกนำมาใช้เป็นเครื่องมือหลักในการดำเนินงาน เพื่อสนับสนุนการทำงานจากระยะไกล การจัดการข้อมูลแบบเรียลไทม์ และการเปลี่ยนผ่านสู่ระบบดิจิทัลอย่างเต็มรูปแบบ (อรสา เตติวัฒน์, 2567; พรรณี สวนเพลง, 2566)

ระบบสารสนเทศมีบทบาทสำคัญในการสนับสนุนการตัดสินใจ การควบคุมการดำเนินงานและการเพิ่มประสิทธิภาพขององค์กร โดยมีงานวิจัยทั้งในประเทศและต่างประเทศที่สนับสนุนความสำคัญของระบบสารสนเทศมีผลเชิงบวกต่อองค์กรหรือการดำเนินงานด้านต่าง ๆ ตัวอย่างงานวิจัยแสดงดังตารางที่ 2-1

ตารางที่ 2-1 ตัวอย่างงานวิจัยที่สนับสนุนความสำคัญของระบบสารสนเทศที่มีผลเชิงบวกต่อองค์กรหรือการดำเนินงาน

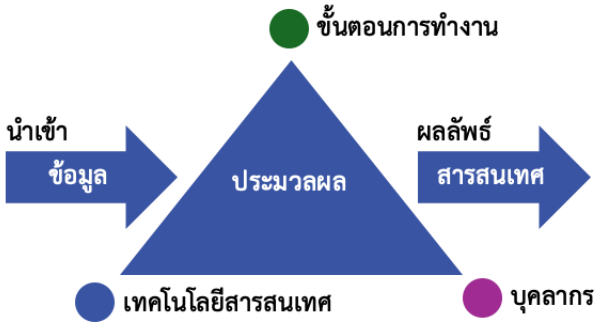
ที่	ชื่องานวิจัย	ผลการวิจัย
1	การพัฒนาระบบสารสนเทศเพื่อการบริหารองค์การ โดยใช้หลักการจัดการการไหลของสารสนเทศโลจิสติกส์ กรณีศึกษาคณะโลจิสติกส์และดิจิทัลซัพพลายเชน มหาวิทยาลัยนเรศวร (เจษฎา โพธิ์จันทร์, สุนันท์ ธาติ, ณัฐชยา ปันชุม, และ สุดารัตน์ จันทิมา, 2565, หน้า 169)	ระบบสารสนเทศเพื่อการบริหารองค์การให้ข้อมูลที่สามารถนำไปใช้ในการวางแผน การกำหนดมาตรการ/นโยบายในการบริการจัดการองค์กรได้ สามารถช่วยให้การจัดการเวลาและการบริหารจัดการทรัพยากรขององค์กรมีประสิทธิภาพมากขึ้น
2	การพัฒนาระบบสารสนเทศโรงพยาบาลสัตว์ โดย Power BI Desktop คณะสัตวแพทยศาสตร์ มหาวิทยาลัยขอนแก่น (ฐานิตย์ อริยะกอง, 2568, หน้า 16)	ผู้ใช้ระบบสารสนเทศสามารถใช้สารสนเทศที่เป็นประโยชน์ในด้านต่าง ๆ จากระบบสารสนเทศในการกำกับติดตามผลการดำเนินงานของโรงพยาบาลสัตว์ ใช้ประกอบการตัดสินใจ วางแผน ปรับปรุงกระบวนการ และพัฒนาประสิทธิภาพการให้บริการด้านสุขภาพสัตว์
3	Leveraging Information Systems for Strategic Management: Enhancing Decision-Making and Organizational Performance (Hera, et al., 2024, p. 1060)	การใช้ระบบสารสนเทศในเชิงกลยุทธ์มีผลเชิงบวกต่อการตัดสินใจของผู้บริหาร และส่งผลให้ประสิทธิภาพขององค์กรโดยรวมเพิ่มขึ้นอย่างมีนัยสำคัญตามการวิเคราะห์เชิงปริมาณและแบบจำลองถดถอยเชิงเส้น
4	การพัฒนาระบบสารสนเทศสำหรับการบริหารจัดการโครงการ คณะวิทยาการจัดการ มหาวิทยาลัยสงขลานครินทร์ (อนุวัฒน์ พัฒนเชียร และ อรญา สุวรรณโณ, 2566, หน้า 116)	ระบบสารสนเทศสามารถเพิ่มความสะดวกรวดเร็วเข้าถึงได้ทุกที่ทุกเวลาและทุกอุปกรณ์ สามารถเชื่อมโยงการทำงานของหน่วยงานภายในบูรณาการข้อมูลข้ามหน่วยงานผ่านระบบดิจิทัล ทำให้ผู้บริหาร
5	The Impact of Management Information Systems on Decision-Making Efficiency (Hamdat, Ceskakusumadewi, Samalam, Rizal, & Lawalata, 2024, pp. 67-68)	การบูรณาการระบบสารสนเทศเพื่อการจัดการเข้ากับโครงสร้างขององค์กรช่วยเพิ่มประสิทธิภาพของกระบวนการตัดสินใจ ทำให้กระบวนการตัดสินใจมีความคล่องตัวมากขึ้น ส่งเสริมกลยุทธ์ที่ขับเคลื่อนด้วยข้อมูล และส่งผลต่อการยกระดับประสิทธิภาพขององค์กร

ตารางที่ 2-1 (ต่อ)

ที่	ชื่องานวิจัย	ผลการวิจัย
6	Assessing the effectiveness of accounting information systems in the era of COVID-19 pandemic (Al-Okaily, 2024, pp. 169-170)	ความมีประสิทธิภาพของระบบสารสนเทศทางการบัญชีส่งผลเชิงบวกต่อประสิทธิภาพของบุคลากรในองค์กร และความสามารถของหน่วยงานในการดำเนินงานด้านการเงินอย่างมีประสิทธิภาพและเป็นระบบ ซึ่งส่งผลโดยตรงต่อความสำเร็จขององค์กร
7	Going organic or staying traditionalistic? The role of agriculture information system (Doanh, Quynh, & Pham, 2022)	ระบบสารสนเทศด้านการเกษตรมีผลเชิงบวกโดยตรงต่อความตั้งใจของเกษตรกรในการเปลี่ยนจากการทำเกษตรแบบดั้งเดิมไปสู่เกษตรอินทรีย์ และยังมีบทบาทเป็นตัวแปรเสริมแรงความสัมพันธ์ระหว่างการรับรู้ถึงประโยชน์และพฤติกรรมการตัดสินใจของเกษตรกร

1.3.2 องค์ประกอบของระบบสารสนเทศ

ระบบสารสนเทศที่ใช้เทคโนโลยีสารสนเทศเป็นเครื่องมือหลักในการทำงานสามารถจำแนกองค์ประกอบได้เป็น 3 องค์ประกอบ แสดงดังภาพที่ 2.4



ภาพที่ 2.4 องค์ประกอบของระบบสารสนเทศ

ที่มา: ปรับปรุงจาก Roch, Fowler, Smith, & Bourgeois (2022)

1) ขั้นตอนการทำงาน

ขั้นตอนการทำงาน (Procedures) เป็นองค์ประกอบทางด้านการดำเนินงานที่เกี่ยวข้องกับระบบสารสนเทศทั้งหมด เพื่อใช้กำกับและควบคุมให้ระบบสารสนเทศสร้างผลลัพธ์เป็นไปตามวัตถุประสงค์ขององค์กร ขั้นตอนการทำงานประกอบด้วยนโยบาย กลยุทธ์ และกฎระเบียบการใช้ระบบสารสนเทศที่รวมถึงขั้นตอนการปฏิบัติงาน วิธีการบำรุงรักษา และการควบคุมความปลอดภัยของระบบสารสนเทศ (โอภาส เอี่ยมสิริวงศ์, 2566)

2) เทคโนโลยีสารสนเทศ

เทคโนโลยีสารสนเทศ (Information Technology: IT) เป็นองค์ประกอบที่บูรณาการการทำงานร่วมกันระหว่างเครื่องมือ กระบวนการ เทคโนโลยีคอมพิวเตอร์และเทคโนโลยีการสื่อสารโทรคมนาคมเพื่อให้ได้มาซึ่งสารสนเทศ (พรรณี สวนเพลง, 2566; โอภาส เอี่ยมสิริวงศ์, 2566) ประกอบด้วย

- **ฮาร์ดแวร์**

ฮาร์ดแวร์ (Hardware) หมายถึงอุปกรณ์ในระบบคอมพิวเตอร์ที่ทำงานร่วมกันเพื่อการนำเข้าข้อมูล ประมวลผลและแสดงผล

- **ซอฟต์แวร์**

ซอฟต์แวร์ (Software) หมายถึง โปรแกรมหรือชุดของโปรแกรมคอมพิวเตอร์ที่นำมาใช้เพื่อควบคุมการทำงานของฮาร์ดแวร์และการประมวลผล

- **ฐานข้อมูล**

ฐานข้อมูล (Database) หมายถึง ระบบจัดการและจัดเก็บข้อมูลในข้อมูลอย่างมีระเบียบแบบแผนเพื่อให้สามารถเรียกใช้ ค้นหา และจัดการข้อมูลได้อย่างมีประสิทธิภาพ ฐานข้อมูลที่ดีจะช่วยให้ข้อมูลและสารสนเทศเป็นไปตามกรอบคุณภาพ เช่น ลดความซ้ำซ้อนของข้อมูล เพิ่มความถูกต้องของข้อมูล ควบคุมการเข้าถึงข้อมูล เป็นต้น

- **ระบบเครือข่ายคอมพิวเตอร์**

ระบบเครือข่ายคอมพิวเตอร์ (Computer Networking) หมายถึง ระบบการเชื่อมต่อของคอมพิวเตอร์และอุปกรณ์ที่ทำหน้าที่ในการสื่อสารโทรคมนาคมเพื่อเผยแพร่แบ่งปันการใช้ข้อมูลและสารสนเทศ

3) บุคลากร

บุคลากร (People) เป็นองค์ประกอบที่มีความสำคัญที่สุดของระบบสารสนเทศ เนื่องจากระบบสารสนเทศไม่สามารถเกิดขึ้นมาในองค์กรเองได้จำเป็นจะต้องมีผู้วางแผน ออกแบบ พัฒนา ดูแลรักษา ไปจนถึงผู้ใช้ระบบสารสนเทศ บุคลากรที่เกี่ยวข้องกับระบบสารสนเทศแสดงดังตารางที่ 2-2 (Roch, Fowler, Smith, & Bourgeois, 2022)

ตารางที่ 2-2 บุคลากรที่เกี่ยวข้องกับระบบสารสนเทศ

กลุ่มผู้บริหาร	
ประเภทบุคลากร	**บทบาทและหน้าที่**
ประธานเจ้าหน้าที่ฝ่ายสารสนเทศ (Chief Information Officer: CIO)	ผู้บริหารระดับสูงขององค์กรที่มีหน้าที่กำหนดทิศทางวางกลยุทธ์ และดูแลการใช้ระบบภายในองค์กรเพื่อให้สนับสนุนเป้าหมายทางธุรกิจ (Business-IT Alignment)

ตารางที่ 2-2 (ต่อ)

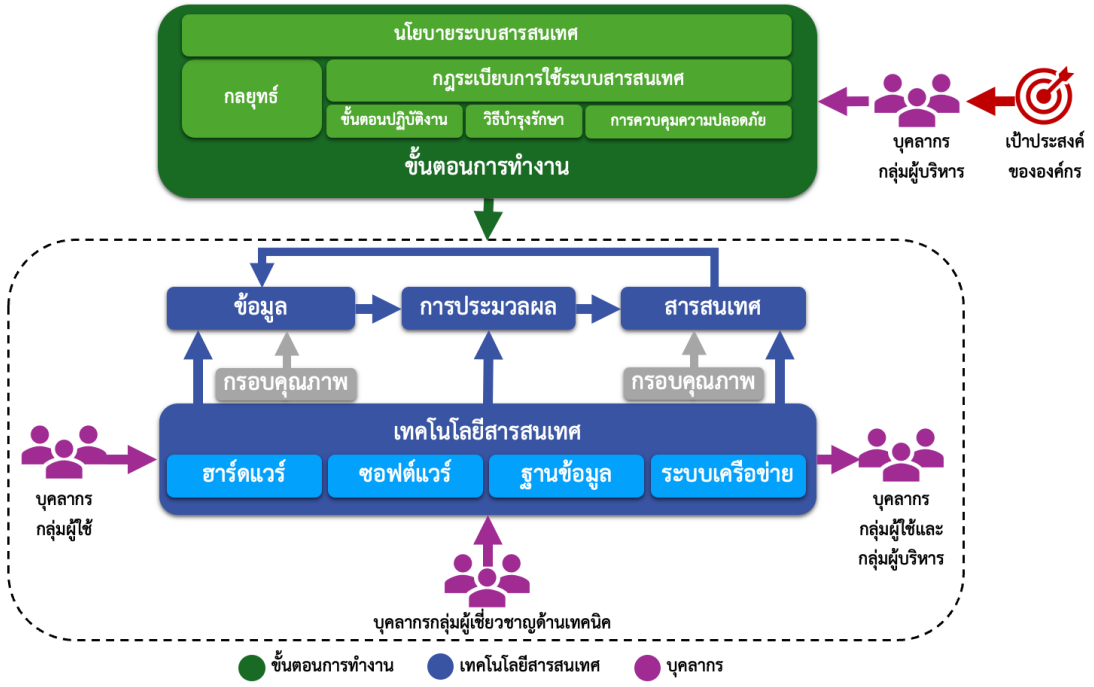
ประเภทบุคลากร	บทบาทและหน้าที่
ผู้บริหารที่ใช้ระบบสำหรับการตัดสินใจ (Executive Users)	ผู้บริหารระดับสูงที่ใช้สารสนเทศจากระบบสารสนเทศเพื่อวิเคราะห์สถานการณ์ กำหนดกลยุทธ์ และตัดสินใจเชิงนโยบาย
ที่ปรึกษาระบบ (IT Consultant)	ผู้ที่ให้คำแนะนำเชิงกลยุทธ์เกี่ยวกับการเลือกใช้ออกแบบ หรือปรับปรุงระบบสารสนเทศ เพื่อเพิ่มประสิทธิภาพในการดำเนินงานขององค์กร
ผู้จัดการโครงการ IT (IT Project Manager)	เป็นผู้ที่วางแผน ควบคุม และติดตามการดำเนินโครงการพัฒนาระบบสารสนเทศให้เป็นไปตามเวลา งบประมาณ และขอบเขตที่กำหนด
หัวหน้าแผนก IT (IT Department Head / IT Manager)	เป็นผู้ที่บริหารงานโดยรวมของฝ่าย IT ดูแลทรัพยากรทั้งคน งบประมาณ และเทคโนโลยี รวมถึงประสานกับหน่วยงานอื่น ๆ
กลุ่มผู้เชี่ยวชาญด้านเทคนิค	
ประเภทบุคลากร	**บทบาทและหน้าที่**
นักวิเคราะห์และออกแบบระบบ (Systems Analyst and Designer)	ผู้ที่ทำหน้าที่ศึกษาความต้องการของผู้ใช้งาน ออกแบบโครงสร้างระบบ และประสานงานระหว่างผู้ใช้กับทีมพัฒนา นักวิเคราะห์ระบบ มีบทบาทสำคัญในการแปลงความต้องการทางธุรกิจเป็นข้อกำหนดทางเทคนิค
นักพัฒนาซอฟต์แวร์ / นักเขียนโปรแกรม (Software Developer / Programmer)	ผู้ที่ทำหน้าที่เขียนโปรแกรมหรือพัฒนาซอฟต์แวร์ตามแบบจำลองหรือความต้องการที่นักวิเคราะห์ระบบกำหนดโดยใช้ภาษาคอมพิวเตอร์
ผู้ดูแลระบบ (Operations Administrator)	ผู้ที่รับผิดชอบในการติดตั้ง บำรุงรักษา และแก้ไขปัญหาเกี่ยวกับระบบคอมพิวเตอร์และเครื่องผู้ให้บริการเพื่อให้ระบบพร้อมใช้งานอยู่เสมอ
ผู้ดูแลเครือข่าย (Network Administrator)	ผู้ที่ดูแลโครงสร้างพื้นฐานด้านเครือข่ายคอมพิวเตอร์ เช่น การเชื่อมต่อเครือข่ายภายในและภายนอกองค์กร อุปกรณ์เราเตอร์ และระบบรักษาความปลอดภัยเครือข่าย

ตารางที่ 2-2 (ต่อ)

ประเภทบุคลากร	บทบาทและหน้าที่
วิศวกร DevOps (DevOps Engineer)	ผู้เชี่ยวชาญที่รับผิดชอบการออกแบบระบบอัตโนมัติเพื่อเชื่อมโยงทีมพัฒนาซอฟต์แวร์ (Development) และทีมผู้ดูแลระบบ (Operations) โดยประยุกต์ใช้แนวทาง CI/CD ร่วมกับเครื่องมือและกระบวนการที่สนับสนุนการทำงานร่วมกันระหว่างทั้งสองทีมอย่างมีประสิทธิภาพและต่อเนื่อง
ผู้ดูแลฐานข้อมูล (Database Administrator: DBA)	ผู้ที่จัดการ ดูแล และรักษาความปลอดภัยของฐานข้อมูล รวมถึงการสำรองข้อมูล การกู้คืน และการปรับปรุงประสิทธิภาพการเข้าถึงข้อมูล
เจ้าหน้าที่สนับสนุนด้านเทคโนโลยีสารสนเทศ (IT Support Staff)	ผู้ให้บริการช่วยเหลือผู้ใช้งานระบบสารสนเทศในระดับเบื้องต้น เช่น การแก้ไขปัญหาการใช้งานทั่วไป การติดตั้งโปรแกรม การตั้งค่าอุปกรณ์ และการส่งต่อปัญหาที่ซับซ้อนให้ทีมเทคนิคที่เกี่ยวข้อง
กลุ่มผู้ใช้งาน	
ประเภทบุคลากร	**บทบาทและหน้าที่**
พนักงานที่ใช้ระบบในการทำงานประจำ (Operational Users)	ผู้ปฏิบัติงานในระดับหน้างานที่ใช้ระบบในการป้อนข้อมูล ค้นหาข้อมูล หรือรายงานผล เช่น เจ้าหน้าที่บัญชี พนักงานคลังสินค้า
ลูกค้าที่ใช้ระบบสำหรับการติดต่อหรือสั่งซื้อสินค้า (External Users / Customers)	ผู้ใช้งานภายนอก เช่น ลูกค้า คู่ค้า หรือประชาชนทั่วไป ที่เข้าถึงระบบผ่านเว็บไซต์ แอปพลิเคชัน หรือระบบบริการอัตโนมัติ

1.3.3 ความสัมพันธ์ขององค์ประกอบของระบบสารสนเทศ

การแสดงความสัมพันธ์ขององค์ประกอบต่าง ๆ ของระบบสารสนเทศ แสดงดังภาพที่ 2.5 ซึ่งประกอบด้วยขั้นตอนการทำงาน เทคโนโลยีสารสนเทศและบุคลากร กรอบของระบบสารสนเทศถูกกำหนดจากนโยบายระบบสารสนเทศขององค์กรโดยบุคลากรกลุ่มผู้บริหารระบบสารสนเทศ ซึ่งนโยบายระบบสารสนเทศนี้มีความสอดคล้องและประสานกันระหว่างเป้าประสงค์ขององค์กรหรือเป้าหมายทางธุรกิจ (Business Goals) กับกลยุทธ์และการใช้งานระบบสารสนเทศ ซึ่งมีเป้าหมายหลักในการทำให้ระบบสารสนเทศไม่ใช่เป็นเพียงเครื่องมือสนับสนุนการทำงานในองค์กรเท่านั้น แต่เป็นกลไกขับเคลื่อนการดำเนินงานขององค์กรให้บรรลุตามเป้าประสงค์รวมทั้งทำให้เกิดความได้เปรียบในการแข่งขัน

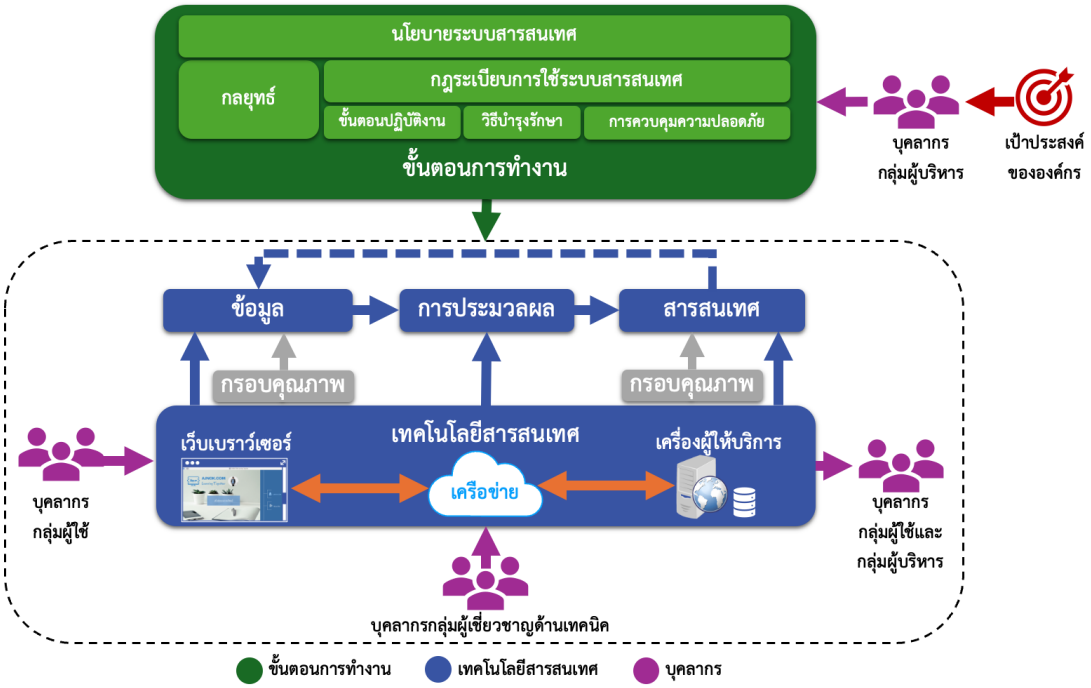


ภาพที่ 2.5 แบบจำลองความสัมพันธ์ขององค์ประกอบของระบบสารสนเทศ

การนำนโยบายไปสู่การสร้างระบบสารสนเทศเริ่มตั้งแต่การวิเคราะห์และออกแบบระบบ การพัฒนาระบบสารสนเทศ การติดตั้งระบบสารสนเทศและการบำรุงรักษา จะเป็นบทบาทหน้าที่ของบุคลากรกลุ่มผู้เชี่ยวชาญด้านเทคนิค ระบบสารสนเทศที่ได้จะต้องสอดคล้องกับนโยบายสารสนเทศ โดยกิจกรรมหลักของระบบสารสนเทศอันประกอบด้วยการนำข้อมูลเข้า การประมวลผล และการนำเสนอผลลัพธ์ในรูปแบบของสารสนเทศจะใช้เทคโนโลยีสารสนเทศเป็นเครื่องมือหลักในการดำเนินการ บุคลากรผู้ใช้งานทั้งในระดับบริหารและในระดับปฏิบัติการก็จะใช้เทคโนโลยีสารสนเทศเป็นเครื่องมือในการใช้งาน

2. ความสัมพันธ์ของระบบสารสนเทศและเว็บแอปพลิเคชัน

จากการศึกษาในบทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน จะเห็นได้ว่าเว็บแอปพลิเคชันเป็นรูปแบบหนึ่งของเทคโนโลยีสารสนเทศที่มีส่วนประกอบในการทำงานครอบคลุมทั้งฮาร์ดแวร์ ซอฟต์แวร์ ฐานข้อมูลและระบบเครือข่าย โดยมีความสามารถในการรับข้อมูลจากผู้ใช้ผ่านเว็บเบราว์เซอร์เพื่อนำไปประมวลผลในเครื่องผู้ให้บริการและจัดการกับข้อมูลในรูปของฐานข้อมูล ตลอดจนการแสดงผลลัพธ์ในรูปของสารสนเทศกลับไปยังผู้ใช้ผ่านเว็บเบราว์เซอร์เช่นกัน ในบริบทของระบบสารสนเทศ เว็บแอปพลิเคชันทำหน้าในฐานะเทคโนโลยีสารสนเทศที่เชื่อมโยงระหว่างข้อมูล บุคลากรและขั้นตอนการทำงาน ภายใต้กรอบนโยบายและเป้าประสงค์ขององค์กร โดยมีบทบาทสำคัญในการสนับสนุนการดำเนินงาน การบริหารจัดการ และการตัดสินใจอย่างมีประสิทธิภาพในยุคดิจิทัล แบบจำลองการบูรณาการเว็บแอปพลิเคชันในระบบสารสนเทศองค์กรแสดงดังภาพที่ 2.6



ภาพที่ 2.6 แบบจำลองการบูรณาการเว็บแอปพลิเคชันในระบบสารสนเทศ

2.1 เว็บแอปพลิเคชันในฐานะองค์ประกอบของเทคโนโลยีสารสนเทศ

เว็บแอปพลิเคชันเป็นแพลตฟอร์มหนึ่งของเทคโนโลยีสารสนเทศที่ทำหน้าที่เป็นเครื่องมือในการรับและประมวลผลข้อมูล จัดระเบียบข้อมูลด้วยฐานข้อมูล และนำเสนอข้อมูลให้เป็นสารสนเทศที่มีประโยชน์ต่อผู้ใช้งาน การทำงานของเว็บแอปพลิเคชันอยู่ในชั้นของเทคโนโลยีสารสนเทศ ซึ่งประกอบด้วย

2.1.1 ฮาร์ดแวร์

โครงสร้างการทำงานของเว็บแอปพลิเคชันต้องอาศัยฮาร์ดแวร์ที่ประกอบด้วยคอมพิวเตอร์ของผู้ใช้งาน เครื่องผู้ให้บริการเว็บและอุปกรณ์ด้านระบบเครือข่ายคอมพิวเตอร์ ฮาร์ดแวร์เหล่านี้ทำหน้าที่เป็นโครงสร้างพื้นฐานที่สำคัญในการรองรับการทำงานของเว็บแอปพลิเคชัน โดยคอมพิวเตอร์หรืออุปกรณ์ของผู้ใช้งานจะต้องมีความสามารถในการใช้เว็บเบราว์เซอร์ ในขณะที่เครื่องผู้ให้บริการเว็บต้องมีสมรรถนะในการให้บริการเว็บประมวลผลและจัดเก็บข้อมูลได้อย่างมีประสิทธิภาพ

2.1.2 ซอฟต์แวร์

เว็บแอปพลิเคชันเป็นรูปแบบหนึ่งของซอฟต์แวร์ที่ทำงานร่วมกับซอฟต์แวร์หรือโปรแกรมอื่นๆ เพื่อให้มีความสามารถการทำงานบนเว็บเทคโนโลยีที่ประกอบด้วยเว็บเบราว์เซอร์ โปรแกรมให้บริการเว็บ ตัวแปลภาษา และระบบจัดการฐานข้อมูล ซอฟต์แวร์เหล่านี้ทำงานร่วมกันเป็นระบบนิเวศที่สมบูรณ์เพื่อให้เว็บแอปพลิเคชันสามารถให้บริการได้ตามที่ออกแบบไว้

2.1.3 ฐานข้อมูล

เว็บแอปพลิเคชันสามารถรับข้อมูลจากผู้ใช้แล้วนำมาจัดเก็บในเครื่องคอมพิวเตอร์ผู้ให้บริการอย่างเป็นระบบระเบียบในรูปแบบของฐานข้อมูลเพื่อให้สามารถค้นหาและจัดการข้อมูลได้อย่างมีประสิทธิภาพ ทั้งในด้านความถูกต้อง ความปลอดภัย และความรวดเร็วในการเรียกใช้งาน

2.1.4 ระบบเครือข่าย

ระบบเครือข่ายคอมพิวเตอร์ทำหน้าที่เป็นตัวกลางในการเชื่อมโยงระหว่างผู้ใช้งานและเครื่องผู้ให้บริการเว็บ ทำให้สามารถรับส่งข้อมูลและสารสนเทศระหว่างกันได้อย่างต่อเนื่องไม่ว่าจะอยู่ในเครือข่ายภายในองค์กรหรือบนเครือข่ายอินเทอร์เน็ต ระบบเครือข่ายจึงเป็นองค์ประกอบที่ทำให้เว็บแอปพลิเคชันสามารถเข้าถึงได้จากทุกที่ทุกเวลา

2.2 เว็บแอปพลิเคชันกับกระบวนการแปลงข้อมูลเป็นสารสนเทศ

เว็บแอปพลิเคชันมีบทบาทสำคัญในการเป็นเครื่องมือที่ใช้ในกระบวนการประมวลผลข้อมูลให้เป็นสารสนเทศ มีความสามารถในการดำเนินกิจกรรมของระบบสารสนเทศ 3 กิจกรรม ประกอบด้วย การรับข้อมูล การประมวลผล และการแสดงผล

2.2.1 การรับข้อมูล

เว็บแอปพลิเคชันสามารถรับข้อมูลจากผู้ใช้เพื่อนำไปประมวลผลหรือจัดเก็บข้อมูลในฐานข้อมูลผ่านฟอร์มที่สร้างขึ้นด้วยภาษา HTML และ CSS สามารถตรวจสอบความถูกต้องของข้อมูลด้วยภาษา JavaScript ก่อนนำส่งไปประมวลผลที่เครื่องผู้ให้บริการ การรับข้อมูลนี้เป็นจุดเริ่มต้นที่สำคัญของกระบวนการแปลงข้อมูลเป็นสารสนเทศ เนื่องจากข้อมูลที่รับเข้ามาจะต้องมีคุณภาพและความถูกต้องเพื่อให้การประมวลผลในขั้นตอนถัดไปเป็นไปอย่างมีประสิทธิภาพ

2.2.2 การประมวลผล

เว็บแอปพลิเคชันมีการประมวลผลข้อมูลตามตรรกะทางธุรกิจในเครื่องผู้ให้บริการ โดยใช้ภาษาประมวลผลในเครื่องคอมพิวเตอร์ผู้ให้บริการ เช่น ภาษา PHP ภาษา Python และภาษา Ruby เป็นต้น นอกจากนี้เว็บแอปพลิเคชันยังมีความสามารถในการจัดเก็บข้อมูลในฐานข้อมูลได้อีกด้วย

2.2.3 การแสดงผล

เว็บแอปพลิเคชันมีกลไกการนำข้อมูลที่ได้จากการประมวลผลหรือข้อมูลจากฐานข้อมูลเพื่อแสดงผลในรูปของสารสนเทศลักษณะต่าง ๆ เช่น ข้อความ ตาราง และกราฟ บนเว็บเบราว์เซอร์ของผู้ใช้ การแสดงผลนี้เป็นขั้นตอนสุดท้ายที่ทำให้ผู้ใช้ได้รับสารสนเทศที่ต้องการในรูปแบบที่เข้าใจได้และสามารถนำไปใช้ประโยชน์

2.3 บทบาทของเว็บแอปพลิเคชันในการสนับสนุนระบบสารสนเทศ

เว็บแอปพลิเคชันมีบทบาทสำคัญในการเสริมสร้างศักยภาพของระบบสารสนเทศในองค์กร ทั้งในด้านการใช้งาน การสื่อสาร การจัดการข้อมูล และความมั่นคงปลอดภัย โดยเว็บแอปพลิเคชันทำหน้าที่เป็นกลไกกลางที่เชื่อมโยงผู้ใช้งานเข้ากับข้อมูลและกระบวนการทำงานภายในระบบผ่านแพลตฟอร์มเว็บ ซึ่งสามารถรองรับการใช้

งานได้หลากหลายสถานการณ์และอุปกรณ์ การสนับสนุนของเว็บแอปพลิเคชันต่อระบบสารสนเทศสามารถจำแนกได้เป็น 4 ด้าน ดังนี้

2.3.1 การเข้าถึงข้อมูลและสารสนเทศ

เว็บแอปพลิเคชันสามารถใช้งานได้จากทุกที่ทุกเวลาผ่านเครือข่ายอินเทอร์เน็ต ไม่จำกัดเฉพาะคอมพิวเตอร์ภายในองค์กร ผู้ใช้งานสามารถเข้าถึงระบบผ่านเว็บเบราว์เซอร์บนคอมพิวเตอร์ สมาร์ตโฟน รวมทั้งแท็บเล็ตได้อย่างสะดวก ความสามารถนี้ช่วยสนับสนุนการทำงานที่มีความยืดหยุ่น รองรับการทำงานแบบไร้พรมแดน และเพิ่มประสิทธิภาพในการให้บริการขององค์กรในยุคดิจิทัล

2.3.2 การประมวลผลแบบเรียลไทม์

เว็บแอปพลิเคชันสามารถประมวลผลข้อมูลได้ทันทีที่ผู้ใช้ป้อนเข้าระบบ โดยข้อมูลจะถูกส่งไปยังเครื่องผู้ให้บริการเพื่อดำเนินการตามตรรกะทางธุรกิจ และแสดงผลกลับมายังผู้ใช้ในเวลาอันสั้น การประมวลผลแบบเรียลไทม์นี้มีความสำคัญต่อการตัดสินใจที่ต้องอาศัยข้อมูลที่ทันสมัย เช่น ระบบการเงิน การจองสินค้า หรือการติดตามสถานการณ์ปฏิบัติงาน เป็นต้น

2.3.3 การทำงานร่วมกัน

เว็บแอปพลิเคชันช่วยให้ผู้ใช้งานหลายคนสามารถเข้าถึงข้อมูลเดียวกันและทำงานร่วมกันได้ในระบบเดียวแบบออนไลน์ ทั้งการเรียกดู การเพิ่ม การแก้ไข และการลบข้อมูล การทำงานร่วมกันนี้ช่วยลดความซ้ำซ้อน เพิ่มความโปร่งใส และส่งเสริมการสื่อสารระหว่างหน่วยงานหรือทีมงานภายในองค์กร

2.3.4 ความปลอดภัยและการจัดเก็บข้อมูล

เว็บแอปพลิเคชันมีความสามารถในการจัดเก็บข้อมูลไว้ในระบบฐานข้อมูล สามารถกำหนดสิทธิ์การเข้าถึงได้ตามบทบาทของผู้ใช้ นอกจากนี้ยังสามารถนำเทคโนโลยีด้านความมั่นคงปลอดภัย เช่น การเข้ารหัสข้อมูล (Encryption) หรือระบบพิสูจน์ตัวตน (Authentication) มาใช้ร่วมด้วย เพื่อเสริมสร้างความมั่นใจในการจัดการข้อมูลที่มีความละเอียดอ่อน

2.4 ตัวอย่างการใช้เว็บแอปพลิเคชันในระบบสารสนเทศ

เพื่อให้เห็นภาพของการนำเว็บแอปพลิเคชันมาประยุกต์ใช้เป็นกลไกด้านเทคโนโลยีของระบบสารสนเทศ หัวข้อนี้จึงเป็นการนำเสนอตัวอย่างการใช้เว็บแอปพลิเคชันในระบบสารสนเทศ โดยมีวัตถุประสงค์เพื่อแสดงให้เห็นถึงความหลากหลายของการใช้งานและประโยชน์ที่ได้รับจากการพัฒนาระบบสารสนเทศในรูปแบบเว็บแอปพลิเคชัน ตัวอย่างที่นำเสนอประกอบด้วย 3 ระบบที่มีลักษณะและวัตถุประสงค์ที่แตกต่างกัน ได้แก่ ระบบการจัดการธุรกิจในระดับชุมชน ระบบสารสนเทศภูมิศาสตร์เพื่อการเกษตร และระบบส่งเสริมการท่องเที่ยวด้วยเทคโนโลยีสมัยใหม่ ในแต่ละตัวอย่างจะครอบคลุมการวิเคราะห์ 5 ด้านสำคัญ ได้แก่ ลักษณะงานของระบบสารสนเทศ ระบบย่อยและการทำงาน องค์ประกอบทางเทคโนโลยีสารสนเทศ การสะท้อนกระบวนการแปลงข้อมูลเป็นสารสนเทศและผลลัพธ์การใช้ระบบ เพื่อให้ผู้อ่านสามารถวิเคราะห์ เปรียบเทียบ และประยุกต์ใช้เว็บแอปพลิเคชันในระบบสารสนเทศได้อย่างเหมาะสม

ตัวอย่างที่ 1	ระบบสารสนเทศจำหน่ายน้ำดื่มบ้านเพชรไพรวัลย์

ระบบสารสนเทศจำหน่ายน้ำดื่มบ้านเพชรไพรวัลย์ (อนันตพร อรุณฉาย และ ฆัมภิชา ตันติสันติสม, 2567) ได้รับการพัฒนาเพื่อบริหารจัดการข้อมูลที่เกี่ยวข้องกับการผลิตและจำหน่ายน้ำดื่มในระดับชุมชน โดยมีเป้าหมายเพื่อทดแทนระบบเดิมที่ใช้การจดบันทึกด้วยกระดาษที่มีข้อจำกัดด้านความแม่นยำ ประสิทธิภาพในการค้นหาและสรุปข้อมูล

- **ลักษณะงานของระบบสารสนเทศ**

  ระบบฯ สร้างขึ้นเพื่อใช้ในงานด้านการบริหารจัดการข้อมูลการผลิตและการจำหน่ายน้ำดื่ม

- **ระบบย่อยและการทำงาน**

ระบบฯ มี 8 ระบบย่อย ได้แก่ การจัดการข้อมูลสมาชิก การจัดการคำสั่งซื้อ การจัดการชำระเงิน การจัดการสินค้า การจัดการจัดส่งสินค้า การจัดการผู้ดูแลระบบ การจัดการรายจ่ายและการออกรายงาน ระบบมีการจำแนกสิทธิ์ผู้ใช้งานตามบทบาทหน้าที่โดยแบ่งผู้ใช้งานเป็น 2 กลุ่ม คือ กลุ่มสมาชิก และกลุ่มผู้ดูแลระบบ มีความสามารถในการควบคุมการเข้าถึงข้อมูลอย่างเหมาะสม

- **องค์ประกอบทางเทคโนโลยีสารสนเทศ**

ระบบฯ สร้างขึ้นเป็นเว็บแอปพลิเคชันที่เรียกใช้งานผ่านเบราว์เซอร์ ใช้ภาษา PHP เป็นภาษาประมวลผลฝั่งผู้ให้บริการเว็บ และใช้โปรแกรม MySQL เป็นระบบจัดการฐานข้อมูล

- **การสะท้อนกระบวนการแปลงข้อมูลเป็นสารสนเทศ**

ระบบฯ ได้รับข้อมูลจากสมาชิก เช่น รายการคำสั่งซื้อและข้อมูลการชำระเงิน จากนั้นประมวลผลเพื่อแสดงรายงานสถานะสินค้า รายงานยอดขาย และรายงานการจัดส่งสินค้าอย่างมีระบบ ช่วยให้ผู้ดูแลระบบสามารถตัดสินใจได้แม่นยำขึ้น และลดระยะเวลาในการเข้าถึงข้อมูล

- **ผลลัพธ์การใช้ระบบ**

ระบบฯ ช่วยลดเวลาในการค้นหาข้อมูลได้ 83.33% ช่วยลดข้อผิดพลาดในการบันทึกข้อมูลได้ 70% และช่วยลดเวลาการสรุปข้อมูลต่าง ๆ ได้ 75% ผู้ใช้มีความพึงพอใจในการใช้งานระบบอยู่ในระดับมาก สามารถนำระบบฯ ไปใช้ในบริบทอื่นๆ

ตัวอย่างที่ 2	ระบบสารสนเทศภูมิศาสตร์สำหรับกลุ่มข้าวภูเขาไฟบ้านโคกเมือง

ระบบสารสนเทศภูมิศาสตร์สำหรับกลุ่มข้าวภูเขาไฟบ้านโคกเมือง ตำบลจรเข้มาก อำเภอประโคนชัย จังหวัดบุรีรัมย์ (ณัฐวุฒิ ทะนันไธสง และ ณัฐพล วงษ์รัมย์, 2566) เพื่อจัดเก็บ วิเคราะห์ และแสดงผลข้อมูลเชิงพื้นที่ของเกษตรกรผู้ปลูกข้าวภูเขาไฟ มีวัตถุประสงค์เพื่อแก้ปัญหาการจัดเก็บข้อมูลที่ยังไม่เป็นระบบ และเพื่อสนับสนุนการตัดสินใจของกลุ่มเกษตรกรในการวางแผนการผลิตพืชตามลักษณะสมบัติดินและความต้องการใช้น้ำ

- **ลักษณะงานของระบบสารสนเทศ**

  ระบบฯ สร้างขึ้นเพื่อใช้ในงานด้านการสนับสนุนการจัดการเกษตรกรรมระดับชุมชน

- **ระบบย่อยและการทำงาน**

ระบบฯ มี 6 ส่วนหลัก ได้แก่ หน้าหลัก การจัดการข้อมูลผู้บันทึก การจัดการข้อมูลสมาชิก การจัดการข้อมูลเพาะปลูก การจัดการข้อมูลเขตความเหมาะสมของพืชเศรษฐกิจ และระบบรายงานข้อมูล มีการแบ่งสิทธิ์การเข้าถึงข้อมูลเป็น 2 กลุ่ม คือ กลุ่มผู้ดูแลระบบและกลุ่มสมาชิก

- **องค์ประกอบทางเทคโนโลยีสารสนเทศ**

ระบบฯ สร้างขึ้นเป็นเว็บแอปพลิเคชันโดยใช้ภาษา HTML ภาษา CSS และภาษา JavaScript ในการสร้างส่วนติดต่อกับผู้ใช้ที่แสดงผลในเว็บเบราว์เซอร์ ใช้ภาษา PHP เป็นภาษาประมวลผลฝั่งผู้ให้บริการเว็บทำงานร่วมกับโปรแกรม ArcGIS สำหรับวิเคราะห์ข้อมูลเชิงพื้นที่ รวมถึง Google Maps API ในการแสดงแผนที่และตำแหน่งแปลงเกษตร และใช้โปรแกรม Microsoft SQL Server เป็นระบบจัดการฐานข้อมูล นอกจากนี้ยังใช้อุปกรณ์กำหนดตำแหน่งบนพื้นโลกในการจัดเก็บข้อมูลเชิงพื้นที่

- **การสะท้อนกระบวนการแปลงข้อมูลเป็นสารสนเทศ**

ระบบฯ นำเข้าข้อมูลสมาชิกประกอบด้วย ชื่อ-สกุล ที่อยู่ พื้นที่การปลูก ผลผลิตประจำปีและข้อมูลการจำหน่าย รวมทั้งข้อมูลจากหน่วยงานราชการ ได้แก่ ข้อมูลบ่งชี้ทางภูมิศาสตร์ข้าวหอมมะลิดินภูเขาไฟ ข้อมูลเขตความเหมาะสมพืชเศรษฐกิจตามลักษณะสมบัติดินสำหรับการปลูกข้าว มันสำปะหลัง อ้อยและยางพารา แสดงผลลัพธ์เป็นรายงานในรูปแบบเชิงพื้นที่ด้วยแผนที่ภาพถ่ายดาวเทียม รายงานผลในรูปแบบตารางและสามารถส่งออกรายงานในรูปแบบของไฟล์เอกสารตารางคำนวณ เช่น รายงานพื้นที่ในครอบครอง และรายงานความต้องการปริมาณน้ำของพืชในแต่ละฤดูปลูก เป็นต้น

- **ผลลัพธ์การใช้ระบบ**

ผู้ใช้และผู้ดูแลระบบสามารถใช้สารสนเทศจากระบบฯ เพื่อแสดงถึงความเชื่อมโยงข้อมูลเชิงบรรยายและข้อมูลเชิงพื้นที่ของแปลงเกษตรกร ระบบฯ คำนวณปริมาณความต้องการน้ำเพื่อนำไปใช้ในการส่งเสริมและวางแผนการใช้น้ำสำหรับเพาะปลูกข้าวภูเขาไฟ นอกจากนี้ ระบบฯ ได้รับการประเมินประสิทธิภาพอยู่ในระดับดีมาก และผู้ใช้ระบบมีความพึงพอใจต่อการใช้ระบบในภาพรวมอยู่ในระดับมาก

ตัวอย่างที่ 3 ระบบส่งเสริมการท่องเที่ยวจังหวัดตากโดยใช้เทคโนโลยีเสมือนจริง

ระบบส่งเสริมการท่องเที่ยวจังหวัดตากโดยใช้เทคโนโลยีเสมือนจริง (วันชนะ จูบรรจง, วิไรวรรณ แสนชะนะ, รุ่งหมู่ล้อม, และ อภิชัย ซื่อสัตย์สกุลชัย, 2564) เป็นระบบสารสนเทศที่ออกแบบขึ้นเพื่อนำเสนอข้อมูลและสร้างประสบการณ์เสมือนด้วยเว็บแอปพลิเคชันให้กับนักท่องเที่ยวผ่านเทคโนโลยีภาพเสมือนจริงแบบ 360 องศาและโมเดลสามมิติ ได้รับการพัฒนาเพื่อส่งเสริมและประชาสัมพันธ์การท่องเที่ยวและเพิ่มโอกาสทางเศรษฐกิจให้แก่ผู้ประกอบการในจังหวัด

- **ลักษณะงานของระบบสารสนเทศ**

ระบบฯ สร้างขึ้นเพื่อใช้ในงานด้านการส่งเสริมการท่องเที่ยวและวัฒนธรรมท้องถิ่น

- **ระบบย่อยและการทำงาน**

ระบบฯ ประกอบด้วยข้อมูล 2 ส่วน ได้แก่ ส่วนสื่อดิจิทัลเสมือนจริงที่แสดงภาพแบบ 360 องศาและโมเดล 3 มิติ และส่วนข้อมูลประกอบของสถานที่ท่องเที่ยวเชิงบรรยาย

- **องค์ประกอบทางเทคโนโลยีสารสนเทศ**

ระบบฯ พัฒนาโดยใช้โปรแกรม Unity ร่วมกับอุปกรณ์ประมวลผลภาพเสมือนจริงสร้างภาพและโมเดลสามมิติ ในส่วนเว็บแอปพลิเคชันใช้ภาษา HTML ภาษา CSS และภาษา JavaScript ในการสร้างส่วนติดต่อกับผู้ใช้ ใช้ภาษา PHP เป็นภาษาประมวลผลในเครื่องผู้ให้บริการเว็บ และใช้โปรแกรม MySQL เป็นระบบจัดการฐานข้อมูล

- **การสะท้อนกระบวนการแปลงข้อมูลเป็นสารสนเทศ**

ระบบฯ นำเข้าข้อมูลเชิงวัฒนธรรมของแหล่งท่องเที่ยวและสร้างสื่อในรูปแบบภาพเสมือนจริงแบบ 360 องศาและโมเดล 3 มิติ จากนั้นประมวลผลและนำเสนอร่วมกับข้อเท็จจริงและเกร็ดความรู้เชิงวัฒนธรรมของแหล่งท่องเที่ยวด้วยเว็บแอปพลิเคชัน

- **ผลลัพธ์การใช้ระบบ**

ระบบฯ สามารถสร้างการรับรู้ ประสบการณ์ และแรงจูงใจให้กับนักท่องเที่ยวได้อย่างมีประสิทธิภาพทั้งในเชิงการออกแบบ การใช้งาน และข้อมูลเนื้อหา ผู้ใช้สามารถเห็นภาพสถานที่ท่องเที่ยวได้สมจริงกว่าการรับรู้ผ่านข้อมูลเชิงบรรยาย ในส่วนของการประเมินประสิทธิภาพของระบบฯ และการประเมินความพึงพอใจในการใช้งาน ประสิทธิภาพของระบบฯ ได้รับการประเมินอยู่ในระดับดีและผู้ใช้ระบบฯ มีความพึงพอใจต่อระบบฯ ในระดับมาก

3. วงจรการพัฒนาระบบ

วงจรการพัฒนาระบบเป็นกรอบแนวคิดที่สำคัญในการพัฒนาระบบสารสนเทศ รวมถึงเว็บแอปพลิเคชัน เพื่อให้ได้ระบบสารสนเทศที่ตรงกับความต้องการใช้งาน ตอบสนองต่อวัตถุประสงค์ทางธุรกิจ มีประสิทธิภาพ และสามารถควบคุมคุณภาพได้ การทำความเข้าใจวงจรการพัฒนาระบบจึงเป็นพื้นฐานสำคัญสำหรับผู้ที่จะพัฒนาเว็บแอปพลิเคชันเพื่อใช้เป็นเครื่องมือในสร้างระบบสารสนเทศ

3.1 ความหมายและความสำคัญของวงจรการพัฒนาระบบ

ความหมายและความสำคัญของวงจรการพัฒนาระบบ มีดังต่อไปนี้

3.1.1 ความหมายของวงจรการพัฒนาระบบ

วงจรการพัฒนาระบบ (System Development Life Cycle: SDLC) หมายถึง แนวทางหรือกระบวนการที่มีลำดับขั้นตอนอย่างเป็นระบบ ซึ่งใช้ในการพัฒนาระบบสารสนเทศหรือซอฟต์แวร์ให้สำเร็จ โดยมุ่งเน้นให้การพัฒนาระบบเป็นไปอย่างมีประสิทธิภาพ มีคุณภาพ ตอบสนองความต้องการของผู้ใช้และสอดคล้องกับเป้าหมายขององค์กร ช่วยควบคุมความเสี่ยงและเพิ่มโอกาสความสำเร็จในการพัฒนาระบบได้ (โอภาส เอี่ยมสิริวงศ์, 2566; พรรณี สวนเพลง, 2566; ปานใจ ธารทัศนวงศ์, 2565; สุวรรณี อัศวกุลชัย, 2561; Roch, Fowler, Smith, & Bourgeois, 2022; Dennis, Wixom, & Roth, 2021) จากองค์ประกอบของระบบสารสนเทศในหัวข้อ

ที่ผ่านมาจะพบว่า ระบบสารสนเทศมีองค์ประกอบที่เกี่ยวข้องหลายองค์ประกอบ ดังนั้นการพัฒนาระบบจึงไม่ได้หมายถึงการเขียนโปรแกรมเพียงอย่างเดียว แต่เป็นการพิจารณาถึงองค์ประกอบต่าง ๆ ของระบบสารสนเทศที่เกี่ยวข้อง

3.1.2 ความสำคัญของวงจรการพัฒนาระบบ

การลงทุนในระบบสารสนเทศก็ไม่ต่างจากการลงทุนประเภทอื่นขององค์กร เป้าหมายของการลงทุนไม่ใช่การได้มาซึ่งเครื่องมือ แต่เป็นการนำระบบสารสนเทศมาใช้เพื่อให้องค์กรสามารถดำเนินงานได้ดีขึ้น สร้างผลกำไรที่มากขึ้น หรือให้บริการแก่ผู้มีส่วนได้ส่วนเสียอย่างมีประสิทธิภาพมากขึ้น วงจรการพัฒนาระบบจึงมีความสำคัญต่อการพัฒนาระบบสารสนเทศ ดังนี้ (Amazon Web Services, n.d.; Dennis, Wixom, & Roth, 2021)

1) การจัดการความซับซ้อน

การพัฒนาระบบสารสนเทศสมัยใหม่มีความซับซ้อนสูง เนื่องจากต้องพิจารณาปัจจัยหลายด้าน เช่น ความต้องการของผู้ใช้ เทคโนโลยีที่เหมาะสม การรักษาความปลอดภัย และการบูรณาการกับระบบเดิม วงจรการพัฒนาระบบช่วยแบ่งความซับซ้อนออกเป็นขั้นตอนย่อยที่จัดการได้ง่ายขึ้น

2) การควบคุมคุณภาพ

แต่ละขั้นตอนในวงจรการพัฒนาระบบมีจุดตรวจสอบ (Checkpoint) และเกณฑ์การประเมินที่ชัดเจน ช่วยให้สามารถควบคุมคุณภาพของระบบตั้งแต่เริ่มต้นจนเสร็จสิ้น ลดโอกาสเกิดข้อผิดพลาดและต้นทุนที่ต้องใช้ในการแก้ไข

3) การจัดการทรัพยากร

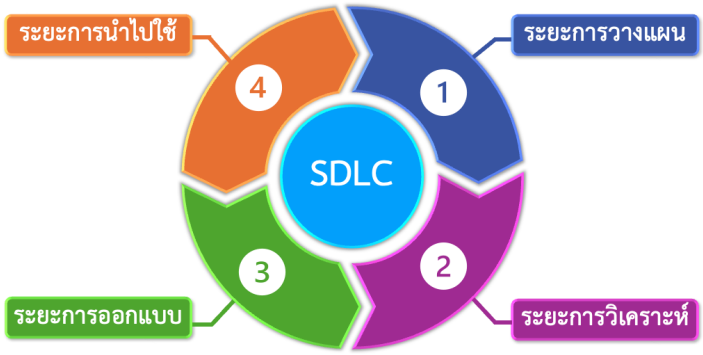
วงจรการพัฒนาระบบช่วยให้สามารถวางแผนการใช้ทรัพยากร ทั้งบุคลากร งบประมาณ และเวลา ได้อย่างมีประสิทธิภาพ โดยสามารถประมาณการได้ว่าแต่ละขั้นตอนจะใช้ทรัพยากรเท่าใด

4) การสื่อสารและการประสานงาน

วงจรการพัฒนาระบบให้กรอบในการสื่อสารระหว่างผู้มีส่วนเกี่ยวข้องกับระบบ เช่น ผู้ใช้งาน นักพัฒนา ผู้บริหาร และลูกค้า ทำให้ทุกฝ่ายเข้าใจบทบาทหน้าที่และความคาดหวังของตนเอง

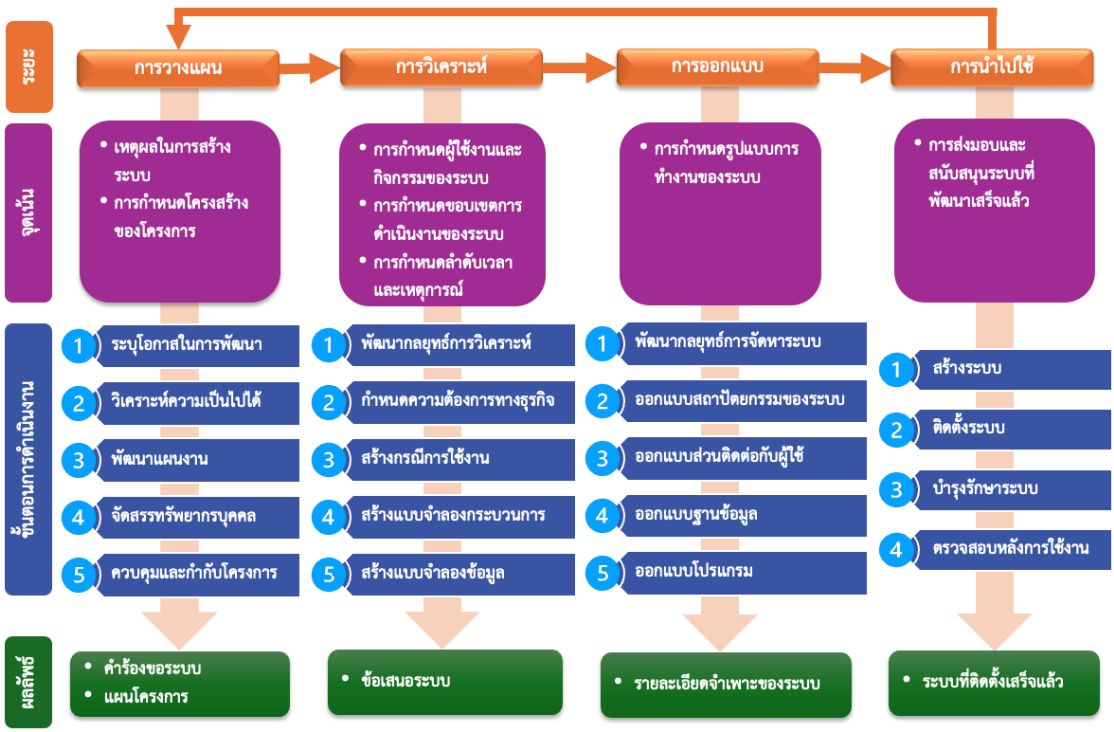
3.2 ระยะและขั้นตอนการดำเนินงานของวงจรการพัฒนาระบบ

วงจรการพัฒนาระบบแบ่งระยะการดำเนินการได้เป็น 4 ระยะ (Phase) ประกอบด้วยระยะการวางแผน ระยะการวิเคราะห์ ระยะการออกแบบ และระยะการนำไปใช้ (Dennis, Wixom, & Roth, 2021) แสดงดังภาพที่ 2.7 วงจรการพัฒนาระบบมีลักษณะคล้ายกับวงจรชีวิตของสิ่งมีชีวิต (Life Cycle) กล่าวคือ ระบบสารสนเทศมีจุดเริ่มต้นของกระบวนการจนกระทั่งได้ระบบตามที่ต้องการ เมื่อระบบถูกนำไปใช้งานในช่วงระยะเวลาหนึ่ง ก็อาจประสบกับปัญหาด้านประสิทธิภาพ ความล้าสมัย หรือไม่ตอบสนองต่อความต้องการใหม่ ๆ จึงต้องมีการปรับปรุงหรือพัฒนาระบบใหม่ขึ้นมาทดแทน เกิดเป็นวงจรซ้ำอย่างต่อเนื่อง (โอภาส เอี่ยมสิริวงศ์, 2566)



ภาพที่ 2.7 ระยะต่าง ๆ ของวงจรการพัฒนาระบบ

ในแต่ละระยะของวงจรการพัฒนาระบบจะมีจุดเน้น ชุดของขั้นตอนดำเนินงานและผลลัพธ์ที่ได้หลังจากที่ดำเนินการในระยะนั้นเสร็จสิ้น ขั้นตอนการพัฒนาระบบตามแบบแผนของวงจรการพัฒนาระบบจัดเป็นวิธีการพัฒนาระบบแบบดั้งเดิม มีเป้าหมาย กรอบการทำงานที่มีโครงสร้างและลำดับที่ชัดเจน (โอภาส เอี่ยมสิริวงศ์, 2566) รายละเอียดการดำเนินการในแต่ละระยะของการวงจรการพัฒนาระบบแสดงดังภาพที่ 2.8



ภาพที่ 2.8 รายละเอียดการดำเนินการในแต่ละระยะของการวงจรการพัฒนาระบบ

ที่มา: ปรับปรุงจาก Dennis, Wixom และ Roth (2021)

จากภาพที่ 2.8 เป็นการแสดงให้เห็นว่าระยะต่าง ๆ ของวงจรการพัฒนาระบบมีลำดับการดำเนินงานที่เป็นระบบ เริ่มจากขั้นตอนแรกของระยะการวางแผนไปจนถึงขั้นตอนสุดท้ายของระยะการนำไปใช้ ในโครงการพัฒนาระบบสารสนเทศบางระบบอาจใช้รูปแบบการดำเนินการแบบลำดับ (Consecutive) หรืออาจเลือกใช้รูปแบบอื่นๆ เช่น การพัฒนาระบบแบบเพิ่มทีละส่วน (Incremental) หรือการพัฒนาระบบแบบวนซ้ำ (Iterative) เป็นต้น ทั้งนี้ ขึ้นอยู่กับลักษณะของโครงการและเป้าหมายการพัฒนา แม้ว่าแต่ละโครงการจะมีวิธีดำเนินงานในแต่ละระยะต่างกัน แต่โดยทั่วไปแล้ว ทุกโครงการล้วนมีองค์ประกอบของทั้ง 4 ระยะของวงจรการพัฒนาระบบอยู่เสมอ โดยระยะของวงจรการพัฒนาระบบ มีดังนี้ (Dennis, Wixom, & Roth, 2021)

3.2.1 ระยะวางแผน

ระยะวางแผน (Planning Phase) เป็นกระบวนการเบื้องต้นที่มีความสำคัญต่อการพัฒนาระบบสารสนเทศ โดยมุ่งเน้นไปที่การตอบคำถามหลักสองประการ คือ เหตุผลในการสร้างระบบนี้คืออะไร (Why) และโครงการจะถูกดำเนินการอย่างไร จากภาพที่ 2.8 ระยะวางแผนมีขั้นตอนการดำเนินงานประกอบด้วย การระบุโอกาสในการพัฒนา การวิเคราะห์ความเป็นไปได้ การพัฒนาแผนงาน การจัดสรรทรัพยากรบุคคล และการควบคุมกำกับโครงการ การดำเนินการในระยะนี้สามารถแบ่งออกเป็น 2 ส่วน ได้แก่ ส่วนการเริ่มต้นโครงการ และ ส่วนการบริหารจัดการโครงการ

1) ส่วนการเริ่มต้นโครงการ

ส่วนการเริ่มต้นโครงการ (Project Initiation) เป็นกลุ่มของขั้นตอนที่ ระบบจะได้รับการพิจารณาจากมุมมองของคุณค่าทางธุรกิจที่ระบบจะส่งมอบให้แก่องค์กรหากมีการนำระบบมาใช้ เช่น การลดต้นทุนหรือการเพิ่มรายได้ เป็นต้น โดยทั่วไป ความคิดริเริ่มในการพัฒนาระบบมักมาจากหน่วยงานธุรกิจภายในองค์กรซึ่งจะยื่นคำร้องขอระบบ (System Request) ที่สรุปปัญหาหรือความต้องการทางธุรกิจ พร้อมระบุว่าระบบสารสนเทศสามารถช่วยตอบสนองความต้องการดังกล่าวได้อย่างไร หลังจากยื่นคำร้องขอระบบจะเข้าสู่กระบวนการวิเคราะห์ความเป็นไปได้ (Feasibility Analysis) โดยทีมระบบสารสนเทศขององค์กรจะทำงานร่วมกับผู้เสนอคำร้องขอระบบซึ่งเรียกผู้เสนอคำร้องขอระบบนี้ว่า ผู้สนับสนุนโครงการ (Project Sponsor) เพื่อประเมินประเด็นหลัก 3 ด้าน ได้แก่

- **ความเป็นไปได้ทางเทคนิค**

ความเป็นไปได้ทางเทคนิค (Technical Feasibility) เป็นการประเมินว่าองค์กรมีความสามารถเชิงเทคนิคในการพัฒนาระบบหรือไม่

- **ความเป็นไปได้ทางเศรษฐศาสตร์**

ความเป็นไปได้ทางเศรษฐศาสตร์ (Economic Feasibility) เป็นการประเมินว่าระบบจะสร้างมูลค่าทางธุรกิจได้เพียงใด

- **ความเป็นไปได้เชิงองค์กร**

ความเป็นไปได้เชิงองค์กร (Organizational Feasibility) เป็นการประเมินว่าหากพัฒนาระบบแล้วจะมีการนำไปใช้งานจริงหรือไม่

ผลลัพธ์ของส่วนเริ่มต้นโครงการนี้ ได้แก่ คำร้องขอระบบ และ รายงานการวิเคราะห์ความเป็นไปได้ ซึ่งจะถูกนำเสนอต่อคณะกรรมการพิจารณาระบบสารสนเทศ (Information Systems Approval Committee หรือ Steering Committee) เพื่อพิจารณาว่าควรดำเนินโครงการหรือไม่

2) ส่วนการบริหารจัดการโครงการ (Project Management)

ส่วนการบริหารจัดการโครงการ (Project Management) เป็นกลุ่มของขั้นตอนที่จะดำเนินการเมื่อโครงการพัฒนาระบบสารสนเทศได้รับการอนุมัติ ระบบจะเข้าสู่กระบวนการวางแผนและบริหารโครงการอย่างเป็นทางการ ผู้จัดการโครงการจะทำหน้าที่กำหนดแผนการทำงาน (Work Plan) จัดสรรทรัพยากรบุคคล และเลือกใช้เทคนิคการจัดการที่เหมาะสม เช่น การควบคุมความเสี่ยง การติดตามความก้าวหน้า และการสื่อสารภายในทีม ผลลัพธ์ของส่วนการบริหารจัดการโครงการคือแผนโครงการ (Project Plan) ซึ่งแสดงรายละเอียดเกี่ยวกับการดำเนินงานของทีมพัฒนาในทุกระยะของวงจรการพัฒนาระบบ

3.2.2 ระยะการวิเคราะห์

ระยะการวิเคราะห์ (Analysis Phase) เป็นระยะที่สองของวงจรการพัฒนาระบบที่มีเป้าหมายเพื่อกำหนดรายละเอียดของระบบที่จะพัฒนา มีจุดเน้นที่การศึกษากระบวนการทำงานของระบบเดิมและการรวบรวมข้อมูลเพื่อพัฒนาแนวคิดของระบบใหม่ที่สามารถปรับปรุงประสิทธิภาพหรือแก้ไขข้อจำกัดที่มีอยู่ในระบบเดิม

ระยะการวิเคราะห์นี้เน้นการตอบคำถามหลักที่ว่า ใครเป็นผู้ใช้งานระบบ (Who) ระบบควรทำงานอะไร (What) ใช้งานที่ใด (Where) และในช่วงเวลาใด (When) เพื่อให้สามารถออกแบบระบบได้อย่างสอดคล้องกับความต้องการทางธุรกิจ ผลลัพธ์ของระยะการวิเคราะห์จะทำให้ได้ข้อเสนอระบบ (System Proposal) ซึ่งรวบรวมข้อมูลจากการวิเคราะห์ระบบเดิม แนวคิดของระบบใหม่ ความต้องการของผู้ใช้ และแบบจำลองต่าง ๆ เข้าไว้ด้วยกัน ข้อเสนอระบบนี้จะถูกนำเสนอแก่ผู้บริหารหรือคณะกรรมการพิจารณาโครงการเพื่อประกอบการตัดสินใจว่าจะดำเนินโครงการต่อในระยะการออกแบบหรือไม่

จากภาพที่ 2.8 ระยะการวิเคราะห์มีขั้นตอนการดำเนินงานประกอบด้วย การพัฒนากลยุทธ์การวิเคราะห์ การกำหนดความต้องการทางธุรกิจ การสร้างกรณีการใช้งาน การสร้างแบบจำลองกระบวนการและการสร้างแบบจำลองข้อมูล ขั้นตอนการดำเนินงานเหล่านี้สามารถจัดกลุ่มออกเป็น 3 ส่วน ดังนี้

1) ส่วนการกำหนดกลยุทธ์การวิเคราะห์

ในส่วนนี้ของการดำเนินงานเป็นขั้นตอนของการพัฒนากลยุทธ์การวิเคราะห์ ทีมพัฒนาจะเริ่มจากการศึกษาระบบเดิมที่ใช้อยู่ในปัจจุบัน (As-Is System) และวิเคราะห์จุดอ่อนหรือปัญหาที่เกิดขึ้นในระบบที่ใช้อยู่นั้น จากนั้นจึงออกแบบแนวคิดของระบบใหม่ในเชิงแนวทาง (To-Be System) เพื่อเปรียบเทียบและมองเห็นช่องทางในการปรับปรุง การกำหนดกลยุทธ์การวิเคราะห์สามารถพิจารณาจากแนวทางต่าง ๆ ได้แก่

- **การเพิ่มประสิทธิภาพโดยใช้ระบบอัตโนมัติ**

การเพิ่มประสิทธิภาพโดยใช้ระบบอัตโนมัติ (Business Process Automation: BPA) เป็นการใช้เทคโนโลยีเข้าไปแทนที่การทำงานที่เป็นขั้นตอนซ้ำ ๆ หรือการทำงานด้วยมือ โดยมีเป้าหมายเพื่อลดภาระงานของบุคลากร เพิ่มความเร็ว และลดข้อผิดพลาด

- **การปรับกระบวนการให้ดีขึ้น**

การปรับกระบวนการให้ดีขึ้น (Business Process Improvement: BPI) เป็นการวิเคราะห์และปรับโครงสร้างของกระบวนการ เพื่อให้มีประสิทธิภาพมากขึ้น โดยไม่เพียงแค่ทำให้เป็นอัตโนมัติเท่านั้น แต่ยังอาจปรับขั้นตอนให้สั้นลง ลดค่าใช้จ่าย หรือลดระยะเวลาในการทำงาน

- **การสร้างเป็นระบบใหม่ทั้งหมด**

การสร้างเป็นระบบใหม่ทั้งหมด (Business Process Reengineering: BPR) เป็นการเปลี่ยนแปลงกระบวนการทางธุรกิจด้วยการออกแบบกระบวนการใหม่ทั้งหมดโดยไม่ยึดติดกับขั้นตอนเดิม เพื่อทำให้เกิดการเปลี่ยนแปลงเชิงกลยุทธ์ เช่น เปลี่ยนวิธีให้บริการลูกค้าทั้งหมดจากออฟไลน์ไปเป็นออนไลน์เต็มรูปแบบ

2) ส่วนการกำหนดความต้องการทางธุรกิจ

ขั้นตอนส่วนนี้ ทีมพัฒนาระบบจะจัดเก็บข้อมูลจากผู้ใช้งานและผู้เกี่ยวข้องโดยใช้เทคนิคที่หลากหลายเป็นเครื่องมือในการเก็บข้อมูล เช่น การสัมภาษณ์ การจัดประชุมกลุ่ม หรือการใช้แบบสอบถาม เป็นต้น เพื่อนำมาวิเคราะห์และแปลงเป็นความต้องการเชิงระบบ (System Requirements) มีขั้นตอนการดำเนินงานประกอบด้วยการกำหนดความต้องการทางธุรกิจ (Business Requirement) และการสร้างกรณีการใช้งานระบบ (Use Case)

3) ส่วนการจัดทำแบบจำลองระบบ

ขั้นตอนส่วนนี้เป็นขั้นตอนของการสร้างแบบจำลองข้อมูล ซึ่งเป็นการนำข้อมูลที่ได้จากขั้นตอนก่อนหน้ามาจัดทำเป็นแบบจำลองเชิงธุรกิจ เช่น แผนภาพการไหลของข้อมูล (Data Flow Diagram: DFD) หรือ แบบจำลองความสัมพันธ์ของข้อมูล (Entity Relationship Model: ER Model) ซึ่งช่วยให้เห็นภาพชัดเจนว่ากระบวนการในระบบใหม่ควรดำเนินการอย่างไร

3.2.3 ระยะการออกแบบ

ระยะการออกแบบ (Design Phase) เป็นระยะที่สามของวงจรการพัฒนาระบบ โดยมีเป้าหมายสำคัญในการแปลงความต้องการเชิงระบบ (System Requirements) ที่ได้จากระยะการวิเคราะห์ให้เป็นแบบพิมพ์เขียวของระบบ (System Blueprint) ที่สามารถนำไปใช้ในการพัฒนาและติดตั้งระบบจริงได้อย่างเป็นรูปธรรม ระยะนี้มุ่งเน้นการกำหนดรูปแบบการทำงานของระบบโดยตอบคำถามหลักที่ว่า ระบบจะทำงานอย่างไร (How) ควรใช้เทคโนโลยีใด (What Technology) และโครงสร้างการทำงานของระบบควรเป็นอย่างไร (System Architecture and Workflow) เพื่อให้ทีมพัฒนาสามารถดำเนินการสร้างระบบได้อย่างถูกต้อง มีประสิทธิภาพ และสอดคล้องกับความต้องการทางธุรกิจ

ผลลัพธ์ของระยะการออกแบบจะได้เอกสารรายละเอียดจำเพาะของระบบ (System Specification) ซึ่งถือเป็นแบบพิมพ์เขียวของระบบที่ครอบคลุมทั้ง โครงสร้างระบบ สถาปัตยกรรมเทคโนโลยี ส่วนติดต่อผู้ใช้ ฐานข้อมูลและไฟล์ข้อมูล รวมถึงการออกแบบโปรแกรม โดยเอกสารฉบับนี้จะทำหน้าที่เป็นคู่มือหลักสำหรับทีมพัฒนาในการสร้างและติดตั้งระบบจริง นอกจากนี้ยังใช้เป็นเครื่องมือตรวจสอบเพื่อทบทวนและวิเคราะห์ความเป็นไปได้ของโครงการ ก่อนที่จะเข้าสู่ระยะการนำไปใช้เต็มรูปแบบ

จากภาพที่ 2.8 ระยะการออกแบบมีขั้นตอนการดำเนินงาน 5 ขั้นตอน ประกอบด้วยการพัฒนากลยุทธ์การจัดหาระบบ การออกแบบสถาปัตยกรรมของระบบ การออกแบบส่วนปฏิสัมพันธ์กับผู้ใช้ การออกแบบฐานข้อมูลและการออกแบบโปรแกรม โดยสามารถจัดกลุ่มการดำเนินงานออกเป็น 4 ส่วน ดังนี้

1) ส่วนการกำหนดกลยุทธ์การออกแบบ

ส่วนแรกของระยะการออกแบบเป็นขั้นตอนของการกำหนดกลยุทธ์การจัดหาระบบ โดยเป็นการตัดสินใจเลือกแนวทางในการพัฒนาด้านเทคนิคของระบบ ซึ่งเป็นการกำหนดทิศทางที่จะส่งผลต่อการออกแบบในขั้นตอนถัดไป กลยุทธ์การจัดหาระบบมีทางเลือกอยู่ 3 แนวทาง ดังนี้

- **การพัฒนาระบบโดยทีมงานภายใน**

การพัฒนาระบบโดยทีมงานภายใน (In-house Development) เป็นการใช้นักพัฒนาขององค์กรเองในการสร้างระบบตั้งแต่เริ่มต้น แนวทางนี้ให้การควบคุมที่สูงและความเข้าใจธุรกิจที่ลึกซึ้ง แต่ต้องใช้เวลานานและต้องการทักษะเฉพาะจากทีมงานพัฒนาระบบ

- **การจ้างพัฒนาจากภายนอก**

การจ้างพัฒนาจากภายนอก (Outsourcing Development) เป็นการว่าจ้างองค์กรภายนอกให้พัฒนาระบบตามข้อกำหนดที่กำหนด แนวทางนี้ช่วยลดภาระงานภายในและได้ความเชี่ยวชาญเฉพาะด้าน แต่ต้องจัดการด้านการสื่อสารและการควบคุมคุณภาพระหว่างทีมงานภายในองค์กรและทีมงานผู้ถูกว่าจ้าง

- **การจัดหาซอฟต์แวร์สำเร็จรูป**

การจัดหาซอฟต์แวร์สำเร็จรูป (Package Software) เป็นการเลือกใช้ระบบที่มีการพัฒนาไว้แล้วในตลาด โดยอาจมีการปรับแต่งบางส่วนให้เข้ากับความต้องการ แนวทางนี้ประหยัดเวลาและต้นทุน แต่อาจมีข้อจำกัดในการปรับแต่งให้เข้ากับความต้องการขององค์กร

2) ส่วนการออกแบบสถาปัตยกรรมและส่วนปฏิสัมพันธ์กับผู้ใช้

ส่วนการดำเนินการนี้เกี่ยวข้องการกำหนดโครงสร้างพื้นฐานของระบบและการออกแบบส่วนที่ผู้ใช้จะมีปฏิสัมพันธ์ด้วย ประกอบด้วย 2 ขั้นตอน คือ การออกแบบสถาปัตยกรรมระบบและการออกแบบส่วนติดต่อผู้ใช้

- **การออกแบบสถาปัตยกรรมระบบ**

การออกแบบสถาปัตยกรรมระบบ (Architecture Design) เป็นการกำหนดโครงสร้างโดยรวมของระบบ รวมถึงการเลือกใช้ฮาร์ดแวร์ ซอฟต์แวร์ และโครงสร้างพื้นฐานด้านเครือข่าย โดยส่วนใหญ่แล้วระบบใหม่จะเข้ามาเสริมหรือเปลี่ยนแปลงโครงสร้างพื้นฐานที่มีอยู่แล้วในองค์กร

- **การออกแบบส่วนติดต่อผู้ใช้**

การออกแบบส่วนติดต่อผู้ใช้ (User Interface Design) เป็นการกำหนดวิธีการที่ผู้ใช้จะมีปฏิสัมพันธ์กับระบบ รวมถึงการนำทาง รูปแบบของฟอร์ม และรายงานที่ระบบจะใช้ องค์ประกอบของการออกแบบส่วนติดต่อผู้ใช้ เช่น การนำทาง (Navigation) แบบฟอร์มการป้อนข้อมูล (Input Forms) และการแสดงผลข้อมูล (Output Display) ในลักษณะต่าง ๆ

3) ส่วนการออกแบบฐานข้อมูลและไฟล์

ขั้นตอนนี้เป็นการกำหนดรายละเอียดของข้อมูลที่จะจัดเก็บในระบบ เช่น ชนิดของฐานข้อมูล โครงสร้างฐานข้อมูล ชนิดของข้อมูล ความสัมพันธ์ระหว่างข้อมูลและตำแหน่งในการจัดเก็บข้อมูล ซึ่งอาจรวมถึงการกำหนดนโยบายด้านความปลอดภัย การเข้าถึงข้อมูล การสำรองข้อมูล และการจัดการประสิทธิภาพ เพื่อให้มั่นใจว่าข้อมูลในระบบมีความถูกต้อง ปลอดภัยและสอดคล้องกับกรอบคุณภาพของข้อมูลและสารสนเทศ

- **การออกแบบฐานข้อมูล**

การออกแบบฐานข้อมูล (Database Design) เป็นการแปลงแบบจำลองข้อมูลที่ได้จากระยะการวิเคราะห์ให้เป็นโครงสร้างฐานข้อมูลจริง และเป็นส่วนของการพิจารณาและเลือกชนิดของฐานข้อมูลที่เหมาะสมกับลักษณะข้อมูลและความต้องการใช้งาน เช่น เช่น อาจเลือกใช้ฐานข้อมูลเชิงสัมพันธ์ (Relational Database) สำหรับระบบงานธุรกิจทั่วไป หรือเลือกใช้ฐานข้อมูลแบบเอกสาร (Document Database) สำหรับเว็บแอปพลิเคชันที่จัดเก็บข้อมูลไม่เป็นโครงสร้างโดยการเลือกฐานข้อมูลจะส่งผลต่อทั้งประสิทธิภาพ ความยืดหยุ่นและการบำรุงรักษาของระบบในระยะยาว เป็นต้น

- **การออกแบบไฟล์**

การออกแบบไฟล์ (File Specification) เป็นการกำหนดไฟล์ประเภทต่าง ๆ ที่ระบบต้องการ เช่น ไฟล์รูปภาพ เอกสาร และไฟล์กำหนดค่าต่าง ๆ รวมทั้งการกำหนดนโยบายด้านความปลอดภัยในการเข้าถึงไฟล์

4) ส่วนการออกแบบโปรแกรม

ส่วนการออกแบบโปรแกรมเป็นขั้นตอนสุดท้ายของระยะการออกแบบมีจุดมุ่งหมายในการกำหนดรายละเอียดของโปรแกรมที่จะพัฒนาขึ้น โดยเป็นการจัดทำรายละเอียดในสองมิติ คือ ด้านโครงสร้างของโปรแกรมและด้านฟังก์ชันการทำงานของแต่ละโปรแกรมเพื่อให้สามารถนำผลลัพธ์จากการออกแบบโปรแกรมไปใช้ในการพัฒนาได้อย่างเป็นระบบและตรงกับความต้องการ

- **การออกแบบโครงสร้างโปรแกรม**

การออกแบบโครงสร้างโปรแกรม (Program Structure Design) เป็นการกำหนดว่าจะมีโปรแกรมหรือโมดูลใดบ้าง และแต่ละส่วนจะทำหน้าที่อะไรโดยใช้แผนผังโครงสร้างโปรแกรม (Program Structure Chart) เป็นเครื่องมือในการออกแบบ

- **การกำหนดฟังก์ชันของแต่ละโปรแกรม**

การกำหนดฟังก์ชันของแต่ละโปรแกรม (Program Specification) เป็นการระบุรายละเอียดว่าแต่ละโปรแกรมจะรับข้อมูลอะไรเข้า ประมวลผลอย่างไร และส่งผลลัพธ์อะไรออกมาจากโปรแกรม

3.2.4 ระยะการนำไปใช้

ระยะการนำไปใช้ (Implementation Phase) เป็นระยะสุดท้ายของวงจรการพัฒนาระบบ มีเป้าหมายหลักในการแปลงแบบพิมพ์เขียวของระบบที่ได้จากระยะการออกแบบให้เป็นระบบที่ทำงานได้จริงและพร้อมใช้งาน ระยะนี้ถือเป็นการสร้างรูปธรรมของระบบสารสนเทศที่ได้วางแผนและออกแบบไว้ โดยมุ่งเน้นการตอบคำถามสำคัญว่า ระบบจะถูกสร้างขึ้นอย่างไร (How to Build) จะทำการติดตั้งและเปลี่ยนผ่านจากระบบเก่าไปยังระบบใหม่อย่างไร (How to Deploy) และจะจัดการดูแลรักษาระบบอย่างไรให้ทำงานได้อย่างต่อเนื่องและมีประสิทธิภาพ (How to Maintain)

ระยะการนำไปใช้มักจะเป็นระยะที่ใช้เวลานานที่สุดและมีค่าใช้จ่ายสูงที่สุดในกระบวนการพัฒนาระบบ เนื่องจากเป็นการดำเนินงานที่เกี่ยวข้องกับการเขียนโปรแกรม การทดสอบอย่างละเอียด การฝึกอบรมผู้ใช้ และการจัดการการเปลี่ยนแปลงขององค์กร ผลลัพธ์สุดท้ายของระยะนี้คือระบบที่ติดตั้งเสร็จสมบูรณ์และพร้อมให้บริการผู้ใช้งานตามวัตถุประสงค์ที่กำหนดไว้

จากภาพที่ 2.8 ระยะการนำไปใช้ประกอบด้วยขั้นตอนการดำเนินงาน 4 ขั้นตอน ได้แก่ การสร้างระบบ การติดตั้งระบบ การดูแลรักษาระบบ และการประเมินผลหลังการนำไปใช้ โดยสามารถจัดกลุ่มการดำเนินงานออกเป็น 3 ส่วน ดังนี้

1) ส่วนการสร้างระบบ

ส่วนแรกของระยะการนำไปใช้เป็นขั้นตอนของการสร้างระบบตามแบบพิมพ์เขียวที่ได้จากระยะการออกแบบ โดยเป็นกระบวนการแปลงเอกสารรายละเอียดจำเพาะของระบบให้เป็นระบบงานที่สามารถใช้งานได้จริง เอกสารผลลัพธ์ที่ได้จากการดำเนินการขั้นตอนส่วนนี้ ได้แก่ แผนการทดสอบโปรแกรม (Test Plan) โปรแกรมที่พัฒนาเสร็จสมบูรณ์ และ เอกสารประกอบโปรแกรม ส่วนการสร้างระบบนี้ประกอบด้วยกิจกรรมสำคัญ 3 กิจกรรม ดังนี้

- **การเขียนโปรแกรม**

การเขียนโปรแกรม (Programming) เป็นกระบวนการแปลงการออกแบบโปรแกรมให้เป็นรหัสโปรแกรมที่สามารถทำงานได้จริง โดยทีมเขียนโปรแกรมจะใช้ภาษาโปรแกรม ซอฟต์แวร์และฮาร์ดแวร์ต่าง ๆ ตามที่กำหนดไว้ในสถาปัตยกรรมระบบ กิจกรรมนี้รวมถึงการสร้างโมดูลต่าง ๆ ของระบบ การเชื่อมต่อฐานข้อมูล การพัฒนาส่วนติดต่อผู้ใช้ และการผนวกรวมโมดูลต่าง ๆ เข้าด้วยกันให้ทำงานเป็นระบบเดียวกัน

- **การทดสอบซอฟต์แวร์**

การทดสอบซอฟต์แวร์ (Software Testing) เป็นกระบวนการตรวจสอบความถูกต้องและครบถ้วนของระบบ โดยมีวัตถุประสงค์เพื่อค้นหาและแก้ไขข้อผิดพลาดก่อนที่ระบบจะถูกนำไปใช้งานจริง การทดสอบมีหลายระดับ ตั้งแต่การทดสอบหน่วยย่อย (Unit Testing) การทดสอบการผนวกรวม (Integration

Testing) การทดสอบระบบ (System Testing) และการทดสอบการยอมรับ (User Acceptance Testing) เนื่องจากต้นทุนในการแก้ไขข้อผิดพลาดหลังจากระบบเริ่มใช้งานแล้วจะสูงมาก องค์กรส่วนใหญ่จึงให้ความสำคัญกับการทดสอบไม่น้อยไปกว่าการเขียนโปรแกรม

- **การทดสอบประสิทธิภาพ**

การทดสอบประสิทธิภาพ (Performance Testing) เป็นการประเมินความสามารถของระบบในการรองรับปริมาณงานและจำนวนผู้ใช้ตามที่คาดการณ์ไว้ รวมถึงการทดสอบความเร็วในการตอบสนอง ความเสถียรของระบบภายใต้ภาระงานสูง และความสามารถในการขยายตัวของระบบ การทดสอบนี้จะช่วยให้มั่นใจว่าระบบสามารถรองรับการใช้งานจริงได้อย่างมีประสิทธิภาพ

2) ส่วนการติดตั้งและเปลี่ยนผ่านระบบ

ส่วนที่สองเป็นขั้นตอนการนำระบบที่สร้างเสร็จแล้วไปติดตั้งในสภาพแวดล้อมการใช้งานจริง และจัดการการเปลี่ยนผ่านจากระบบเก่าไปยังระบบใหม่ ขั้นตอนนี้ถือเป็นจุดสำคัญที่ต้องการการวางแผนอย่างละเอียดเพื่อลดความเสี่ยงและผลกระทบต่อการดำเนินธุรกิจ เอกสารสำคัญที่ได้จากส่วนการดำเนินงานนี้ประกอบด้วย แผนการย้ายระบบ (Migration Plan) แผนฝึกอบรม และคู่มือการใช้งานระบบ ส่วนการติดตั้งและเปลี่ยนผ่านระบบประกอบด้วยกิจกรรมหลัก 3 กิจกรรม ดังนี้

- **การเลือกกลยุทธ์การเปลี่ยนผ่าน**

การเลือกกลยุทธ์การเปลี่ยนผ่าน (Conversion Strategy Selection) เป็นการกำหนดแนวทางในการเปลี่ยนจากระบบเก่าไปยังระบบใหม่ ตัวอย่างของกลยุทธ์การเปลี่ยนผ่านมีหลายกลยุทธ์ เช่น การเปลี่ยนผ่านแบบขนาน (Parallel Conversion) เป็นการเปลี่ยนผ่านที่ใช้ระบบเก่าและใหม่พร้อมกันเป็นระยะเวลาหนึ่ง การเปลี่ยนผ่านแบบทันทีทันใด (Direct Conversion) เป็นการเปลี่ยนผ่านที่หยุดระบบเก่าและเริ่มใช้ระบบใหม่ทันที เป็นต้น

- **การจัดทำแผนฉุกเฉิน**

การจัดทำแผนฉุกเฉิน (Business Contingency Plan) เป็นกระบวนการในการจัดทำแผนรองรับหรือแนวปฏิบัติในกรณีระบบใหม่เกิดปัญหา เพื่อไม่ให้ความผิดพลาดทางเทคโนโลยีส่งผลกระทบเชิงลบต่อธุรกิจ การจัดทำแผนฉุกเฉินมีเป้าหมายเพื่อรักษาความต่อเนื่องของงานและลดความเสียหายที่อาจเกิดขึ้น ไม่เพียงแต่ช่วยลดผลกระทบทางเทคนิคเท่านั้น แต่ยังช่วยสร้างความมั่นใจให้กับผู้บริหารและผู้ใช้งานว่าการนำระบบใหม่มาใช้งานจะไม่ก่อให้เกิดความเสียหายร้ายแรงต่อธุรกิจหลักขององค์กร

- **การฝึกอบรม**

การฝึกอบรม (Training) เป็นกระบวนการถ่ายทอดความรู้และทักษะการใช้งานที่จำเป็นต้องใช้ในระบบใหม่ให้กับผู้ใช้ โดยครอบคลุมทั้งการอบรมเชิงเทคนิคสำหรับผู้ดูแลระบบและการอบรมการใช้งานสำหรับผู้ใช้ทั่วไป การฝึกอบรมที่มีประสิทธิภาพจะช่วยลดการต่อต้านต่อการเปลี่ยนแปลงและเพิ่มอัตราการยอมรับระบบใหม่ ซึ่งเป็นหัวใจสำคัญของการจัดการการเปลี่ยนแปลง (Change Management)

3) ส่วนการดูแลรักษาและประเมินผล

ส่วนสุดท้ายของระยะการนำไปใช้เป็นขั้นตอนของการบำรุงรักษาระบบและการตรวจสอบหลังการใช้งาน เป็นการจัดการระบบหลังจากเริ่มใช้งานแล้วเพื่อให้มั่นใจว่าระบบจะทำงานได้อย่างต่อเนื่องและตอบสนองความต้องการที่เปลี่ยนแปลงไป ประกอบด้วยกิจกรรมสำคัญ 4 กิจกรรม ดังนี้

- **การเลือกรูปแบบการสนับสนุน**

การเลือกรูปแบบการสนับสนุน (Support Selection) เป็นการกำหนดวิธีการและช่องทางการให้ความช่วยเหลือแก่ผู้ใช้ระหว่างการใช้งานระบบ ซึ่งอาจเป็นการจัดตั้งศูนย์ช่วยเหลือ (Help Desk) การจัดทำคู่มือผู้ใช้ การมอบหมายผู้เชี่ยวชาญประจำหน่วยงาน หรือการจัดหาบริการสนับสนุนจากผู้ให้บริการภายนอก

- **การดูแลรักษาระบบ**

การดูแลรักษาระบบ (System Maintenance) เป็นกิจกรรมต่อเนื่องที่มีวัตถุประสงค์เพื่อรักษาระบบให้ทำงานได้อย่างปกติและปรับปรุงประสิทธิภาพ รวมถึงการแก้ไขข้อผิดพลาด การปรับปรุงฟังก์ชันการทำงาน การเพิ่มความสามารถใหม่ และการปรับแต่งระบบให้เหมาะสมกับความต้องการที่เปลี่ยนแปลง

- **การประเมินโครงการ**

การประเมินโครงการ (Project Assessment) เป็นการทบทวนผลสำเร็จของโครงการพัฒนาระบบโดยเปรียบเทียบกับเป้าหมายที่กำหนดไว้ตั้งแต่เริ่มต้น รวมถึงการประเมินด้านเวลา งบประมาณ คุณภาพ และความพึงพอใจของผู้ใช้

- **การตรวจสอบหลังการนำไปใช้**

การตรวจสอบหลังการนำไปใช้ (Post-implementation Audit) เป็นการประเมินผลอย่างเป็นทางการหลังจากระบบทำงานได้อย่างมีเสถียรภาพ โดยมีจุดมุ่งหมายเพื่อระบุบทเรียนที่ได้รับ ประเมินประโยชน์ที่เกิดขึ้นจริง และค้นหาโอกาสในการปรับปรุงระบบหรือกระบวนการพัฒนาในอนาคต ผลลัพธ์จากการตรวจสอบนี้จะถูกจัดทำเป็นรายงานการตรวจสอบหลังการนำไปใช้ (Post-implementation Audit Report) เพื่อเป็นข้อมูลสำหรับการพัฒนาโครงการในอนาคต

4. การพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ

วงจรการพัฒนาระบบที่นำเสนอในหัวข้อที่ 3 เป็นกรอบกระบวนการที่ครอบคลุมองค์ประกอบทั้งหมดของระบบสารสนเทศ ได้แก่ ฮาร์ดแวร์ ซอฟต์แวร์ ข้อมูล บุคลากร และกระบวนการทำงาน โดยในโครงการระบบสารสนเทศขนาดใหญ่หรือระบบที่มีความซับซ้อนสูง มักมีการดำเนินงานอย่างเป็นลำดับตามทุกระยะของวงจรการพัฒนาระบบ อย่างไรก็ตาม สำหรับโครงการพัฒนาซอฟต์แวร์ขนาดเล็ก เช่น เว็บแอปพลิเคชันที่มีเป้าหมายเฉพาะด้าน อาจเลือกประยุกต์ใช้เพียงบางระยะที่จำเป็นอย่างระยะการวิเคราะห์ ระยะการออกแบบ และระยะการนำไปใช้ เป็นต้น (โอภาส เอี่ยมสิริวงศ์, 2566)

ทั้งนี้ หากมุ่งเน้นเฉพาะการพัฒนาซอฟต์แวร์ให้มีคุณภาพสูง ใช้งานได้ตรงความต้องการและลดโอกาสที่จะเกิดความล้มเหลวในการพัฒนาซอฟต์แวร์ สามารถนำแนวคิดจากวิศวกรรมซอฟต์แวร์มาประยุกต์ใช้ในรูปแบบของวงจรการพัฒนาซอฟต์แวร์ (Software Development Life Cycle: SDLC) ซึ่งมุ่งจัดการกระบวนการพัฒนาในเชิงเทคนิคอย่างเป็นระบบ ดังมีขั้นตอนแสดงดังภาพที่ 2.9 (ปานใจ ธารทัศนวงศ์, 2565; สุวรรณี อัศวกุลชัย, 2561) ทั้งนี้ แม้ว่าแต่ละขั้นตอนจะถูกนำเสนอแยกจากกันอย่างชัดเจน แต่ในความเป็นจริง การดำเนินงานในแต่ละขั้นตอนนั้นไม่ได้เป็นกระบวนการที่แยกจากกันโดยสิ้นเชิง เพราะหลายกิจกรรมสามารถดำเนินพร้อมกันได้ และบางกิจกรรมอาจจำเป็นต้องทำซ้ำอีกครั้ง (Kendall & Kendall, 2014)



ภาพที่ 2.9 ขั้นตอนการพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ

4.1 ขั้นตอนการกำหนดปัญหาและการศึกษาความเป็นไปได้ระบบ

ขั้นตอนการกำหนดปัญหาและการศึกษาความเป็นได้ของระบบ (Problem Definition and Feasibility Study) ไปได้เป็นขั้นตอนเริ่มต้นที่กำหนดทิศทางของโครงการพัฒนาเว็บแอปพลิเคชัน เป็นขั้นตอนที่อยู่ในระยะการวางแผนตามวงจรการพัฒนาระบบและการกำหนดเป้าหมายและขอบเขตของโครงการ มีจุดเน้นในการระบุและกำหนดปัญหาทางที่ต้องการแก้ไขด้วยเว็บแอปพลิเคชัน การประเมินความเป็นไปได้ของโครงการในมิติต่าง ๆ รวมทั้งการกำหนดเป้าหมายและขอบเขตของโครงการ

เทคนิคที่ใช้ในการรวบรวมปัญหา เช่น การวิเคราะห์เอกสาร การสัมภาษณ์ผู้มีส่วนได้ส่วนเสีย การใช้แบบสอบถาม การประชุมเชิงปฏิบัติการ และการสังเกต เป็นต้น การศึกษาความเป็นไปได้ครอบคลุมถึงความเป็นไปได้ทางเศรษฐศาสตร์ ความเป็นไปได้ทางเทคนิค ความเป็นไปได้เชิงองค์กรและความเป็นไปได้เชิงกฎหมาย

(สุวรรณี อัศวกุลชัย, 2561) ผลลัพธ์ที่ได้จากขั้นตอนนี้ ได้แก่ เอกสารคำร้องขอพัฒนาระบบ รายงานการวิเคราะห์ความเป็นไปได้ และร่างแนวคิดของโครงการและขอบเขตเบื้องต้น

4.2 ขั้นตอนการวิเคราะห์ความต้องการ

ขั้นตอนการวิเคราะห์ความต้องการ (Requirement Analysis) เป็นขั้นตอนที่ศึกษาและทำความเข้าใจความต้องการของผู้ใช้และธุรกิจอย่างละเอียดและครบถ้วน เป็นขั้นตอนที่อยู่ในระยะการวิเคราะห์ตามวงจรการพัฒนาระบบ จุดเน้นของขั้นตอนนี้อยู่ที่การเข้าใจพฤติกรรมผู้ใช้ (User Behavior) สภาพแวดล้อมในการใช้งาน และข้อกำหนดที่จำเป็นต้องมีทั้งที่เป็นหน้าที่หลักของเว็บแอปพลิเคชัน (Functional Requirements) เช่น การบันทึกข้อมูล การจัดทำรายงาน เป็นต้น และไม่ใช่หน้าที่หลัก (Non-functional Requirements) เช่น ความเร็วในการโหลด ความปลอดภัย ความสามารถในการรองรับผู้ใช้จำนวนมาก เป็นต้น

เทคนิคที่ใช้ในการรวบรวมความต้องการ ประกอบด้วย การสัมภาษณ์ผู้ใช้งาน การสำรวจความต้องการ การสังเกตการณ์ทำงาน การจัดทำแบบสอบถาม และการวิเคราะห์กรณีการใช้งาน (Use Case Analysis) ผลลัพธ์ที่ได้จากขั้นตอนนี้ ได้แก่ เอกสารความต้องการของผู้ใช้ เอกสารข้อกำหนดของระบบ เอกสารกรณีการใช้งาน แผนผังการทำงานของระบบ และข้อกำหนดด้านการใช้งานเฉพาะสำหรับเว็บแอปพลิเคชัน

ข้อสังเกตของขั้นตอนนี้ทีมผู้พัฒนาจำเป็นต้องมีทักษะในการสื่อสารกับผู้ใช้โดยใช้ภาษาที่เข้าใจง่าย ไม่ใช้ศัพท์ทางเทคนิคมากเกินไป นอกจากนี้ควรมีการจำแนกความต้องการของผู้ใช้โดยแบ่งออกเป็นความต้องการที่จำเป็น (Mandatory Requirement) และความต้องการที่ปรารถนา (Desirable Requirement)

4.3 ขั้นตอนการออกแบบเว็บแอปพลิเคชัน

ขั้นตอนการออกแบบเว็บแอปพลิเคชันเป็นขั้นตอนที่แปลงความต้องการให้เป็นพิมพ์เขียวสำหรับการพัฒนาเว็บแอปพลิเคชัน เป็นขั้นตอนที่อยู่ในระยะการออกแบบตามวงจรการพัฒนาระบบ มีจุดเน้นในการออกแบบโครงสร้าง รูปลักษณ์ และการทำงานของเว็บแอปพลิเคชันให้สอดคล้องกับความต้องการของผู้ใช้และข้อกำหนดของระบบ

การออกแบบในขั้นตอนนี้ประกอบด้วยหลายมิติ ได้แก่ การออกแบบประสบการณ์ผู้ใช้ (User Experience: UX) การออกแบบส่วนติดต่อผู้ใช้ (User Interface: UI) การออกแบบการนำทาง (Navigation) การออกแบบระบบและฐานข้อมูล (System and Database Design) รวมถึงการกำหนดเทคโนโลยีที่ใช้ในการพัฒนา

เทคนิคและเครื่องมือที่ใช้ในขั้นตอนนี้ ได้แก่ การสร้างโครงร่างหน้าจอ (Wireframe) การออกแบบต้นแบบใช้งาน (Prototype) การใช้โปรแกรมออกแบบ UI/UX การสร้างแบบจำลองข้อมูล (Data Modeling) และการออกแบบสถาปัตยกรรมระบบที่เหมาะสมกับลักษณะการใช้งานของเว็บแอปพลิเคชัน

ผลลัพธ์ที่ได้จากขั้นตอนนี้ ได้แก่ เอกสารการออกแบบส่วนติดต่อผู้ใช้ (UI Design Document) แบบร่างโครงสร้างหน้าเว็บ ต้นแบบการใช้งาน เอกสารการออกแบบระบบ (System Design Specification) และแบบจำลองฐานข้อมูล (Database Schema) ซึ่งใช้เป็นแนวทางสำคัญสำหรับทีมพัฒนาในการพัฒนาเว็บแอปพลิเคชัน

4.4 ขั้นตอนการเขียนรหัสคำสั่ง

ขั้นตอนการเขียนรหัสคำสั่ง (Coding) เป็นกระบวนการแปลงแบบแปลนระบบที่ได้จากขั้นตอนการออกแบบให้เป็นเว็บแอปพลิเคชันที่สามารถทำงานได้จริง เป็นขั้นตอนที่อยู่ในระยะการนำไปใช้ตามวงจรการพัฒนาระบบ มีจุดเน้นในการสร้างระบบที่สามารถประมวลผลและตอบสนองต่อการใช้งานของผู้ใช้ได้อย่างถูกต้อง มีประสิทธิภาพ และสามารถบำรุงรักษาได้ง่าย

กระบวนการเขียนโปรแกรมในเว็บแอปพลิเคชันมักแยกออกเป็นสองส่วน ได้แก่ การพัฒนาส่วนแสดงผลแก่ผู้ใช้ (Frontend Development) และการพัฒนาส่วนประมวลผลและจัดการข้อมูล (Backend Development) โดยส่วนแสดงผลแก่ผู้ใช้จะใช้ภาษาหลักในการสร้างเว็บ ประกอบด้วย ภาษา HTML ภาษา CSS และภาษา JavaScript สำหรับส่วนประมวลผลและจัดการข้อมูลจะใช้ภาษาประมวลผลในเครื่องผู้ให้บริการเว็บ เช่น ภาษา PHP ภาษา Python และภาษา JavaScript เป็นต้น รวมทั้งภาษาในการจัดการฐานข้อมูล เช่น ภาษา SQL ภาษา OQL และภาษา MQL เป็นต้น

เทคนิคที่ใช้ในการเขียนรหัสคำสั่งประกอบด้วยการเขียนโปรแกรมเชิงโครงสร้าง การเขียนโปรแกรมเชิงวัตถุ รวมทั้งการเขียนโปรแกรมด้วยสถาปัตยกรรมแบบแยกส่วนการทำงาน เช่น Model-View-Controller (MVC) เป็นต้น นอกจากนี้ในการพัฒนาเว็บแอปพลิเคชันยังต้องอาศัยการสร้างสภาพแวดล้อมการพัฒนา (Development Environment) ให้สอดคล้องกับกระบวนการทำงานของเว็บแอปพลิเคชันที่มีลักษณะเป็น Client/Server

ผลลัพธ์ที่ได้จากขั้นตอนนี้ ได้แก่ รหัสโปรแกรมของเว็บแอปพลิเคชันที่สามารถนำไปทดสอบและใช้งานได้ รวมถึงเอกสารประกอบการเขียนโปรแกรม เช่น คำอธิบายฟังก์ชัน (Function Documentation) แผนผังโครงสร้างรหัสคำสั่ง และคำอธิบายวิธีการติดตั้งและใช้งานระบบเบื้องต้น

4.5 ขั้นตอนการทดสอบเว็บแอปพลิเคชัน

ขั้นตอนการทดสอบเว็บแอปพลิเคชัน (Web Application Testing) เป็นขั้นตอนสำคัญที่ช่วยรับรองว่าระบบที่พัฒนาขึ้นสามารถทำงานได้ตรงตามข้อกำหนด มีเสถียรภาพ ปลอดภัย และสร้างประสบการณ์ใช้งานที่ดีแก่ผู้ใช้ เป็นขั้นตอนที่อยู่ในระยะการนำไปใช้ตามวงจรการพัฒนาระบบ มีจุดเน้นที่การตรวจสอบความถูกต้องของการทำงาน การตรวจสอบประสิทธิภาพ และการประเมินความพร้อมของระบบก่อนเผยแพร่จริง

เทคนิคที่ใช้ในการทดสอบเว็บแอปพลิเคชันสามารถแบ่งออกได้หลายระดับ ได้แก่ การทดสอบหน่วยย่อย (Unit Testing) เป็นการทดสอบการทำงานของแต่ละฟังก์ชันหรือโมดูลการทดสอบการผนวกรวม (Integration Testing) เป็นการทดสอบการทำงานร่วมกันของแต่ละส่วน การทดสอบระบบ (System Testing) เป็นการทดสอบระบบทั้งหมดในสภาพแวดล้อมที่ใกล้เคียงกับการใช้งานจริง และการทดสอบการยอมรับจากผู้ใช้ (User Acceptance Testing: UAT) เป็นการทดสอบความพึงพอใจของผู้ใช้และความสอดคล้องกับความต้องการ นอกจากนี้ ยังมีการทดสอบเฉพาะทาง เช่น การทดสอบการทำงานข้ามเว็บเบราว์เซอร์ (Cross-browser Testing) การทดสอบความปลอดภัย (Security Testing) และการทดสอบการใช้งานบนอุปกรณ์ต่าง ๆ (Responsive Testing)

ผลลัพธ์ที่ได้จากขั้นตอนนี้ ได้แก่ รายงานผลการทดสอบ (Test Report) รายงานข้อผิดพลาด (Bug Report) และข้อเสนอแนะในการปรับปรุงระบบ ซึ่งจะถูกนำไปใช้ในการปรับแก้ไขรหัสคำสั่งก่อนเข้าสู่ขั้นตอนการติดตั้งใช้งานจริง

4.6 ขั้นตอนการติดตั้งเว็บแอปพลิเคชัน

ขั้นตอนการติดตั้งเว็บแอปพลิเคชัน (Web Application Deployment) เป็นกระบวนการนำระบบที่ผ่านการพัฒนาและทดสอบแล้วเข้าสู่สภาพแวดล้อมจริงที่ผู้ใช้สามารถเข้าถึงได้ผ่านเครือข่ายอินเทอร์เน็ตหรืออินทราเน็ต เป็นขั้นตอนที่อยู่ในระยะการนำไปใช้ตามวงจรการพัฒนาระบบ (Implementation Phase) มีจุดเน้นที่การปรับระบบให้เหมาะสมกับสภาพแวดล้อมของเครื่องผู้ให้บริการเว็บ การตั้งค่าความปลอดภัย และการเตรียมความพร้อมด้านเทคนิคให้สามารถรองรับผู้ใช้งานจริงได้อย่างมีประสิทธิภาพ

การติดตั้งในขั้นตอนนี้เป็นการนำเว็บแอปพลิเคชันที่อยู่ในสภาพแวดล้อมการพัฒนาไปสู่สภาพแวดล้อมการใช้งานจริง (Production Environment) โดยต้องคำนึงถึงประสิทธิภาพ ความเสถียร ความปลอดภัย และความสามารถในการขยายตัว การติดตั้งต้องมีการวางแผนอย่างละเอียดเพื่อลดความเสี่ยงและผลกระทบต่อการดำเนินธุรกิจ รวมทั้งการเตรียมแผนสำรองข้อมูลและระบบตรวจสอบการทำงาน

ผลลัพธ์ที่ได้จากขั้นตอนนี้ ได้แก่ เว็บแอปพลิเคชันที่เข้าถึงได้ผ่านอินเทอร์เน็ตตลอดเวลา เครื่องผู้ให้บริการเว็บและฐานข้อมูลที่มีประสิทธิภาพและพร้อมใช้งาน ระบบสำรองข้อมูลที่ทำงานอัตโนมัติ ระบบตรวจสอบการทำงานและการแจ้งเตือน เอกสารการติดตั้งและการกำหนดค่า และคู่มือการใช้งานสำหรับผู้ใช้

4.7 ขั้นตอนการบำรุงรักษา

ขั้นตอนการบำรุงรักษา (Maintenance Phase) เป็นกระบวนการที่เกิดขึ้นหลังจากเว็บแอปพลิเคชันถูกติดตั้งและเริ่มใช้งานจริงแล้ว เป็นขั้นตอนที่อยู่ในระยะการนำไปใช้ตามวงจรการพัฒนาระบบเช่นเดียวกับขั้นตอนการติดตั้ง แต่มีจุดเน้นที่การดูแลรักษาให้ระบบทำงานได้อย่างต่อเนื่อง ปลอดภัย และสอดคล้องกับความต้องการที่เปลี่ยนแปลงไปของผู้ใช้งาน

กระบวนการบำรุงรักษาเว็บแอปพลิเคชันสามารถจำแนกออกเป็น 4 ประเภท ได้แก่การบำรุงรักษาเพื่อแก้ไขข้อผิดพลาด (Corrective Maintenance) เป็นการปรับปรุงหรือแก้ไขจุดบกพร่องหรือข้อผิดพลาดที่พบหลังจากระบบเริ่มใช้งานจริง การบำรุงรักษาเพื่อปรับปรุงประสิทธิภาพ (Perfective Maintenance) เป็นการปรับปรุงฟังก์ชันหรือประสิทธิภาพของระบบ เช่น ลดเวลาโหลด เพิ่มความเร็วในการประมวลผล หรือปรับ UI/UX ให้ดียิ่งขึ้น

การบำรุงรักษาเพื่อรองรับการเปลี่ยนแปลง (Adaptive Maintenance) เป็นการปรับปรุงระบบให้สอดคล้องกับการเปลี่ยนแปลงของสภาพแวดล้อม เช่น การเปลี่ยนรุ่นของระบบปฏิบัติการ ฐานข้อมูล หรือภาษาการเขียนโปรแกรม และการบำรุงรักษาเพื่อป้องกัน (Preventive Maintenance) เป็นการดำเนินการเชิงรุก เช่น การตรวจสอบช่องโหว่ด้านความปลอดภัย การทดสอบระบบตามรอบเวลา การสำรองข้อมูล และการตรวจสอบบันทึก (Log) เพื่อลดความเสี่ยงของปัญหาที่อาจเกิดขึ้น

เทคนิคและเครื่องมือที่ใช้ในขั้นตอนนี้ได้แก่ ระบบตรวจสอบสถานะ (Monitoring Tools) เช่น New Relic, Datadog หรือ Grafana เป็นต้น ระบบสำรองข้อมูลอัตโนมัติ (Automated Backup) ระบบจัดการข้อผิดพลาด (Error Tracking Tools) และระบบควบคุมรุ่นของรหัสคำสั่ง (Version Control) เพื่อให้สามารถย้อนกลับหรือปรับปรุงซอฟต์แวร์ได้อย่างมีประสิทธิภาพ

ผลลัพธ์ของขั้นตอนการบำรุงรักษา ได้แก่ ระบบที่มีเสถียรภาพต่อเนื่อง มีประสิทธิภาพตามที่กำหนดไว้ และสามารถตอบสนองต่อความต้องการใหม่ๆ ได้อย่างทันท่วงที รายงานการตรวจสอบประสิทธิภาพและแนวโน้มการใช้งาน รายงานการใช้งานและพฤติกรรมผู้ใช้ การปรับปรุงความปลอดภัยและฟีเจอร์ใหม่อย่างสม่ำเสมอ เอกสารการแก้ไขปัญหาและคำถามที่พบบ่อย แผนการบำรุงรักษาระยะยาว ข้อมูลสำรองที่ปลอดภัยและทันสมัย และระบบสนับสนุนผู้ใช้ที่มีประสิทธิภาพและตอบสนองรวดเร็ว

5. สรุปท้ายบท

ระบบสารสนเทศ หมายถึง ระบบที่ประกอบด้วยองค์ประกอบต่าง ๆ ซึ่งทำงานร่วมกันอย่างบูรณาการ ได้แก่ บุคลากร ซอฟต์แวร์ ฮาร์ดแวร์ ข้อมูล และกระบวนการทำงาน เพื่อให้สามารถจัดการข้อมูลให้กลายเป็นสารสนเทศที่มีความหมายและคุณค่าสอดคล้องกับวัตถุประสงค์ขององค์กร โดยระบบสารสนเทศจะอาศัยข้อมูลนำเข้ากระบวนการ และข้อมูลส่งออกภายใต้กรอบคุณภาพที่ชัดเจน และมีการควบคุม การประเมิน รวมถึงปฏิกิริยาย้อนกลับเพื่อปรับปรุงระบบให้มีประสิทธิภาพสูงสุด องค์ประกอบของระบบสารสนเทศประกอบด้วยขั้นตอนการทำงาน เทคโนโลยีสารสนเทศและบุคลากร

เว็บแอปพลิเคชันมีบทบาทสำคัญในฐานะเทคโนโลยีสารสนเทศที่ผสมผสานองค์ประกอบหลักทั้งฮาร์ดแวร์ ซอฟต์แวร์ ฐานข้อมูล และระบบเครือข่าย โดยทำหน้าที่เชื่อมโยงข้อมูล บุคลากร และกระบวนการทำงานขององค์กรให้สามารถดำเนินการได้อย่างมีประสิทธิภาพ

เว็บแอปพลิเคชันเป็นเครื่องมือที่สำคัญในการแปลงข้อมูลให้กลายเป็นสารสนเทศที่มีประโยชน์ โดยมีกระบวนการรับข้อมูลจากผู้ใช้ผ่านเว็บเบราว์เซอร์ การประมวลผลและจัดการข้อมูลด้วยภาษาโปรแกรมและฐานข้อมูล และการนำเสนอสารสนเทศกลับไปยังผู้ใช้ผ่านการแสดงผลที่เข้าใจได้ง่าย นอกจากนี้ ยังสามารถสนับสนุนระบบสารสนเทศองค์กรได้ในหลายมิติ ทั้งในแง่ของการเข้าถึงข้อมูลได้ทุกที่ทุกเวลา การประมวลผลแบบเรียลไทม์ การทำงานร่วมกันระหว่างบุคลากร และความปลอดภัยในการจัดเก็บข้อมูล

วงจรการพัฒนาระบบเป็นกระบวนการเชิงระบบที่กำหนดขั้นตอนการดำเนินงานเพื่อสร้างระบบสารสนเทศให้มีประสิทธิภาพ ตอบสนองความต้องการของผู้ใช้ และสอดคล้องกับเป้าหมายองค์กร โดยประกอบด้วย 4 ระยะ ได้แก่ ระยะวางแผน ระยะวิเคราะห์ ระยะออกแบบ และระยะการนำไปใช้ ทั้งนี้ กระบวนการพัฒนาระบบดังกล่าวมีจุดประสงค์เพื่อจัดการความซับซ้อน ควบคุมคุณภาพ จัดสรรทรัพยากร และเสริมสร้างการสื่อสารระหว่างผู้มีส่วนเกี่ยวข้อง ทำให้การพัฒนาระบบเป็นไปอย่างมีประสิทธิผลและสามารถปรับปรุงได้อย่างต่อเนื่องในระยะยาว

การพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ เป็นการประยุกต์ใช้แนวคิดของ SDLC เพื่อให้เหมาะสมกับลักษณะเฉพาะของเว็บแอปพลิเคชัน ประกอบด้วย 7 ขั้นตอนสำคัญ ได้แก่ การกำหนดปัญหาและ

ศึกษาความเป็นไปได้ การวิเคราะห์ความต้องการผู้ใช้ การออกแบบระบบและส่วนติดต่อผู้ใช้ การเขียนรหัสคำสั่ง การทดสอบระบบ การติดตั้งใช้งาน และการบำรุงรักษาระบบ โดยแต่ละขั้นตอนมีเป้าหมายเพื่อให้ระบบทำงานได้อย่างมีคุณภาพ ตรงตามความต้องการ และสามารถพัฒนาได้อย่างต่อเนื่องในอนาคต

6. กิจกรรมท้ายบท

6.1 จงอธิบายความหมายของระบบสารสนเทศ และบอกองค์ประกอบของระบบสารสนเทศพร้อมอธิบายแต่ละองค์ประกอบพอสังเขป

6.2 เว็บแอปพลิเคชันมีความสัมพันธ์และมีบทบาทอย่างไรในระบบสารสนเทศ? จงอธิบายพอสังเขป

6.3 วงจรการพัฒนาระบบคืออะไร? และมีความสำคัญอย่างไรต่อระบบสารสนเทศ? จงอธิบายพอสังเขป

6.4 วงจรการพัฒนาระบบแบ่งได้เป็น 4 ระยะ ประกอบด้วยระยะใดบ้าง? และแต่ละระยะของวงจรการพัฒนาระบบมีจุดเน้นและผลลัพธ์อะไรบ้าง? จงอธิบายพอสังเขป

6.5 จงบอกขั้นตอนการพัฒนาเว็บแอปพลิเคชันตามวงจรการพัฒนาระบบ พร้อมอธิบายแต่ละขั้นตอนพอสังเขป

6.6 จงยกตัวอย่างเว็บแอปพลิเคชันที่พบในชีวิตประจำวันมาหนึ่งรายการ โดยระบุข้อมูลนำเข้า กระบวนการทำงาน และผลลัพธ์ของระบบ พร้อมอธิบายว่าเว็บแอปพลิเคชันดังกล่าวสนับสนุนการตัดสินใจของผู้ใช้อย่างไร โดยอ้างอิงแนวคิดจากระบบสารสนเทศในบทนี้



สภาพแวดล้อมการพัฒนา



สภาพแวดล้อมเว็บแอปพลิเคชัน และการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูป



การสร้างกลไกผู้ให้บริการเว็บด้วยแนวคิดการจำลอง-เสมือนในสภาพแวดล้อมการพัฒนา



สภาพแวดล้อมเว็บแอปพลิเคชันและการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูป

บทที่ 3 สภาพแวดล้อมเว็บแอปพลิเคชันและการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูป มุ่งเน้นให้ผู้อ่านความเข้าใจถึงความแตกต่างระหว่างสภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริง รวมทั้งศึกษาองค์ประกอบหลักที่จำเป็นต่อการสร้างกลไกผู้ให้บริการเว็บในเครื่องคอมพิวเตอร์ของผู้พัฒนา เพื่อให้ผู้อ่านสามารถเตรียมพร้อมสำหรับการพัฒนาเว็บแอปพลิเคชันได้อย่างมีประสิทธิภาพและเป็นระบบ ในบทนี้ยังนำเสนอวิธีการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรม XAMPP ซึ่งเป็นหนึ่งในสองวิธีที่จะกล่าวถึงในหนังสือเล่มนี้ พร้อมทั้งนำเสนอข้อดีและข้อจำกัดเพื่อประกอบการพิจารณาการเลือกวิธีการสร้างกลไกผู้ให้บริการเว็บ เนื้อหาในบทนี้มีดังนี้

1. สภาพแวดล้อมเว็บแอปพลิเคชัน
2. องค์ประกอบหลักของสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชัน
3. ชื่อเรียกชุดโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บ
4. การสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูปในสภาพแวดล้อมการพัฒนา
5. สรุปท้ายบท
6. กิจกรรมท้ายบท

1. สภาพแวดล้อมเว็บแอปพลิเคชัน

สภาพแวดล้อมเว็บแอปพลิเคชัน (Web Application Environment) หมายถึง ระบบนิเวศของเทคโนโลยีที่ผสานองค์ประกอบด้านฮาร์ดแวร์ ซอฟต์แวร์ ฐานข้อมูล ระบบเครือข่าย และบริการต่าง ๆ ที่จำเป็นต่อการทำงานของเว็บแอปพลิเคชัน ระบบนิเวศดังกล่าวเปรียบเสมือนห้องปฏิบัติการของเว็บแอปพลิเคชันที่มีเครื่องมือต่าง ๆ ที่จำเป็นต่อการสนับสนุนการทำงานของเว็บแอปพลิเคชัน โดยเทคโนโลยีที่นำมาใช้ในระบบนิเวศของเว็บแอปพลิเคชันได้มาจากระยะการออกแบบซึ่งเป็นระยะที่ 3 ตามกรอบแนวคิดของวงจรการพัฒนาระบบดังที่ได้กล่าวถึงในบทที่ 2 ซึ่งมุ่งเน้นการกำหนดโครงสร้าง สถาปัตยกรรมและเทคโนโลยีที่จำเป็นต่อการทำงานของระบบ โดยทั่วไปสภาพแวดล้อมเว็บแอปพลิเคชันแบ่งได้เป็น 2 ลักษณะ คือ สภาพแวดล้อมการพัฒนา และสภาพแวดล้อมการใช้งานจริง (Monnikendam, 2020; IBM, 2025)

1.1 สภาพแวดล้อมการพัฒนา

ความหมายและความสำคัญของสภาพแวดล้อมการพัฒนามีดังต่อไปนี้

1.1.1 ความหมายของสภาพแวดล้อมการพัฒนา

สภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชัน (Development Environment) หมายถึง ชุดของเครื่องมือ ซอฟต์แวร์ ฮาร์ดแวร์ และการกำหนดค่าต่าง ๆ ที่จำเป็นสำหรับการสร้าง ทดสอบ และพัฒนาเว็บแอปพลิเคชัน ก่อนที่จะนำไปติดตั้งใช้งานจริงบนเครื่องผู้ให้บริการเว็บ

จากการศึกษาในบทที่ 1 เกี่ยวกับกลไกการทำงานของเว็บแอปพลิเคชัน จะเห็นว่าการพัฒนาเว็บแอปพลิเคชันต้องอาศัยการทำงานร่วมกันของหลายองค์ประกอบ ซึ่งประกอบด้วย เว็บเบราว์เซอร์ เครื่องผู้ให้บริการเว็บ โปรแกรมให้บริการเว็บ ตัวแปลภาษาฝั่งเครื่องผู้ให้บริการ และระบบจัดการฐานข้อมูล ดังนั้นสภาพแวดล้อมการพัฒนาจึงต้องสร้างองค์ประกอบเหล่านี้ให้สามารถทำงานร่วมกันได้บนเครื่องคอมพิวเตอร์ของนักพัฒนา การสร้างสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันนี้จะถูกนำไปใช้ในขั้นตอนการสร้างระบบของระยะการนำไปใช้ซึ่งเป็นระยะที่ 4 ตามกรอบแนวคิดของวงจรการพัฒนาระบบ และเป็นการดำเนินการในขั้นตอนการเขียนรหัสคำสั่งและการทดสอบตามกระบวนการพัฒนาเว็บแอปพลิเคชันที่ได้นำเสนอในบทที่ 2

ในบางกรณี โดยเฉพาะในโครงการขนาดกลางถึงขนาดใหญ่ สภาพแวดล้อมสำหรับการพัฒนาเว็บแอปพลิเคชันอาจมีการแยกย่อยเพิ่มเติมเป็นสภาพแวดล้อมสำหรับการทดสอบ (Testing Environment) เพื่อใช้ในการทดสอบรหัสคำสั่งหรือฟีเจอร์ที่พัฒนาใหม่ โดยไม่จำเป็นต้องสร้างสภาพแวดล้อมตามการใช้งานจริงทั้งหมด การทดสอบในสภาพแวดล้อมนี้จะเน้นการตรวจสอบการทำงานเฉพาะส่วน โดยอยู่บนสมมติฐานทางเทคนิคที่สามารถควบคุมได้ในระดับหนึ่ง นอกจากนี้ ยังอาจมีการสร้างสภาพแวดล้อมเตรียมการใช้งาน (Staging Environment) ซึ่งสร้างระบบทั้งหมดให้เหมือนกับสภาพแวดล้อมการใช้งานจริงมากที่สุด เพื่อทำการทดสอบแบบองค์รวมก่อนนำขึ้นระบบจริง

1.1.2 ความสำคัญของสภาพแวดล้อมการพัฒนา

สภาพแวดล้อมการพัฒนาที่เหมาะสมมีความสำคัญต่อกระบวนการพัฒนาเว็บแอปพลิเคชันในหลายประการ ดังนี้

1) เพิ่มประสิทธิภาพในการพัฒนา

สภาพแวดล้อมการพัฒนาที่ดีช่วยให้นักพัฒนาสามารถเขียนรหัสคำสั่ง ทดสอบ และแก้ไขข้อผิดพลาดได้อย่างรวดเร็ว โดยไม่ต้องอัปโหลดไฟล์ไปยังเครื่องผู้ให้บริการเว็บทุกครั้งที่มีการเปลี่ยนแปลงรหัสคำสั่ง

2) ลดความเสี่ยงต่อระบบที่ใช้งานจริง

การทดสอบและแก้ไขรหัสคำสั่งในสภาพแวดล้อมการพัฒนาช่วยป้องกันไม่ให้เกิดข้อผิดพลาดหรือปัญหาที่อาจส่งผลกระทบต่อระบบที่กำลังให้บริการผู้ใช้งานจริง

3) สนับสนุนการทำงานเป็นทีม

เมื่อทีมพัฒนามีสภาพแวดล้อมการพัฒนาที่เป็นมาตรฐานเดียวกัน จะช่วยให้การแบ่งปันรหัสคำสั่ง การแก้ไขปัญหาร่วมกัน และการบูรณาการงานของสมาชิกในทีมเป็นไปได้อย่างราบรื่นและมีทิศทางเดียวกัน

4) เอื้อต่อการเรียนรู้และทดลอง

นักพัฒนาสามารถทดลองใช้เทคนิคใหม่ ๆ ทดสอบแนวคิด หรือเรียนรู้วิธีการเขียนรหัสคำสั่งได้อย่างอิสระ โดยไม่ต้องกังวลว่าจะส่งผลกระทบต่อการทำงานของระบบที่ใช้งานจริง

5) จำลองสภาพแวดล้อมจริง

สภาพแวดล้อมการพัฒนาที่ดีควรสามารถจำลองการทำงานของระบบจริงได้ใกล้เคียงที่สุด ทั้งในส่วนของโปรแกรมให้บริการเว็บ ตัวแปลภาษา และระบบจัดการฐานข้อมูล ตามที่ได้ศึกษาองค์ประกอบเหล่านี้ในบทที่ 1

1.2 สภาพแวดล้อมการใช้งานจริง

ความหมายของสภาพแวดล้อมการใช้งานจริงและปัจจัยสำคัญของการสร้างสภาพแวดล้อมการใช้งานจริงมีดังต่อไปนี้

1.2.1 ความหมายของสภาพแวดล้อมการใช้งานจริง

สภาพแวดล้อมการใช้งานจริง (Production Environment) หมายถึง ชุดของเครื่องมือซอฟต์แวร์ ฮาร์ดแวร์ และการกำหนดค่าต่าง ๆ ที่ถูกจัดเตรียมไว้เพื่อรองรับการให้บริการเว็บแอปพลิเคชันแก่ผู้ใช้งานทั่วไปในสภาพแวดล้อมจริง สภาพแวดล้อมนี้แตกต่างจากสภาพแวดล้อมการพัฒนาในด้านความเสถียร ความปลอดภัย และประสิทธิภาพในการรองรับผู้ใช้จำนวนมากพร้อมกันอย่างต่อเนื่อง

การสร้างสภาพแวดล้อมการใช้งานจริงจะเกิดขึ้นภายหลังจากที่เว็บแอปพลิเคชันผ่านการพัฒนาและทดสอบในสภาพแวดล้อมการพัฒนาแล้ว การสร้างสภาพแวดล้อมการใช้งานจริงนี้จะเกิดขึ้นในขั้นตอน

การติดตั้งของระยะการนำไปใช้ซึ่งเป็นระยะที่ 4 ตามกรอบแนวคิดของวงจรการพัฒนาระบบและเป็นการดำเนินการในขั้นตอนการติดตั้งตามกระบวนการพัฒนาเว็บแอปพลิเคชันที่ได้นำเสนอในบทที่ 2

1.2.2 ปัจจัยสำคัญของการสร้างสภาพแวดล้อมการใช้งานจริง

สภาพแวดล้อมการใช้งานจริงเป็นปลายทางสุดท้ายที่เว็บแอปพลิเคชันจะถูกติดตั้งและให้บริการแก่ผู้ใช้งานจริง สภาพแวดล้อมการใช้งานจริงจึงมีความสำคัญอย่างยิ่งต่อความเชื่อมั่นของผู้ใช้งานและความสำเร็จโดยรวมของระบบเว็บแอปพลิเคชัน ปัจจัยสำคัญของการสร้างสภาพแวดล้อมการใช้งานจริงที่มีผลต่อความสำเร็จของเว็บแอปพลิเคชันประกอบด้วย ความสามารถในการให้บริการอย่างต่อเนื่อง การรักษาความปลอดภัยของข้อมูลและระบบ การรองรับปริมาณผู้ใช้งานพร้อมกันเป็นจำนวนมาก ระบบสำรองข้อมูลและกู้คืนกรณีเกิดปัญหา รวมทั้งการเฝ้าระวังและบำรุงรักษาระบบ

1.3 ความแตกต่างระหว่างสภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริง

สภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริงมีความแตกต่างกันในหลายด้าน ดังแสดงในตารางที่ 3-1

ตารางที่ 3-1 การเปรียบเทียบสภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริง

มิติความแตกต่าง	สภาพแวดล้อมการพัฒนา	สภาพแวดล้อมการใช้งานจริง
วัตถุประสงค์	เพื่อการพัฒนา ทดสอบ และแก้ไขเว็บแอปพลิเคชัน	เพื่อให้บริการแก่ผู้ใช้งานจริง
ผู้เข้าถึง	นักพัฒนาและทีมพัฒนา	ผู้ใช้งานทั่วไป
การแสดงข้อผิดพลาด	แสดงข้อผิดพลาดทั้งหมด	ซ่อนข้อผิดพลาดทางเทคนิค
ประสิทธิภาพการทำงาน	ไม่เน้นประสิทธิภาพสูงสุด	ต้องมีประสิทธิภาพสูง
ความปลอดภัย	การตั้งค่าความปลอดภัยอาจไม่เข้มงวด	ต้องมีความปลอดภัยสูงสุด
การปรับปรุงองค์ประกอบ	สามารถเปลี่ยนแปลงได้บ่อย	ต้องวางแผนอย่างรอบคอบ
ข้อมูล	ใช้ข้อมูลทดสอบ	ใช้ข้อมูลจริงของผู้ใช้งาน

จากตารางที่ 3.1 เป็นการเปรียบเทียบความแตกต่างระหว่างสภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริงในหลายด้าน โดยมีความแตกต่างดังนี้

1.3.1 ด้านวัตถุประสงค์

สภาพแวดล้อมการพัฒนามีวัตถุประสงค์เพื่อการพัฒนา ทดสอบ และแก้ไขรหัสคำสั่ง ในขณะที่สภาพแวดล้อมการใช้งานจริงมุ่งเน้นการให้บริการแก่ผู้ใช้งานทั่วไป

1.3.2 ด้านผู้เข้าถึง

สภาพแวดล้อมการพัฒนามีการจำกัดการใช้งานเฉพาะทีมพัฒนาซึ่งสามารถเข้าถึงองค์ประกอบภายในของสภาพแวดล้อมการพัฒนาได้ทั้งหมด ส่วนสภาพแวดล้อมการใช้งานจริงมีกลุ่มเป้าหมายหลักเป็นผู้ใช้ทั่วไปเพื่อให้สามารถใช้งานระบบได้จริง แต่จะเข้าถึงได้เฉพาะส่วนที่เป็นเว็บแอปพลิเคชันเท่านั้น

1.3.3 ด้านข้อผิดพลาด

ในสภาพแวดล้อมการพัฒนาจะมีการแสดงรายละเอียดของข้อผิดพลาดที่เกิดจากการทำงานขององค์ประกอบภายในสภาพแวดล้อมการพัฒนาอย่างครบถ้วน เพื่อให้ผู้พัฒนาสามารถหาข้อผิดพลาดและแก้ไขได้ง่าย ในขณะที่สภาพแวดล้อมการใช้งานจริงจะซ่อนข้อผิดพลาดทางเทคนิคจากผู้ใช้เพื่อป้องกันความเสี่ยงด้านความปลอดภัย

1.3.4 ด้านประสิทธิภาพการทำงาน

ระบบที่ใช้ในสภาพแวดล้อมการพัฒนาอาจไม่จำเป็นต้องมีประสิทธิภาพสูงสุด แต่ยังคงระดับประสิทธิภาพขั้นต่ำตามข้อกำหนดของระบบไว้ เนื่องจากต้องการความยืดหยุ่นในการพัฒนา ในขณะที่สภาพแวดล้อมการใช้งานจริงต้องมีความสามารถในการให้บริการอย่างต่อเนื่อง มีประสิทธิภาพสูง และสามารถรองรับผู้ใช้งานจำนวนมากพร้อมกันได้

1.3.5 ด้านการรักษาความปลอดภัย

สภาพแวดล้อมการพัฒนามักมีการตั้งค่าความปลอดภัยที่ยืดหยุ่นกว่าเพื่อความสะดวกในการทดสอบและการพัฒนา เช่น การเข้าระบบด้วยรหัสผ่านอย่างง่าย เป็นต้น ส่วนสภาพแวดล้อมการใช้งานจริงต้องมีการกำหนดค่าความปลอดภัยในระดับสูง

1.3.6 ด้านการปรับปรุงองค์ประกอบ

สภาพแวดล้อมการพัฒนาสามารถปรับปรุงองค์ประกอบภายในสภาพแวดล้อมให้เป็นรุ่นล่าสุดและเปลี่ยนแปลงได้อย่างต่อเนื่อง ในขณะที่สภาพแวดล้อมการใช้งานจริงต้องมีการวางแผนและควบคุมการปรับปรุงองค์ประกอบของสภาพแวดล้อมอย่างรอบคอบเพื่อไม่ให้กระทบต่อการให้บริการ

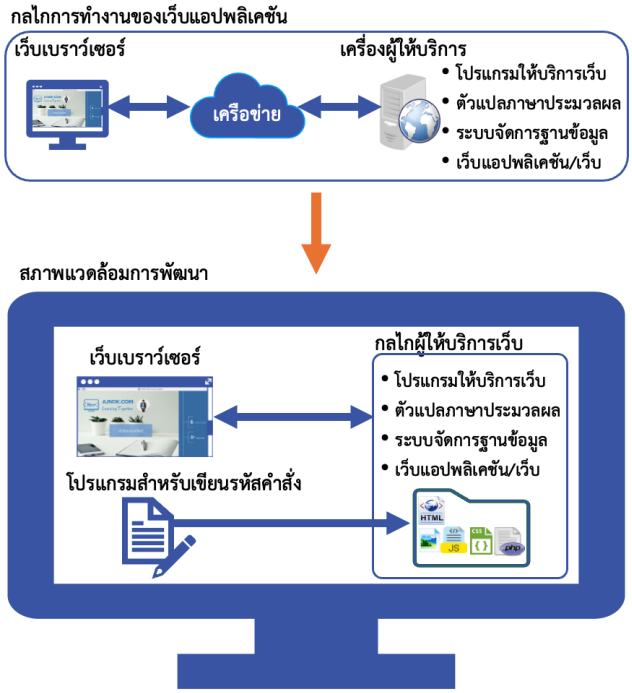
1.3.7 ด้านข้อมูล

สภาพแวดล้อมการพัฒนาจะใช้ข้อมูลจำลองหรือข้อมูลทดสอบการใช้งานกับเว็บแอปพลิเคชันเพื่อป้องกันผลกระทบต่อข้อมูลจริง และมีการใช้ข้อมูลที่หลากหลายเพื่อทดสอบการทำงานของเว็บแอปพลิเคชันเพื่อหาทางป้องกันข้อผิดพลาดที่เกิดจากการป้อนข้อมูลของผู้ใช้ ในขณะที่สภาพแวดล้อมการใช้งานจริงจะใช้ข้อมูลจริงซึ่งต้องมีการจัดการอย่างปลอดภัยและสอดคล้องตามกรอบคุณภาพของข้อมูล

2. องค์ประกอบหลักของสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชัน

จากกลไกการทำงานของเว็บแอปพลิเคชันที่ได้นำเสนอในบทที่ 1 แสดงให้เห็นว่า เว็บแอปพลิเคชันประกอบด้วยองค์ประกอบหลักทั้งในฝั่งผู้ใช้และฝั่งเครื่องผู้ให้บริการเว็บ โดยในฝั่งผู้ใช้จะเข้าถึงเว็บแอปพลิเคชันผ่านเว็บเบราว์เซอร์ ส่วนฝั่งเครื่องผู้ให้บริการจะประกอบด้วยโปรแกรมให้บริการเว็บ ตัวแปลภาษาประมวลผล

ฝั่งผู้ให้บริการ ระบบจัดการฐานข้อมูล และไฟล์ที่เกี่ยวข้องกับเว็บแอปพลิเคชัน ซึ่งทำงานร่วมกันในการตอบสนองคำร้องขอของผู้ใช้ อย่างไรก็ตาม การสร้างระบบเพื่อการพัฒนาเว็บแอปพลิเคชันตามกลไกการทำงานจริงอาจทำได้ไม่สะดวกนัก เนื่องจากต้องใช้ทรัพยากรในการพัฒนาเป็นจำนวนมากทั้งในเครื่องคอมพิวเตอร์ของผู้ใช้ เครื่องคอมพิวเตอร์ผู้ให้บริการ โปรแกรมต่าง ๆ และการเชื่อมต่อผ่านเครือข่าย ดังนั้น ในการพัฒนาเว็บแอปพลิเคชันจึงต้องมีการสร้างกลไกผู้ให้บริการเว็บภายในเครื่องคอมพิวเตอร์ของผู้พัฒนาเอง แสดงดังภาพที่ 3.1



ภาพที่ 3.1 การสร้างกลไกการทำงานของผู้ให้บริการเว็บภายใต้สภาพแวดล้อมการพัฒนา

จากภาพที่ 3.1 แสดงถึงการสร้างกลไกการทำงานของผู้ให้บริการเว็บแทนการใช้เครื่องผู้ให้บริการจริงภายใต้สภาพแวดล้อมการพัฒนาโดยมีองค์ประกอบหลักดังนี้

2.1 เครื่องคอมพิวเตอร์ของผู้พัฒนา

เครื่องคอมพิวเตอร์ของผู้พัฒนาเป็นเครื่องที่ใช้ในการสร้างและทดสอบการทำงานของเว็บแอปพลิเคชันโดยคุณสมบัติทางฮาร์ดแวร์และระบบปฏิบัติการของคอมพิวเตอร์ผู้พัฒนาจะต้องรองรับการติดตั้งและการใช้งานโปรแกรมต่าง ๆ ที่เป็นองค์ประกอบหลักของสภาพแวดล้อมการพัฒนา ทั้งนี้ ข้อแนะนำโดยทั่วไปสำหรับเครื่องคอมพิวเตอร์ของผู้พัฒนาควรมีขนาดพื้นที่ในหน่วยความจำหลัก (RAM) อย่างน้อย 8 GB หรือ 16 GB ขึ้นไป เนื่องจากในการพัฒนาจำเป็นต้องเปิดใช้งานหลายโปรแกรมพร้อมกัน และหากนำเทคโนโลยีคอนเทนเนอร์มาใช้ในการสร้างสภาพแวดล้อมการพัฒนาก็จะใช้หน่วยความจำหลักมากขึ้นตามไปด้วย

2.2 โปรแกรมสำหรับเขียนรหัสคำสั่ง

โปรแกรมสำหรับเขียนรหัสคำสั่งเป็นโปรแกรมสำคัญของผู้พัฒนาที่ใช้สำหรับสร้าง แก้ไขและจัดการรหัสคำสั่งของภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนาเว็บแอปพลิเคชัน เปรียบเสมือนเป็นปากกาและกระดาษสำหรับการเขียนของผู้เขียน โปรแกรมสำหรับเขียนรหัสคำสั่งแบ่งได้เป็น 2 ลักษณะ ดังนี้

2.2.1 โปรแกรมที่ใช้สำหรับแก้ไขข้อความธรรมดา

โปรแกรมแก้ไขข้อความธรรมดา (Text Editor) เป็นโปรแกรมที่ใช้สำหรับสร้าง แก้ไขและจัดการไฟล์ข้อความ (Text File) โดยทำงานกับข้อความแบบธรรมดาที่ไม่ได้มีการจัดรูปแบบ (Plain Text) หรืออาจมีการจัดรูปแบบบ้างแต่ไม่ซับซ้อน เช่น การเปลี่ยนรูปแบบตัวอักษร สีของข้อความ หรือขนาดตัวอักษร เป็นต้น ทั้งนี้ไฟล์รหัสคำสั่งภาษาคอมพิวเตอร์ไม่ได้คำนึงถึงรูปแบบของข้อความแต่ให้ความสำคัญกับไวยากรณ์และโครงสร้างของภาษาเป็นสำคัญ

ตัวอย่างโปรแกรมแก้ไขข้อความธรรมดา เช่น โปรแกรม Notepad โปรแกรม TextEdit และโปรแกรม Vim ซึ่งใช้ในระบบปฏิบัติการ Microsoft Windows ระบบปฏิบัติการ macOS และระบบปฏิบัติการ Linux ตามลำดับ นอกจากนี้ บางโปรแกรมมีความสามารถเพิ่มเติมในการเน้นสีไวยากรณ์ในรหัสคำสั่ง เพื่ออำนวยความสะดวกแก่ผู้พัฒนาในการแยกส่วนที่เป็นคำสั่งของภาษานั้น ๆ ออกจากส่วนที่ไม่ใช่คำสั่งของภาษา รวมทั้งสามารถติดตั้งส่วนเสริมการทำงานเพิ่มเติมได้ เช่น โปรแกรม Notepad++ และโปรแกรม Atom เป็นต้น

2.2.2 โปรแกรมที่มีสภาพแวดล้อมการพัฒนาแบบบูรณาการ

โปรแกรมที่มีสภาพแวดล้อมการพัฒนาแบบบูรณาการ (Integrated Development Environment: IDE) เป็นโปรแกรมที่รวมเอาเครื่องมือหลักที่จำเป็นสำหรับการเขียนโปรแกรมมาไว้ด้วยกันอย่างครบวงจร เช่น ส่วนการเขียนรหัสคำสั่ง ส่วนการแปลรหัสคำสั่งเป็นภาษาเครื่อง ส่วนการสั่งให้คอมพิวเตอร์ทำงานตามรหัสคำสั่ง ส่วนการตรวจสอบและแก้ไขข้อผิดพลาดจากการทำงานตามรหัสคำสั่ง ส่วนจัดการไฟล์และโครงสร้างการจัดการไฟล์ ส่วนการควบคุมรุ่นของรหัสคำสั่ง เป็นต้น

นอกเหนือจากจะมีการเน้นสีไวยากรณ์ในรหัสคำสั่งและความสามารถในการติดตั้งส่วนเสริม ซึ่งเป็นการอำนวยความสะดวกขั้นพื้นฐานแล้ว ยังมีความสามารถในการแสดงคำแนะนำระหว่างการพิมพ์รหัสคำสั่งในแต่ละภาษาที่โปรแกรมรองรับ ช่วยให้ผู้พัฒนาลดเวลาในการพิมพ์และลดข้อผิดพลาดจากการพิมพ์ได้อีกด้วย ตัวอย่างโปรแกรมลักษณะนี้ที่สามารถนำมาใช้ในการพัฒนาเว็บแอปพลิเคชัน เช่น โปรแกรม Visual Studio Code (VSCode) โปรแกรม NetBeans และโปรแกรม PhpStorm เป็นต้น

ปัจจุบัน โปรแกรมที่มีสภาพแวดล้อมการพัฒนาแบบบูรณาการหลายโปรแกรมได้เพิ่มขีดความสามารถด้วยการนำเทคโนโลยีปัญญาประดิษฐ์ (Artificial Intelligence: AI) เข้ามาช่วยในการเขียนรหัสคำสั่ง เช่น GitHub Copilot, Codeium และ Tabnine เป็นต้น โดย AI เหล่านี้สามารถสร้างชุดรหัสคำสั่งตามคำอธิบายของผู้พัฒนา วิเคราะห์รหัสคำสั่งและให้คำแนะนำที่เหมาะสม ตรวจหาข้อผิดพลาดพร้อมแนะนำแนวทางแก้ไข ปรับปรุงโครงสร้างของรหัสคำสั่งให้เหมาะสมโดยอัตโนมัติ รวมถึงการคาดการณ์คำสั่งที่ผู้พัฒนากำลังพิมพ์เพื่อเพิ่มความแม่นยำและลดเวลาในการพัฒนา

2.3 เว็บเบราว์เซอร์

เว็บเบราว์เซอร์เป็นโปรแกรมสำหรับการท่องเว็บที่ติดตั้งในเครื่องคอมพิวเตอร์ของผู้พัฒนา ซึ่งผู้พัฒนาควรติดตั้งเว็บเบราว์เซอร์หลักที่เป็นที่นิยมในการใช้งานให้ครอบคลุม เช่น โปรแกรม Google Chrome โปรแกรม Mozilla Firefox โปรแกรม Safari และ โปรแกรม Microsoft Edge เป็นต้น ทั้งนี้เนื่องจากจะใช้ในการทดสอบการทำงานของเว็บแอปพลิเคชันให้รองรับการทำงานและการแสดงผลในเว็บเบราว์เซอร์ได้หลากหลาย ซึ่งผู้ใช้แต่ละคนอาจใช้เว็บเบราว์เซอร์ที่แตกต่างกันไปตามความถนัด

2.4 กลไกผู้ให้บริการเว็บ

ส่วนของกลไกผู้ให้บริการเว็บตามภาพที่ 3.1 จะมีการติดตั้งโปรแกรมที่ทำงานร่วมกันในลักษณะที่เรียกว่า Software Stack กล่าวคือ เป็นชุดของซอฟต์แวร์หรือโปรแกรมคอมพิวเตอร์ที่ทำงานร่วมกันเป็นลำดับชั้น (Layer) เพื่อสนับสนุนการทำงานของเว็บแอปพลิเคชันซึ่งชุดของซอฟต์แวร์หรือโปรแกรมนี้อาจรวมถึงระบบปฏิบัติการ ชุดฟังก์ชันสำเร็จรูป สภาพแวดล้อมขณะทำงานของโปรแกรม และระบบจัดการฐานข้อมูล (Kirvan, Semilof, & Montgomery, 2024) กลไกผู้ให้บริการเว็บนี้จะประกอบด้วยโปรแกรมหลัก 3 โปรแกรม คือ โปรแกรมให้บริการเว็บ ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ และระบบจัดการฐานข้อมูล

2.4.1 โปรแกรมให้บริการเว็บ

โปรแกรมให้บริการเว็บเป็นโปรแกรมที่ติดตั้งลงในเครื่องคอมพิวเตอร์ของผู้พัฒนาให้มีความสามารถในการให้บริการเว็บ เมื่อทำการติดตั้งแล้วผู้พัฒนาก็สามารถร้องขอบริการเว็บจากเครื่องคอมพิวเตอร์ของผู้พัฒนาเองได้โดยไม่ต้องร้องขอผ่านระบบเครือข่าย ทำให้ไม่ต้องใช้เครื่องคอมพิวเตอร์ผู้ให้บริการเว็บจริง ตัวอย่างของโปรแกรมให้บริการเว็บ เช่น โปรแกรม Nginx โปรแกรม Apache HTTP Server และ Microsoft IIS เป็นต้น ทั้งนี้โปรแกรม Nginx และ Apache HTTP Server เป็นโปรแกรมประเภทเปิดเผยรหัส (Open Source) ที่นิยมใช้อย่างแพร่หลายและมีรุ่นที่สามารถใช้งานได้โดยไม่เสียค่าใช้จ่าย

2.4.2 ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ

ตัวแปลภาษาประมวลผลเป็นโปรแกรมที่ทำหน้าที่ในการแปลและทำงานตามรหัสคำสั่งภาษาประมวลผลในฝั่งผู้ให้บริการ โดยจะต้องทำงานร่วมกันได้กับโปรแกรมให้บริการเว็บ เช่น ตัวแปลภาษา PHP ตัวแปลและสภาพแวดล้อมการทำงานของภาษา JavaScript ในเครื่องผู้ให้บริการ และตัวแปลภาษา Python เป็นต้น

2.4.3 ระบบจัดการฐานข้อมูล

ระบบจัดการฐานข้อมูลเป็นโปรแกรมที่มีหน้าที่ในการจัดการกับข้อมูลที่บันทึกในหน่วยบันทึกข้อมูลของคอมพิวเตอร์ในรูปแบบของฐานข้อมูล เช่น โปรแกรม MySQL โปรแกรม MariaDB และโปรแกรม Microsoft SQL Server เป็นต้น โดยโปรแกรม MySQL และ MariaDB เป็นโปรแกรมประเภทเปิดเผยรหัสที่ได้รับความนิยมในการนำมาใช้กับฐานข้อมูลเชิงสัมพันธ์ นอกจากนี้หากต้องการสั่งการระบบจัดการฐานข้อมูลผ่านหน้าเว็บเพื่อลดความยุ่งยากในการสั่งการด้วยบรรทัดคำสั่ง (Command-Line) ก็สามารถติดตั้งเว็บแอปพลิเคชันที่ใช้

เป็นส่วนเชื่อมต่อระหว่างผู้ใช้และระบบจัดการฐานข้อมูลผ่านเว็บ (Web Interface) ลงในโปรแกรมให้บริการเว็บได้ เช่น PhpMyAdmin และ Adminer ซึ่งเป็นส่วนเชื่อมต่อสำหรับโปรแกรม MySQL และ MariaDB เป็นต้น

3. ชื่อเรียกชุดโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บ

จากโปรแกรมหลักของกลไกผู้ให้บริการเว็บที่กล่าวถึงในหัวข้อที่ 2.4 หากมีการเปลี่ยนแปลงโปรแกรมให้บริการเว็บหรือระบบปฏิบัติการที่ใช้งาน ชื่อเรียกของ Software Stack ก็จะเปลี่ยนไปตามโปรแกรมให้บริการเว็บและระบบปฏิบัติการนั้น ซึ่งแต่ละชุดโปรแกรมมีโครงสร้างหลักที่คล้ายคลึงกัน แต่แตกต่างกันที่ระบบปฏิบัติการและโปรแกรมบางส่วนภายใน (ธนัชชา วีรวัฒนโยธิน, 2566; IBM Cloud Education Team, 2022) ดังนี้

3.1 LAMP Stack

LAMP เป็นคำที่ย่อมาจากคำว่า **L**inux, **A**pache, **M**ySQL และ **P**HP เป็นชุดของโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บ ประกอบด้วย โปรแกรม Apache เป็นโปรแกรมให้บริการเว็บ ทำงานร่วมกับตัวแปลภาษา PHP ในการแปลและประมวลผลภาษาที่ทำงานในฝั่งผู้ให้บริการ และมีโปรแกรม MySQL หรือ MariaDB สำหรับเป็นระบบจัดการฐานข้อมูล โดยทั้ง 3 โปรแกรมนี้ทำงานภายใต้สภาพแวดล้อมของระบบปฏิบัติการ Linux

3.2 LEMP Stack

LEMP เป็นคำที่ย่อมาจากคำว่า **L**inux, **N**ginx, **M**ySQL และ **P**HP เป็นชุดของโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บที่ใช้โปรแกรมและสภาพแวดล้อมของระบบปฏิบัติการเช่นเดียวกับ LAMP Stack ยกเว้นโปรแกรมให้บริการเว็บ ซึ่ง LEMP Stack จะใช้โปรแกรม Nginx (อ่านว่า Engine-x) เป็นโปรแกรมให้บริการเว็บทำงานร่วมกับตัวแปลภาษา PHP

3.3 WAMP Stack

WAMP เป็นคำที่ย่อมาจากคำว่า **W**indows, **A**pache, **M**ySQL และ **P**HP เป็นชุดของโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บที่ใช้โปรแกรมเช่นเดียวกับ LAMP Stack แต่จะทำงานภายใต้สภาพแวดล้อมของระบบปฏิบัติการ Windows

3.4 MAMP Stack

MAMP เป็นคำที่ย่อมาจากคำว่า **m**acOS, **A**pache, **M**ySQL และ **P**HP เป็นชุดของโปรแกรมที่ใช้สร้างกลไกผู้ให้บริการเว็บที่ใช้โปรแกรมเช่นเดียวกับ LAMP Stack และ WAMP Stack แต่จะทำงานภายใต้สภาพแวดล้อมของระบบปฏิบัติการ macOS

3.5 MEAN Stack

MEAN เป็นคำที่ย่อมาจากคำว่า **M**ongoDB, **E**xpress.js, **A**ngula.js และ **N**ode.js เป็นชุดของโปรแกรมและกรอบงาน (Framework) ที่ใช้สร้างกลไกผู้ให้บริการเว็บที่ทำงานกับภาษา JavaScript เป็นการเฉพาะ ดังนี้

- MongoDB เป็นโปรแกรมระบบจัดการฐานข้อมูลที่จัดเก็บข้อมูลในรูปแบบเอกสาร (Document-Oriented Database)
- Express.js ทำงานร่วมกับ Node.js ในการทำงานเป็นผู้ให้บริการเว็บในลักษณะการรับส่งข้อมูลระหว่างฝั่งผู้ใช้และฝั่งผู้ให้บริการ
- AngularJS เป็นเฟรมเวิร์กของภาษา JavaScript ที่ทำงานในเว็บเบราว์เซอร์ซึ่งใช้สำหรับสร้างการติดต่อกับผู้ใช้ในเว็บเบราว์เซอร์
- Node.js เป็นสภาพแวดล้อมการทำงานและการประมวลผลภาษา JavaScript ในฝั่งเครื่องผู้ให้บริการเว็บ ตามปกติแล้วภาษา JavaScript จะประมวลผลในเว็บเบราว์เซอร์ แต่ด้วยกรอบงาน Node.js ทำให้ภาษา JavaScript ประมวลผลในเครื่องผู้ให้บริการเว็บได้

4. การสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูปในสภาพแวดล้อมการพัฒนา

การสร้างกลไกผู้ให้บริการเว็บในสภาพแวดล้อมการพัฒนาโดยใช้โปรแกรมสำเร็จรูป (Package Software) เป็นวิธีการหนึ่งที่ผู้พัฒนาสามารถทำให้เครื่องคอมพิวเตอร์ของผู้พัฒนาสามารถทำงานตามกลไกของผู้ให้บริการเว็บตามที่ได้อธิบายถึงในหัวข้อที่ 2.4 โดยผู้พัฒนาสามารถพิจารณาเลือกใช้โปรแกรมสำเร็จรูปที่ทำหน้าที่ในการติดตั้งโปรแกรมหลักของกลไกผู้ให้บริการเว็บทั้ง 3 โปรแกรมพร้อมกันในครั้งเดียวเพื่ออำนวยความสะดวกและลดเวลาในการติดตั้ง ตัวอย่างของโปรแกรมสำเร็จรูปแสดงดังตารางที่ 3-2

ตารางที่ 3-2 ตัวอย่างโปรแกรมสำเร็จรูปสำหรับการติดตั้งชุดโปรแกรมกลไกผู้ให้บริการเว็บ

ชื่อโปรแกรม	ระบบปฏิบัติการที่สนับสนุน	องค์ประกอบที่ติดตั้ง
XAMPP	Windows/macOS/Linux	Apache + MariaDB + PHP + Perl
WampServer	Windows	Apache + MySQL + PHP
MAMP	macOS/Windows	Apache/Nginx + MySQL + PHP
AMPPS	Windows/macOS	Apache + MySQL + PHP
LAMP Stack	Linux	Apache + MariaDB + PHP/Python/Perl

ในหัวข้อนี้จะนำเสนอการใช้โปรแกรม XAMPP เพื่อติดตั้งชุดโปรแกรมกลไกผู้ให้บริการเว็บในระบบปฏิบัติการ Microsoft Windows โดยมีข้อมูลทั่วไปเกี่ยวกับ XAMPP และขั้นตอนการติดตั้งดังต่อไปนี้

4.1 ข้อมูลทั่วไปเกี่ยวกับ XAMPP

XAMPP เป็นชุดโปรแกรมที่รวมโปรแกรมให้บริการเว็บ ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการและระบบจัดการฐานข้อมูลมาไว้ในแพ็กเกจเดียวกัน ได้รับการพัฒนาโดย Apache Friends และเป็นโปรแกรมประเภทเปิดเผยรหัสที่สามารถดาวน์โหลดและใช้งานได้โดยไม่เสียค่าใช้จ่าย ชื่อ XAMPP มาจากตัวอักษรย่อดังนี้

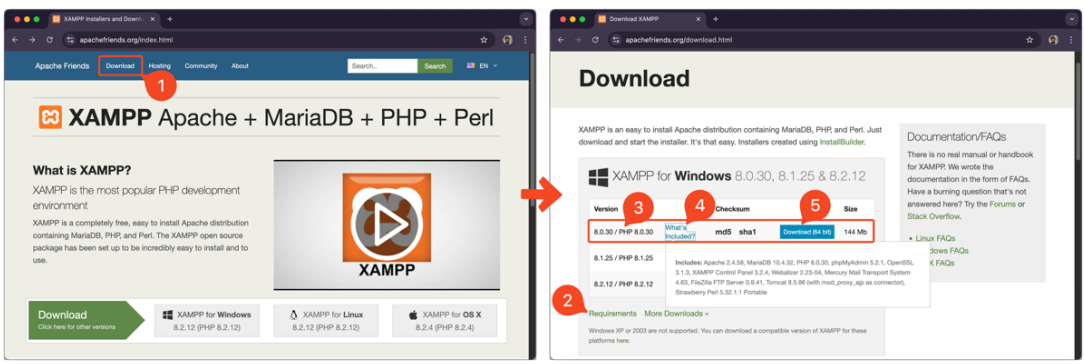
- X ใช้แทนคำว่า Cross-platform หมายถึงความสามารถในการทำงานข้ามแพลตฟอร์มระบบปฏิบัติการ โดยสามารถติดตั้งและใช้งานได้ทั้งในระบบปฏิบัติการ Microsoft Windows ระบบปฏิบัติการ macOS และระบบปฏิบัติการ Linux
- A ใช้แทนคำว่า Apache HTTP Server ซึ่งเป็นชื่อของโปรแกรมให้บริการเว็บแบบเปิดเผยรหัส
- M ใช้แทนคำว่า MariaDB ซึ่งเป็นชื่อของระบบจัดการฐานข้อมูลเชิงสัมพันธ์ (Relational Database Management System: RDBMS) แบบเปิดเผยรหัส
- P ใช้แทนคำว่า PHP ย่อมาจากคำว่า PHP: Hypertext Preprocessor เป็นภาษาประมวลผลฝั่งผู้ให้บริการ โดย XAMPP จะมีการติดตั้งตัวแปลภาษา PHP ให้สามารถทำงานร่วมกับ Apache HTTP Server และระบบจัดการฐานข้อมูล MariaDB
- P ใช้แทนคำว่า Perl เป็นภาษาประมวลผลฝั่งผู้ให้บริการเช่นเดียวกับภาษา PHP โดย XAMPP จะมีการติดตั้งตัวแปลภาษา Perl มาพร้อมด้วยทำให้ผู้พัฒนาสามารถใช้ภาษา Perl ในการสร้างเว็บแอปพลิเคชันได้เช่นกัน

4.2 การติดตั้งและการใช้งาน XAMPP

การติดตั้งและการใช้งาน XAMPP มีขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1) ดาวน์โหลดไฟล์สำหรับการติดตั้ง XAMPP

เริ่มต้นจากการเข้าไปยังเว็บไซต์อย่างเป็นทางการของ Apache Friends ซึ่งเป็นผู้พัฒนาโปรแกรม XAMPP เข้าถึงได้จาก https://www.apachefriends.org จากนั้นคลิกที่เมนู Download ดังแสดงในตำแหน่งที่ 1 ของภาพที่ 3.2 เป็นการเข้าสู่หน้าเว็บสำหรับการดาวน์โหลดไฟล์สำหรับการติดตั้ง XAMPP โดยแบ่งหมวดหมู่ตามระบบปฏิบัติการที่ผู้พัฒนาใช้อยู่



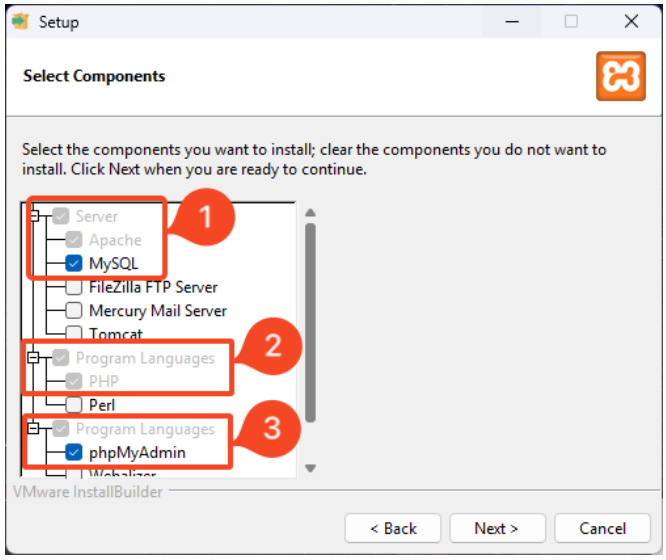
ภาพที่ 3.2 เว็บไซต์ทางการของผู้พัฒนาโปรแกรม XAMPP

จากตัวอย่างดังภาพที่ 3.2 เป็นไฟล์ติดตั้งโปรแกรม XAMPP สำหรับระบบปฏิบัติการ Microsoft Windows มีข้อมูลสำคัญที่เกี่ยวข้องในแต่ละตำแหน่งของภาพที่ 3.2 ประกอบด้วย ตำแหน่งที่ 2 แสดงข้อกำหนดเกี่ยวกับระบบปฏิบัติการที่รองรับ ตำแหน่งที่ 3 แสดงรุ่นของโปรแกรม XAMPP และรุ่นของภาษา PHP ตำแหน่งที่ 4 แสดงรายการและรุ่นของโปรแกรมที่เกี่ยวข้องกับกลไกผู้ให้บริการเว็บ และตำแหน่งที่ 5 เป็นปุ่มสำหรับการดาวน์โหลดไฟล์ติดตั้งโปรแกรม XAMPP

ขั้นตอนที่ 2) ติดตั้ง XAMPP

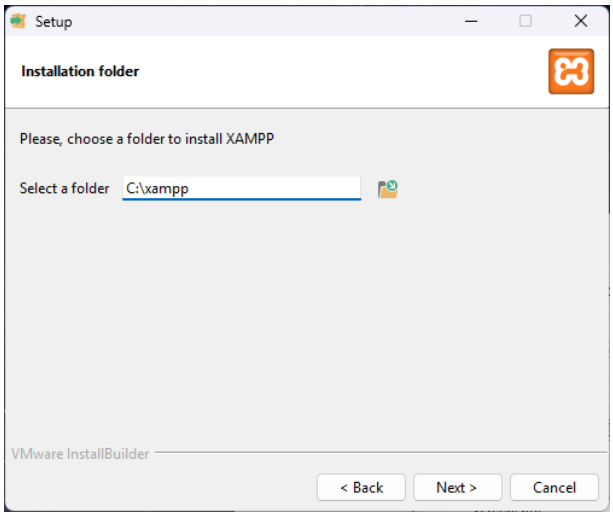
การติดตั้ง XAMPP ในระบบปฏิบัติ Microsoft Windows มีส่วนดำเนินการที่สำคัญดังนี้

- ดับเบิลคลิกไฟล์สำหรับการติดตั้งที่ดาวน์โหลดมาจากขั้นตอนที่ 1 เมื่อโปรแกรมติดตั้งเริ่มทำงาน จะมีการแสดงหน้าต้อนรับพร้อมข้อมูลเบื้องต้นเกี่ยวกับ XAMPP ให้คลิกปุ่ม Next เพื่อดำเนินการต่อไป
- ขั้นตอนถัดไปจะเป็นการเลือกองค์ประกอบ (Component) ที่ต้องการติดตั้งแสดงดังภาพที่ 3.3 โดยในการพัฒนาเว็บแอปพลิเคชันทั่วไปที่ใช้ภาษา PHP และระบบจัดการฐานข้อมูล MySQL (MariaDB) ให้ทำการเลือกองค์ประกอบ MySQL และ phpMyAdmin (เว็บแอปพลิเคชันที่ช่วยในการจัดการฐานข้อมูล) เป็นอย่างน้อย ส่วนองค์ประกอบ Apache และ PHP ตามภาพที่ 3.3 นั้น จะไม่สามารถยกเลิกการเลือกได้เนื่องจากเป็นองค์ประกอบหลักที่ต้องใช้ในกลไกผู้ให้บริการเว็บ



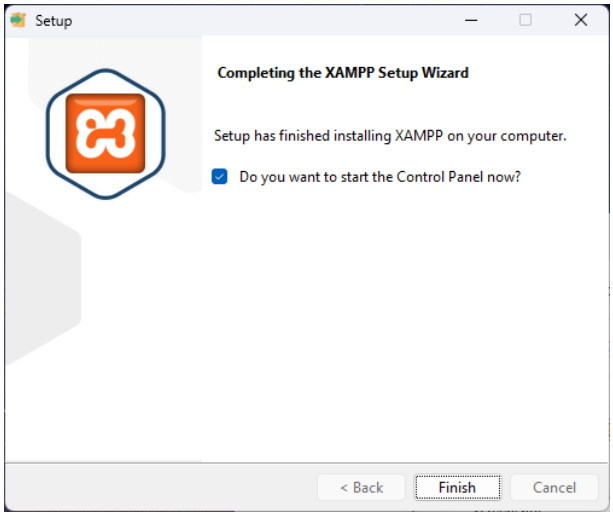
ภาพที่ 3.3 การเลือกองค์ประกอบที่ต้องการติดตั้ง

- หลังจากเลือกองค์ประกอบที่ต้องการติดตั้งแล้ว ระบบจะให้กำหนดตำแหน่งที่จัดเก็บไฟล์และโฟลเดอร์ที่เกี่ยวข้องกับการทำงานของ XAMPP รวมทั้งองค์ประกอบที่ได้เลือกทั้งหมดจากขั้นตอนก่อนหน้า โดยค่าเริ่มต้นจะเป็น C:\xampp แสดงดังภาพที่ 3.4 ให้คลิกปุ่ม Next เพื่อดำเนินการต่อไป



ภาพที่ 3.4 การกำหนดตำแหน่งที่จัดเก็บไฟล์และโฟลเดอร์ที่เกี่ยวข้องกับการทำงานของ XAMPP

- ขั้นตอนต่อไปอาจมีการแจ้งเตือนเกี่ยวกับการอนุญาตให้องค์ประกอบต่าง ๆ ที่ได้เลือกจากขั้นตอนก่อนหน้าให้สามารถเข้าถึงระบบเครือข่ายคอมพิวเตอร์ เมื่อการติดตั้งเสร็จสิ้นจะแสดงผลดังภาพที่ 3.5 โดยหากคลิกปุ่ม Finish จะเป็นการเปิดหน้าต่างควบคุมการทำงานของ XAMPP (XAMPP Control Panel)



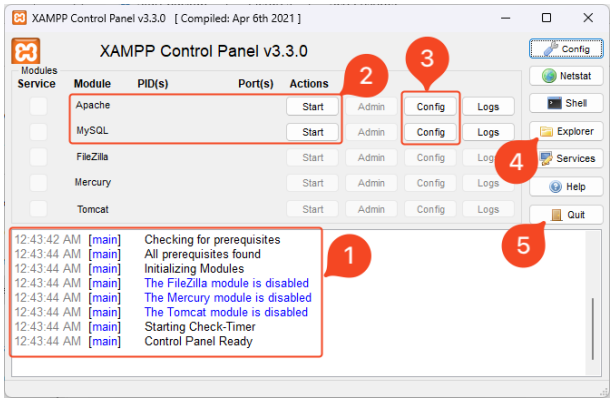
ภาพที่ 3.5 การแสดงสถานะการติดตั้ง XAMPP เสร็จสมบูรณ์

ขั้นตอนที่ 3) เปิดและปิดการทำงานขององค์ประกอบต่าง ๆ ที่ได้ติดตั้ง

จากหน้าต่างควบคุมการทำงานของ XAMPP แสดงดังภาพที่ 3.6 มีส่วนประกอบที่สำคัญในการใช้งานดังนี้

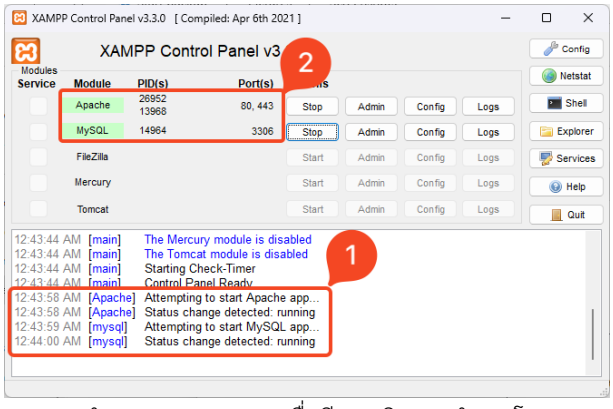
- ตำแหน่งที่ 1 เป็นส่วนแสดงบันทึกการทำงาน (Log) ของ XAMPP

- ตำแหน่งที่ 2 เป็นส่วนที่ใช้เปิดและปิดการทำงานของ Apache และ MySQL พร้อมทั้งแสดงหมายเลขโปรเซส (PID) และพอร์ตการทำงาน
- ตำแหน่งที่ 3 เป็นส่วนที่ใช้ในการกำหนดค่าการทำงานของ Apache และ MySQL ผ่านไฟล์กำหนดค่า (Configuration File) ของแต่ละโปรแกรม
- ตำแหน่งที่ 4 เป็นส่วนที่ใช้ในการเปิดตำแหน่งที่จัดเก็บไฟล์และโฟลเดอร์ที่เกี่ยวข้องกับการทำงานของ XAMPP ซึ่งจากขั้นตอนการติดตั้งจะมีตำแหน่งอยู่ที่ C:\xampp เป็นค่าเริ่มต้น รวมทั้งที่เก็บไฟล์ในการให้บริการเว็บแอปพลิเคชันจะอยู่ที่ C:\xampp\htdocs
- ตำแหน่งที่ 5 เป็นการปิดโปรแกรมควบคุมการทำงานของ XAMPP ทั้งนี้การคลิกที่ปุ่ม X ที่อยู่มุมบนด้านขวามือของหน้าต่างโปรแกรม จะเป็นการซ่อนโปรแกรมไว้ที่ถาดระบบ (System Tray) ซึ่งแตกต่างจากโปรแกรมโดยทั่วไปที่จะเป็นการปิดโปรแกรม



ภาพที่ 3.6 หน้าต่างควบคุมการทำงานของ XAMPP

เมื่อคลิกปุ่มเปิดการทำงาน (Start) ของโปรแกรม Apache และ MySQL จะแสดงผลดังภาพที่ 3.7 โดยมีข้อมูลที่สำคัญดังนี้

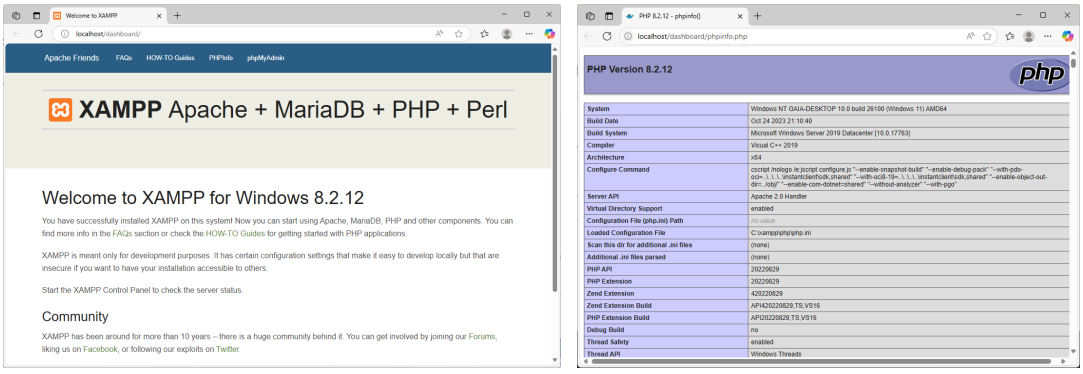


ภาพที่ 3.7 หน้าต่างควบคุมการทำงานของ XAMPP เมื่อมีการเปิดการทำงานโปรแกรม Apache และ MySQL

- ตำแหน่งที่ 1 แสดงข้อความว่ามีการเปิดการทำงานของโปรแกรมแต่ละโปรแกรม พร้อมแสดงสถานะการทำงานเป็น running หมายความว่าโปรแกรมนั้น ๆ กำลังทำงานอยู่
- ตำแหน่งที่ 2 แสดงสถานการณ์ทำงานของแต่ละโปรแกรม โดยโปรแกรมที่กำลังทำงานอยู่จะแสดงพื้นหลังชื่อโปรแกรมเป็นสีเขียว รวมทั้งแสดงหมายเลขโปรเซสและพอร์ตการทำงาน จากภาพจะเห็นว่าโปรแกรม Apache มีหมายเลขโปรเซสเป็น 26952 และ 13968 มีพอร์ตการทำงานที่หมายเลข 80 และ 443 ตามลำดับ ส่วนโปรแกรม MySQL มีหมายเลขโปรเซสเป็น 14964 และพอร์ตการทำงานที่หมายเลข 3306

ขั้นตอนที่ 4) การทดสอบการทำงานของ Apache

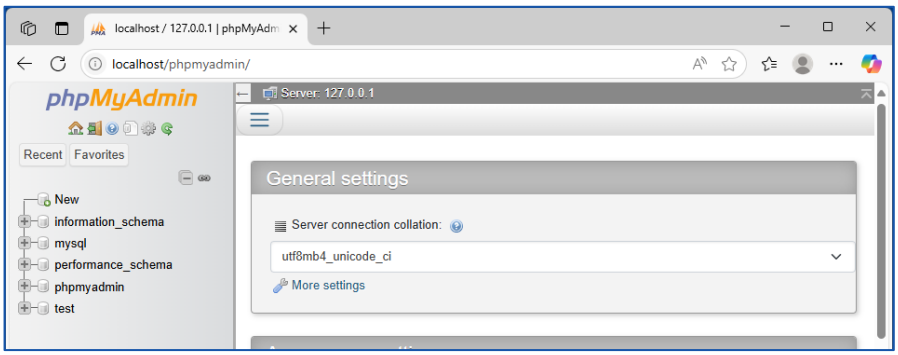
หลังจากที่ได้เปิดการทำงาน Apache จากหน้าต่างควบคุมการทำงานของ XAMPP แล้ว สามารถทดสอบการทำงานของ Apache ซึ่งเป็นโปรแกรมในการให้บริการเว็บและมีการทำงานร่วมกับตัวแปลภาษา PHP ได้ 2 วิธี คือ 1) การคลิกที่ปุ่ม Admin ในหน้าต่างควบคุมการทำงานของ XAMPP และ 2) การเปิดโปรแกรมเว็บเบราว์เซอร์และเข้าถึงด้วยที่อยู่ http://localhost หรือ http://127.0.0.1 ซึ่งทั้งสองวิธีจะนำเข้าสู่หน้าเว็บดังหน้าต่างทางด้านซ้ายมือของภาพที่ 3.8 โดยไฟล์ต่าง ๆ ที่เกี่ยวข้องกับหน้าเว็บนี้จะถูกจัดเก็บไว้ที่ C:\xampp\htdocs ซึ่งเป็นตำแหน่งที่เก็บทรัพยากรตามกลไกการให้บริการเว็บดังที่กล่าวถึงในบทที่ 1 นอกจากนี้ในการทดสอบว่า Apache สามารถทำงานร่วมกับตัวแปลภาษา PHP ได้หรือไม่ สามารถดูข้อมูลได้จากการคลิกที่เมนู PHPInfo ของภาพที่ 3.8 จะปรากฏข้อมูลของภาษา PHP ที่ใช้ในกลไกให้บริการเว็บ



ภาพที่ 3.8 หน้าเว็บทดสอบการทำงานของ Apache

ขั้นตอนที่ 5) การทดสอบการทำงานของ MySQL

หลังจากที่ได้เปิดการทำงาน MySQL แล้ว สามารถทดสอบการทำงานของ MySQL ได้โดยการตรวจสอบจากเว็บแอปพลิเคชัน phpMyAdmin ซึ่งสามารถเข้าถึงได้ 2 วิธี คือ 1) การคลิกที่ปุ่ม Admin ของ MySQL ในหน้าต่างควบคุมการทำงานของ XAMPP และ 2) การเปิดโปรแกรมเว็บเบราว์เซอร์และเข้าถึงด้วยที่อยู่ http://localhost/phpmyadmin ผลลัพธ์ที่ได้จากทั้ง 2 วิธีแสดงดังภาพที่ 3.9



ภาพที่ 3.9 หน้าเว็บแอปพลิเคชัน phpMyAdmin

4.3 ข้อดีและข้อจำกัดของการใช้ XAMPP

การเลือกใช้ XAMPP เป็นเครื่องมือในการสร้างกลไกผู้ให้บริการเว็บภายใต้สภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันมีทั้งข้อดีและข้อจำกัดที่ผู้พัฒนาควรพิจารณา ดังนี้ (Apache Friends, n.d.)

4.3.1 ข้อดีของการใช้ XAMPP

ข้อดีของการใช้ XAMPP มีดังนี้

1) ความสะดวกในการติดตั้งและใช้งานเป็นจุดเด่นหลักของ XAMPP ผู้พัฒนาสามารถติดตั้งองค์ประกอบสำคัญทั้งหมดของสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันได้ในขั้นตอนเดียว โดยไม่จำเป็นต้องติดตั้งและกำหนดค่าโปรแกรมแต่ละตัวแยกจากกัน ซึ่งช่วยประหยัดเวลาและลดความซับซ้อนในการเตรียมสภาพแวดล้อมการพัฒนาอย่างมีนัยสำคัญ โดยเฉพาะสำหรับผู้เริ่มต้นเรียนรู้การพัฒนาเว็บแอปพลิเคชันที่อาจยังไม่คุ้นเคยกับการติดตั้งและกำหนดค่าโปรแกรมในเครื่องผู้ให้บริการต่าง ๆ

2) การรวมองค์ประกอบที่จำเป็นไว้ในแพ็กเกจเดียวทำให้ XAMPP มีความครบถ้วนสำหรับการพัฒนาเว็บแอปพลิเคชันทั่วไป ประกอบด้วย Apache HTTP Server สำหรับการให้บริการเว็บ MySQL และ MariaDB สำหรับการจัดการฐานข้อมูล PHP สำหรับการประมวลผลฝั่งผู้ให้บริการ รวมถึงเครื่องมือเสริมอื่น ๆ เช่น phpMyAdmin สำหรับการจัดการฐานข้อมูลผ่านหน้าเว็บ การรวมองค์ประกอบเหล่านี้ไว้ในแพ็กเกจเดียวช่วยให้ผู้พัฒนาสามารถเริ่มต้นการทำงานได้ทันทีหลังจากติดตั้งเสร็จสิ้น

3) หน้าต่างควบคุมการทำงานของ XAMPP เป็นเครื่องมือที่ออกแบบมาให้ใช้งานง่าย ผู้พัฒนาสามารถเริ่ม หยุด และตรวจสอบสถานะของบริการต่าง ๆ ได้ผ่านส่วนติดต่อผู้ใช้แบบกราฟิก โดยไม่จำเป็นต้องใช้บรรทัดคำสั่งหรือมีความรู้เชิงลึกเกี่ยวกับการจัดการเครื่องผู้ให้บริการ

4.3.2 ข้อจำกัดของการใช้ XAMPP

ข้อจำกัดของการใช้ XAMPP ที่ควรคำนึงถึงมีดังนี้

1) XAMPP ถูกออกแบบมาเพื่อใช้งานในสภาพแวดล้อมการพัฒนาเท่านั้น ไม่เหมาะสมสำหรับการใช้งานจริงในระบบที่ให้บริการแก่ผู้ใช้ทั่วไป เนื่องจากการกำหนดค่าเริ่มต้นของ XAMPP มุ่งเน้นความสะดวกในการพัฒนามากกว่าความปลอดภัย ทำให้ระบบมีช่องโหว่ที่อาจถูกใช้ประโยชน์ในทางที่ผิดได้หากเชื่อมต่อกับเครือข่ายสาธารณะหรืออินเทอร์เน็ต

2) ประสิทธิภาพและความยืดหยุ่นในการปรับการทำงานอาจเป็นข้อจำกัดสำหรับเว็บแอปพลิเคชันที่มีความซับซ้อนหรือการรองรับผู้ใช้จำนวนมาก โดย XAMPP ถูกออกแบบมาเพื่อความสะดวกในการใช้งานทั่วไป ไม่ใช่เพื่อประสิทธิภาพสูงสุดหรือการปรับแต่งที่ซับซ้อน

3) การจัดการหลายเว็บแอปพลิเคชันพร้อมกันอาจมีความยุ่งยากเพิ่มขึ้น โดยเฉพาะเมื่อแต่ละเว็บแอปพลิเคชันต้องการรุ่นของภาษา PHP หรือการตั้งค่าฐานข้อมูลที่แตกต่างกัน แม้ว่า XAMPP จะมีความสามารถในการสลับรุ่นของภาษา PHP แต่การจัดการอาจไม่สะดวกเท่ากับการใช้เครื่องมือเฉพาะสำหรับการจัดการสภาพแวดล้อมการพัฒนาหลายเว็บแอปพลิเคชัน เช่น เทคโนโลยีคอนเทนเนอร์และเทคโนโลยีเครื่องเสมือน

4) ปัญหาการขัดแย้งของหมายเลขพอร์ตเป็นอุปสรรคที่พบบ่อยในการใช้งาน XAMPP โดยเฉพาะในเครื่องคอมพิวเตอร์ที่มีการติดตั้งโปรแกรมอื่นที่ใช้พอร์ต 80, 443 หรือ 3306 การแก้ไขปัญหานี้อาจต้องใช้ความรู้เพิ่มเติมเกี่ยวกับการกำหนดค่าระบบเครือข่ายและอาจส่งผลต่อการทำงานของโปรแกรมอื่นในระบบ นอกจากนี้ โปรแกรมป้องกันไวรัสมักตรวจจับไฟล์บางส่วนของ XAMPP ว่าเป็นภัยคุกคาม ทำให้เกิดการขัดจังหวะในการทำงานและต้องใช้เวลาในการปรับแต่งการตั้งค่าโปรแกรมป้องกันไวรัส

5) ปัญหาความสอดคล้องระหว่างเครื่องพัฒนาและเครื่องที่ให้บริการจริงเป็นประเด็นที่สำคัญในการใช้ XAMPP สภาพแวดล้อมการพัฒนาที่สร้างด้วย XAMPP มักมีการกำหนดค่าที่แตกต่างจากเครื่องผู้ให้บริการที่ใช้งานจริงอย่างมีนัยสำคัญ เช่น รุ่นของภาษา PHP ระบบปฏิบัติการ การตั้งค่าของโปรแกรมให้บริการเว็บและการกำหนดค่าฐานข้อมูล ความแตกต่างเหล่านี้อาจทำให้เว็บแอปพลิเคชันที่ทำงานได้ปกติในเครื่องพัฒนากลับเกิดข้อผิดพลาดเมื่อนำไปติดตั้งบนเครื่องผู้ให้บริการจริง

6) ระบบไฟล์และการจัดการสิทธิ์ของไฟล์ใน XAMPP ที่ติดตั้งบนระบบปฏิบัติการ Microsoft Windows มักแตกต่างจากระบบปฏิบัติการ Linux ที่ใช้ในการให้บริการจริง ความแตกต่างนี้ทำให้เกิดปัญหาเกี่ยวกับเส้นทางการเข้าถึงไฟล์ สิทธิ์การเข้าถึงไฟล์ และการทำงานของ script ที่เกี่ยวข้องกับระบบไฟล์ ซึ่งอาจส่งผลต่อการทำงานของเว็บแอปพลิเคชันเมื่อย้ายจากสภาพแวดล้อมการพัฒนาไปสู่การใช้งานจริง

5. สรุปท้ายบท

สภาพแวดล้อมเว็บแอปพลิเคชันเป็นระบบนิเวศของเทคโนโลยีที่รวมองค์ประกอบด้านฮาร์ดแวร์ ซอฟต์แวร์ ฐานข้อมูล ระบบเครือข่าย และบริการต่าง ๆ ที่จำเป็นต่อการทำงานของเว็บแอปพลิเคชัน โดยแบ่งออกเป็น 2 ลักษณะหลัก คือ สภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริง ซึ่งมีความแตกต่างกันในหลายมิติทั้งด้านวัตถุประสงค์ ผู้เข้าถึง การแสดงข้อผิดพลาด ประสิทธิภาพการทำงาน ความปลอดภัย การปรับปรุงองค์ประกอบ และการใช้ข้อมูล

สภาพแวดล้อมการพัฒนาที่เหมาะสมมีความสำคัญต่อกระบวนการพัฒนาเว็บแอปพลิเคชันในหลายประการ ได้แก่ ช่วยเพิ่มประสิทธิภาพในการพัฒนา ลดความเสี่ยงต่อระบบที่ใช้งานจริง สนับสนุนการทำงานเป็นทีม เอื้อต่อการเรียนรู้และทดลอง และสามารถจำลองสภาพแวดล้อมจริงได้ใกล้เคียง

องค์ประกอบหลักของสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันประกอบด้วย เครื่องคอมพิวเตอร์ของผู้พัฒนา โปรแกรมสำหรับเขียนรหัสคำสั่ง เว็บเบราว์เซอร์ และกลไกผู้ให้บริการเว็บ ซึ่งกลไกผู้ให้บริการเว็บจะทำงานในลักษณะ Software Stack ที่ประกอบด้วยโปรแกรมให้บริการเว็บ ตัวแปลภาษาประมวลผลฝั่งผู้ให้บริการ และระบบจัดการฐานข้อมูล โดยมีชื่อเรียกต่าง ๆ ตามองค์ประกอบและระบบปฏิบัติการที่ใช้ เช่น LAMP, LEMP, WAMP, MAMP และ MEAN Stack เป็นต้น

การใช้โปรแกรมสำเร็จรูปในการสร้างกลไกผู้ให้บริการเว็บเป็นวิธีการที่ช่วยอำนวยความสะดวกในการติดตั้งและใช้งาน โดย XAMPP เป็นตัวอย่างโปรแกรมสำเร็จรูปที่ได้รับความนิยม ซึ่งรวมองค์ประกอบสำคัญของกลไกผู้ให้บริการเว็บไว้ในแพ็กเกจเดียว ประกอบด้วย Apache HTTP Server, MariaDB, PHP และ Perl สามารถทำงานข้ามแพลตฟอร์มระบบปฏิบัติการ รวมทั้งมีหน้าต่างควบคุมการทำงานที่ใช้งานง่าย อย่างไรก็ตาม การใช้ XAMPP มีทั้งข้อดีและข้อจำกัดที่ควรพิจารณา โดยข้อดีที่สำคัญ ได้แก่ ความสะดวกในการติดตั้ง การรวมองค์ประกอบครบถ้วนและการใช้งานที่ง่าย ส่วนข้อจำกัดที่สำคัญประกอบด้วย โปรแกรมถูกออกแบบมาเพื่อใช้ในสภาพแวดล้อมการพัฒนา ปัญหาความสอดคล้องกับระบบจริง และข้อจำกัดในการจัดการหลายโครงการ

จากข้อดีและข้อจำกัดของการใช้ XAMPP ที่ได้นำเสนอ ผู้พัฒนาอาจต้องการทางเลือกอื่นที่สามารถแก้ไขข้อจำกัดเหล่านี้ได้ โดยเฉพาะในด้านความสอดคล้องระหว่างสภาพแวดล้อมการพัฒนาและการใช้งานจริง ซึ่งวิธีการสร้างกลไกผู้ให้บริการเว็บที่จะช่วยลดข้อจำกัดที่สำคัญนี้จะได้กล่าวถึงในบทที่ 4 การสร้างกลไกผู้ให้บริการเว็บด้วยแนวคิดการจำลองเสมือนในสภาพแวดล้อมการพัฒนา ซึ่งเป็นแนวทางสมัยใหม่ที่ได้รับความนิยมในอุตสาหกรรมการพัฒนาซอฟต์แวร์และสามารถแก้ไขข้อจำกัดหลายประการของโปรแกรมสำเร็จรูป นอกจากนี้ความรู้ที่ได้จากทั้งสองบทจะช่วยให้ผู้อ่านสามารถเปรียบเทียบและเลือกใช้วิธีการที่เหมาะสมที่สุดสำหรับการพัฒนาเว็บแอปพลิเคชันในแต่ละสถานการณ์

6. กิจกรรมท้ายบท

6.1 จงอธิบายความหมายของสภาพแวดล้อมเว็บแอปพลิเคชัน

6.2 สภาพแวดล้อมเว็บแอปพลิเคชันแบ่งได้เป็น 2 ลักษณะ มีอะไรบ้าง? จงให้คำนิยามและอธิบายความแตกต่างของสภาพแวดล้อมทั้งสองพอสังเขป

6.3 องค์ประกอบของสภาพแวดล้อมการพัฒนามีอะไรบ้าง? และแต่ละองค์ประกอบมีบทบาทอย่างไรในการสร้างกลไกผู้ให้บริการเว็บ?

6.4 Software Stack ที่ใช้ในการสร้างกลไกผู้ให้บริการเว็บคืออะไร? ประกอบด้วยโปรแกรมหลักอะไรบ้าง? ยกตัวอย่างชื่อเรียก Software Stack มาอย่างน้อย 3 ชื่อ พร้อมระบุระบบปฏิบัติการที่รองรับ

6.5 การสร้างกลไกผู้ให้บริการเว็บในสภาพแวดล้อมการพัฒนาโดยใช้โปรแกรมสำเร็จรูปมีข้อดีและข้อจำกัดอย่างไรบ้าง? จงอธิบายพอสังเขป และระบุสถานการณ์ที่เหมาะสมต่อการใช้งาน

6.6 ให้ติดตั้ง XAMPP แล้วเปิดใช้งาน Apache และ MySQL จากนั้นทดสอบการทำงานโดยเข้าถึง http://localhost และ http://localhost/phpmyadmin ผ่านเว็บเบราว์เซอร์



การสร้างกลไกผู้ให้บริการเว็บด้วยแนวคิดการจำลองเสมือนในสภาพแวดล้อมการพัฒนา

บทที่ 4 มุ่งนำเสนอแนวทางการสร้างกลไกผู้ให้บริการเว็บด้วยเทคโนโลยีคอนเทนเนอร์ ซึ่งเป็นเทคโนโลยีที่อยู่ภายใต้แนวคิดการจำลองเสมือน โดยเป็นอีกทางเลือกหนึ่งในการสร้างกลไกผู้ให้บริการเว็บ นอกเหนือจากการใช้โปรแกรมสำเร็จรูปที่ได้กล่าวไว้ในบทก่อนหน้า โดยบทนี้จะเน้นการใช้ Docker Desktop ซึ่งเป็นเครื่องมือที่ได้รับความนิยมในอุตสาหกรรมซอฟต์แวร์ สำหรับการจัดการคอนเทนเนอร์และการจำลองสภาพแวดล้อมการพัฒนา ผู้อ่านจะได้เรียนรู้การใช้งาน Docker Desktop และ Docker Compose ผ่านกรณีศึกษาการสร้าง LAMP Stack ซึ่งมีความสามารถในการให้บริการเว็บเช่นเดียวกับการใช้โปรแกรมสำเร็จรูป แต่มีความยืดหยุ่นและสอดคล้องกับแนวโน้มของอุตสาหกรรมซอฟต์แวร์สมัยใหม่มากยิ่งขึ้น เนื้อหาในบทนี้มีดังนี้

1. แนวคิดการจำลองเสมือน
2. หลักการของ Docker Desktop
3. กรณีศึกษา: การสร้าง LAMP Stack ด้วย Docker Desktop
4. สรุปท้ายบท
5. กิจกรรมท้ายบท

1. แนวคิดการจำลองเสมือน

แนวคิดการจำลองเสมือน (Virtualization Approach) เป็นแนวคิดในการสร้างสภาพแวดล้อมการทำงานที่แยกอิสระจากกันในเครื่องคอมพิวเตอร์เครื่องเดียวกันเพื่อเพิ่มประสิทธิภาพ ความยืดหยุ่นและความปลอดภัยในการใช้งาน โดยในการสร้างกลไกผู้ให้บริการเว็บในเครื่องคอมพิวเตอร์ผู้พัฒนาสามารถนำแนวคิดนี้ไปใช้ในการแยกสภาพแวดล้อมในเครื่องของผู้พัฒนาสำหรับการพัฒนาเว็บแอปพลิเคชันได้ โดยเทคโนโลยีหลักที่เกี่ยวข้องกับแนวคิดการจำลองเสมือนประกอบด้วยเทคโนโลยีเครื่องเสมือนและเทคโนโลยีคอนเทนเนอร์

1.1 เทคโนโลยีเครื่องเสมือน

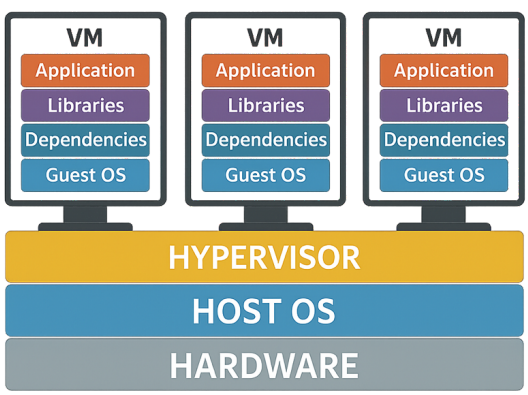
เทคโนโลยีเครื่องเสมือนเป็นการสร้างสภาพแวดล้อมที่แยกส่วนจากระบบคอมพิวเตอร์หลักในระดับฮาร์ดแวร์ (Nair, 2024) โดยมีความหมายและสถาปัตยกรรมเทคโนโลยีเครื่องเสมือน ดังนี้

1.1.1 ความหมายของเทคโนโลยีเครื่องเสมือน

เทคโนโลยีเครื่องเสมือน (Virtual Machine Technology) หมายถึง เทคโนโลยีที่ใช้ในการสร้างเครื่องหรือคอมพิวเตอร์เสมือน (Virtual Machine: VM) ขึ้นมาภายในเครื่องหรือคอมพิวเตอร์หลัก (Host Machine) โดยอาศัยกระบวนการจำลองฮาร์ดแวร์ (Hardware Virtualization) ซึ่งเป็นการแบ่งทรัพยากรต่าง ๆ ของคอมพิวเตอร์หลักมาใช้กับเครื่องเสมือน เช่น หน่วยประมวลผลกลาง หน่วยความจำหลัก และหน่วยบันทึกข้อมูล เป็นต้น โดยเครื่องเสมือนจะมีการติดตั้งระบบปฏิบัติการของตนเองแยกต่างหากจากระบบปฏิบัติการของเครื่องหลัก รวมทั้งองค์ประกอบด้านซอฟต์แวร์ทั้งหมด ทำให้เสมือนว่าเป็นระบบแยกอิสระออกจากเครื่องคอมพิวเตอร์หลัก (Susnjara & Smalley, 2025; Red Hat, 2024)

1.1.2 สถาปัตยกรรมเทคโนโลยีเครื่องเสมือน

สถาปัตยกรรมเทคโนโลยีเครื่องเสมือนมีการแบ่งส่วนการทำงานออกเป็นระดับชั้น ประกอบด้วย ฮาร์ดแวร์ (Hardware) ระบบปฏิบัติการโฮสต์ (Host Operating System: Host OS) ซอฟต์แวร์บริหารจัดการเครื่องเสมือน (Hypervisor) และเครื่องเสมือน แสดงดังภาพที่ 4.1



ภาพที่ 4.1 สถาปัตยกรรมเทคโนโลยีเครื่องเสมือน

ที่มา: ปรับปรุงจาก Nutanix (2024) และ Red Hat (2024)

จากภาพที่ 4.1 เป็นสถาปัตยกรรมเทคโนโลยีเครื่องเสมือน ซึ่งสภาพแวดล้อมของแต่ละเครื่องเสมือนจะได้รับการจัดสรรทรัพยากรทางกายภาพจากเครื่องหลัก และมีการติดตั้งระบบปฏิบัติการรวมทั้งซอฟต์แวร์ต่าง ๆ เสมือนว่าเป็นระบบที่แยกออกจากกัน (Isolated System) แอปพลิเคชันรวมทั้งองค์ประกอบด้านซอฟต์แวร์ภายในเครื่องเสมือนจะใช้เคอร์เนลของระบบปฏิบัติการภายในเครื่องเสมือนเอง โดยระบบปฏิบัติการที่ติดตั้งในเครื่องเสมือนนี้เรียกว่าระบบปฏิบัติการเกสต์ (Guest Operating System: Guest OS) ส่วนระบบปฏิบัติการที่ติดตั้งในเครื่องหลักเรียกว่าระบบปฏิบัติการโฮสต์ การบริหารจัดการเครื่องเสมือนและการจัดสรรทรัพยากรทางกายภาพนี้จะดำเนินการผ่านซอฟต์แวร์ที่เรียกว่า Hypervisor หรือเรียกอีกอย่างหนึ่งว่า Virtual Machine Monitor (VMM) ซึ่งมี 2 ประเภท ดังนี้ (Red Hat, 2024)

1) Type-1 Hypervisor

Type-1 Hypervisor หรือเรียกว่า Bare-Metal Hypervisor เป็นซอฟต์แวร์ที่ทำงานโดยตรงกับฮาร์ดแวร์ของเครื่องหลัก และประสานการทำงานกับระบบปฏิบัติการเกสต์ในเครื่องเสมือนโดยไม่ต้องทำงานผ่านระบบปฏิบัติการโฮสต์ Hypervisor ประเภทนี้ทำหน้าที่แทนระบบปฏิบัติการโฮสต์ในการจัดการทรัพยากรให้กับเครื่องเสมือน และเป็นผู้จัดสรรทรัพยากรของเครื่องหลักไปยังเครื่องเสมือนโดยตรง มักนิยมใช้ในศูนย์ข้อมูลขององค์กร (Enterprise Data Center) หรือในสภาพแวดล้อมที่ใช้เซิร์ฟเวอร์เป็นหลัก ตัวอย่างของ Hypervisor ประเภทนี้ เช่น VMware ESXi, Microsoft Hyper-V และ Kernel-based Virtual Machine (KVM) เป็นต้น

2) Type-2 Hypervisor

Type-2 Hypervisor หรือเรียกว่า Hosted Hypervisor เป็นซอฟต์แวร์ที่ทำงานบนระบบปฏิบัติการโฮสต์เช่นเดียวกับแอปพลิเคชันทั่วไป Hypervisor ประเภทนี้จะทำหน้าที่แยกระบบปฏิบัติการเกสต์ออกจากระบบปฏิบัติการโฮสต์ และจัดสรรทรัพยากรผ่านระบบปฏิบัติการโฮสต์ก่อนส่งต่อไปยังฮาร์ดแวร์ เหมาะสำหรับผู้ใช้ทั่วไปที่ต้องการใช้งานระบบปฏิบัติการหลายระบบบนคอมพิวเตอร์เครื่องเดียว ตัวอย่างของ Hypervisor ประเภทนี้ เช่น Oracle Virtual Box, VMware Workstation และ Parallels Desktop เป็นต้น

1.1.3 ประโยชน์ของเทคโนโลยีเครื่องเสมือน

ประโยชน์ของการนำเทคโนโลยีเครื่องเสมือนไปใช้ในบริบททั่วไป รวมทั้งการใช้งานในระดับองค์กรหรือการจัดการระบบขนาดใหญ่ มีดังนี้ (Susnjara & Smalley, 2025; IBM, 2023)

1) การเพิ่มประสิทธิภาพการใช้ทรัพยากร

การใช้เทคโนโลยีเครื่องเสมือนช่วยให้สามารถแบ่งทรัพยากรของเครื่องคอมพิวเตอร์หลักออกเป็นหลายเครื่องเสมือนที่ใช้งานร่วมกันได้ ซึ่งส่งผลให้การใช้ทรัพยากรฮาร์ดแวร์มีประสิทธิภาพมากยิ่งขึ้น และลดการปล่อยทรัพยากรให้ว่างเปล่าโดยไม่จำเป็น

2) การประหยัดต้นทุนและลดการใช้พลังงาน

องค์กรสามารถลดค่าใช้จ่ายในการจัดซื้อและบำรุงรักษาเครื่องคอมพิวเตอร์โดยเฉพาะเครื่องคอมพิวเตอร์ผู้ให้บริการ จากเดิมที่ต้องจัดซื้อและบำรุงรักษาหลายเครื่อง แต่การใช้เทคโนโลยีเครื่องเสมือน

จะช่วยลดจำนวนเครื่องคอมพิวเตอร์และอุปกรณ์ที่ต้องใช้ลงได้ อีกทั้งยังช่วยลดค่าสาธารณูปโภคที่ต้องใช้กับคอมพิวเตอร์เหล่านี้ได้ นอกจากนี้ยังช่วยลดปัญหาสิ่งแวดล้อมอันเนื่องมาจากการลดใช้พลังงานไฟฟ้าอีกด้วย

3) การโยกย้ายที่รวดเร็ว

ด้วยการที่เครื่องเสมือนมีการกำหนดค่าด้วยซอฟต์แวร์ สภาพแวดล้อมของแต่ละเครื่องเสมือนจะถูกสร้างเป็นไฟล์ ผู้ดูแลระบบสามารถโอนย้ายหรือสำเนาไฟล์ดังกล่าวไปยังคอมพิวเตอร์เครื่องอื่น หรือย้ายไปยังระบบคลาวด์ได้โดยไม่ต้องปรับแต่งการตั้งค่าใหม่ทั้งหมด ซึ่งเป็นข้อดีอย่างยิ่งในกรณีที่ต้องเปลี่ยนระบบหรือทำการสำรองข้อมูล

4) การกู้คืนระบบจากภาวะฉุกเฉิน

เทคโนโลยีเครื่องเสมือนสามารถทำสำเนาระบบไว้ล่วงหน้าหรือตามช่วงเวลาที่กำหนด (Snapshot) รวมทั้งสามารถย้อนกลับ (Rollback) หรือกู้คืน (Restore) สภาพแวดล้อมเดิมในภายหลังหากเกิดข้อผิดพลาดหรือความเสียหายได้อย่างรวดเร็ว ช่วยลดเวลาในการหยุดทำงานของระบบ

5) การแยกสภาพแวดล้อมเพื่อความปลอดภัย

เทคโนโลยีเครื่องเสมือนมีความเหมาะสมกับงานที่ต้องการความมั่นคงปลอดภัยในระดับสูง เช่น ระบบขององค์กรที่เกี่ยวข้องกับข้อมูลที่มีความอ่อนไหว หรือส่วนงานที่ไม่เกี่ยวข้องกัน จะแยกงานนั้นไว้ในเครื่องเสมือนที่แยกจากกัน เพราะเครื่องเสมือนแต่ละเครื่องทำงานแยกกันอย่างอิสระ

6) การจัดการที่ง่ายและมีประสิทธิภาพ

เครื่องเสมือนสามารถสร้าง ลบ คัดลอก หรือควบคุมการทำงานได้จากระยะไกลผ่านเครื่องมือจัดการแบบอัตโนมัติ ทำให้ทีมพัฒนาและผู้ดูแลระบบสามารถควบคุมและดูแลระบบจำนวนมากได้จากศูนย์กลาง

7) การลดความแตกต่างในการจัดการเรียนการสอนและการฝึกอบรม

เทคโนโลยีเครื่องเสมือนสามารถนำมาใช้ในการจัดการเรียนการสอนและการฝึกอบรม ในกรณีที่เครื่องของผู้เรียนมีความแตกต่างกัน เช่น ระบบปฏิบัติการแตกต่างกัน การปรับแต่งระบบปฏิบัติการหรือซอฟต์แวร์ไม่เหมือนกัน เป็นต้น

1.1.4 ข้อจำกัดของการใช้เทคโนโลยีเครื่องเสมือนในสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชัน

แม้ว่าเทคโนโลยีเครื่องเสมือนจะมีประโยชน์ในหลายด้าน แต่ในบริบทของผู้พัฒนาเว็บแอปพลิเคชันที่ใช้เครื่องคอมพิวเตอร์ทั่วไปในการพัฒนา ยังมีข้อจำกัดหลายประการที่ควรพิจารณา ดังนี้

1) ประสิทธิภาพการทำงานของเครื่องหลักลดลง

การพัฒนาเว็บแอปพลิเคชันในเครื่องเสมือนทำให้ประสิทธิภาพการทำงานของเครื่องหลักลดลง เนื่องจากเครื่องหลักจะต้องจัดสรรทรัพยากรโดยเฉพาะหน่วยประมวลผลกลางและหน่วยความจำหลักไปให้กับเครื่องเสมือนแต่ละเครื่อง ทำให้เครื่องหลักมีทรัพยากรในการใช้งานลดลงตามไปด้วย ซึ่งส่งผลต่อประสิทธิภาพโดยรวมของระบบปฏิบัติการโฮสต์ การตอบสนองของระบบที่ช้าลงนี้ส่งผลกระทบต่อประสิทธิภาพการทำงานของนักพัฒนา

2) การใช้ทรัพยากรสูง

เทคโนโลยีเครื่องเสมือนมีความต้องการทรัพยากรจากเครื่องหลักเป็นจำนวนมาก เนื่องจากแต่ละเครื่องเสมือนจะต้องทำงานภายใต้ระบบปฏิบัติการเกสต์ที่แยกออกจากเครื่องหลักโดยสมบูรณ์ ทำให้ต้องจัดสรรหน่วยความจำหลัก หน่วยประมวลผลกลาง และพื้นที่จัดเก็บข้อมูลให้กับแต่ละเครื่องเสมือนอย่างเพียงพอ เช่น เครื่องเสมือนที่ติดตั้งระบบปฏิบัติการ Microsoft Windows Server 2025 มีข้อกำหนดขั้นต่ำทางด้านฮาร์ดแวร์ประกอบด้วย หน่วยความจำหลักอย่างน้อย 2 GB หน่วยประมวลผลกลางแบบ 64 บิตที่มีความถี่สัญญาณนาฬิกาไม่น้อยกว่า 1.4 GHz และมีพื้นที่ว่างในฮาร์ดดิสก์อย่างน้อย 32 GB เป็นต้น ดังนั้น คุณสมบัติของเครื่องคอมพิวเตอร์ของผู้พัฒนาก็จะต้องมีการพิจารณาข้อกำหนดขั้นต่ำเหล่านี้ รวมกับการใช้งานโดยทั่วไปของผู้พัฒนาด้วย

3) เวลาเริ่มต้นที่ช้า (Slow Startup Time)

เครื่องเสมือนต้องใช้เวลาระยะเวลาหนึ่งในการเริ่มการทำงานของระบบปฏิบัติการเกสต์ (Boot) เช่นเดียวกับระบบปฏิบัติการในเครื่องหลัก ในการพัฒนาเว็บแอปพลิเคชันที่ต้องมีการทดสอบอย่างต่อเนื่อง การต้องรอให้เครื่องเสมือนเริ่มต้นทำงานในทุกครั้งที่เปิดใช้งาน อาจกลายเป็นข้อจำกัดที่ส่งผลกระทบต่อประสิทธิภาพในการทำงาน และทำให้ระยะเวลาการพัฒนาโดยรวมเพิ่มขึ้นอย่างไม่จำเป็น

4) ค่าใช้จ่ายและข้อจำกัดด้านสิทธิการใช้งาน

การใช้งานเทคโนโลยีเครื่องเสมือนมักมีค่าใช้จ่ายแฝงที่เกิดจากการจัดซื้อสิทธิการใช้งานซอฟต์แวร์ (Software Licensing) ทั้งในส่วนของระบบปฏิบัติการเกสต์ในแต่ละเครื่องเสมือน ตลอดจนแอปพลิเคชันที่อาจมีการคิดค่าลิขสิทธิ์แยกตามจำนวนการใช้ (Instance-Based Licensing) ซึ่งเป็นภาระเพิ่มเติมจากลิขสิทธิ์ของเครื่องหลัก

5) ข้อจำกัดด้านประสบการณ์ของนักพัฒนา

การพัฒนาเว็บแอปพลิเคชันภายในเครื่องเสมือนอาจมีข้อจำกัดในด้านประสบการณ์ของนักพัฒนา โดยเฉพาะเมื่อเทียบกับการพัฒนาบนเครื่องหลักโดยใช้โปรแกรมสำเร็จรูปในสภาพแวดล้อมการดังที่ได้กล่าวถึงในหัวข้อที่ 3 หรือการใช้เทคโนโลยีคอนเทนเนอร์ซึ่งจะได้กล่าวถึงในหัวข้อต่อไป ตัวอย่างของข้อจำกัดที่พบบ่อย ได้แก่ การตอบสนองของเครื่องมือพัฒนาที่ล่าช้าหรือไม่ราบรื่น การแสดงผลกราฟิกที่มีความหน่วง นอกจากนี้ยังอาจพบปัญหาในการทำงานร่วมกันระหว่างระบบปฏิบัติการโฮสต์และระบบปฏิบัติการเกสต์ เช่น การคัดลอก-วางข้อมูล (Copy-Paste) ข้ามระบบที่อาจไม่สมบูรณ์ เป็นต้น ข้อจำกัดเหล่านี้ส่งผลกระทบต่อความสะดวกในการพัฒนา ความต่อเนื่องของกระบวนการทำงาน และลดประสิทธิภาพการพัฒนาเว็บแอปพลิเคชันในภาพรวม

1.2 เทคโนโลยีคอนเทนเนอร์

เทคโนโลยีคอนเทนเนอร์เป็นการสร้างสภาพแวดล้อมที่แยกส่วนจากระบบคอมพิวเตอร์หลักในระดับระบบปฏิบัติการ (Nair, 2024) จากข้อจำกัดของการใช้เทคโนโลยีเครื่องเสมือนในสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันที่ได้กล่าวถึงก่อนหน้า การนำเทคโนโลยีคอนเทนเนอร์มาใช้สร้างกลไกผู้ให้บริการในเครื่องผู้พัฒนาจึง

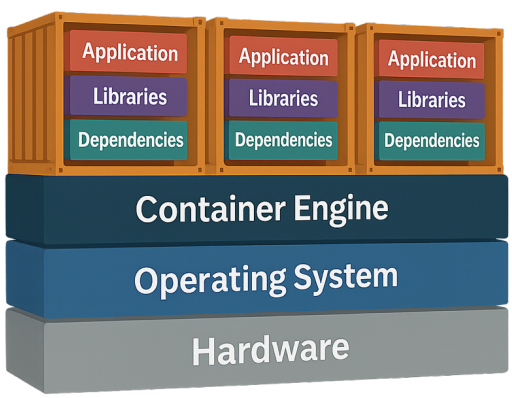
เป็นอีกทางเลือกหนึ่งที่ช่วยลดข้อจำกัดต่าง ๆ อีกทั้งเป็นเทคโนโลยีที่ได้รับความนิยมในการพัฒนาเว็บแอปพลิเคชันอีกด้วย เทคโนโลยีคอนเทนเนอร์มีความหมายและสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์ ดังนี้

1.2.1 ความหมายของเทคโนโลยีคอนเทนเนอร์

เทคโนโลยีคอนเทนเนอร์ (Container Technology) หมายถึง เทคโนโลยีที่ใช้ในการจัดการและใช้งานซอฟต์แวร์ในรูปแบบที่แยกตัวออกจากระบบปฏิบัติการโดยตรง โดยอาศัยกลไกการจำลองระดับระบบปฏิบัติการ (Operating System Virtualization) เพื่อสร้างหน่วยปฏิบัติการของซอฟต์แวร์ที่เรียกว่า คอนเทนเนอร์ (Container) ซึ่งเป็นหน่วยย่อยที่สามารถดำเนินการได้ด้วยตนเอง และมีความสามารถในการบรรจุส่วนประกอบของแอปพลิเคชันทั้งหมด เช่น รหัสคำสั่ง (Source Code) ไลบรารี (Library) และการพึ่งพาต่าง ๆ (Dependency) ไว้อย่างครบถ้วนในหน่วยเดียว (Susnjara & Smalley, 2024; Nutanix, 2024; DigitalOcean, 2024; ณัฐโชติ พรหมฤกษ์, 2563) จากความหมายข้างต้นกล่าวโดยสรุปได้ว่า เทคโนโลยีคอนเทนเนอร์เป็นเทคนิคการบรรจุซอฟต์แวร์หรือแอปพลิเคชันและทุกองค์ประกอบที่จำเป็นสำหรับการทำงานของแอปพลิเคชันนั้นไว้ในหน่วยเดียว โดยเรียกหน่วยดังกล่าวว่าคอนเทนเนอร์

1.2.2 สถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์

สถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์มีการแบ่งส่วนการทำงานออกเป็นระดับชั้น ประกอบด้วย ฮาร์ดแวร์ (Hardware) ระบบปฏิบัติการ (Operating System: OS) กลไกคอนเทนเนอร์ (Container Engine) และคอนเทนเนอร์ (Container) แสดงดังภาพที่ 4.2



ภาพที่ 4.2 สถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์

ที่มา: ปรับปรุงจาก Nutanix (2024) และณัฐโชติ พรหมฤกษ์ (2563)

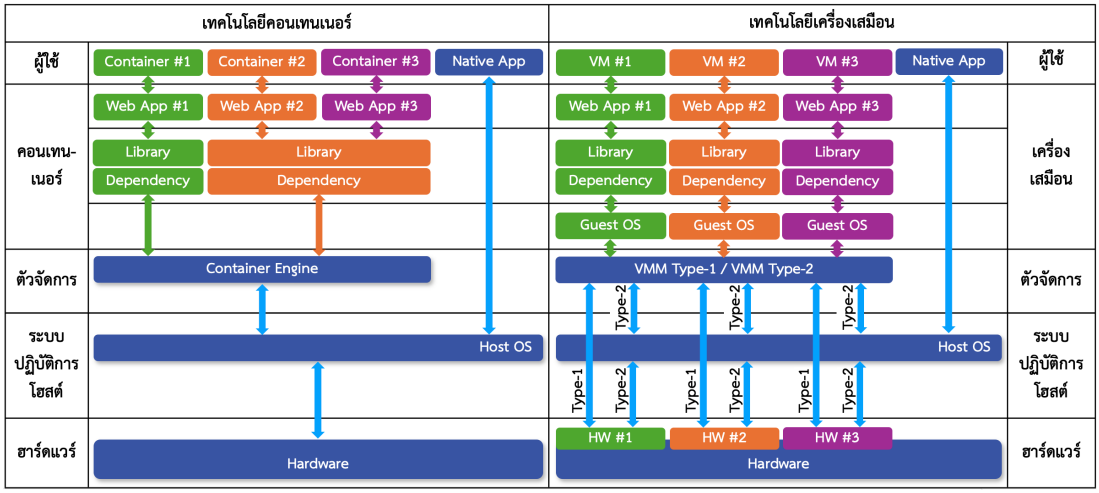
จากภาพที่ 4.2 แสดงถึงสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์ ซึ่งคอนเทนเนอร์สามารถทำงานได้หลายหน่วยพร้อมกันบนคอมพิวเตอร์เครื่องเดียวกัน และแต่ละหน่วยจะทำงานเป็นกระบวนที่แยกจากกัน (Isolated Process) ภายในแต่ละคอนเทนเนอร์จะบรรจุเฉพาะไฟล์รหัสคำสั่งของแอปพลิเคชัน ไลบรารี ไฟล์กำหนดค่า และไฟล์พึ่งพาที่แอปพลิเคชันจำเป็นต้องใช้โดยไม่มีเคอร์เนลของระบบปฏิบัติการ (OS Kernel) สำหรับ

การติดต่อกับฮาร์ดแวร์อยู่ภายในคอนเทนเนอร์ อย่างไรก็ตาม แต่ละคอนเทนเนอร์จะใช้เคอร์เนลของระบบปฏิบัติการโฮสต์ร่วมกันโดยอาศัยกลไกคอนเทนเนอร์ (Container Engine) ซึ่งเป็นซอฟต์แวร์ที่มีหน้าที่ในการบริหารจัดการและประสานการทำงานของคอนเทนเนอร์ทั้งหมด ด้วยเหตุที่คอนเทนเนอร์ไม่มีเคอร์เนลของระบบปฏิบัติการอยู่ภายในนี้เอง ทำให้คอนเทนเนอร์มีขนาดเล็กและเริ่มต้นการทำงานได้รวดเร็ว นอกจากนี้ ถ้าองค์ประกอบในคอนเทนเนอร์ใดมีการพึ่งพาเคอร์เนลของระบบปฏิบัติการ Linux ในการประมวลผล จะเรียกคอนเทนเนอร์นั้นว่าคอนเทนเนอร์ Linux (Linux Container) และถ้าองค์ประกอบในคอนเทนเนอร์พึ่งพาเคอร์เนลของระบบปฏิบัติการ Windows ในการประมวลผล ก็จะเรียกคอนเทนเนอร์นั้นว่าคอนเทนเนอร์ Windows (Windows Container)

กลไกคอนเทนเนอร์จะอาศัยความสามารถของเคอร์เนลในระบบปฏิบัติการโฮสต์ เพื่อแยกกระบวนการและควบคุมการเข้าถึงฮาร์ดแวร์ของเครื่องคอมพิวเตอร์อย่างมีประสิทธิภาพและปลอดภัยโดยไม่เกิดการรบกวนกันระหว่างคอนเทนเนอร์แต่ละหน่วย ตัวอย่างของความสามารถของเคอร์เนลในระบบปฏิบัติการโฮสต์ เช่น ความสามารถของ namespaces และ cgroups ที่มีอยู่ในระบบปฏิบัติการ Linux หรือความสามารถของ silos และ job objects ในระบบปฏิบัติการ Microsoft Windows เป็นต้น

1.2.3 การเปรียบเทียบสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์และเทคโนโลยีเครื่องเสมือน

การเปรียบเทียบสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์และเทคโนโลยีเครื่องเสมือนจะทำให้เห็นถึงความแตกต่างของเทคโนโลยีทั้งสองเทคโนโลยี โดยแสดงดังภาพที่ 4.3



ภาพที่ 4.3 การเปรียบเทียบสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์และเทคโนโลยีเครื่องเสมือน

ที่มา: ปรับปรุงจาก Microsoft (2025)

จากภาพที่ 4.3 แสดงการเปรียบเทียบโครงสร้างสถาปัตยกรรมของเทคโนโลยีคอนเทนเนอร์ (Container Technology) และเทคโนโลยีเครื่องเสมือน (Virtual Machine Technology) ซึ่งเป็นแนวทางในการจำลองสภาพแวดล้อมการทำงานที่นิยมใช้ในระบบคอมพิวเตอร์สมัยใหม่

ในฝั่งซ้ายของภาพที่ 4.3 แสดงถึงเครื่องคอมพิวเตอร์ของผู้พัฒนาที่ใช้เทคโนโลยีคอนเทนเนอร์ ซึ่งแต่ละคอนเทนเนอร์ (Container #1, Container #2 และ Container #3) ประกอบไปด้วยเว็บแอปพลิเคชัน (หรืออาจจะเป็นแอปพลิเคชันลักษณะอื่น) ไลบรารีและไฟล์พึ่งพาโดยไม่มีระบบปฏิบัติการภายในของตนเอง แต่จะใช้เคอร์เนลของระบบปฏิบัติการโฮสต์ร่วมกันผ่านซอฟต์แวร์ที่เรียกว่ากลไกคอนเทนเนอร์ (Container Engine) เช่น Docker หรือ containerd เป็นต้น ด้วยเหตุนี้ทำให้คอนเทนเนอร์มีขนาดเบา ใช้ทรัพยากรน้อย และเริ่มทำงานได้อย่างรวดเร็วเนื่องจากไม่มีการเริ่มต้นการทำงานของระบบปฏิบัติการภายในคอนเทนเนอร์

ในขณะที่ฝั่งขวาของภาพที่ 4.3 แสดงถึงเครื่องคอมพิวเตอร์ของผู้พัฒนาที่ใช้เทคโนโลยีเครื่องเสมือน โดยแต่ละเครื่องเสมือน (VM #1, VM #2 และ VM #3) จะมีระบบปฏิบัติการของตนเอง ทำให้มีลักษณะเป็นระบบคอมพิวเตอร์ที่สมบูรณ์ในตัวเอง โดย Hypervisor (VMM Type-1 หรือ Type-2) ทำหน้าที่เป็นตัวกลางในการจัดสรรทรัพยากรจากระบบปฏิบัติการโฮสต์หรือจากฮาร์ดแวร์โดยตรง เทคโนโลยีนี้มีความสามารถในการแยกอิสระได้อย่างสมบูรณ์ แต่ใช้ทรัพยากรระบบมากกว่าและเริ่มทำงานช้ากว่าเมื่อเทียบกับเทคโนโลยีคอนเทนเนอร์

นอกจากนี้ ในภาพที่ 4.3 ยังนำเสนอให้เห็นว่าเครื่องคอมพิวเตอร์ของผู้พัฒนาทั้งสองยังสามารถทำงานร่วมกับแอปพลิเคชันที่ติดตั้งบนระบบปฏิบัติการโฮสต์โดยตรง (Native Application: Native App) ได้อีกด้วย เพื่อสะท้อนบริบทของการใช้งานในเครื่องคอมพิวเตอร์ในฐานะผู้พัฒนาที่ใช้เทคโนโลยีในการแยกสภาพแวดล้อมและในฐานะผู้ใช้คอมพิวเตอร์ทั่วไปที่ยังใช้แอปพลิเคชันอื่น ๆ ได้ตามปกติ

1.2.4 ประโยชน์ของเทคโนโลยีคอนเทนเนอร์

การนำเทคโนโลยีคอนเทนเนอร์มาใช้ในงานด้านโครงสร้างพื้นฐานทางเทคโนโลยีสารสนเทศระบบคลาวด์ และการพัฒนาเว็บแอปพลิเคชันมีแนวโน้มเพิ่มสูงขึ้นอย่างต่อเนื่อง เนื่องจากมีจุดแข็งหลายประการเมื่อเทียบกับเทคโนโลยีเครื่องเสมือน โดยเฉพาะในด้านความยืดหยุ่น ความเร็ว และการใช้ทรัพยากรอย่างคุ้มค่า ประโยชน์ที่สำคัญของเทคโนโลยีคอนเทนเนอร์มีดังนี้ (Nutanix, 2024; DigitalOcean, 2024)

การใช้ทรัพยากรอย่างมีประสิทธิภาพ

คอนเทนเนอร์ไม่จำเป็นต้องติดตั้งระบบปฏิบัติการแยกภายในแต่ละหน่วยเหมือนเครื่องเสมือน จึงช่วยลดการใช้หน่วยประมวลผลกลาง หน่วยความจำหลัก และพื้นที่จัดเก็บข้อมูลได้อย่างมีประสิทธิภาพ สามารถสั่งให้คอนเทนเนอร์หลายหน่วยทำงานพร้อมกันบนคอมพิวเตอร์เครื่องเดียวได้โดยไม่เกิดการใช้ทรัพยากรสูงเหมือนเครื่องเสมือน ส่งผลให้เหมาะสมกับเครื่องคอมพิวเตอร์ระดับผู้ใช้งานทั่วไปและการใช้งานในคลาวด์แบบพื้นฐาน

1) ความรวดเร็วในการเริ่มต้นและหยุดการทำงาน

คอนเทนเนอร์สามารถเริ่มต้นการทำงานและหยุดการทำงานโดยใช้เวลาน้อย เนื่องจากไม่ต้องผ่านกระบวนการเริ่มต้นระบบปฏิบัติการแบบเต็มรูปแบบ ช่วยเพิ่มความคล่องตัวในการพัฒนา ทดสอบและปรับใช้งานระบบโดยเฉพาะในสภาพแวดล้อมที่ต้องการการเปลี่ยนแปลงอย่างรวดเร็ว

2) ความสามารถในการพกพาและใช้งานข้ามสภาพแวดล้อม

จุดเด่นที่สำคัญของสถาปัตยกรรมเทคโนโลยีคอนเทนเนอร์คือคุณลักษณะด้านการพกพา (Portability) และการใช้งานข้ามสภาพแวดล้อม กล่าวคือ ภายในคอนเทนเนอร์แต่ละหน่วยจะมีเฉพาะไฟล์รหัส

คำสั่งของแอปพลิเคชัน และไฟล์ประกอบต่าง ๆ ที่จะทำให้แอปพลิเคชันทำงานได้เท่านั้น ซึ่งทั้งหมดที่กล่าวมานี้อยู่ภายในหน่วยคอนเทนเนอร์เดียวกัน มีทุกสิ่งพร้อมที่จะทำให้แอปพลิเคชันในคอนเทนเนอร์ทำงานได้ด้วยตนเอง และใช้ความสามารถของเคอร์เนลในระบบปฏิบัติการโฮสต์เพื่อติดต่อกับฮาร์ดแวร์ ทำให้สามารถนำคอนเทนเนอร์ไปติดตั้งและใช้งานในเครื่องคอมพิวเตอร์เครื่องอื่นหรือในสภาพแวดล้อมคลาวด์ (Cloud Environment) ที่มีกลไกคอนเทนเนอร์ติดตั้งอยู่ได้โดยไม่เกิดปัญหาความขัดแย้ง

คอนเทนเนอร์สามารถทำงานได้บนหลากหลายแพลตฟอร์ม ไม่ว่าจะเป็นในเครื่องพัฒนา เครื่องทดสอบ หรือระบบปฏิบัติการที่แตกต่างกัน โดยไม่จำเป็นต้องปรับแต่งรหัสคำสั่งหรือการตั้งค่าใหม่ ช่วยลดปัญหาความไม่เข้ากันของสภาพแวดล้อม (Environment Mismatch) และเพิ่มความมั่นใจว่าระบบจะทำงานได้เหมือนกันทุกที่

3) ความยืดหยุ่นในการปรับใช้งาน

เทคโนโลยีคอนเทนเนอร์สามารถสร้างคอนเทนเนอร์ใหม่เฉพาะในเวลาที่ต้องการใช้ได้ (Use Only What You Use) และกำหนดให้เริ่ม-หยุด-ทำลาย (Start-Stop-Destroy) คอนเทนเนอร์ได้โดยใช้เครื่องมือจัดการ เช่น Docker Desktop, Docker CLI หรือ Kubernetes เป็นต้น ช่วยให้สามารถจัดการระบบได้อย่างมีประสิทธิภาพทั้งในระดับบุคคลและองค์กร

4) ความปลอดภัยในการแยกสภาพแวดล้อม

แม้ว่าคอนเทนเนอร์จะใช้เคอร์เนลของระบบปฏิบัติการโฮสต์ร่วมกัน แต่สามารถแยกกระบวนการทำงานออกจากกันได้อย่างชัดเจน (Process-Level Isolation) โดยอาศัยความสามารถของ namespaces และ cgroups ซึ่งเป็นกลไกของระบบปฏิบัติการ Linux หรือ silos และ job objects ในระบบ Windows ช่วยลดผลกระทบที่เกิดจากคอนเทนเนอร์อื่นทำงานผิดพลาดหรือหยุดการทำงาน ความสามารถนี้ช่วยให้สามารถออกแบบระบบที่มีความทนทานสูง (Resilient Architecture) และสามารถฟื้นตัวได้รวดเร็วในกรณีเกิดความผิดพลาดในบางส่วนของระบบ อันเป็นการเพิ่มความทนทานของระบบ (Enhanced Fault Tolerance)

5) ความสามารถในการสร้างต้นแบบและจัดการรุ่น

เทคโนโลยีคอนเทนเนอร์ช่วยให้สามารถสร้างต้นแบบการติดตั้งระบบซอฟต์แวร์ในรูปแบบของไฟล์คำสั่งสำหรับสร้างต้นแบบที่ผู้สร้างสภาพแวดล้อมเป็นผู้สร้างขึ้น เช่น Dockerfile เป็นต้น ภายในไฟล์ประกอบด้วยคำสั่งและขั้นตอนในการติดตั้งแอปพลิเคชัน ไลบรารี และไฟล์พึ่งพาต่าง ๆ ที่จำเป็นต้องมีในคอนเทนเนอร์ ซึ่งไฟล์คำสั่งสำหรับสร้างต้นแบบจะมีการกำหนดขั้นตอนการติดตั้งแอปพลิเคชันและส่วนประกอบที่เกี่ยวข้องอย่างเป็นลำดับ

ผลลัพธ์จากการประมวลผลไฟล์คำสั่งสำหรับสร้างต้นแบบจะทำให้ได้ไฟล์ต้นแบบของคอนเทนเนอร์ที่เรียกว่าไฟล์อิมเมจ (Image File) ซึ่งเปรียบเสมือนเป็นพิมพ์เขียวในการสร้างบ้าน ในการสร้างคอนเทนเนอร์ก็จะสร้างจากไฟล์อิมเมจนี้ ทำให้คอนเทนเนอร์ที่สร้างขึ้นด้วยไฟล์อิมเมจเดียวกันจะมีคุณลักษณะเหมือนกันทุกครั้ง (Repeatable Image Build) นอกจากนี้ การกำหนดไฟล์อิมเมจให้มีการระบุรุ่น (Versioned Image) ยังช่วยในการย้อนกลับ (Rollback) หรือปรับใช้รุ่นต่าง ๆ ได้ง่าย และสามารถจัดเก็บในระบบจัดการอิมเมจ เช่น

Docker Hub หรือ GitLab Container Registry ได้อย่างเป็นระบบ ส่งผลให้การพัฒนา การทดสอบ และการนำส่งซอฟต์แวร์ (Deployment) มีความเป็นอัตโนมัติและควบคุมได้ดีขึ้น

6) การสนับสนุนกระบวนการ DevOps และวงจรการพัฒนาแบบต่อเนื่อง

เทคโนโลยีคอนเทนเนอร์ช่วยส่งเสริมวงจรการพัฒนาและการส่งมอบแบบต่อเนื่อง (Continuous Integration / Continuous Delivery: CI/CD) ซึ่งเป็นแนวปฏิบัติตามแนวทาง DevOps (มาจากคำว่า Development และ Operations) โดยสามารถสร้างสภาพแวดล้อมใหม่เพื่อพัฒนา ตรวจสอบ ทดสอบ และปรับใช้งานระบบได้อย่างรวดเร็วโดยไม่รบกวนระบบหลัก ช่วยให้ทีมพัฒนาและทีมปฏิบัติการทำงานร่วมกันได้อย่างมีประสิทธิภาพ

1.2.5 ข้อจำกัดของการใช้เทคโนโลยีคอนเทนเนอร์ในการพัฒนาเว็บแอปพลิเคชัน

แม้ว่าเทคโนโลยีคอนเทนเนอร์จะมีจุดเด่นในหลายด้านและได้รับความนิยมอย่างกว้างขวางในการพัฒนาเว็บแอปพลิเคชันสมัยใหม่ แต่ในทางปฏิบัติยังมีข้อจำกัดบางประการที่ควรพิจารณา โดยเฉพาะเมื่อใช้ในบริบทของเครื่องพัฒนาระดับบุคคล หรือทีมขนาดเล็กที่มีข้อจำกัดด้านทรัพยากร ความเชี่ยวชาญ หรือระบบพื้นฐานอื่น ๆ ข้อจำกัดที่สำคัญมีดังนี้

1) ความซับซ้อนในการตั้งค่าและการจัดการระบบ

แม้ว่าคอนเทนเนอร์จะสามารถใช้งานได้อย่างยืดหยุ่น แต่การเริ่มต้นใช้งานในระดับผู้พัฒนาอาจต้องทำความเข้าใจคำสั่งและโครงสร้างของไฟล์คำสั่งสำหรับสร้างต้นแบบ เช่น Dockerfile และไฟล์ Compose เป็นต้น รวมทั้งการสร้างระบบเครือข่ายเสมือนเพื่อเชื่อมต่อการทำงานคอนเทนเนอร์ต่าง ๆ และเชื่อมโยง (Map) หมายเลขพอร์ตของคอนเทนเนอร์และเครื่องหลัก ซึ่งอาจเป็นอุปสรรคสำหรับผู้เริ่มต้นหรือผู้ที่ไม่มีพื้นฐานด้านระบบ

2) ประเด็นด้านความมั่นคงปลอดภัย

เนื่องจากคอนเทนเนอร์จะทำงานร่วมกันภายใต้เคอร์เนลของระบบปฏิบัติการเดียวกัน หากมีการกำหนดสิทธิ์การเข้าถึง (Permission) ไม่เหมาะสม อาจนำไปสู่การเข้าถึงทรัพยากรร่วมโดยไม่ได้รับอนุญาต ถึงแม้จะสามารถใช้กลไก cgroups และ namespace ในการควบคุมได้ ก็ต้องอาศัยความเข้าใจเชิงเทคนิคนอกเหนือไปจากทักษะในการเขียนโปรแกรม

3) การบริโภคทรัพยากรของระบบเมื่อมีจำนวนมาก

แม้ว่าคอนเทนเนอร์แต่ละหน่วยจะมีขนาดเล็ก แต่เมื่อเปิดใช้งานหลายหน่วยพร้อมกันโดยไม่มีการจัดสรรทรัพยากรอย่างเหมาะสม อาจทำให้ระบบทำงานช้าลง โดยเฉพาะในเครื่องของผู้พัฒนาที่มีหน่วยความจำและการประมวลผลจำกัด เช่น ในการสั่งให้คอนเทนเนอร์หลายหน่วยพร้อมกันอาจทำให้มีการใช้หน่วยความจำหลักและหน่วยประมวลผลกลางมากขึ้นตามไปด้วย ทั้งนี้ก็ขึ้นอยู่กับคุณลักษณะของแอปพลิเคชันและองค์ประกอบต่าง ๆ ภายในคอนเทนเนอร์ รวมทั้งการตั้งค่าการจำกัดการใช้ทรัพยากรของกลไกคอนเทนเนอร์

4) คอนเทนเนอร์ไม่สามารถใช้งานข้ามระบบปฏิบัติการที่มีเคอร์เนลต่างกันได้โดยตรง

จากการที่เทคโนโลยีคอนเทนเนอร์ทำงานโดยอาศัยการจำลองในระดับระบบปฏิบัติการ ซึ่งคอนเทนเนอร์แต่ละหน่วยจะใช้เคอร์เนลร่วมกันกับระบบปฏิบัติการโฮสต์ ส่งผลให้ไม่สามารถเรียกใช้งานแอปพลิเคชันที่ต้องการเคอร์เนลของระบบปฏิบัติการที่แตกต่างกันได้โดยตรง ตัวอย่างเช่น ในกรณีที่เครื่องคอมพิวเตอร์ของผู้พัฒนามีระบบปฏิบัติการโฮสต์เป็น Windows หากผู้พัฒนาสร้างคอนเทนเนอร์ที่ภายในคอนเทนเนอร์มีการติดตั้งแอปพลิเคชัน Nginx ซึ่งถูกพัฒนาให้ใช้กับระบบปฏิบัติการ Linux เท่านั้น จากกรณีนี้ คอนเทนเนอร์ดังกล่าวจะไม่สามารถทำงานได้โดยตรง เนื่องจาก Nginx จำเป็นต้องใช้เคอร์เนลของ Linux ในการประมวลผล ในขณะที่ระบบปฏิบัติการโฮสต์คือ Windows (ไม่มีเคอร์เนลของ Linux อยู่ภายในระบบปฏิบัติการ) ดังนั้น คอนเทนเนอร์ที่ถูกสร้างขึ้นมาในระบบปฏิบัติการโฮสต์ที่เป็น Windows ก็จะไม่สามารถทำงานได้

สำหรับแนวทางการแก้ไขสามารถทำได้โดยการใช้เทคโนโลยีอย่าง Windows Subsystem for Linux (WSL) ติดตั้งในระบบปฏิบัติการโฮสต์ที่เป็น Windows เพื่อให้ระบบปฏิบัติการโฮสต์สามารถใช้งานเคอร์เนลของ Linux ได้ หรืออีกแนวหนึ่งคือการนำเทคโนโลยีเครื่องเสมือนมาใช้ร่วมกับเทคโนโลยีคอนเทนเนอร์

5) ปัญหาด้านการตรวจหาข้อผิดพลาด

การตรวจสอบและแก้ไขข้อผิดพลาด (Debugging) ภายในคอนเทนเนอร์อาจทำได้ยากกว่าการทำงานบนระบบทั่วไป เนื่องจากโครงสร้างไฟล์และกระบวนการต่าง ๆ ภายในคอนเทนเนอร์ถูกแยกออกจากเครื่องหลัก ผู้พัฒนาจึงต้องใช้คำสั่งเฉพาะ เช่น docker exec หรือ docker logs ในการเข้าถึงกระบวนการทำงานภายในคอนเทนเนอร์ หรือใช้เครื่องมือเพิ่มเติม เช่น Docker Desktop ซึ่งมีส่วนติดต่อผู้ใช้แบบกราฟิก (GUI) เพื่ออำนวยความสะดวกในการตรวจสอบ สังเกตการทำงาน และวิเคราะห์ปัญหาได้อย่างมีประสิทธิภาพมากขึ้น

6) ความเสี่ยงจากการสูญหายของข้อมูลภายในคอนเทนเนอร์

โดยปกติ ข้อมูลที่ถูกสร้างหรือแก้ไขภายในคอนเทนเนอร์จะถูกเก็บไว้ในระบบไฟล์ภายในของคอนเทนเนอร์นั้นเอง ซึ่งเมื่อคอนเทนเนอร์ถูกลบทิ้ง (Deleted) หรือสร้างขึ้นใหม่ ข้อมูลเหล่านั้นจะหายไปทันที ส่งผลให้ไม่สามารถเข้าถึงไฟล์ที่ใช้ในการพัฒนา เช่น ไฟล์รหัสคำสั่ง ฐานข้อมูล หรือไฟล์กำหนดค่าได้อีก แม้ในกรณีที่คอนเทนเนอร์ถูกหยุดการทำงานชั่วคราว (Stopped) โดยยังไม่ถูกลบทิ้ง ข้อมูลภายในคอนเทนเนอร์จะยังคงอยู่ แต่ไม่สามารถเข้าถึงจากระบบโฮสต์ได้โดยตรงจนกว่าจะสั่งให้คอนเทนเนอร์กลับมาทำงานอีกครั้ง ดังนั้นเพื่อป้องกันความเสี่ยงจากการสูญหายของข้อมูล ควรใช้เทคนิคการเชื่อมโยงไดเรกทอรีระหว่างคอนเทนเนอร์และระบบไฟล์ของเครื่องหลัก เช่น การใช้ Volume หรือ Bind Mount เพื่อให้ข้อมูลสามารถเข้าถึงได้ตลอดเวลาและคงอยู่แม้ว่าคอนเทนเนอร์จะถูกลบทิ้งไปแล้วก็ตาม

จากข้อจำกัดที่กล่าวถึงข้างต้น แม้ว่าเทคโนโลยีคอนเทนเนอร์จะมีข้อจำกัดบางประการโดยเฉพาะในด้านการตั้งค่าการทำงานของคอนเทนเนอร์ ความมั่นคงปลอดภัย หรือการจัดการข้อมูล แต่ด้วยคุณสมบัติด้านความยืดหยุ่น ความรวดเร็วในการทำงาน และประสิทธิภาพในการใช้ทรัพยากร คอนเทนเนอร์จึง

ยังคงเป็นเทคโนโลยีที่เหมาะสมสำหรับการสร้างกลไกผู้ให้บริการเว็บเพื่อการพัฒนาเว็บแอปพลิเคชัน และยังสอดคล้องกับคุณลักษณะของสภาพแวดล้อมการพัฒนาที่ได้กล่าวถึงมาแล้วในหัวข้อที่ 1.3 จุดเด่นที่สำคัญอีกประการหนึ่งคือ ความสามารถในการสร้างสภาพแวดล้อมที่ใกล้เคียงกับการใช้งานจริง ซึ่งช่วยลดความขัดแย้งของสภาพแวดล้อมระหว่างการพัฒนา การทดสอบ และการนำไปใช้ตามวงจรการพัฒนาระบบได้อย่างมีนัยสำคัญ อีกทั้งยังสามารถทำให้ทีมผู้พัฒนาหลายคนทำงานภายใต้สภาพแวดล้อมเดียวกันโดยไม่เกิดความแตกต่างของระบบ ซึ่งช่วยลดปัญหาที่เกิดจากที่ว่า “ทำงานในเครื่องหนึ่งได้ แต่ไม่ทำงานในอีกเครื่องหนึ่ง”

1.3 การเปรียบเทียบการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูป เทคโนโลยีเครื่องเสมือน และเทคโนโลยีคอนเทนเนอร์

แนวทางในการสร้างกลไกผู้ให้บริการเว็บในเครื่องคอมพิวเตอร์ของผู้พัฒนามีความหลากหลาย ทั้งการใช้โปรแกรมสำเร็จรูป การใช้เทคโนโลยีเครื่องเสมือนและการใช้เทคโนโลยีคอนเทนเนอร์ โดยแต่ละแนวทางมีจุดเด่นและข้อจำกัดที่แตกต่างกันไปตามลักษณะของระบบ ทรัพยากรของเครื่องคอมพิวเตอร์ และระดับความเชี่ยวชาญของผู้พัฒนา หัวข้อนี้จึงมุ่งเปรียบเทียบแนวทางทั้งสามในประเด็นสำคัญด้านโครงสร้าง ประสิทธิภาพการใช้งาน และความเหมาะสม เพื่อเป็นข้อมูลประกอบการตัดสินใจเลือกใช้งานเทคโนโลยีให้สอดคล้องกับบริบทของการพัฒนาเว็บแอปพลิเคชัน การเปรียบเทียบการสร้างกลไกผู้ให้บริการเว็บแสดงดังตารางที่ 4-1

ตารางที่ 4-1 ตารางเปรียบเทียบการสร้างกลไกผู้ให้บริการเว็บด้วยวิธีต่าง ๆ

ประเด็นการเปรียบเทียบ	โปรแกรมสำเร็จรูป	เทคโนโลยีเครื่องเสมือน	เทคโนโลยีคอนเทนเนอร์
ระดับของการแยกสภาพแวดล้อมการพัฒนาออกจากเครื่องหลัก	ทำไม่ได้	ทำได้ในระดับฮาร์ดแวร์	ทำได้ในระดับระบบปฏิบัติการ
โอกาสเกิดความขัดแย้งกับสภาพแวดล้อมการใช้งานจริง	ปานกลางถึงสูง	ต่ำ	ต่ำ
การแยกสภาพแวดล้อมสำหรับแต่ละโครงการพัฒนา	ยุ่งยาก	ทำได้	ทำได้
ความสามารถในการพกพา	ต่ำ	ปานกลาง	สูง
ความยืดหยุ่นในการปรับแต่ง	ต่ำ	สูง	สูง
ความเร็วในการเริ่มระบบ	เร็วมาก	ช้า	เร็ว

ตารางที่ 4-1 (ต่อ)

ประเด็นการเปรียบเทียบ	โปรแกรมสำเร็จรูป	เทคโนโลยีเครื่องเสมือน	เทคโนโลยีคอนเทนเนอร์
การใช้ทรัพยากร	ต่ำ	สูง	ต่ำถึงปานกลาง
ระดับความยากง่ายในการเริ่มต้นเรียนรู้และใช้งาน	ง่าย	ปานกลางถึงยาก	ปานกลางถึงยาก
แหล่งสนับสนุนการเรียนรู้และเอกสารประกอบ	ปานกลาง	สูง	สูง

จากตารางที่ 4-1 แสดงการเปรียบเทียบแนวทางการสร้างกลไกผู้ให้บริการเว็บด้วยโปรแกรมสำเร็จรูป เทคโนโลยีเครื่องเสมือน และเทคโนโลยีคอนเทนเนอร์ โดยมีประเด็นการเปรียบเทียบดังนี้

1.3.1 ระดับการแยกสภาพแวดล้อมการพัฒนาออกจากเครื่องหลัก

โปรแกรมสำเร็จรูปที่ใช้สร้างกลไกผู้ให้บริการเว็บ จะไม่สามารถแยกสภาพแวดล้อมการพัฒนาออกจากระบบของเครื่องหลักได้อย่างแท้จริง เนื่องจากโปรแกรมจะทำงานภายใต้ระบบปฏิบัติการของเครื่องผู้พัฒนาโดยตรง ในขณะที่เทคโนโลยีเครื่องเสมือนสามารถแยกสภาพแวดล้อมในระดับฮาร์ดแวร์ โดยสร้างระบบปฏิบัติการแยกออกจากเครื่องหลัก ส่วนเทคโนโลยีคอนเทนเนอร์ แม้จะยังใช้ระบบปฏิบัติการร่วมกับเครื่องหลัก แต่สามารถแยกพื้นที่การทำงานในระดับระบบปฏิบัติการได้อย่างมีประสิทธิภาพ ด้วยกลไกที่ช่วยให้คอนเทนเนอร์แต่ละหน่วยทำงานอย่างเป็นอิสระ ส่งผลให้สามารถควบคุมสภาพแวดล้อมการพัฒนาได้ยืดหยุ่นและปลอดภัยยิ่งขึ้น

1.3.2 โอกาสเกิดความขัดแย้งกับสภาพแวดล้อมการใช้งานจริง

ความขัดแย้งระหว่างสภาพแวดล้อมการพัฒนาและสภาพแวดล้อมการใช้งานจริงเป็นปัจจัยสำคัญที่ส่งผลต่อความสำเร็จในระยะการนำไปใช้ของวงจรการพัฒนาระบบ การใช้โปรแกรมสำเร็จรูปในการสร้างกลไกผู้ให้บริการเว็บจะขึ้นอยู่กับแพลตฟอร์มของเครื่องผู้พัฒนา เช่น ระบบปฏิบัติการ รุ่นของซอฟต์แวร์ และการตั้งค่าต่าง ๆ ซึ่งอาจแตกต่างจากเครื่องใช้งานจริง ประกอบกับไม่สามารถสร้างสภาพแวดล้อมการพัฒนาแยกออกจากเครื่องผู้พัฒนาได้ ส่งผลให้เว็บแอปพลิเคชันที่ทำงานได้ดีในเครื่องพัฒนาอาจเกิดข้อผิดพลาดเมื่อย้ายไปใช้งานจริง เทคโนโลยีเครื่องเสมือนสามารถจำลองสภาพแวดล้อมที่ใกล้เคียงกับระบบจริงได้มากที่สุด เนื่องจากการเครื่องเสมือนใช้ระบบปฏิบัติการแยกออกจากเครื่องหลัก

ในขณะที่เทคโนโลยีคอนเทนเนอร์สามารถระบุเวอร์ชันของซอฟต์แวร์และการตั้งค่าต่าง ๆ ไว้ชัดเจนในไฟล์อิมเมจของคอนเทนเนอร์ ทำให้ลดความเสี่ยงจากความไม่สอดคล้องขององค์ประกอบต่าง ๆ แต่ยังต้องคำนึงถึงความเข้ากันได้ของระบบปฏิบัติการระหว่างสภาพแวดล้อมการพัฒนาและการใช้งานจริง เช่น ถ้าเครื่องที่จะใช้ติดตั้งจริงเป็นระบบปฏิบัติการ Linux คอนเทนเนอร์ที่ใช้ก็ต้องเป็น Linux คอนเทนเนอร์ด้วย

1.3.3 การแยกสภาพแวดล้อมสำหรับแต่ละโครงการพัฒนา

ผู้พัฒนาอาจมีการพัฒนาเว็บแอปพลิเคชันหลายโครงการในเครื่องผู้พัฒนาเดียวกัน ซึ่งแต่ละโครงการอาจมีความต้องการขององค์ประกอบด้านซอฟต์แวร์ที่แตกต่างกันออกไป เช่น เวอร์ชันของภาษาประมวลผล โปรแกรมให้บริการเว็บ หรือระบบจัดการฐานข้อมูลที่แตกต่างกัน โปรแกรมสำเร็จรูปมักไม่เอื้อต่อการแยกสภาพแวดล้อมดังกล่าวอย่างชัดเจน อาจทำให้เกิดปัญหาเรื่องโฟลเดอร์ทำงาน พอร์ตการเชื่อมต่อทางซอฟต์แวร์ หรือความขัดแย้งของการตั้งค่าระบบระหว่างแต่ละโครงการ

ในขณะที่เทคโนโลยีเครื่องเสมือนสามารถสร้างเครื่องเสมือนแยกตามแต่ละโครงการได้ ทำให้แต่ละโครงการมีระบบปฏิบัติการและการกำหนดค่าของตนเองอย่างสมบูรณ์ แต่ก็จะใช้ทรัพยากรของเครื่องหลักในการสร้างเครื่องเสมือนมากตามไปด้วย ส่วนเทคโนโลยีคอนเทนเนอร์มีจุดเด่นด้านความคล่องตัวในการจัดการหลายโครงการ โดยสามารถสร้างและควบคุมคอนเทนเนอร์แยกต่างหากสำหรับแต่ละโครงการได้อย่างอิสระ โดยใช้ทรัพยากรของเครื่องที่น้อยกว่าเทคโนโลยีเครื่องเสมือน ช่วยให้ผู้พัฒนาสามารถทดสอบ แก้ไข และสั่งให้เว็บแอปพลิเคชันของแต่ละโครงการทำงานพร้อมกันได้อย่างไม่รบกวนกัน

1.3.4 ความสามารถในการพกพา

ความสามารถในการพกพา หมายถึง ความสะดวกในการย้ายสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันจากเครื่องหนึ่งไปยังอีกเครื่องหนึ่ง หรือจากเครื่องพัฒนาไปสู่เครื่องใช้งานจริงได้อย่างราบรื่น โปรแกรมสำเร็จรูปมักมีข้อจำกัดในด้านนี้ เนื่องจากการติดตั้งขึ้นอยู่กับระบบปฏิบัติการและการกำหนดค่าบนเครื่องพัฒนา ทำให้ไม่สามารถย้ายชุดโปรแกรมไปใช้งานในเครื่องอื่นได้โดยตรง ต้องอาศัยการติดตั้งใหม่และกำหนดค่าระบบใหม่ทุกครั้ง เทคโนโลยีเครื่องเสมือนมีความสามารถในการพกพาที่ดีกว่า โดยสามารถสร้างอิมเมจของเครื่องเสมือนที่ประกอบด้วยระบบปฏิบัติการและซอฟต์แวร์ทั้งหมด แล้วนำไปใช้งานในเครื่องอื่นที่รองรับ Hypervisor เดียวกันได้ ในขณะที่เทคโนโลยีคอนเทนเนอร์มีจุดเด่นที่สุดในด้านนี้ เนื่องจากสามารถจัดเก็บทุกองค์ประกอบของระบบในรูปแบบไฟล์อิมเมจ และนำไปใช้งานข้ามแพลตฟอร์มได้อย่างสะดวกโดยไม่ต้องพึ่งพา Hypervisor

1.3.5 ความยืดหยุ่นในการปรับแต่ง

ความยืดหยุ่นในการปรับแต่ง หมายถึง ความสามารถในการแก้ไของค์ประกอบหรือกำหนดค่าต่าง ๆ ของกลไกผู้ให้บริการเว็บให้เหมาะสมกับความต้องการเฉพาะของแต่ละโครงการ โปรแกรมสำเร็จรูปมักมีข้อจำกัดด้านนี้ เนื่องจากองค์ประกอบต่าง ๆ ถูกกำหนดไว้ล่วงหน้าจากผู้พัฒนาโปรแกรมสำเร็จรูปนั้น ๆ ในขณะที่เทคโนโลยีเครื่องเสมือนมีความยืดหยุ่นสูง เพราะสามารถติดตั้งระบบปฏิบัติการและซอฟต์แวร์ได้ตามต้องการเช่นเดียวกับเครื่องจริง แต่ก็อาจต้องใช้เวลาในการติดตั้งและปรับแต่งระบบมากขึ้น

เทคโนโลยีคอนเทนเนอร์มีจุดเด่นด้านการกำหนดค่าที่ชัดเจนและควบคุมได้ผ่านไฟล์อิมเมจ ซึ่งช่วยให้สามารถปรับแต่งค่ากำหนดต่าง ๆ ได้อย่างเป็นระบบ นำไปสร้างสภาพแวดล้อมลักษณะเดียวกันซ้ำได้ และควบคุมการเปลี่ยนแปลงในแต่ละเวอร์ชันได้อย่างมีประสิทธิภาพ โดยเฉพาะในกระบวนการพัฒนาแบบทีมที่ต้องการความสอดคล้องของสภาพแวดล้อม

1.3.6 ความเร็วในการเริ่มระบบ

ความเร็วในการเริ่มระบบ หมายถึง ระยะเวลาที่ใช้ในการเปิดใช้งานกลไกผู้ให้บริการเว็บจากสถานะปิดหรือยังไม่พร้อมใช้งาน ไปสู่สถานะที่สามารถให้บริการเว็บแอปพลิเคชันได้อย่างสมบูรณ์ โปรแกรมสำเร็จรูปมักมีความเร็วในการเริ่มระบบสูง เนื่องจากติดตั้งมาในลักษณะแพ็กเกจเดียวที่สามารถเปิดใช้งานได้ทันทีผ่านส่วนควบคุมแบบกราฟิก อย่างไรก็ตาม อาจมีข้อจำกัดในการรองรับการทำงานหลายระบบพร้อมกัน เทคโนโลยีเครื่องเสมือนต้องใช้เวลาในการเริ่มระบบมากที่สุด เนื่องจากต้องบูตระบบปฏิบัติการในเครื่องเสมือนก่อนที่จะเริ่มการทำงานองค์ประกอบของเว็บแอปพลิเคชันได้ ซึ่งอาจใช้เวลาหลายวินาทีถึงนาที ขึ้นอยู่กับขนาดของเครื่องเสมือนและทรัพยากรของเครื่องหลัก

เทคโนโลยีคอนเทนเนอร์มีจุดเด่นด้านความเร็วในการเริ่มต้น เนื่องจากไม่มีขั้นตอนการบูตระบบปฏิบัติการ และสามารถสั่งให้บริการต่าง ๆ ภายในคอนเทนเนอร์เริ่มทำงานได้ทันทีคล้ายกับโปรแกรมสำเร็จรูป จึงเหมาะสมกับงานที่ต้องการการเปิด-ปิดระบบบ่อยครั้ง หรืองานที่ต้องการปรับแต่งขนาดของระบบ (Scale) อย่างรวดเร็ว

1.3.7 การใช้ทรัพยากร

การใช้ทรัพยากร หมายถึง ปริมาณการใช้หน่วยประมวลผลกลาง หน่วยความจำหลัก พื้นที่จัดเก็บข้อมูล และฮาร์ดแวร์อื่น ๆ ของเครื่องพัฒนาในการทำงานของกลไกผู้ให้บริการเว็บ โปรแกรมสำเร็จรูปมักใช้ทรัพยากรน้อยที่สุด เนื่องจากติดตั้งและทำงานภายใต้ระบบปฏิบัติการของเครื่องพัฒนาโดยตรง โดยไม่มีการจำลองหรือแยกระบบเพิ่มเติม เทคโนโลยีเครื่องเสมือนใช้ทรัพยากรมากที่สุด เพราะต้องแบ่งทรัพยากรจากเครื่องผู้พัฒนาให้กับระบบปฏิบัติการของเครื่องเสมือนในแต่ละเครื่อง ซึ่งอาจส่งผลให้เครื่องพัฒนาทำงานช้าลงเมื่อมีการเปิดใช้งานเครื่องเสมือนหลายเครื่องพร้อมกัน โดยเฉพาะเมื่อเครื่องผู้พัฒนาเปิดใช้งานโปรแกรมอื่นควบคู่ไปด้วย เช่น โปรแกรมเขียนรหัสคำสั่งหรือโปรแกรมตกแต่งภาพ ซึ่งล้วนใช้ทรัพยากรร่วมกันทั้งหมด

เทคโนโลยีคอนเทนเนอร์มีความได้เปรียบด้านนี้ เนื่องจากสามารถแยกกระบวนการทำงานโดยใช้เคอร์เนลร่วมกับระบบปฏิบัติการโฮสต์ ทำให้ใช้ทรัพยากรน้อยกว่าเครื่องเสมือน และสามารถสั่งให้คอนเทนเนอร์หลายหน่วยทำงานพร้อมกันได้อย่างมีประสิทธิภาพ จึงเหมาะสมกับเครื่องพัฒนาที่มีทรัพยากรจำกัดหรือจำเป็นต้องเปิดหลายบริการพร้อมกัน

1.3.8 รายการความรู้พื้นฐานที่จำเป็นสำหรับการเริ่มต้น

การเลือกใช้เทคโนโลยีเพื่อสร้างกลไกผู้ให้บริการเว็บในเครื่องพัฒนามีความสัมพันธ์กับความรู้พื้นฐานของผู้พัฒนาอย่างมีนัยสำคัญ เนื่องจากแต่ละแนวทางมีระดับความซับซ้อนที่แตกต่างกัน โปรแกรมสำเร็จรูปเหมาะสำหรับผู้เริ่มต้นที่มีพื้นฐานทั่วไปเกี่ยวกับการใช้งานคอมพิวเตอร์ เช่น การติดตั้งโปรแกรม การจัดการโฟลเดอร์ และการใช้งานอินเทอร์เฟซแบบกราฟิก โดยไม่จำเป็นต้องมีความรู้ด้านระบบปฏิบัติการหรือเครือข่ายเชิงลึก ขณะที่เทคโนโลยีเครื่องเสมือนต้องการความรู้พื้นฐานด้านโครงสร้างระบบคอมพิวเตอร์ การติดตั้งระบบปฏิบัติการ การกำหนดค่าระบบ และความเข้าใจด้านการจัดการเครือข่ายในระดับหนึ่ง

เทคโนโลยีคอนเทนเนอร์ ผู้พัฒนาควรมีความเข้าใจเกี่ยวกับแนวคิดของ container การจัดการคำสั่งผ่านบรรทัดคำสั่ง (command-line interface) การกำหนดค่าด้วยไฟล์ เช่น Dockerfile หรือ docker-compose.yml และคำสั่งพื้นฐานในการจัดการไฟล์และโฟลเดอร์ในระบบปฏิบัติการ Linux ซึ่งถือเป็นพื้นฐานสำคัญในการบริหารจัดการคอนเทนเนอร์ให้มีประสิทธิภาพ

1.3.9 ระดับความยากง่ายในการเริ่มต้นเรียนรู้และใช้งาน

ระดับความยากง่ายในการเริ่มต้นเรียนรู้และใช้งานของเทคโนโลยีแต่ละประเภทในการสร้างกลไกผู้ให้บริการเว็บมีความสัมพันธ์กับความรู้พื้นฐานและทักษะของผู้พัฒนาอย่างมีนัยสำคัญ โปรแกรมสำเร็จรูปเหมาะสำหรับผู้เริ่มต้นที่มีความรู้ทั่วไปเกี่ยวกับการใช้งานคอมพิวเตอร์ เช่น การติดตั้งโปรแกรม การจัดการโฟลเดอร์ และการใช้งานส่วนติดต่อผู้ใช้แบบกราฟิกโดยอาจไม่จำเป็นต้องเข้าใจระบบปฏิบัติการหรือคำสั่งเชิงเทคนิค ในขณะที่เทคโนโลยีเครื่องเสมือนต้องการความเข้าใจในระดับที่สูงขึ้น เช่น การติดตั้งระบบปฏิบัติการ การจัดการทรัพยากร และการตั้งค่าเครือข่ายเบื้องต้น

เทคโนโลยีคอนเทนเนอร์ต้องการความเข้าใจเชิงแนวคิดเกี่ยวกับ container การสั่งงานกลไกคอนเทนเนอร์ผ่านบรรทัดคำสั่ง (Command-Line: CLI) การใช้ไฟล์อิมเมจ และการจัดการไฟล์ในระบบปฏิบัติการ Linux ซึ่งแม้จะมีระดับความยากเริ่มต้นสูงกว่าทางเลือกอื่น แต่ก็ให้ความยืดหยุ่นและเป็นเทคโนโลยีที่ได้รับความนิยมในอุตสาหกรรมการพัฒนาเว็บแอปพลิเคชันในปัจจุบัน โดยเฉพาะในงานที่เกี่ยวข้องกับระบบคลาวด์ การพัฒนาตามแนวทาง DevOps และการส่งมอบซอฟต์แวร์อย่างต่อเนื่อง ดังนั้น การพัฒนาทักษะด้านนี้จึงไม่เพียงตอบโจทย์การเรียนรู้เท่านั้น แต่ยังเป็นการเตรียมความพร้อมสำหรับการทำงานในสายวิชาชีพด้านเทคโนโลยีสารสนเทศที่เปลี่ยนแปลงอย่างรวดเร็ว

1.3.10 แหล่งสนับสนุนการเรียนรู้และเอกสารประกอบ

การเข้าถึงแหล่งสนับสนุนการเรียนรู้และเอกสารประกอบเป็นปัจจัยสำคัญที่ช่วยสนับสนุนการเริ่มต้นใช้งานเทคโนโลยีในการสร้างกลไกผู้ให้บริการเว็บได้อย่างมีประสิทธิภาพ โปรแกรมสำเร็จรูปมักมีเอกสารประกอบในรูปแบบคู่มือผู้ใช้จากผู้ผลิต วิดีโอแนะนำการใช้งาน และกระดานถามตอบจากผู้ใช้ทั่วไปในระดับเบื้องต้น ซึ่งเพียงพอสำหรับผู้เริ่มต้นที่ต้องการเรียนรู้ด้วยตนเองโดยไม่จำเป็นต้องมีความรู้เชิงลึก แหล่งสนับสนุนการเรียนรู้สำหรับเทคโนโลยีเครื่องเสมือน มักอยู่ในรูปของบทความเชิงเทคนิค หนังสือคู่มือ และเอกสารจากผู้ผลิตซอฟต์แวร์ ซึ่งเนื้อหามีความละเอียดและเหมาะสำหรับผู้ใช้ที่มีประสบการณ์ด้านระบบปฏิบัติการ

เทคโนโลยีคอนเทนเนอร์ โดยเฉพาะ Docker มีแหล่งข้อมูลและเอกสารสนับสนุนจำนวนมาก ทั้งจากเว็บไซต์อย่างเป็นทางการของ Docker เอกสารบนแพลตฟอร์ม GitHub และคอร์สออนไลน์จากผู้ให้บริการด้านเทคโนโลยีชั้นนำ อีกทั้งยังมีชุมชนนักพัฒนาและเอกสารแบบเปิดที่อัปเดตอยู่เสมอ ซึ่งช่วยให้ผู้พัฒนาสามารถค้นคว้า แก้ไขปัญหา และเรียนรู้เพิ่มเติมได้อย่างต่อเนื่อง

2. หลักการของ Docker Desktop

จากการเปรียบเทียบแนวทางการสร้างกลไกผู้ให้บริการเว็บด้วยวิธีการต่าง ๆ จะเห็นว่าเทคโนโลยีคอนเทนเนอร์มีข้อได้เปรียบหลายประการ ทั้งด้านการแยกสภาพแวดล้อมการพัฒนาออกจากเครื่องหลัก การลดภาระของระบบและการรองรับโครงการพัฒนาหลายระบบพร้อมกัน หัวข้อนี้จึงนำเสนอการใช้ Docker ซึ่งเป็นแพลตฟอร์มเทคโนโลยีคอนเทนเนอร์ที่ได้รับการยอมรับในอุตสาหกรรมซอฟต์แวร์ และรองรับการใช้งานบนระบบปฏิบัติการหลัก ได้แก่ Windows, macOS และ Linux

ในหนังสือเล่มนี้จะนำเสนอการใช้งาน Docker Desktop บนระบบปฏิบัติการ Windows เพื่อให้สอดคล้องกับหัวข้อก่อนหน้าที่กล่าวถึงการใช้โปรแกรมสำเร็จรูป และเพื่อลดความซับซ้อนจากการสั่งงานผ่านบรรทัดคำสั่ง อย่างไรก็ตาม ผู้ใช้ระบบปฏิบัติการ macOS และ Linux ก็สามารถประยุกต์ใช้แนวทางเดียวกันได้ เนื่องจาก Docker Desktop มีหลักการทำงาน ส่วนติดต่อผู้ใช้แบบกราฟิก และรูปแบบการจัดการที่คล้ายคลึงกันในทุกแพลตฟอร์ม

2.1 ความรู้เบื้องต้นเกี่ยวกับ Docker

Docker เป็นแพลตฟอร์มแบบเปิด (Open Platform) ที่ออกแบบมาเพื่อรองรับกระบวนการพัฒนา (Develop) ส่งมอบ (Ship) และเรียกใช้งานแอปพลิเคชัน (Run) อย่างมีประสิทธิภาพโดยใช้เทคโนโลยีคอนเทนเนอร์ ซึ่งช่วยให้สามารถแยกการทำงานของแอปพลิเคชันออกจากโครงสร้างพื้นฐานได้อย่างชัดเจน ส่งผลให้การจัดการซอฟต์แวร์เป็นไปอย่างคล่องตัว ลดระยะเวลาระหว่างการพัฒนาโปรแกรม (Development) และการนำไปใช้งานจริง (Production) ตลอดจนสามารถจัดการโครงสร้างพื้นฐานในลักษณะเดียวกับการจัดการแอปพลิเคชัน (Stoneman, 2020; Docker Inc., n.d.c) โดย Docker เปิดตัวครั้งแรกในปี พ.ศ. 2556 ซึ่งพัฒนาต่อยอดจากเทคโนโลยี Linux Containers หรือ LXC และในเวลาต่อมาได้พัฒนาเป็นไลบรารีของตนเอง ส่งผลให้เทคโนโลยีคอนเทนเนอร์ได้รับความนิยมอย่างแพร่หลายจนถึงปัจจุบัน (Randal, 2020; Stack Overflow, 2024)

2.2 การติดตั้งโปรแกรม Docker Desktop

Docker Desktop เป็นแอปพลิเคชันที่อำนวยความสะดวกให้แก่ผู้พัฒนาในการจัดการคอนเทนเนอร์ซึ่งสามารถติดตั้งได้ทั้งในระบบปฏิบัติการ Microsoft Windows, macOS และ Linux โดยก่อนการติดตั้งผู้พัฒนาจะต้องตรวจสอบเครื่องคอมพิวเตอร์ที่จะติดตั้งก่อน สำหรับคุณสมบัติที่สำคัญของคอมพิวเตอร์ที่จะใช้ในการติดตั้ง Docker Desktop แสดงดังตารางที่ 4-2 (Docker Inc., n.d.a)

ตารางที่ 4-2 คุณสมบัติขั้นพื้นฐานที่สำคัญของเครื่องคอมพิวเตอร์ที่จะติดตั้ง Docker Desktop

ระบบปฏิบัติการ	คุณสมบัติขั้นต่ำของเครื่องคอมพิวเตอร์ที่ใช้ติดตั้ง
Microsoft Windows	• Windows 11 และ Windows 10 แบบ 64 บิต รุ่นHome, Pro, Enterprise หรือ Education เวอร์ชัน 22H2 ขึ้นไป

ตารางที่ 4-2 (ต่อ)

ระบบปฏิบัติการ	คุณสมบัติขั้นต่ำของเครื่องคอมพิวเตอร์ที่ใช้ติดตั้ง
	• หน่วยประมวลผลกลางแบบ 64 บิตที่รองรับเทคโนโลยี Second Level Address Translation (SLAT) • หน่วยความจำหลักอย่างน้อย 4 GB • เปิดการใช้งาน Virtualization ใน BIOS/UEFI • รองรับทั้ง WSL และ Hyper-V
macOS	• ต้องเป็น macOS เวอร์ชันที่รองรับ โดยตรวจสอบได้จากเว็บไซต์ของ Docker โดยตรง • หน่วยความจำหลักอย่างน้อย 4 GB
Linux	• หน่วยประมวลผลแบบ 64 บิต ที่รองรับ Virtualization • ระบบปฏิบัติการที่มีเคอร์เนลรองรับ Virtualization • QEMU เวอร์ชัน 5.2 ขึ้นไป • หน่วยความจำหลักอย่างน้อย 4 GB • สภาพแวดล้อมของ Desktop รองรับเฉพาะ GNOME, KDE หรือ MATE • ต้องติดตั้งในระบบปฏิบัติการ Linux โดยตรงเท่านั้น (Native Environment)

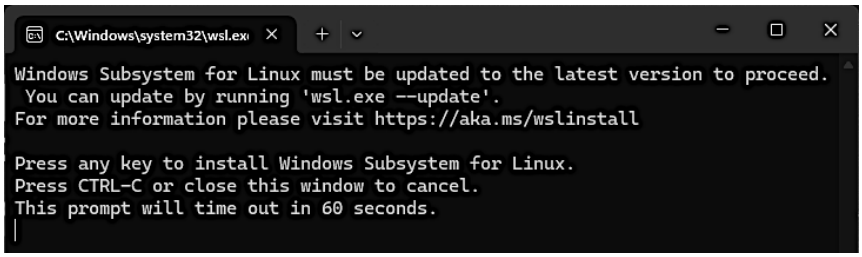
หลังจากที่ได้ตรวจสอบคุณสมบัติที่จำเป็นของเครื่องคอมพิวเตอร์ที่จะติดตั้งแล้ว การติดตั้งและการใช้งาน Desktop มีขั้นตอนที่สำคัญดังต่อไปนี้

ขั้นตอนที่ 1) ดาวน์โหลดไฟล์สำหรับการติดตั้ง Docker Desktop

เริ่มต้นจากการเข้าไปยังเว็บไซต์อย่างเป็นทางการของ Docker โดยเข้าถึงได้จาก https://docs.docker.com/get-started/get-docker/ จากนั้นเลือกระบบปฏิบัติการโฮสต์ที่ใช้งานอยู่ และคลิกลิงก์สำหรับดาวน์โหลดไฟล์

ขั้นตอนที่ 2) เรียกใช้งานไฟล์ติดตั้ง

เรียกใช้งานไฟล์ที่ดาวน์โหลดและทำตามขั้นตอนที่ปรากฏบนหน้าจอ ในกรณีที่ทำการติดตั้ง Docker Desktop ในระบบปฏิบัติการ Microsoft Windows หากยังไม่มีการติดตั้ง WSL 2 จะมีการเสนอให้ติดตั้ง WSL 2 โดยมีหน้าต่างแสดงข้อความดังภาพที่ 4.7 ให้กดปุ่มใดๆ ในหน้าจอดังกล่าวเพื่อยอมรับการติดตั้ง WSL 2 โดยอัตโนมัติ



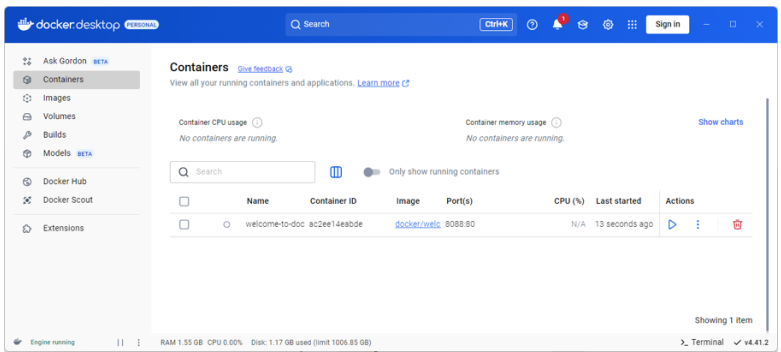
ภาพที่ 4.4 หน้าจอสอบถามการติดตั้ง WSL 2

ขั้นตอนที่ 3) เสร็จสิ้นการติดตั้ง

หลังจากที่ติดตั้งเสร็จสิ้นแล้ว อาจมีการแจ้งให้ทำการเริ่มต้นระบบใหม่ (Restart) ในขั้นตอนนี้ให้ทำการเริ่มต้นระบบใหม่เป็นอันเสร็จสิ้นการติดตั้ง

ขั้นตอนที่ 4) เปิดและทดสอบโปรแกรม

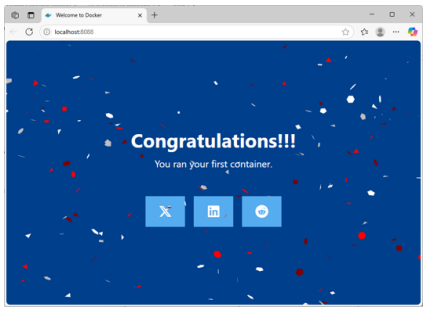
ทำการเปิดโปรแกรม Docker Desktop ที่ได้ติดตั้ง ซึ่งก่อนเข้าใช้โปรแกรมอาจมีการให้สร้างบัญชีผู้ใช้สำหรับการเข้าถึงระบบ Docker Hub ซึ่งเป็น Docker Registry สาธารณะของ Docker รวมทั้งการสำรวจการใช้งานซึ่งสามารถข้ามขั้นตอนนี้ได้ จากนั้นจะปรากฏหน้าต่างโปรแกรม Docker Desktop ดังภาพที่ 4.5 เมื่อคลิกที่เมนู Containers จะพบว่าระบบมีการสร้างคอนเทนเนอร์ให้แล้วจำนวน 1 หน่วยเพื่อใช้ในการทดสอบชื่อ welcome-to-doc



ภาพที่ 4.5 หน้าต่างโปรแกรม Docker Desktop

ขั้นตอนที่ 5) ทดสอบการทำงานของคอนเทนเนอร์

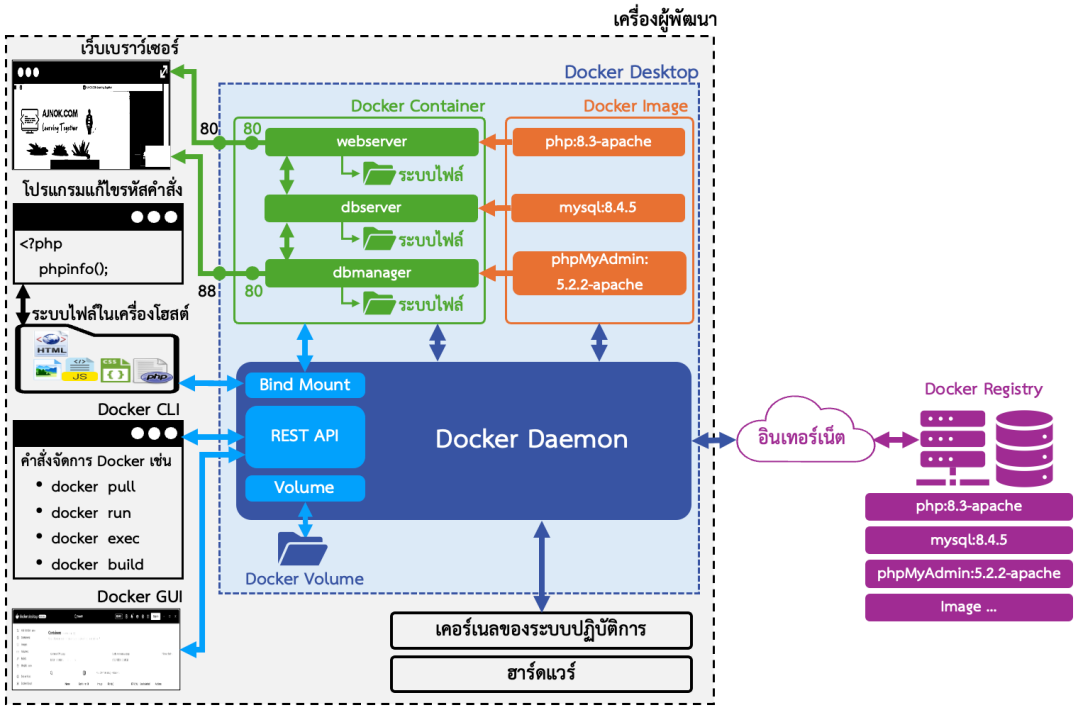
ให้ทำคลิกที่ปุ่มคำสั่ง Run (▷) ที่ด้านท้ายของรายการคอนเทนเนอร์ชื่อ welcome-to-doc เพื่อทำการสั่งให้คอนเทนเนอร์ทำงาน หากคอนเทนเนอร์ทำงานได้จะพบว่ามีการแจ้งสถานะการใช้หน่วยประมวลผลกลางและหน่วยความจำหลักของคอนเทนเนอร์ (Container CPU usage และ Container memory usage) นอกจากนี้คอนเทนเนอร์ทดสอบนี้เปิดให้เข้าถึงจากภายนอกคอนเทนเนอร์ผ่านพอร์ต 8088 ให้ทำการคลิกที่ลิงก์ 8088:80 หรือ เปิดเว็บเบราว์เซอร์แล้วพิมพ์ URL เป็น http://localhost:8088 จะทำให้ได้หน้าผลลัพธ์การทำงานของคอนเทนเนอร์ทดสอบซึ่งเป็นหน้าเว็บลักษณะดังภาพที่ 4.6



ภาพที่ 4.6 หน้าจอผลลัพธ์การทดสอบคอนเทนเนอร์ที่ได้จากการติดตั้ง Docker Desktop

2.2.1 สถาปัตยกรรมและกลไกการทำงานของ Docker Desktop

สถาปัตยกรรมของ Docker โดยรวมถูกออกแบบในลักษณะ Client/Server ซึ่งภายในจะมีองค์ประกอบสำคัญ ได้แก่ Docker Client, Docker Daemon และ Docker Registry (Jayaprakash, 2024) โดยองค์ประกอบเหล่านี้จะทำงานร่วมกันอย่างสอดคล้องผ่านการสื่อสารที่มีประสิทธิภาพผ่าน API ภายในเครื่องหรือผ่านเครือข่ายอินเทอร์เน็ต ได้มีการพัฒนา Docker Desktop ให้เป็นแอปพลิเคชันที่สามารถติดตั้งได้ง่ายบนระบบปฏิบัติการ Windows, macOS และ Linux เพื่ออำนวยความสะดวกสำหรับผู้พัฒนาให้สามารถสั่งงาน Docker โดยใช้ส่วนติดต่อกับผู้ใช้แบบกราฟิกแทนการสั่งการด้วยบรรทัดคำสั่ง (Docker Inc., n.d.a; Stoneman, 2020) หลังจากที่มีการติดตั้ง Docker Desktop ในเครื่องผู้พัฒนาแล้ว จะทำให้เกิดกลไกการทำงานของ Docker แสดงดังภาพที่ 4.7



ภาพที่ 4.7 กลไกการทำงานของ Docker Desktop บนเครื่องผู้พัฒนา

2.2.2 สภาพแวดล้อม Docker Desktop

เมื่อผู้พัฒนาติดตั้ง Docker Desktop ลงในเครื่องคอมพิวเตอร์ ระบบจะสร้างพื้นที่การทำงานสำหรับ Docker ขึ้นบนระบบปฏิบัติการโฮสต์ ซึ่งเรียกว่า Docker Host ภายใน Docker Host จะมี Docker Daemon เป็นบริการหลักที่คอยประมวลผลคำสั่งจากผู้ใช้งาน ทำงานร่วมกับเคอร์เนลของระบบปฏิบัติการโฮสต์ดึงและเผยแพร่แม่แบบของคอนเทนเนอร์ และจัดการกับส่วนประกอบหลักของ Docker ดังนี้

1) Docker Daemon

Docker Daemon คือหัวใจหลักของระบบ ทำหน้าที่ในการประมวลผลคำสั่งทั้งหมดที่ได้รับจาก Docker Client โดยมีความสามารถในการจัดการส่วนต่าง ๆ ของ Docker ประกอบด้วย การจัดการแม่แบบของคอนเทนเนอร์ การจัดการคอนเทนเนอร์ การจัดการพื้นที่จัดเก็บข้อมูลที่แยกจากระบบไฟล์ของคอนเทนเนอร์ การจัดการระบบเครือข่ายเสมือนที่ใช้เชื่อมโยงระหว่างคอนเทนเนอร์ให้สามารถติดต่อกันได้ การเชื่อมโยงพอร์ต การดึงและเผยแพร่แม่แบบของคอนเทนเนอร์ รวมทั้งทำงานร่วมกับเคอร์เนลของระบบปฏิบัติการโฮสต์เพื่อติดต่อกับฮาร์ดแวร์

2) แม่แบบของคอนเทนเนอร์

แม่แบบของคอนเทนเนอร์หรือเรียกว่า Docker Image หรือเรียกสั้นๆ ว่า Image เป็นไฟล์ไบนารีที่ไม่สามารถแก้ไขเพิ่มเติมได้ ซึ่งจะถูกนำไปใช้เป็นแม่แบบในการสร้างคอนเทนเนอร์ โดย Docker Image จะถูกสร้างมาจากไฟล์ที่ชื่อว่า Dockerfile ซึ่งภายในไฟล์ประกอบด้วยลำดับขั้นตอนของการสร้าง Docker Image มีการกำหนดลำดับขั้นตอนของการติดตั้งระบบที่จำเป็นสำหรับการทำงานของแอปพลิเคชันที่จะสร้างขึ้นในคอนเทนเนอร์ เช่น ไลบรารี ไฟล์พึ่งพา คำสั่งเริ่มต้น และไฟล์การกำหนดค่า เป็นต้น ผู้พัฒนาสามารถสร้าง Docker Image ขึ้นเอง หรือดึง (Pull) มาจากแหล่งเก็บแม่แบบ โดยการดึงนี้สามารถดำเนินการผ่าน Docker CLI หรือส่วนติดตั้งกับผู้ใช้ของ Docker Desktop ได้ จากภาพที่ 4.7 จะเห็นว่ามีการดึง Docker Image จากแหล่งเก็บแม่แบบของคอนเทนเนอร์จำนวน 3 รายการ ดังนี้

- php:8.3-apache เป็น Image ที่รวมตัวแปลภาษา PHP เวอร์ชัน 8.3.3 พร้อมกับโปรแกรมให้บริการเว็บ Apache
- mysql:8.4.5 เป็น Image ของระบบจัดการฐานข้อมูล MySQL เวอร์ชัน 8.4.5
- phpmyadmin:5.2.2-apache เป็น Image ที่รวมเว็บแอปพลิเคชัน phpMyAdmin เวอร์ชัน 5.2.2 ซึ่งใช้ในการบริหารจัดการฐานข้อมูล 5.2.2 พร้อมกับโปรแกรมให้บริการเว็บ Apache

3) คอนเทนเนอร์

เมื่อมีการดึงหรือสร้าง Docker Image แล้ว สามารถสร้างคอนเทนเนอร์ขึ้นมาได้ โดย Docker Image เดียวกันสามารถสร้างคอนเทนเนอร์ได้หลายคอนเทนเนอร์ โดยแต่ละคอนเทนเนอร์เป็นหน่วยทำงานเป็นอิสระและมีชั้นข้อมูลระบบไฟล์ที่สามารถจัดเก็บข้อมูลไว้ในคอนเทนเนอร์ขณะทำงานได้ อย่างไรก็ตาม

ที่จัดเก็บข้อมูลนี้จะถูกลบเมื่อคอนเทนเนอร์ถูกทำลาย (Destroy) และหากคอนเทนเนอร์หยุดการทำงาน ผู้พัฒนาก็ไม่สามารถเข้าถึงที่จัดเก็บข้อมูลนี้ได้ จากภาพที่ 4.7 มีการสร้างคอนเทนเนอร์จำนวน 3 หน่วย ดังนี้

- **webserver**

webserver เป็นคอนเทนเนอร์ที่สร้างจากแม่แบบชื่อ php:8.3-apache ซึ่งแม่แบบนี้จะติดตั้งโปรแกรมผู้ให้บริการเว็บ Apache ในคอนเทนเนอร์ด้วย ทำให้ผู้พัฒนาสามารถเข้าถึงคอนเทนเนอร์นี้ในฐานะผู้ให้บริการเว็บที่ทำงานร่วมกับตัวแปลภาษา PHP โดยเปิดให้บริการผ่านพอร์ต 80 และมีการเชื่อมโยงพอร์ต (Port Mapping) ไปยังพอร์ต 80 ของเครื่องโฮสต์ หมายความว่า ถ้าผู้ใช้เปิดโปรแกรมเว็บเบราว์เซอร์และพิมพ์ URL เป็น http://localhost:80 หรือ http://localhost ก็จะสามารถเปิดหน้าโฮมเพจของผู้ให้บริการเว็บในคอนเทนเนอร์นี้ได้

- **dbserver**

dbserver เป็นคอนเทนเนอร์ที่สร้างจากแม่แบบชื่อ mysql:8.45 ทำให้คอนเทนเนอร์นี้ทำหน้าที่ในการเป็นผู้ให้บริการฐานข้อมูล MySQL ซึ่งหากสร้างฐานข้อมูลในคอนเทนเนอร์นี้แล้ว ฐานข้อมูลจะถูกจัดเก็บไว้ในระบบไฟล์ของคอนเทนเนอร์เองเช่นเดียวกับคอนเทนเนอร์ก่อนหน้า

จากภาพที่ 3.16 จะเห็นว่าคอนเทนเนอร์ dbserver สามารถติดต่อกับคอนเทนเนอร์ webserver และ dbmanager ได้ ซึ่งเป็นผลจากกลไกระบบเครือข่ายเสมือนของ Docker ที่ใช้เชื่อมโยงคอนเทนเนอร์ต่าง ๆ ให้สามารถติดต่อกันได้โดยใช้ชื่อของคอนเทนเนอร์แทนที่อยู่ IP เช่น คอนเทนเนอร์ webserver สามารถเชื่อมต่อไปยังคอนเทนเนอร์ฐานข้อมูลได้ด้วยการใช้ชื่อคอนเทนเนอร์ dbserver เป็นชื่อเครื่อง (Host Name) ในการติดต่อ เป็นต้น

- **dbmanager**

dbmanager เป็นคอนเทนเนอร์ที่สร้างจากชื่อ phpmyadmin:5.2.2-apache ซึ่งแม่แบบนี้จะติดตั้งเว็บแอปพลิเคชันชื่อ phpMyAdmin ซึ่งมาพร้อมกับโปรแกรมผู้ให้บริการเว็บ Apache ในคอนเทนเนอร์ด้วย คอนเทนเนอร์นี้ทำหน้าที่ในการเข้าถึงเว็บแอปพลิเคชันสำหรับการจัดการฐานข้อมูลได้ เป็นการช่วยอำนวยความสะดวกให้แก่ผู้พัฒนาโดยแทนที่จะใช้บรรทัดคำสั่งในคอนเทนเนอร์ dbserver ในการจัดการฐานข้อมูล ผู้พัฒนาก็สามารถเข้าถึงหน้าเว็บจากคอนเทนเนอร์ dbmanager นี้แทน ซึ่งฐานข้อมูลจะถูกเก็บไว้ที่คอนเทนเนอร์ dbserver แต่ด้วยกลไกด้านระบบเครือข่ายเสมือนทำให้เว็บแอปพลิเคชัน phpMyAdmin เข้าถึงฐานข้อมูลที่อยู่ต่างคอนเทนเนอร์กันได้

ด้วยผู้ให้บริการเว็บของคอนเทนเนอร์นี้มีพอร์ตให้บริการเป็น 80 เช่นเดียวกับคอนเทนเนอร์แรก จึงทำให้เกิดการชนกันของพอร์ต (Port Collision) จึงต้องมีการเชื่อมโยงพอร์ตให้เข้าถึงจากหมายเลขพอร์ต 88 แทน ซึ่งจะทำให้ผู้พัฒนาเข้าถึงเว็บแอปพลิเคชัน phpMyAdmin ได้จาก http://localhost:88 นอกจากนี้ ระบบไฟล์ภายในคอนเทนเนอร์นี้จะใช้ในการเก็บค่าสถานะจากการทำงานของ phpMyAdmin แต่ไม่ได้ใช้เก็บฐานข้อมูลแต่อย่างใด

4) การเชื่อมโยงแหล่งเก็บข้อมูลระหว่างโฮสต์และคอนเทนเนอร์

จากที่ได้กล่าวมาแล้วข้างต้นว่าข้อมูลที่อยู่ในคอนเทนเนอร์จะไม่สามารถเข้าถึงได้โดยตรงจากเครื่องโฮสต์ของผู้พัฒนาเนื่องจากจะอยู่ในพื้นที่จัดเก็บของคอนเทนเนอร์ซึ่งแยกเป็นส่วนอิสระจากเครื่องโฮสต์เสมือนการแยกออกจากกันทางกายภาพ ดังนั้นในการเข้าถึงข้อมูลในคอนเทนเนอร์ผู้พัฒนาจะต้องสั่งให้คอนเทนเนอร์ทำงาน (Start) จากนั้นจึงใช้คำสั่ง docker exec เพื่อให้เข้าถึงส่วนรับคำสั่ง (Command-line Prompt) ของคอนเทนเนอร์นั้น ๆ เสมือนการเชื่อมต่อระยะไกล (Remote Connection) ไปยังเครื่องคอมพิวเตอร์เครื่องอื่นจึงจะสามารถจัดการข้อมูลในคอนเทนเนอร์นั้นได้

หากผู้พัฒนาต้องการเชื่อมโยงระบบไฟล์ของเครื่องโฮสต์เข้ากับคอนเทนเนอร์เพื่อให้เข้าถึงข้อมูลในคอนเทนเนอร์ได้โดยตรงจากเครื่องโฮสต์ สามารถใช้กลไกที่เรียกว่า Bind Mount ซึ่งจะทำให้ที่จัดเก็บไฟล์ในคอนเทนเนอร์มีตำแหน่งเดียวกันกับที่จัดเก็บไฟล์ในเครื่องโฮสต์ กล่าวคือ เมื่อมีการเปลี่ยนแปลงไฟล์ในที่จัดเก็บข้อมูลของคอนเทนเนอร์ ไฟล์ในที่จัดเก็บเครื่องโฮสต์ก็จะเปลี่ยนแปลงไปด้วยเช่นกัน และหากมีการเปลี่ยนแปลงไฟล์ในที่จัดเก็บของเครื่องโฮสต์ก็จะมีการเปลี่ยนแปลงในที่จัดเก็บของคอนเทนเนอร์ในทำนองเดียวกัน

จากภาพที่ 3.16 เมื่อมีการเชื่อมโยงแหล่งเก็บข้อมูลระหว่างโฮสต์และคอนเทนเนอร์ ผู้พัฒนาสามารถใช้โปรแกรมแก้ไขรหัสคำสั่งในการสร้าง แก้ไขหรือลบไฟล์ต่าง ๆ ในเครื่องโฮสต์ของผู้พัฒนาได้โดยตรง ดังนั้นหากผู้พัฒนาต้องการเขียนรหัสคำสั่งของไฟล์เว็บแอปพลิเคชันภายใต้สภาพแวดล้อมการพัฒนาจึงควรใช้กลไก Bind Mount ในการเชื่อมโยงระบบไฟล์

5) การจัดเก็บข้อมูลแยกออกจากวงจรชีวิตของคอนเทนเนอร์

การจัดเก็บข้อมูลแยกออกจากวงจรชีวิตของคอนเทนเนอร์เป็นกลไกที่เรียกว่า Docker Volume กล่าวคือ เป็นการสร้างที่เก็บข้อมูลไว้ในพื้นที่การทำงานของ Docker Desktop เอง หากคอนเทนเนอร์ใดมีการสร้าง Docker Volume เพื่อใช้จัดเก็บข้อมูลแทนที่การจัดเก็บในระบบไฟล์ของคอนเทนเนอร์ เมื่อคอนเทนเนอร์นั้นหยุดการทำงานหรือถูกทำลาย ข้อมูลจะยังคงอยู่ในที่เก็บของ Docker Desktop แต่ยังคงไม่เชื่อมโยงโดยตรงกับระบบไฟล์ของเครื่องโฮสต์เหมือน Bind Mount ผู้พัฒนาสามารถเข้าถึงที่เก็บนี้ได้ผ่าน Docker Daemon ด้วยคำสั่งของ Docker หรือส่วนติดต่อผู้ใช้แบบกราฟิกใน Docker Desktop โดยกลไกการจัดเก็บข้อมูลลักษณะนี้จะเหมาะกับสภาพแวดล้อมการทำงานจริง

6) REST API

REST API หรืออินเทอร์เฟซโปรแกรมประยุกต์ (Application Programming Interface: API) แบบ REST (Representational State Transfer) ซึ่งเป็นกลไกในการแลกเปลี่ยนข้อมูลระหว่าง Docker Client และ Docker Daemon โดย Docker Daemon จะรับคำสั่งในรูปแบบ HTTP Requests และส่งคืนผลลัพธ์เป็น JSON Responses

7) Docker Client

Docker Client คือเครื่องมือหลักที่ใช้ในการโต้ตอบหรือสั่งให้ Docker ทำงาน ในกรณีที่ไม่ได้ติดตั้ง Docker Desktop จะต้องสั่งการผ่านบรรทัดคำสั่งของ Docker (Docker CLI) ซึ่งสามารถสั่งงานได้ด้วย

คำสั่งพื้นฐาน เช่น docker build, docker run, docker pull, docker push และ docker exec เป็นต้น นอกจากนี้ Docker Client ยังสามารถอยู่ในรูปแบบของ Docker Desktop ซึ่งมีส่วนติดต่อกับผู้ใช้เป็นกราฟิกช่วยให้ผู้ใช้สามารถสั่งงานผ่านอินเทอร์เฟซที่ใช้งานง่าย

8) Docker Registry

Docker Registry คือแหล่งเก็บแม่แบบของคอนเทนเนอร์ทั้งที่สร้างโดยนักพัฒนาเองและที่เผยแพร่โดยชุมชนนักพัฒนา ซึ่งผู้ใช้สามารถดึงแม่แบบของคอนเทนเนอร์จาก Registry มาใช้งาน หรือเผยแพร่แม่แบบจากคอนเทนเนอร์ที่ตนเองสร้างขึ้นไปเก็บใน Docker Registry โดย Docker Registry สาธารณะอย่างเป็นทางการของ Docker คือ Docker Hub

3. กรณีศึกษา: การสร้าง LAMP Stack ด้วย Docker Desktop

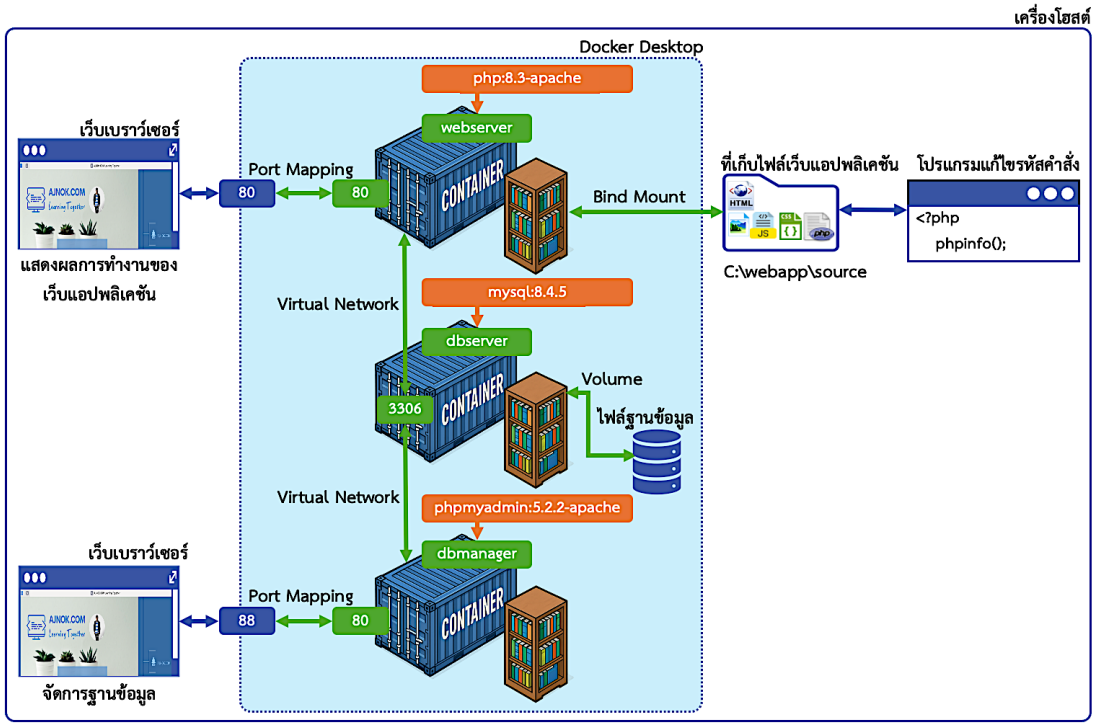
จากที่ได้กล่าวมาในหัวข้อก่อนหน้าเกี่ยวกับหลักการของ Docker Desktop จะเห็นได้ว่า Docker เป็นเครื่องมือที่มีประสิทธิภาพในการจัดการสภาพแวดล้อมของแอปพลิเคชัน โดยเฉพาะเมื่อต้องพัฒนาเว็บแอปพลิเคชันที่ต้องอาศัยองค์ประกอบของกลไกผู้ให้บริการเว็บ ประกอบด้วย ผู้ให้บริการเว็บ ตัวแปลภาษา และระบบจัดการฐานข้อมูล

การใช้ Docker ช่วยให้สามารถแยกแต่ละส่วนของเว็บแอปพลิเคชันให้อยู่ในคอนเทนเนอร์ที่เป็นอิสระจากกัน แต่สามารถเชื่อมโยงกันเพื่อทำงานร่วมกันเป็นระบบกลไกผู้ให้บริการเว็บที่สมบูรณ์ โดยหลักการสำคัญที่ควรยึดถือในการออกแบบคอนเทนเนอร์คือ คอนเทนเนอร์หนึ่งหน่วยควรรับผิดชอบเพียงหนึ่งหน้าที่ (Single Responsibility) และทำหน้าที่นั้นให้ดีที่สุด เป็นแนวทางที่ช่วยลดความซับซ้อนของระบบ ทำให้การดูแลรักษาและปรับปรุงระบบในภายหลังเป็นไปอย่างมีประสิทธิภาพ แม้จะมีข้อยกเว้นในบางกรณี แต่การแยกคอนเทนเนอร์ตามบทบาทหน้าที่ย่อมส่งผลให้ระบบมีความยืดหยุ่นสูง และง่ายต่อการพัฒนา ทดสอบ และปรับแต่ง (Docker Inc., n.d.b)

หัวข้อนี้เป็นกรณีศึกษาการสร้าง LAMP Stack ด้วย Docker Desktop ที่ใช้แนวทางการออกแบบคอนเทนเนอร์ที่กล่าวถึงข้างต้น กรณีศึกษานี้จะใช้โปรแกรม Visual Studio Code หรือ VSCode ในการเตรียมโครงสร้างโฟลเดอร์ การจัดการคอนเทนเนอร์ และการพัฒนาเว็บแอปพลิเคชัน โดยมีกระบวนการดังนี้

3.1 การออกแบบโครงสร้าง LAMP Stack

การออกแบบโครงสร้าง LAMP Stack เป็นการกำหนดบทบาทของแต่ละคอนเทนเนอร์ การพิจารณาแม่แบบของคอนเทนเนอร์ที่จะใช้สร้างคอนเทนเนอร์ การวางแผนการใช้พอร์ต การเชื่อมโยงระหว่างคอนเทนเนอร์ และการพิจารณาการจัดเก็บข้อมูล ในกรณีศึกษานี้ได้มีการออกแบบโครงสร้างลักษณะดังภาพที่ 4.8



ภาพที่ 4.8 การออกแบบโครงสร้าง LAMP Stack

จากภาพที่ 4.8 ได้มีการกำหนดให้มีคอนเทนเนอร์จำนวน 3 หน่วย ประกอบด้วย webserver, dbserver และ dbmanager ดังนี้

3.1.1 webserver

คอนเทนเนอร์ชื่อ webserver ทำหน้าที่เป็นผู้ให้บริการเว็บพร้อมตัวแปลภาษา PHP โดยใช้แม่แบบ php:8.3-apache ซึ่งเป็นแม่แบบที่มีโปรแกรม Apache เป็นผู้ให้บริการเว็บทำงานร่วมกับตัวแปลภาษา PHP เวอร์ชัน 8.3 ทำให้ไม่ต้องสร้างคอนเทนเนอร์แยกอีก โดยมีการกำหนดให้เข้าถึงคอนเทนเนอร์ผู้ให้บริการเว็บจากเครื่องโฮสต์ผ่านทางพอร์ต 80 และมีการเชื่อมโยงกับพอร์ตภายในของคอนเทนเนอร์ที่พอร์ต 80 เช่นกัน นอกจากนี้ยังมีการออกแบบให้เชื่อมโยงแหล่งเก็บข้อมูลระหว่างโฮสต์และคอนเทนเนอร์ไว้ที่ c:\webapp\source เพื่อให้ทั้งสองส่วนสามารถใช้ไฟล์ร่วมกันได้

3.1.2 dbserver

คอนเทนเนอร์ชื่อ dbserver ทำหน้าที่เป็นผู้ให้บริการระบบจัดการฐานข้อมูล ใช้แม่แบบ mysql:8.4.5 ซึ่งเป็นแม่แบบที่มีโปรแกรม MySQL เวอร์ชัน 8.4.5 เป็นผู้ให้บริการระบบจัดการฐานข้อมูล มีการกำหนดให้ใช้พอร์ต 3306 เพื่อให้คอนเทนเนอร์อื่นๆ สามารถติดต่อกับผู้ให้บริการระบบจัดการฐานข้อมูลในคอนเทนเนอร์นี้ได้ นอกจากนี้ มีการกำหนดให้จัดเก็บข้อมูลแยก

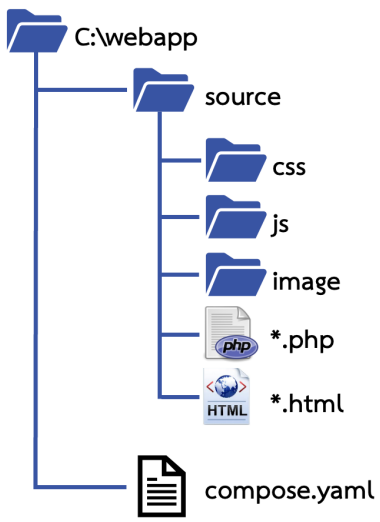
ออกจากวงจรชีวิตของคอนเทนเนอร์ไว้ในพื้นที่การทำงานของ Docker Desktop เพื่อป้องกันไม่ให้ข้อมูลถูกลบเมื่อมีการลบคอนเทนเนอร์

3.1.3 dbmanager

คอนเทนเนอร์ชื่อ dbmanager ทำหน้าที่เป็นเว็บแอปพลิเคชันเพื่ออำนวยความสะดวกในการจัดการฐานข้อมูลผ่านเว็บเบราว์เซอร์ ใช้แม่แบบ phpmyadmin:5.2.2-apache ซึ่งเป็นแม่แบบที่มีโปรแกรม Apache และตัวแปลภาษา PHP เป็นผู้ให้บริการเว็บแอปพลิเคชันชื่อ PhpMyAdmin รุ่น 5.2.2 โดยมีการกำหนดให้เข้าถึงคอนเทนเนอร์ผู้ให้บริการเว็บแอปพลิเคชันนี้จากเครื่องโฮสต์ผ่านทางพอร์ต 88 และมีการเชื่อมโยงกับพอร์ตภายในของคอนเทนเนอร์ที่พอร์ต 80 เนื่องจากไม่สามารถเผยแพร่พอร์ต 80 ให้เครื่องโฮสต์เข้าถึงได้ซึ่งจะซ้ำกับพอร์ตของคอนเทนเนอร์ webserver

3.2 การออกแบบโครงสร้างโฟลเดอร์ของโครงการพัฒนาเว็บแอปพลิเคชัน

หลังจากที่ได้ทำการออกแบบโครงสร้าง LAMP Stack แล้ว จะเป็นการเตรียมโครงสร้างโฟลเดอร์สำหรับโครงการพัฒนาเว็บแอปพลิเคชัน โดยโครงสร้างโฟลเดอร์สำหรับกรณีศึกษานี้แสดงดังภาพที่ 4.9 ประกอบด้วยโฟลเดอร์หลักของโครงการชื่อ webapp อยู่ในไดร์ฟ c: ของระบบปฏิบัติการ Microsoft Windows ซึ่งภายในจะมีโฟลเดอร์สำหรับจัดเก็บไฟล์เว็บแอปพลิเคชันชื่อ source ที่จะใช้เชื่อมโยงเข้ากับคอนเทนเนอร์ webserver เพื่อให้คอนเทนเนอร์และเครื่องโฮสต์สามารถเข้าถึงไฟล์ของเว็บแอปพลิเคชันร่วมกันได้



ภาพที่ 4.9 ตัวอย่างการออกแบบโครงสร้างโฟลเดอร์สำหรับโครงการพัฒนาเว็บแอปพลิเคชัน

จากภาพที่ 4.9 เป็นตัวอย่างการออกแบบโครงสร้างโฟลเดอร์สำหรับโครงการพัฒนาเว็บแอปพลิเคชัน ซึ่งไฟล์ต่าง ๆ ที่จะสร้างขึ้นมาเพื่อใช้ในการทำงานของเว็บแอปพลิเคชัน เช่นไฟล์ที่มีนามสกุลเป็น .php และ .html จะถูกจัดเก็บไว้ใน source ซึ่งยังมีการแบ่งโฟลเดอร์ไว้ภายในเพื่อใช้แยกจัดเก็บไฟล์ชนิดต่าง ๆ ให้เป็นระเบียบ เช่น โฟลเดอร์ css, js และ image ใช้จัดเก็บไฟล์ CSS, JavaScript และรูปภาพตามลำดับ นอกจากนี้ยังมีไฟล์

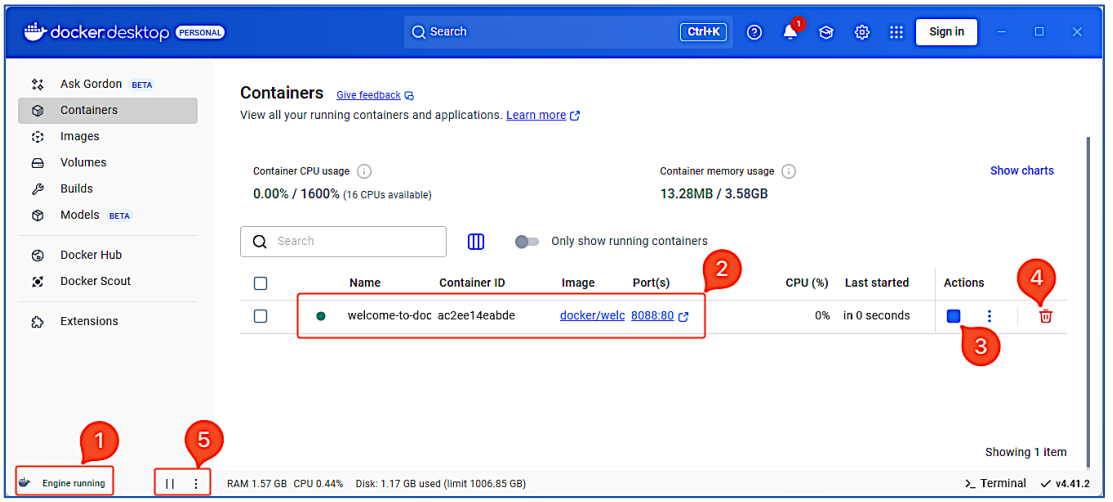
compose.yaml ที่ใช้กับ Docker Compose ซึ่งเป็นเครื่องมือที่ใช้ในการสร้างคอนเทนเนอร์หลายคอนเทนเนอร์ให้ทำงานร่วมกันได้ในรูปแบบของบริการ (Service) และสามารถจัดการการเชื่อมโยงไฟล์ การจัดเก็บข้อมูลแยกจากคอนเทนเนอร์และการกำหนดพอร์ตการทำงานต่าง ๆ ของคอนเทนเนอร์

3.3 การสร้างและเรียกใช้งานคอนเทนเนอร์

หลังจากที่ได้ออกแบบโครงสร้างของ LAMP Stack และโครงสร้างโฟลเดอร์ของโครงการเรียบร้อยแล้ว ขั้นตอนถัดไปคือการลงมือปฏิบัติในการสร้างและเรียกใช้งานคอนเทนเนอร์ตามที่ได้ออกแบบไว้ โดยกรณีศึกษานี้จะใช้โปรแกรม Visual Studio Code (VSCode) ร่วมกับ Docker Desktop ในการจัดการไฟล์โครงการ ตรวจสอบสถานะ และดำเนินการกับคอนเทนเนอร์ต่าง ๆ โดยมีลำดับขั้นตอนการดำเนินงานดังต่อไปนี้

ขั้นตอนที่ 1) เปิดและตรวจสอบสถานะการทำงานของ Docker Desktop

ขั้นตอนแรกเป็นการตรวจสอบการทำงานของ Docker Engine ทำได้โดยการเปิดโปรแกรม Docker Desktop และพิจารณาสถานะการทำงานที่แสดงบริเวณตำแหน่งที่ 1 ของภาพที่ 4.10 ซึ่งเป็นหน้าจอโปรแกรม Docker Desktop หากแสดงข้อความ Engine running แสดงว่า Docker พร้อมใช้งานแล้ว จากนั้นให้ทำการตรวจสอบว่ามีคอนเทนเนอร์ใดที่กำลังทำงานอยู่ ซึ่งอาจใช้พอร์ตเดียวกันกับคอนเทนเนอร์ที่กำลังจะสร้างขึ้น โดยพิจารณาจากตำแหน่งที่ 2 ของภาพที่ 4.10 ซึ่งพบว่ามีคอนเทนเนอร์ชื่อ welcome-to-doc ให้บริการพอร์ต 8080:80 อยู่ หากต้องการหยุดการทำงานของคอนเทนเนอร์ให้คลิกที่ปุ่ม Action ในตำแหน่งหมายเลข 3 และถ้าต้องการลบคอนเทนเนอร์ให้คลิกที่ปุ่มในตำแหน่งหมายเลข 4 ของภาพที่ 4.10 ตามลำดับ ในกรณีที่ต้องการสั่งให้ Docker Engine หยุดทำงานหรือเริ่มต้นการทำงานใหม่ สามารถทำได้โดยการคลิกที่ปุ่มเมนูในตำแหน่งหมายเลข 5 ของภาพที่ 4.10



ภาพที่ 4.10 การตรวจสอบการทำงานของ Docker Engine

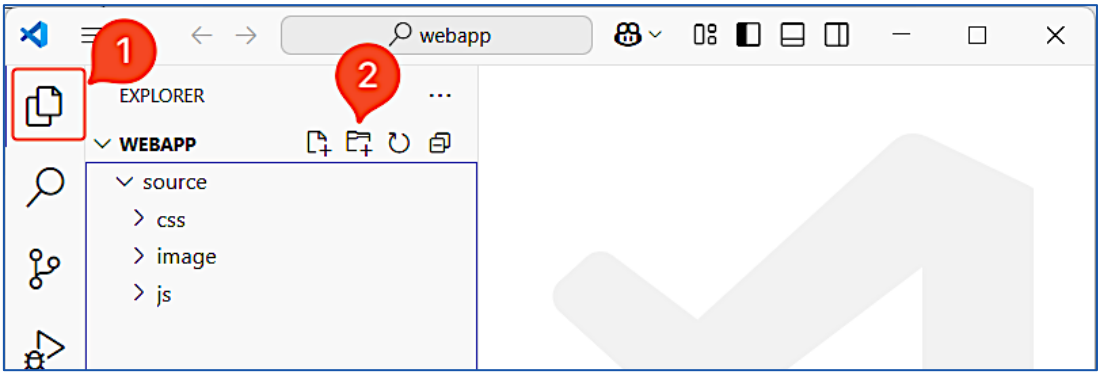
ขั้นตอนที่ 2) สร้างโครงสร้างโฟลเดอร์สำหรับเว็บแอปพลิเคชัน

หลังจากตรวจสอบความพร้อมของ Docker Desktop แล้ว ขั้นตอนถัดไปคือการสร้างโฟลเดอร์ของโครงการพัฒนาเว็บแอปพลิเคชันตามที่ได้ออกแบบไว้ โดยในกรณีศึกษานี้ใช้โปรแกรม VSCode ในการสร้างและจัดการโครงสร้างโฟลเดอร์ ซึ่งมีการดำเนินการดังนี้

1) สร้างโฟลเดอร์ชื่อ webapp ไว้ที่ C:\

2) เปิดโปรแกรม VSCode แล้วคลิกเมนู File > Open Folder... จากนั้นเลือกโฟลเดอร์ webapp ที่ได้สร้างขึ้น

3) สร้างโฟลเดอร์ตามโครงสร้างที่ได้ออกแบบด้วยโปรแกรม VSCode โดยใช้เครื่องมือ Explorer ในตำแหน่งที่ 1 และ 2 ของภาพที่ 4.11



ภาพที่ 4.11 การสร้างโฟลเดอร์ด้วยโปรแกรม VSCode

ขั้นตอนที่ 3) สร้างไฟล์ compose.yaml เพื่อสร้างคอนเทนเนอร์

หลังจากที่ได้สร้างโฟลเดอร์สำหรับโครงการพัฒนาเว็บแอปพลิเคชันตามที่แสดงไว้ในหัวข้อก่อนหน้า ขั้นตอนถัดไปคือการสร้างไฟล์ compose.yaml ไว้ในโฟลเดอร์ webapp ซึ่งเป็นไฟล์หลักในการใช้ Docker Compose สำหรับกำหนดรายละเอียดของคอนเทนเนอร์หลายหน่วย (Multi-container Application) ให้สามารถทำงานร่วมกันได้อย่างเป็นระบบ ซึ่งไฟล์ compose.yaml เป็นไฟล์ข้อความที่ใช้สำหรับการจัดเก็บข้อมูลเชิงโครงสร้างในลักษณะที่สามารถอ่านได้ง่าย โดยไฟล์ที่ใช้ในการสร้างคอนเทนเนอร์ในกรณีศึกษานี้มีรายละเอียดดังนี้

#	ไฟล์ C:\webapp\compose.yaml
1	name: *webapp*
2	services:
3	*web:*
4	image: *php:8.3-apache*

```
5           container_name: webserver
6           ports:
7               - "80:80"
8           volumes:
9               - ./source:/var/www/html
10          depends_on:
11              - db
12      db:
13          image: mysql:8.4.5
14          container_name: dbserver
15          expose:
16              - "3306"
17          volumes:
18              - db_data:/var/lib/mysql
19          environment:
20              MYSQL_ROOT_PASSWORD: rootpass
21              MYSQL_DATABASE: db_webapp
22              MYSQL_USER: webapp_user
23              MYSQL_PASSWORD: webapp_user_pass
24      dbmanager:
25          image: phpmyadmin:5.2.2-apache
26          container_name: dbmanager
27          ports:
28              - "88:80"
29          environment:
30              - PMA_HOST: db
31          depends_on:
32              - db
33  volumes:
34      db_data:
```

จากไฟล์ compose.yaml ข้างต้น จะเห็นได้ว่ามีการใช้ระยะเยื้อง (Indentation) เพื่อบ่งบอกโครงสร้างของบริการต่าง ๆ ที่จะทำงานร่วมกัน โดยมีคำอธิบายดังนี้

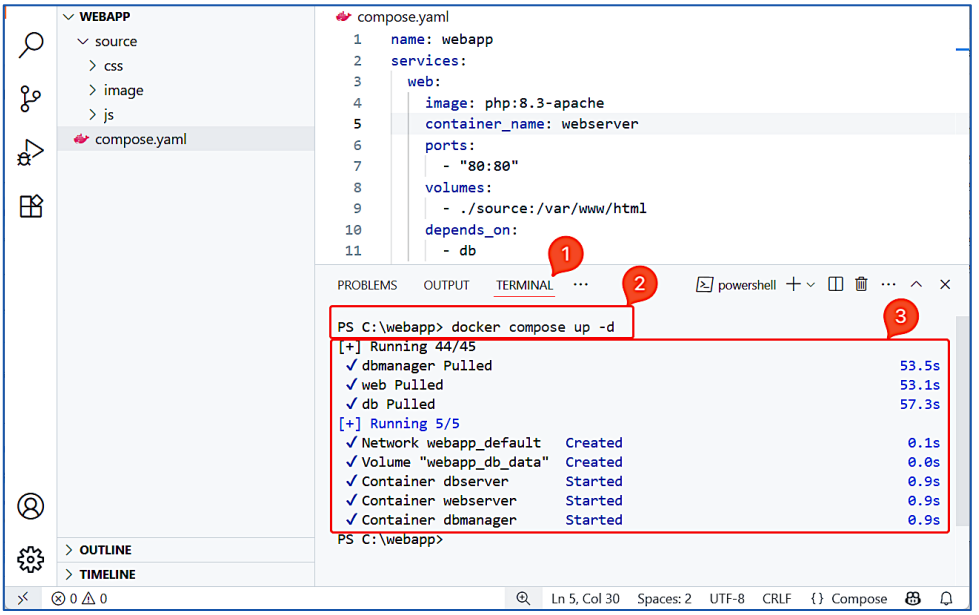
- บรรทัดที่ 1 เป็นองค์ประกอบระดับบนสุด (top-level element) ใช้กำหนดชื่อของโครงการ Docker Compose (Compose Project Name) ซึ่งในที่นี้คือ webapp โดยชื่อนี้จะถูกนำไปใช้เป็นคำนำหน้า (prefix) สำหรับการตั้งชื่อทรัพยากรต่าง ๆ ที่ Docker Compose สร้างขึ้น เช่น คอนเทนเนอร์ (webapp_web_1) เครือข่าย (webapp_default) เป็นต้น หากไม่ระบุค่า name นี้ ระบบจะใช้ชื่อของโฟลเดอร์ที่ไฟล์ compose.yaml อยู่แทน
- บรรทัดที่ 2 เป็นองค์ประกอบระดับบนสุดของไฟล์ที่จำเป็นต้องมีในไฟล์ compose.yaml ใช้สำหรับการกำหนดบริการ (Services) ที่ทำงานร่วมกันภายใต้โครงการ Docker Compose เดียวกัน โดยแต่ละบริการจะถูกนิยามผ่านชื่อที่กำหนดขึ้นเอง เช่น web, db และ dbmanager (ในบรรทัดที่ 3, 12 และ 24 ตามลำดับ) ซึ่งบริการเหล่านี้จะประกอบด้วยคำสั่งและคุณลักษณะต่าง ๆ ที่ใช้ในการสร้างและจัดการคอนเทนเนอร์ที่สนับสนุนการทำงานของบริการนั้นๆ
- บรรทัดที่ 3–11 เป็นการกำหนดบริการชื่อ web ซึ่งทำหน้าที่เป็นผู้ให้บริการเว็บและตัวแปลภาษา PHP โดยใช้แม่แบบคอนเทนเนอร์ php:8.3-apache (บรรทัดที่ 4) พร้อมกำหนดชื่อคอนเทนเนอร์เป็น webserver (บรรทัดที่ 5) เพื่อให้ง่ายต่อการอ้างอิงภายในระบบ มีกำหนดการเชื่อมโยงพอร์ต 80:80 (บรรทัดที่ 6–7) ทำให้เมื่อผู้ใช้เข้าถึง http://localhost:80 คำร้องขอจะถูกส่งไปยังคอนเทนเนอร์ webserver ที่ให้บริการเว็บผ่านพอร์ต 80 สำหรับแอตทริบิวต์ volumes (บรรทัดที่ 8–9) ใช้เพื่อเชื่อมโยงโฟลเดอร์ source ซึ่งอยู่ในตำแหน่งเดียวกับไฟล์ compose.yaml บนเครื่องโฮสต์ กับ /var/www/html ภายในคอนเทนเนอร์ ซึ่งเป็นตำแหน่งที่ Apache ใช้ในการจัดเก็บทรัพยากรในการให้บริการเว็บแก่ผู้ร้องขอ นอกจากนี้ ยังมีการกำหนดแอตทริบิวต์ depends_on (บรรทัดที่ 10–11) เพื่อระบุให้บริการ web เริ่มต้นหลังจากบริการ db ทำงานแล้ว ซึ่งช่วยให้การเรียงลำดับการเริ่มต้นบริการเป็นไปอย่างเหมาะสม
- บรรทัดที่ 12–23 เป็นการกำหนดบริการชื่อ db ซึ่งทำหน้าที่เป็นผู้ให้บริการระบบจัดการฐานข้อมูล โดยใช้แม่แบบคอนเทนเนอร์ mysql:8.4.5 (บรรทัดที่ 13) และระบุชื่อคอนเทนเนอร์ว่า dbserver (บรรทัดที่ 14) สำหรับการเปิดใช้งานพอร์ต 3306 ซึ่งเป็นพอร์ตมาตรฐานของ MySQL นั้น ได้ใช้แอตทริบิวต์ expose (บรรทัดที่ 15–16) เพื่อให้บริการอื่นภายในเครือข่ายเสมือนสามารถเชื่อมต่อได้โดยไม่เปิดเผยพอร์ตสู่เครื่องโฮสต์ ช่วยยกระดับความปลอดภัยของระบบ ในส่วนของแอตทริบิวต์ volumes (บรรทัดที่ 17–18) ใช้สำหรับเชื่อมโยงพื้นที่จัดเก็บข้อมูลภายในคอนเทนเนอร์กับ Docker Volume ชื่อ db_data ซึ่งเป็นพื้นที่จัดเก็บถาวรที่แยกออกจากวงจรชีวิตของคอนเทนเนอร์ ทำให้ข้อมูลยังคงอยู่แม้คอนเทนเนอร์จะหยุดการทำงานหรือลบออก ส่วนแอตทริบิวต์ environment (บรรทัดที่ 19–23) ใช้กำหนดค่าตัวแปรสภาพแวดล้อมที่จำเป็นต่อการเริ่มต้นระบบ ได้แก่ รหัสผ่านของผู้ดูแลระบบ (MYSQL_ROOT_PASSWORD) ชื่อฐานข้อมูลเริ่มต้น (MYSQL_DATABASE) ชื่อผู้ใช้ (MYSQL_USER) และรหัสผ่านของผู้ใช้ (MYSQL_PASSWORD) โดยค่าดังกล่าวจะต้องอ้างอิงจากคำอธิบายการใช้งานของแม่แบบใน Docker Registry ซึ่งอาจแตกต่างกันไปในแต่ละเวอร์ชัน

- บรรทัดที่ 24–32 เป็นการกำหนดบริการชื่อ dbmanager ซึ่งทำหน้าที่เป็นเว็บแอปพลิเคชันสำหรับจัดการฐานข้อมูลผ่านเว็บเบราว์เซอร์ โดยใช้แม่แบบคอนเทนเนอร์ phpmyadmin:5.2.2-apache (บรรทัดที่ 25) พร้อมกำหนดชื่อคอนเทนเนอร์ว่า dbmanager (บรรทัดที่ 26) การเข้าถึงผ่านเว็บเบราว์เซอร์จากเครื่องโฮสต์ได้มีการเชื่อมโยงพอร์ต 88:80 (บรรทัดที่ 27–28) เพื่อหลีกเลี่ยงความขัดแย้งกับพอร์ต 80 ที่บริการ web เปิดให้ใช้งานอยู่ โดยผู้ใช้สามารถเข้าถึง phpMyAdmin ได้จาก http://localhost:88 นอกจากนี้ยังมีการกำหนดตัวแปรสภาพแวดล้อม PMA_HOST เป็น db (บรรทัดที่ 29–30) ซึ่งเป็นชื่อของบริการระบบจัดการฐานข้อมูลที่ phpMyAdmin จะเชื่อมต่อด้วย และสุดท้ายมีการใช้แอตทริบิวต์ depends_on (บรรทัดที่ 31–32) เพื่อระบุให้บริการ dbmanager เริ่มต้นหลังจากที่บริการ db พร้อมใช้งานแล้ว

- บรรทัดที่ 33–34 เป็นการกำหนด volumes ซึ่งเป็นองค์ประกอบระดับบนสุดของไฟล์ compose.yaml ใช้สำหรับกำหนดชื่อของพื้นที่จัดเก็บข้อมูลแบบถาวร (Named Volume) ที่สามารถนำไปใช้ร่วมกันระหว่างบริการต่าง ๆ ภายใต้โครงการ Docker Compose เดียวกันได้ โดยในกรณีศึกษานี้ได้ประกาศชื่อพื้นที่จัดเก็บว่า db_data และมีการนำไปใช้ในบริการ db เพื่อจัดเก็บข้อมูลของระบบฐานข้อมูลให้คงอยู่ แม้ว่าคอนเทนเนอร์จะหยุดทำงานหรือลบออกก็ตาม หากต้องการให้บริการอื่นใช้งานพื้นที่นี้ร่วมกัน ก็สามารถกำหนดเพิ่มเติมภายใต้ แอตทริบิวต์ volumes ของบริการนั้น ๆ ลักษณะเดียวกันกับที่กำหนดในบริการ db ได้

ขั้นตอนที่ 4) สร้างคอนเทนเนอร์ด้วย Docker Compose

หลังจากจัดทำไฟล์ compose.yaml เรียบร้อยแล้ว ขั้นตอนถัดไปคือการสั่งให้ Docker Compose สร้างคอนเทนเนอร์และองค์ประกอบที่เกี่ยวข้อง โดยใช้คำสั่งผ่าน Terminal ซึ่งเป็นเครื่องมือภายในโปรแกรม VSCode สำหรับสั่งการด้วยบรรทัดคำสั่ง



ภาพที่ 4.12 การสั่งให้ Docker Compose ทำการสร้างคอนเทนเนอร์และองค์ประกอบต่าง ๆ

จากภาพที่ 4.12 แสดงการเข้าถึง Terminal ที่ตำแหน่งหมายเลข 1 ของภาพ (หากไม่ปรากฏสามารถเข้าถึงได้จากเมนู Terminal > New Terminal) ตรวจสอบให้แน่ใจว่าตำแหน่งของ Terminal อยู่ที่โฟลเดอร์ C:\webapp ซึ่งเป็นที่เก็บไฟล์ compose.yaml ดังแสดงในตำแหน่งหมายเลข 2 ของภาพ หากยังไม่อยู่ในตำแหน่งดังกล่าว ให้ใช้คำสั่ง cd c:\webapp เพื่อเปลี่ยนไดเรกทอรีการทำงาน เมื่ออยู่ในตำแหน่งที่ถูกต้องแล้ว ให้พิมพ์คำสั่งดังนี้เพื่อเริ่มสร้างและเปิดใช้งานคอนเทนเนอร์ทั้งหมด

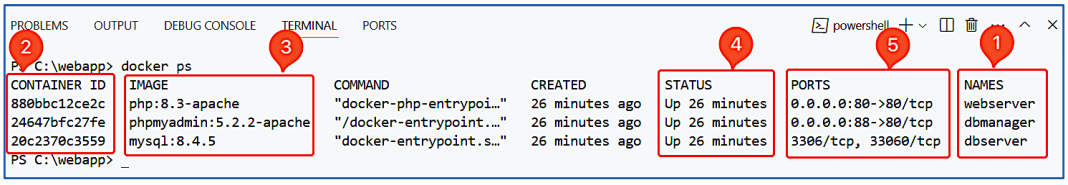
คำสั่ง	docker compose up -d

หากไม่พบข้อผิดพลาดในการทำงานของคำสั่งข้างต้น ระบบจะแสดงสถานะการดำเนินการตามลำดับ เช่น การดึงแม่แบบของคอนเทนเนอร์ การสร้างคอนเทนเนอร์สำหรับบริการ db, web, และ dbmanager การสร้างเครือข่ายเสมือน webapp_default การสร้าง Volume ชื่อ webapp_db_data ตลอดจนการเริ่มต้นคอนเทนเนอร์ทั้งสามหน่วย ดังแสดงในตำแหน่งหมายเลข 3 ของภาพที่ 4.12

ขั้นตอนที่ 5) ตรวจสอบสถานการณ์ทำงานของคอนเทนเนอร์

ภายหลังจากที่ได้สั่งให้ Docker Compose สร้างและเปิดใช้งานคอนเทนเนอร์แล้ว ผู้พัฒนาควรตรวจสอบว่าสถานะของคอนเทนเนอร์ทั้งสามหน่วยอยู่ในสภาวะทำงาน (Up) และไม่มีข้อผิดพลาดเกิดขึ้น โดยสามารถตรวจสอบด้วยการใช้บรรทัดคำสั่งผ่าน Terminal ดังนี้ โดยผลลัพธ์จะแสดงดังภาพที่ 4.13

คำสั่ง	docker ps

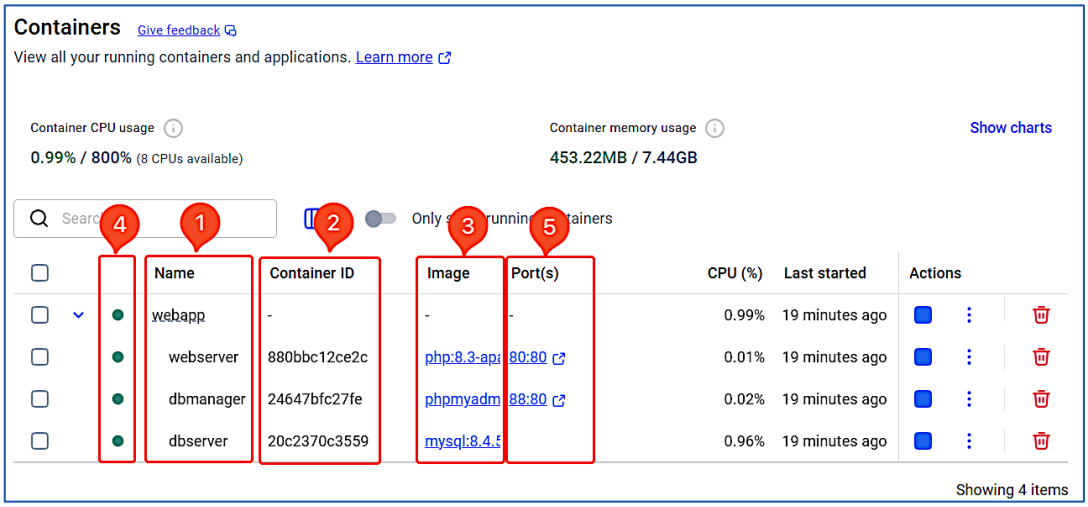


ภาพที่ 4.13 ผลลัพธ์ของคำสั่ง docker ps

จากภาพที่ 4.13 แสดงผลลัพธ์ของคำสั่ง docker ps ซึ่งแสดงรายละเอียดของคอนเทนเนอร์ที่สร้างขึ้น ทั้งนี้ นอกจากจะตรวจสอบด้วยการใช้บรรทัดคำสั่งแล้ว ยังสามารถตรวจสอบได้จากหน้าต่างโปรแกรม Docker Desktop แสดงดังภาพที่ 4.14 โดยมีข้อมูลที่สำคัญในตำแหน่งต่าง ๆ ของทั้งสองภาพ ดังนี้

- ตำแหน่งหมายเลข 1) แสดงข้อมูลชื่อคอนเทนเนอร์
- ตำแหน่งหมายเลข 2) แสดงข้อมูลหมายเลขประจำตัวคอนเทนเนอร์
- ตำแหน่งหมายเลข 3) แสดงข้อมูลแม่แบบที่ใช้สร้างคอนเทนเนอร์
- ตำแหน่งหมายเลข 4) แสดงสถานะการทำงานของคอนเทนเนอร์ ซึ่งทั้งหมดแสดงสถานะเป็น Up หมายความว่าทุกๆ คอนเทนเนอร์กำลังทำงานอยู่

- ตำแหน่งหมายเลข 5) แสดงข้อมูลหมายเลขพอร์ตที่เชื่อมโยง



ภาพที่ 4.14 การแสดงสถานะของคอนเทนเนอร์จากหน้าต่างโปรแกรม Docker Desktop

ขั้นตอนที่ 6) ทดสอบการทำงานคอนเทนเนอร์ผ่านเว็บเบราว์เซอร์

ขั้นตอนสุดท้ายเป็นการทดสอบการทำงานของคอนเทนเนอร์ ซึ่งคอนเทนเนอร์ชื่อ webserver และ dbmanager ผู้พัฒนาสามารถทดสอบโดยใช้เว็บแอปพลิเคชันได้เนื่องจากทั้งสองคอนเทนเนอร์มีการติดตั้งผู้ให้บริการเว็บ Apache ส่วนคอนเทนเนอร์ dbserver นั้นไม่สามารถเข้าถึงจากเครื่องโฮสต์ได้โดยตรงตามการออกแบบที่กำหนดไว้ แต่จะใช้วิธีการทดสอบจากคอนเทนเนอร์ dbmanager ซึ่งเป็นคอนเทนเนอร์ที่มีเว็บแอปพลิเคชันสำหรับการจัดการฐานข้อมูลอยู่ในคอนเทนเนอร์ dbserver แทน

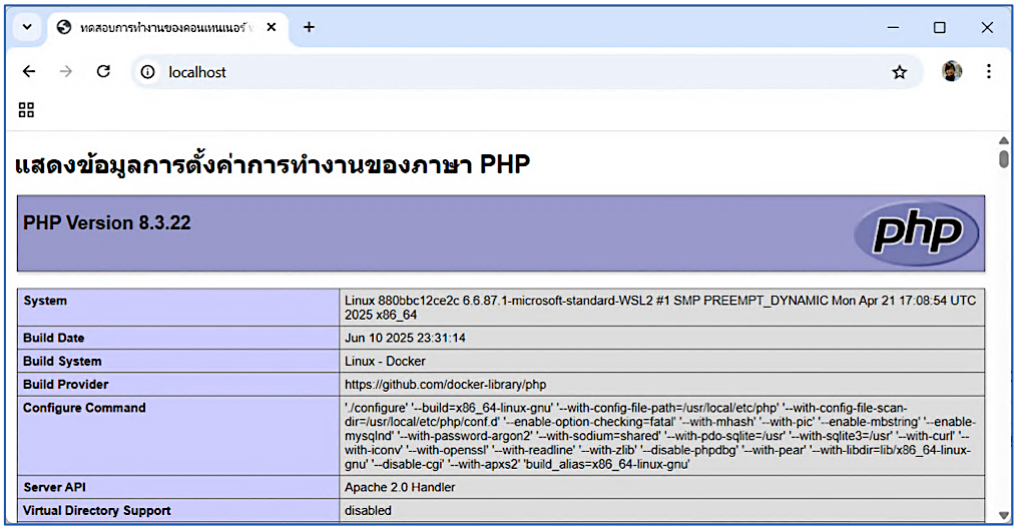
- การทดสอบการทำงานของคอนเทนเนอร์ webserver

การทดสอบส่วนนี้มีวัตถุประสงค์เพื่อยืนยันความสามารถในการให้บริการเว็บและการประมวลผลภาษา PHP ภายในคอนเทนเนอร์ webserver โดยสร้างไฟล์ชื่อ index.php ซึ่งภายในประกอบด้วยภาษา HTML และ PHP ตามที่แสดงด้านล่างนี้ด้วยโปรแกรม VSCode และบันทึกไว้ในโฟลเดอร์ source ที่อยู่ในเครื่องโฮสต์ ทั้งนี้ เนื่องจากในไฟล์ compose.yaml ได้กำหนดการเชื่อมโยงโฟลเดอร์ source เข้ากับไดเรกทอรี /var/www/html ภายในคอนเทนเนอร์ จึงทำให้คอนเทนเนอร์ webserver สามารถเข้าถึงไฟล์ index.php ได้

#	ไฟล์ C:\webapp\source\index.php
1	<!DOCTYPE html>
2	<html>
3	<head>
4	<meta charset="UTF-8">

```
5           <title>ทดสอบการทำงานของคอนเทนเนอร์ webserver</title>
6       </head>
7       <body>
8           <h1>แสดงข้อมูลการตั้งค่าการทำงานของภาษา PHP</h1>
9           <?php
10              phpinfo();
11          ?>
12      </body>
13  </html>
```

เมื่อสร้างไฟล์ index.php แล้ว ผู้พัฒนาสามารถเปิดเว็บเบราว์เซอร์และพิมพ์ URL เป็น http://localhost หรือ http://localhost:80 เพื่อเข้าถึงบริการเว็บที่ให้บริการโดยคอนเทนเนอร์ webserver หากคอนเทนเนอร์ทำงานได้อย่างถูกต้อง จะปรากฏหน้าเว็บที่มีเนื้อหาจากการใช้ภาษา HTML และผลจากฟังก์ชัน phpinfo() ในภาษา PHP ซึ่งแสดงข้อมูลการตั้งค่าของตัวแปลภาษา PHP ที่อยู่ภายในคอนเทนเนอร์ เช่น เวอร์ชันของ PHP, ค่ากำหนดต่าง ๆ และค่าตัวแปรสภาพแวดล้อม เป็นต้น ดังแสดงในภาพที่ 4.15



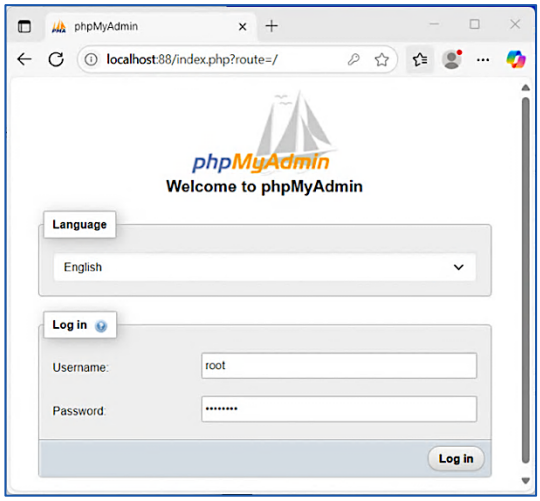
ภาพที่ 4.15 ผลลัพธ์การทำงานของไฟล์ index.php จากคอนเทนเนอร์ webserver

- การทดสอบการทำงานของคอนเทนเนอร์ dbmanager และ dbserver

การทดสอบในส่วนนี้เป็นการตรวจสอบว่าเว็บแอปพลิเคชัน phpMyAdmin ซึ่งให้บริการโดยคอนเทนเนอร์ dbmanager สามารถทำงานได้ถูกต้อง และสามารถเชื่อมต่อกับคอนเทนเนอร์ dbserver ซึ่งให้บริการระบบจัดการฐานข้อมูล MySQL ได้หรือไม่ โดยคอนเทนเนอร์ทั้งสองมีการเชื่อมโยงผ่านระบบเครือข่าย

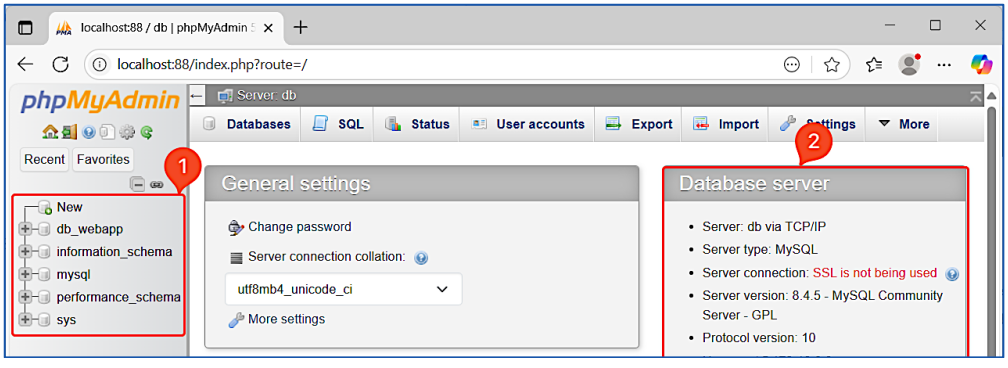
และการตั้งค่าตัวแปรสภาพแวดล้อม PMA_HOST: db ซึ่งเป็นการบ่งถึงการเชื่อมต่อกับบริการชื่อ db ที่กำหนดในไฟล์ compose.yaml

วิธีการทดสอบทำได้โดยเปิดเว็บเบราว์เซอร์ แล้วพิมพ์ URL เป็น http://localhost:88 เพื่อเข้าถึงเว็บแอปพลิเคชัน phpMyAdmin หากคอนเทนเนอร์ dbmanager ทำงานได้อย่างถูกต้อง จะปรากฏหน้าเข้าสู่ระบบของ phpMyAdmin ซึ่งให้กรอกข้อมูลการเข้าสู่ระบบฐานข้อมูลแสดงดังภาพที่ 4.16 โดยใช้ค่าที่กำหนดไว้ในไฟล์ compose.yaml ของบริการ db ได้แก่ ซึ่งในการทดสอบนี้จะใช้ผู้ดูแลระบบสูงสุดของระบบจัดการฐานข้อมูลมีบัญชีผู้ใช้เป็น root และมีรหัสผ่านเป็น rootpass ซึ่งรหัสผ่านนี้มีการกำหนดในค่าตัวแปรสภาพแวดล้อม MYSQL_ROOT_PASSWORD ของคอนเทนเนอร์ชื่อ dbserver ที่อยู่ในไฟล์ compose.yaml นั่นเอง



ภาพที่ 4.16 หน้าจอเข้าสู่ระบบของเว็บแอปพลิเคชัน phpMyAdmin

เมื่อเข้าสู่ระบบสำเร็จ จะปรากฏหน้าเว็บแอปพลิเคชัน phpMyAdmin สำหรับการทำงานร่วมกับระบบฐานข้อมูล MySQL ดังแสดงในภาพที่ 4.17



ภาพที่ 4.17 หน้าเว็บแอปพลิเคชัน phpMyAdmin ที่อยู่ในคอนเทนเนอร์ dbmanager

กลไกการทำงานของเว็บและหน้าเว็บต่าง ๆ เป็นผลมาจากการทำงานของคอนเทนเนอร์ dbmanager ส่วนรายการฐานข้อมูลที่แสดงในตำแหน่งหมายเลข 1 ของภาพรวมทั้งรายละเอียดของผู้ให้บริการระบบจัดการฐานข้อมูลในตำแหน่งหมายเลข 2 ของภาพ เป็นข้อมูลที่มาจากคอนเทนเนอร์ dbserver ซึ่งเป็นการยืนยันว่าคอนเทนเนอร์ dbmanager สามารถเชื่อมต่อกับระบบฐานข้อมูลที่ให้บริการโดยคอนเทนเนอร์ dbserver ได้อย่างถูกต้อง ทั้งนี้ สังเกตได้จากข้อมูลในตำแหน่งหมายเลข 2 ที่ระบุว่า Server: db via TCP/IP ซึ่ง db คือชื่อบริการที่กำหนดไว้ในไฟล์ compose.yaml และข้อความ Server version: 8.4.5 - MySQL Community Server ซึ่งเป็นเวอร์ชันที่กำหนดไว้ในไฟล์ compose.yaml นอกจากนี้ ในตำแหน่งหมายเลข 1 แสดงรายชื่อฐานข้อมูลที่มีอยู่ในระบบ ซึ่งรวมถึงฐานข้อมูล db_webapp ที่ได้กำหนดไว้ในค่าตัวแปรสภาพแวดล้อมของบริการ db ในไฟล์ compose.yaml เช่นกัน

4. สรุปท้ายบท

แนวคิดการจำลองเสมือนเป็นแนวทางสำคัญในการสร้างสภาพแวดล้อมการทำงานที่แยกอิสระจากกันในเครื่องคอมพิวเตอร์เครื่องเดียวกัน เพื่อเพิ่มประสิทธิภาพ ความยืดหยุ่น และความปลอดภัยในการใช้งาน โดยมีเทคโนโลยีหลักสองประเภท ได้แก่ เทคโนโลยีเครื่องเสมือนและเทคโนโลยีคอนเทนเนอร์ ซึ่งแต่ละเทคโนโลยีมีจุดเด่นและข้อจำกัดที่แตกต่างกัน

การเปรียบเทียบแนวทางการสร้างกลไกผู้ให้บริการเว็บแสดงให้เห็นว่าเทคโนโลยีคอนเทนเนอร์มีความเหมาะสมสำหรับการพัฒนาเว็บแอปพลิเคชันมากกว่าโปรแกรมสำเร็จรูปและเทคโนโลยีเครื่องเสมือน โดยเฉพาะในด้านความสามารถในการแยกสภาพแวดล้อมสำหรับแต่ละโครงการ ความยืดหยุ่นในการปรับแต่ง และความสามารถในการพกพา แม้ว่าจะมีความซับซ้อนในการเรียนรู้เริ่มต้นที่สูงกว่าโปรแกรมสำเร็จรูป แต่ก็ให้ประโยชน์ในระยะยาวและสอดคล้องกับโครงสร้างพื้นฐานของอุตสาหกรรมซอฟต์แวร์ในยุคปัจจุบัน

Docker Desktop เป็นแพลตฟอร์มเทคโนโลยีคอนเทนเนอร์ที่ได้รับการยอมรับอย่างกว้างขวางในอุตสาหกรรมซอฟต์แวร์ มีสถาปัตยกรรมแบบ Client/Server ที่ประกอบด้วยองค์ประกอบสำคัญ ได้แก่ Docker Client, Docker Daemon และ Docker Registry ซึ่งทำงานร่วมกันผ่าน REST API เพื่อจัดการแม่แบบของคอนเทนเนอร์ คอนเทนเนอร์ การเชื่อมโยงแหล่งเก็บข้อมูล และการจัดเก็บข้อมูลแยกออกจากวงจรชีวิตของคอนเทนเนอร์ ความสามารถเหล่านี้ทำให้ Docker Desktop เป็นเครื่องมือที่มีประสิทธิภาพในการสร้างสภาพแวดล้อมการพัฒนาที่สอดคล้องกับการใช้งานจริงและรองรับการทำงานแบบทีม

กรณีศึกษาการสร้าง LAMP Stack ด้วย Docker Desktop แสดงให้เห็นถึงประโยชน์ของการใช้หลักการแยกความรับผิดชอบของแต่ละคอนเทนเนอร์ โดยการแบ่งบทบาทออกเป็นคอนเทนเนอร์ webserver สำหรับผู้ให้บริการเว็บและตัวแปลภาษา PHP คอนเทนเนอร์ dbserver สำหรับระบบจัดการฐานข้อมูล และคอนเทนเนอร์ dbmanager สำหรับเว็บแอปพลิเคชันจัดการฐานข้อมูล การใช้ Docker Compose ช่วยให้สามารถกำหนดและจัดการคอนเทนเนอร์หลายหน่วยได้อย่างเป็นระบบผ่านไฟล์ compose.yaml ที่มีโครงสร้างชัดเจนและสามารถควบคุมการเชื่อมโยงพอร์ต การจัดเก็บข้อมูล และการเชื่อมต่อระหว่างคอนเทนเนอร์ได้อย่างมีประสิทธิภาพ

5. กิจกรรมท้ายบท

5.1 แนวคิดการจำลองเสมือนมีลักษณะเป็นอย่างไร? และมีเทคโนโลยีหลักใดบ้างที่อยู่ภายใต้แนวคิดนี้? จงอธิบายพอสังเขป

5.2 จงให้ความหมายของเทคโนโลยีเครื่องเสมือน พร้อมทั้งบอกประโยชน์และข้อจำกัดของเทคโนโลยีเสมือนในการพัฒนาเว็บแอปพลิเคชันพอสังเขป

5.3 จงให้ความหมายของเทคโนโลยีคอนเทนเนอร์ พร้อมทั้งบอกประโยชน์และข้อจำกัดของเทคโนโลยีคอนเทนเนอร์ในการพัฒนาเว็บแอปพลิเคชันพอสังเขป

5.4 เทคโนโลยีเครื่องเสมือนและเทคโนโลยีคอนเทนเนอร์แตกต่างกันอย่างไรในด้านสถาปัตยกรรม การใช้ทรัพยากร และความเหมาะสมกับงานพัฒนาเว็บแอปพลิเคชัน? จงอธิบายพอสังเขป

5.5 จงอธิบายส่วนประกอบของกลไกการทำงานของ Docker Desktop ต่าง ๆ ต่อไปนี้

5.5.1 Docker Daemon

5.5.2 แม่แบบของคอนเทนเนอร์

5.5.3 คอนเทนเนอร์

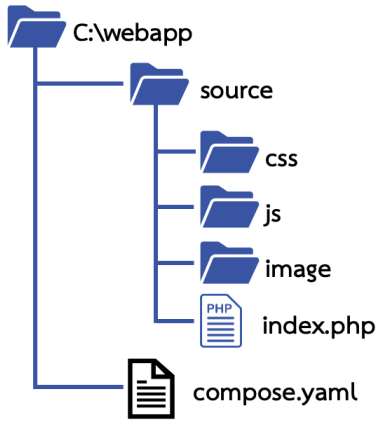
5.5.4 Bind Mount

5.5.5 Docker Volume

5.5.6 Docker Registry

5.6 ทดลองติดตั้งโปรแกรม Docker Desktop และทำการสร้างสภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชันด้วย LAMP Stack ตามขั้นตอนต่อไปนี้

5.6.1 สร้างโฟลเดอร์ให้มีโครงสร้างดังภาพที่ 4.18



ภาพที่ 4.18 โครงสร้างโฟลเดอร์สำหรับกิจกรรมท้ายบท

5.6.2 สร้างไฟล์ compose.yaml ในตำแหน่งดังภาพที่ 4.18 เพื่อใช้ในการสร้างบริการและคอนเทนเนอร์ต่อไปนี้

- บริการชื่อ db ใช้แม่แบบคอนเทนเนอร์ mysql:8.4.5 เพื่อสร้างคอนเทนเนอร์ชื่อ dbserver กำหนดให้มีการเปิดใช้งานพอร์ต 3306 เพื่อให้บริการอื่นสามารถเชื่อมต่อได้โดยไม่เปิดเผยพอร์ตสู่เครื่องโฮสต์ และเชื่อมโยงพื้นที่จัดเก็บข้อมูลภายในคอนเทนเนอร์กับ Docker Volume ชื่อ db_data รวมทั้งกำหนดค่าตัวแปรสภาพแวดล้อมดังนี้
  - รหัสผ่านผู้ดูแลระบบฐานข้อมูลเป็น rootpass
  - ฐานข้อมูลชื่อ db_webapp
  - ผู้ใช้ฐานข้อมูลชื่อ webapp_user
  - รหัสผ่านของผู้ใช้ฐานข้อมูลเป็น userpass
- บริการชื่อ web ใช้แม่แบบคอนเทนเนอร์ php:8.3-apache เพื่อสร้างคอนเทนเนอร์ชื่อ webserver กำหนดให้มีการเชื่อมโยงพอร์ต 80:80 และทำการเชื่อมโยงโฟลเดอร์ source ไปยัง /var/www/html ของคอนเทนเนอร์ webserver รวมทั้งมีการกำหนดให้บริการ web เริ่มต้นหลังจากบริการ db ทำงานแล้ว
- บริการชื่อ dbmanager ใช้แม่แบบคอนเทนเนอร์ phpmyadmin:5.2.2-apache เพื่อสร้างคอนเทนเนอร์ชื่อ phpmyadmin กำหนดให้มีการเชื่อมโยงพอร์ต 88:80 และกำหนดตัวแปรสภาพแวดล้อม PMA_HOST เป็น db ซึ่งเป็นชื่อบริการของคอนเทนเนอร์ฐานข้อมูล รวมทั้งมีการกำหนดให้บริการ dbmanager เริ่มต้นหลังจากบริการ db ทำงานแล้ว

5.6.3 สร้างไฟล์ index.php ในตำแหน่งดังภาพที่ 4.18 เพื่อใช้ในการทดสอบการทำงานของบริการเว็บ เนื้อหาในไฟล์มีดังนี้

#	ไฟล์ C:\webapp\source\index.php
1	<?php
2	echo " ทดสอบการทำงานของ Web Server ";
3	phpinfo();

5.6.4 สร้างคอนเทนเนอร์จากไฟล์ compose.yaml ที่ได้สร้างขึ้น

5.6.5 ตรวจสอบผลการสร้างคอนเทนเนอร์และทดสอบการทำงานของคอนเทนเนอร์ดังนี้

- ทดสอบการทำงานของคอนเทนเนอร์ให้บริการเว็บด้วยการเปิดเว็บเบราว์เซอร์และเข้าถึงด้วย http://localhost
- ทดสอบการทำงานของคอนเทนเนอร์ให้บริการฐานข้อมูลด้วยการเปิดเว็บเบราว์เซอร์และเข้าถึงด้วย http://localhost:88



Frontend: ภาษาสำหรับสร้างส่วนติดต่อผู้ใช้



ภาษา HTML



แท็ก HTML ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน



ภาษา CSS



การกำหนดลักษณะการแสดงผลด้วย CSS



ภาษา JavaScript



การจัดการข้อมูลและอิลิเมนต์ของหน้าเว็บด้วย JavaScript



ภาษา HTML

บทที่ 5 มุ่งเน้นการนำเสนอภาษา HTML ซึ่งเป็นภาษาหลักในการกำหนดโครงสร้างของส่วนหน้าของเว็บแอปพลิเคชัน เพื่อสร้างองค์ประกอบที่สามารถแสดงผลและโต้ตอบกับผู้ใช้ผ่านเว็บเบราว์เซอร์ได้อย่างเป็นระบบและสอดคล้องกับวัตถุประสงค์ของระบบ โดยเนื้อหาในบทนี้จะปูพื้นฐานเกี่ยวกับแนวคิด กลไก และโครงสร้างที่จำเป็นในการเขียนเอกสาร HTML อย่างถูกต้องตามมาตรฐาน เพื่อเป็นพื้นฐานสำหรับการเรียนรู้แท็ก HTML เฉพาะทางในบทถัดไป หัวข้อหลักในบทนี้ประกอบด้วย

1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML
2. การกำกับในภาษา HTML
3. ค่าเริ่มต้นการแสดงผลของอิลิเมนต์
4. โครงสร้างของภาษา HTML
5. การเขียนบรรทัดหมายเหตุ
6. สรุปท้ายบท
7. กิจกรรมท้ายบท

1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML

ความหมาย บทบาทและพัฒนาการของภาษา HTML มีดังต่อไปนี้

1.1 ความหมายและบทบาทของภาษา HTML

ภาษา HTML ย่อมาจาก HyperText Markup Language เป็นภาษามาตรฐานที่ใช้สำหรับการกำหนดโครงสร้างของเอกสารเว็บ โดยทำหน้าที่กำกับ (Mark) ว่าเนื้อหาส่วนใดในเอกสารมีบทบาทหรือหน้าที่อย่างไร เช่น หัวเรื่อง ย่อหน้า รายการ ลิงก์ หรือแบบฟอร์ม เป็นต้น เพื่อให้เว็บเบราว์เซอร์สามารถแสดงผลเนื้อหาเหล่านั้นได้อย่างถูกต้อง (สุพรรณิการ์ ย่องซื่อ, 2563)

คำว่า HyperText ในชื่อของภาษาหมายถึงความสามารถในการสร้างการเชื่อมโยงหรือลิงก์ (Link) ที่ใช้เชื่อมโยงไปยังหน้าเว็บหรือทรัพยากรลักษณะต่าง ๆ เช่น รูปภาพ ไฟล์เอกสาร หรือหน้าเว็บ เป็นต้น ไม่ว่าจะเป็นภายในเว็บไซต์เดียวกันหรือเว็บไซต์อื่น ๆ ซึ่งถือเป็นกลไกหลักของเวิลด์ไวด์เว็บ

ภาษา HTML ไม่ใช่ภาษาที่ใช้ในการประมวลผลแบบภาษาโปรแกรมทั่วไป เช่น ภาษา PHP ภาษา JavaScript หรือภาษา Python แต่เป็นภาษากำกับ (Markup Language) ชนิดหนึ่งที่อยู่ในกลุ่มภาษากำกับเชิงอธิบาย (Descriptive Markup Language) ซึ่งภาษากำกับเป็นภาษาที่ถูกออกแบบมาเพื่อใช้ในการกำหนดและแสดงผลข้อความในเอกสารโดยใช้แท็ก (Tag) เป็นตัวตัวกำกับ ซึ่งมีวัตถุประสงค์เพื่อกำกับโครงสร้าง ความหมาย หรือรูปแบบของเนื้อหานั้น ๆ (Mozilla Developer Network, n.d.a)

ในบริบทของการพัฒนาเว็บแอปพลิเคชัน ภาษา HTML ทำหน้าที่เป็นโครงสร้างหลักในการแสดงผลในเว็บเบราว์เซอร์ ภาษา HTML สามารถทำงานร่วมกับภาษา CSS ในการตกแต่งการแสดงผลและภาษา JavaScript ในการจัดการกับเหตุการณ์หรือการประมวลผลดังที่ได้กล่าวถึงในบทที่ 1 สำหรับการสร้างเอกสารเว็บที่ใช้ภาษา HTML จะใช้โปรแกรมประเภทสำหรับแก้ไขข้อความธรรมดาเขียนโครงสร้างและคำสั่งกำกับภาษา HTML และบันทึกไฟล์ให้มีนามสกุล .html หรือ .htm

1.2 พัฒนาการและมาตรฐานของภาษา HTML

จากเว็บไซต์ของ W3C (Berners-Lee, n.d.) และ WHATWG (WHATWG, 2024) ซึ่งเป็นองค์กรที่เกี่ยวข้องกับการสร้างและการกำกับมาตรฐานของภาษา HTML ได้กล่าวถึงพัฒนาการของภาษา HTML โดยเริ่มต้นขึ้นในปี พ.ศ. 2533 โดย Tim Berners-Lee ขณะปฏิบัติงานที่ศูนย์วิจัย CERN ประเทศสวิตเซอร์แลนด์ โดยมีวัตถุประสงค์เพื่อสร้างรูปแบบเอกสารที่สามารถเชื่อมโยงถึงกันได้ในลักษณะไฮเปอร์ลิงก์ผ่านเครือข่ายอินเทอร์เน็ต นับเป็นจุดเริ่มต้นของเทคโนโลยีที่เรียกว่าเวิลด์ไวด์เว็บ (World Wide Web)

ในช่วงแรก HTML ยังอยู่ในระยะทดลองภายในองค์กร ก่อนที่ในปี พ.ศ. 2538 การพัฒนาจะถูกส่งต่อไปยัง Internet Engineering Task Force (IETF) และต่อมาในปี พ.ศ. 2537 ได้มีการจัดตั้ง World Wide Web Consortium (W3C) เพื่อทำหน้าที่กำหนดมาตรฐานของเทคโนโลยีเว็บอย่างเป็นทางการ

ภายใต้การดูแลของ W3C ได้มีการเผยแพร่มาตรฐาน HTML 3.2 ในปี พ.ศ. 2540 ซึ่งถือเป็นมาตรฐาน HTML ฉบับแรกที่ได้รับการยอมรับอย่างแพร่หลาย และในปีเดียวกันยังได้เผยแพร่ HTML 4.0 ซึ่งรองรับการ

จัดรูปแบบด้วยภาษา CSS และสนับสนุนแนวคิดการจัดโครงสร้างเอกสารอย่างมีระบบ หลังจากนั้นภาษา HTML ได้หยุดการพัฒนาในระยะหนึ่ง เนื่องจากมีความพยายามพัฒนาแนวทางอื่นเพื่อตอบสนองความต้องการของระบบเว็บที่ซับซ้อนยิ่งขึ้น อย่างไรก็ตาม แนวทางดังกล่าวกลับไม่สามารถรองรับการพัฒนาเว็บให้สอดคล้องกับการใช้งานจริงในช่วงเวลานั้นได้อย่างเต็มที่

ในปี พ.ศ. 2547 กลุ่มผู้พัฒนาเทคโนโลยีเว็บจากองค์กรหลัก ได้แก่ Mozilla, Apple และ Opera ได้รวมตัวกันจัดตั้งกลุ่มงานที่ชื่อว่า Web Hypertext Application Technology Working Group หรือ WHATWG โดยมีเป้าหมายเพื่อฟื้นฟูและพัฒนาภาษา HTML ให้สามารถตอบสนองการใช้งานตามความก้าวหน้าทางเทคโนโลยีที่เปลี่ยนแปลงอย่างรวดเร็ว ทั้งในแง่ของความเข้ากันได้กับภาษารุ่นก่อนหน้า การรองรับการใช้งานแบบโต้ตอบการฝังสื่อ และการแสดงผลในอุปกรณ์พกพา แนวทางการพัฒนาของ WHATWG นำไปสู่การกำหนดมาตรฐานที่เรียกว่า HTML5 หรือ HTML รุ่นที่ 5 ซึ่งวางแนวทางการพัฒนาใหม่ให้มีความยืดหยุ่น เข้าใจตรงกันระหว่างเบราว์เซอร์ และสามารถขยายความสามารถได้อย่างต่อเนื่อง

ต่อมา WHATWG ได้ปรับรูปแบบการพัฒนาภาษา HTML ให้เป็นแบบ Living Standard ซึ่งเป็นมาตรฐานที่มีการปรับปรุงและพัฒนาอย่างต่อเนื่องโดยไม่แบ่งแยกเป็นเวอร์ชัน และในปี พ.ศ. 2550 องค์กร W3C ได้เข้าร่วมในการพัฒนาภาษา HTML ร่วมกับ WHATWG จนกระทั่งในปี พ.ศ. 2562 ทั้งสององค์กรได้ลงนามในข้อตกลงความร่วมมือเพื่อใช้แนวทางของ WHATWG เป็นแนวทางหลักในการกำหนดมาตรฐานภาษา HTML ที่ใช้อยู่ในปัจจุบันภายใต้ชื่อว่า HTML Living Standard

WHATWG ได้กำหนดแนวทางการพัฒนาภาษา HTML ในแนวทางที่เรียกว่า Living Standard ซึ่งเป็นแนวทางการพัฒนาที่ปรับปรุงอย่างต่อเนื่องอยู่เสมอโดยไม่แบ่งเป็นเวอร์ชันอย่างชัดเจน ต่อมาในปี พ.ศ. 2550 องค์กร W3C ได้เข้าร่วมในการพัฒนา HTML5 ร่วมกับ WHATWG โดยได้มีการทำงานคู่ขนานกันอยู่ระยะหนึ่ง จนกระทั่งในปี พ.ศ. 2562 ทั้งสององค์กรได้ลงนามข้อตกลงร่วมมือเพื่อรวมการพัฒนา HTML ให้เป็นมาตรฐานเดียว โดยอาศัยแนวทางของ WHATWG เป็นหลักและถูกใช้เป็นมาตรฐานในการสร้างเว็บ ซึ่งมาตรฐาน HTML ที่ใช้อ้างอิงอยู่ในขณะเขียนหนังสือเล่มนี้จึงอยู่ภายใต้ชื่อว่า HTML Living Standard

เนื่องจาก HTML ยังคงมีการปรับปรุงและพัฒนาอย่างต่อเนื่อง ผู้พัฒนาเว็บแอปพลิเคชันจึงควรติดตามความเปลี่ยนแปลงของมาตรฐานภาษา HTML จากแหล่งข้อมูลอย่างเป็นทางการ ทั้ง WHATWG และ W3C เพื่อให้สามารถพัฒนาเว็บแอปพลิเคชันที่มีความทันสมัย ปลอดภัย และรองรับการทำงานในสภาพแวดล้อมที่หลากหลายได้อย่างมีประสิทธิภาพ

2. การกำกับในภาษา HTML

สิ่งที่แสดงในเว็บเบราว์เซอร์เป็นเนื้อหาที่อาจอยู่ในหลายรูปแบบเช่น ข้อความ รูปภาพ ตาราง หัวข้อ หรือลิงก์ เป็นต้น ซึ่งเนื้อหาเหล่านี้จะใช้ภาษา HTML ในการกำกับว่าเนื้อหาส่วนใดมีบทบาทอย่างไรในเอกสารเว็บ จากภาพที่ 5.1 แสดงถึงตัวอย่างของเนื้อหาที่เป็นข้อความ ซึ่งในกรณีที่ไม่ได้มีการกำหนดบทบาทให้กับข้อความแต่ละส่วนจะทำให้การแสดงผลในเว็บเบราว์เซอร์เป็นข้อความที่เรียงต่อกันไปในลักษณะการแสดงผลทางด้านซ้ายของ

ภาพที่ 5.1 แต่หากมีการกำหนดบทบาทให้ข้อความแต่ละส่วน เช่น หัวเรื่องสำคัญระดับ 1 ถึงระดับ 3 หัวข้อแบบลำดับ และย่อหน้า เป็นต้น ก็จะทำให้เว็บเบราว์เซอร์แสดงผลได้เหมาะสมมากยิ่งขึ้นดังแสดงทางด้านขวาของภาพที่ 5.1 การกำหนดบทบาทให้กับเนื้อหานี้จะใช้แท็กในการกำหนด ซึ่งแท็กอาจมีการกำหนดคุณลักษณะเพิ่มเติมโดยการใช้แอตทริบิวต์ เช่น ความกว้าง ความยาว ชื่อ เป็นต้น เนื้อหาที่ถูกกำกับด้วยแท็กจะถือเป็นองค์ประกอบหนึ่งของเอกสารเว็บซึ่งเรียกว่า อิลิเมนต์ (Element)



ภาพที่ 5.1 ตัวอย่างบทบาทของเนื้อหาในเอกสารเว็บ

2.1 แท็ก

แท็ก (Tag) คือ คำสั่งกำกับที่ใช้ในการควบคุมหรือระบุลักษณะของเนื้อหาในเอกสาร HTML โดยคำสั่งกำกับจะถูกเขียนระหว่างเครื่องหมาย < และ > เช่น <h1>, <p> และ
 เป็นต้น แท็กมี 2 ลักษณะคือ แท็กคู่และแท็กเดี่ยว

2.1.1 แท็กคู่

แท็กคู่ (Paired Tag) เป็นแท็กที่ใช้แท็กเปิด (Opening Tag หรือ Start Tag) และแท็กปิด (Closing Tag หรือ End Tag) เพื่อการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของคำสั่งกำกับ โดยแท็กเปิดจะเขียนคำสั่งกำกับระหว่างเครื่องหมาย < และ > เป็นตำแหน่งเริ่มต้นที่ใช้บอกเว็บเบราว์เซอร์ว่าจะต้องดำเนินการอย่างไร และจะดำเนินการสิ่งนั้นจนกว่าจะพบจุดสิ้นสุดของการดำเนินการซึ่งจะเขียนระหว่างเครื่องหมาย </ และ > ซึ่งเรียกว่าแท็กปิด แท็กคู่มีรูปแบบดังต่อไปนี้

แท็กเปิด	<คำสั่งกำกับ>	แท็กปิด	</คำสั่งกำกับ>

ในตัวอย่างที่ 1 เป็นตัวอย่างของการกำหนดบทบาทที่ใช้แท็กเปิดและแท็กปิด (อ้างถึงตัวอย่างในภาพที่ 5.1 ประกอบ) เพื่อกำหนดให้ข้อความ *บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน* มีบทบาทเป็นหัวเรื่องสำคัญระดับ 1 โดยจะต้องมีการกำหนดจุดเริ่มต้นของการเป็นหัวเรื่องสำคัญระดับ 1 ตั้งแต่หน้าคำว่า *บทที่ 1* และมีจุดสิ้นสุดอยู่ที่ท้ายคำว่า *แอป-พลิเคชัน* ซึ่งคำสั่งกำกับที่ใช้ในการระบุว่าเป็นหัวเรื่องสำคัญระดับ 1 คือ h1 (Heading 1) เมื่อเขียนในเอกสารเว็บจะได้ดังตัวอย่างที่ 1

ตัวอย่างที่ 1

```
<h1>บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน</h1>
```

2.1.2 แท็กเดี่ยว

แท็กเดี่ยว (Single Tags หรือ Self-closing Tags) เป็นแท็กที่มีเพียงแท็กเปิดเพียงอย่างเดียวเท่านั้นโดยไม่มีแท็กปิดประกอบ แท็กเดี่ยวเป็นแท็กที่ต้องการให้ผลของคำสั่งกำกับมีผล ณ จุดที่แท็กเดี่ยวปรากฏอยู่ เช่น การขึ้นบรรทัดใหม่ (ใช้แท็ก <br>) เป็นการกระทำที่ไม่จำเป็นต้องมีการกำหนดจุดเริ่มต้นและจุดสิ้นสุด ต้องการเพียงแค่ทราบจุดที่ต้องการขึ้นบรรทัดใหม่เท่านั้น หรือการแสดงเนื้อหาที่เป็นรูปภาพ (ใช้แท็ก <img>) ก็ไม่จำเป็นต้องทราบจุดเริ่มต้นและจุดสิ้นสุดเช่นกัน เป็นต้น ตัวอย่างของการใช้แท็กเดี่ยวแสดงดังตัวอย่างที่ 2 และ 3

ตัวอย่างที่ 2

```
เนื้อหาบรรทัดที่ 1<br>เนื้อหาบรรทัดที่ 2
```

จากตัวอย่างที่ 2 เมื่อเว็บเบราว์เซอร์แสดงผลลัพธ์จะทำให้ข้อความคำว่า *เนื้อหาบรรทัดที่ 1* อยู่คนละบรรทัดกับข้อความคำว่า *เนื้อหาบรรทัดที่ 2*

ตัวอย่างที่ 3

```
รูปภาพประกอบ <img  src="room.jpg">
```

จากตัวอย่างที่ 3 เป็นการแสดงข้อความพร้อมด้วยรูปภาพ โดยเมื่อเว็บเบราว์เซอร์แสดงผลลัพธ์รูปภาพชื่อ webapp.jpg ก็จะแสดงต่อท้ายข้อความคำว่า *รูปภาพประกอบ* (หากพื้นท้ายข้อความมีไม่เพียงพอกับขนาดของรูปภาพก็จะแสดงรูปภาพในบรรทัดถัดไป) ทั้งนี้ส่วน *src="webapp.jpg"* ของแท็ก img เป็นส่วนที่เรียกว่าแอตทริบิวต์ซึ่งจะได้กล่าวถึงในหัวข้อถัดไป

2.2 แอตทริบิวต์

แอตทริบิวต์ (Attribute) คือ คุณลักษณะหรือข้อมูลเพิ่มเติมที่ใช้ประกอบคำสั่งกำกับ เช่น ขนาด สี ตำแหน่ง ลิงก์ ชื่อไฟล์ หรือค่าอื่น ๆ ที่ใช้ประกอบการประมวลผลของเว็บเบราว์เซอร์ แอตทริบิวต์จะเขียนอยู่ในแท็กเปิดเสมอโดยเขียนต่อจากคำสั่งกำกับ ซึ่งมีรูปแบบดังต่อไปนี้

รูปแบบสำหรับแท็กคู่

<คำสั่งกำกับ แอตทริบิวต์_1="ค่า" แอตทริบิวต์_2="ค่า" ...> </คำสั่งกำกับ>

รูปแบบสำหรับแท็กเดี่ยว

<คำสั่งกำกับ แอตทริบิวต์_1="ค่า" แอตทริบิวต์_2="ค่า" ...>

จากรูปแบบของการกำหนดแอตทริบิวต์ให้กับคำสั่งกำกับข้างต้น จะมีการกำหนดที่เหมือนกันทั้งในแท็กคู่และแท็กเดี่ยวซึ่งจะกำหนดในแท็กเปิดเท่านั้น ท้ายคำสั่งกำกับจะมีการระบุชื่อ (Attribute Name) และค่าของแอตทริบิวต์ (Attribute Value) ซึ่งใช้เครื่องหมาย = ในการกำหนดค่าให้กับแอตทริบิวต์ นอกจากนี้ ในแต่ละคำสั่งกำกับสามารถมีแอตทริบิวต์ได้มากกว่า 1 แอตทริบิวต์ ดังแสดงในตัวอย่างที่ 4 และ 5

ตัวอย่างที่ 4

```
<a  href="https://www.ajnok.com"  target="_blank">เว็บ AJNOK.COM</a>
```

จากตัวอย่างที่ 4 เป็นการกำหนดแอตทริบิวต์ให้กับคำสั่งกำกับ a ซึ่งเป็นคำสั่งกำกับที่ใช้สำหรับการสร้างลิงก์ โดยมีการกำหนดแอตทริบิวต์จำนวน 2 แอตทริบิวต์ ดังนี้

- แอตทริบิวต์ชื่อ href ใช้สำหรับระบุ URL ที่ต้องการเชื่อมโยงไปถึง โดยมีการกำหนดค่าของแอตทริบิวต์เป็น https://www.ajnok.com
- แอตทริบิวต์ชื่อ target ใช้สำหรับกำหนดเป้าหมายของการแสดงผลเมื่อผู้ใช้คลิกลิงก์ โดยมีการกำหนดค่าของแอตทริบิวต์เป็น _blank หมายถึงให้เปิดลิงก์ในแท็บใหม่

ตัวอย่างที่ 5

```
<img  src="webapp.jpg"  alt="ภาพของหนังสือการพัฒนาเว็บแอปพลิเคชัน">
```

จากตัวอย่างที่ 5 เป็นการกำหนดแอตทริบิวต์ให้กับคำสั่งกำกับ img ซึ่งเป็นคำสั่งกำกับที่ใช้สำหรับการแสดงรูปภาพ โดยมีการกำหนดแอตทริบิวต์จำนวน 2 แอตทริบิวต์ ดังนี้

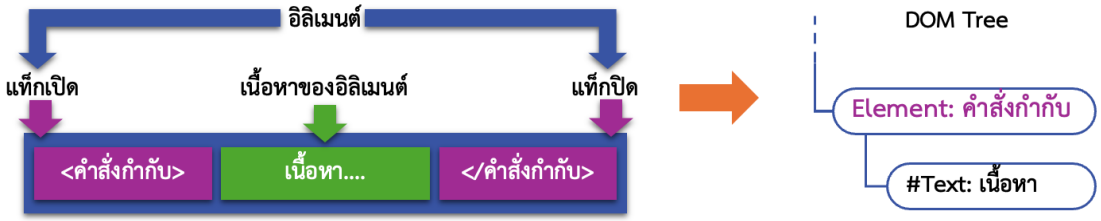
- แอตทริบิวต์ชื่อ src มาจากคำว่า Source ใช้สำหรับกำหนดเส้นทางการเข้าถึงรูปภาพ (Path) โดยมีการกำหนดค่าของแอตทริบิวต์เป็น webapp.jpg
- แอตทริบิวต์ชื่อ alt มาจากคำว่า Alternative Text ใช้สำหรับระบุข้อความสำรองที่จะแสดงแทนรูปภาพในกรณีที่ภาพไม่สามารถโหลดได้ เช่น กรณีอินเทอร์เน็ตช้า หรือผู้ใช้ปิดการแสดงผลภาพ โดยมีการกำหนดค่าของแอตทริบิวต์เป็น ภาพของหนังสือการพัฒนาเว็บแอปพลิเคชัน

2.3 อิลิเมนต์

อิลิเมนต์ (Element) คือ องค์ประกอบที่เกิดขึ้นจากการกำกับเนื้อหาด้วยแท็ก เป็นหน่วยโครงสร้างข้อมูลของเอกสารเว็บที่จะถูกนำไปแปลงให้เป็นโครงสร้างต้นไม้ DOM Tree โดยกลไกการแสดงผลของเว็บเบราว์เซอร์ตามที่ได้กล่าวถึงในบทที่ 1 เพื่อนำไปสู่การแสดงผลในเว็บเบราว์เซอร์ได้อย่างถูกต้อง

2.3.1 อิลิเมนต์ที่มีเนื้อหา

อิลิเมนต์ที่มีเนื้อหาเป็นอิลิเมนต์ที่เกิดขึ้นเมื่อใช้แท็กคู่ในการกำกับเนื้อหา โดยกลไกการแสดงผลของเว็บเบราว์เซอร์จะมีการแปลงให้เป็นโครงสร้างต้นไม้ DOM Tree ลักษณะแสดงดังภาพที่ 5.2



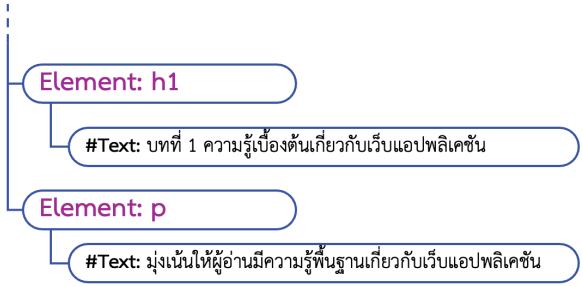
ภาพที่ 5.2 อิลิเมนต์ที่มีเนื้อหา

จากภาพที่ 5.2 เมื่อมีการกำกับเนื้อหาโดยใช้แท็กคู่ กลไกการแสดงผลของเว็บเบราว์เซอร์ก็จะทำการพิจารณาจากแท็กเปิดไปจนถึงแท็กปิด ซึ่งประกอบด้วยแท็กเปิด เนื้อหา และแท็กปิด โดยทั้งหมดนี้ถือเป็นอิลิเมนต์เดียวในมุมมองของกลไกการแสดงผล และจะถูกสร้างเป็นองค์ประกอบหนึ่งในโครงสร้างต้นไม้ DOM Tree ดังตัวอย่างที่ 6 ต่อไปนี้

ตัวอย่างที่ 6

```
<h1>บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน</h1>
<p>มุ่งเน้นให้ผู้อ่านมีความรู้พื้นฐานเกี่ยวกับเว็บแอปพลิเคชัน</p>
```

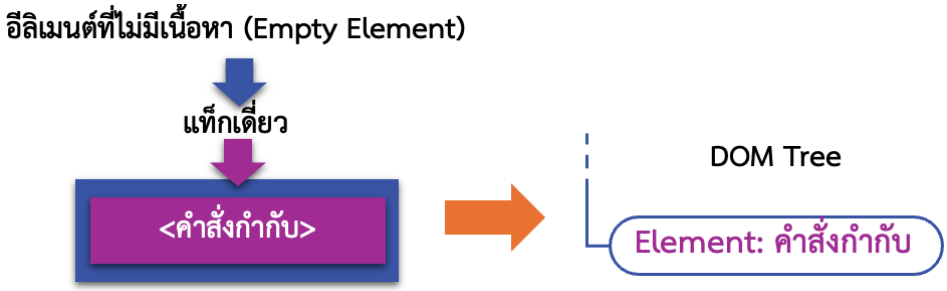
DOM Tree



จากตัวอย่างที่ 6 กลไกการแสดงผลของเว็บเบราว์เซอร์จะทำการสร้างอิลิเมนต์เป็นโครงสร้างต้นไม้ DOM Tree จำนวน 2 อิลิเมนต์ โดยอิลิเมนต์แรกเป็น อิลิเมนต์ h1 ซึ่งมีเนื้อหาเป็นข้อความว่า *บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน* ส่วนอิลิเมนต์ที่สองเป็นอิลิเมนต์ p ที่มีเนื้อหาเป็น *มุ่งเน้นให้ผู้อ่านมีความรู้พื้นฐานเกี่ยวกับเว็บแอปพลิเคชัน*

2.3.2 อิลิเมนต์ที่ไม่มีเนื้อหา

อิลิเมนต์ที่ไม่มีเนื้อหาเป็นอิลิเมนต์ที่เกิดจากการใช้แท็กเดี่ยวในการกำกับเนื้อหา โดยกลไกการแสดงผลของเว็บเบราว์เซอร์จะมีการแปลงให้เป็นโครงสร้างต้นไม้ลักษณะแสดงดังภาพที่ 5.3



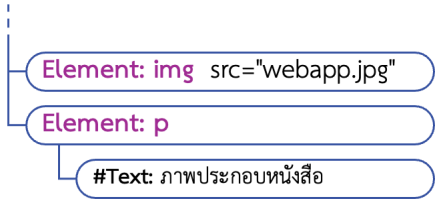
ภาพที่ 5.3 อิลิเมนต์ที่ไม่มีเนื้อหา

จากภาพที่ 5.3 เมื่อมีการกำกับเนื้อหาโดยใช้แท็กเดี่ยว กลไกการแสดงผลของเว็บเบราว์เซอร์ก็จะทำการพิจารณาเฉพาะแท็กเดี่ยวเท่านั้น และจะถูกสร้างเป็นองค์ประกอบหนึ่งในโครงสร้างต้นไม้ DOM Tree ดังตัวอย่างที่ 7 ต่อไปนี้

ตัวอย่างที่ 7

```
<img  src="webapp.jpg>
<p>ภาพประกอบหนังสือ</p>
```

DOM Tree

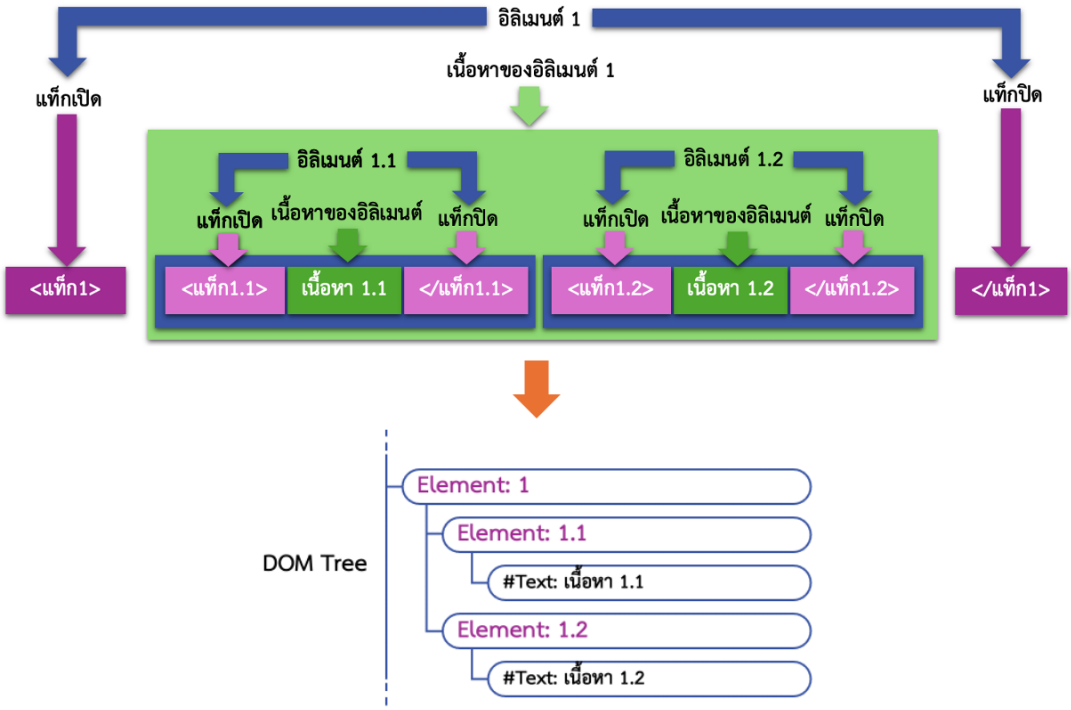


จากตัวอย่างที่ 7 กลไกการแสดงผลของเว็บเบราว์เซอร์จะทำการสร้างอิลิเมนต์เป็นโครงสร้างต้นไม้ DOM Tree จำนวน 2 อิลิเมนต์ โดยอิลิเมนต์แรกเป็นอิลิเมนต์ img จะเห็นว่าเป็นอิลิเมนต์ที่ไม่มีเนื้อหา ส่วนอิลิเมนต์ที่สองเป็นอิลิเมนต์ p ที่มีเนื้อหาเป็น *ภาพประกอบหนังสือ*

2.3.3 อิลิเมนต์ที่ซ้อนกัน

อิลิเมนต์ที่ซ้อนกัน (Nested Element) คือรูปแบบของการเขียนรหัสคำสั่งภาษา HTML ที่มีการวางอิลิเมนต์หนึ่งไว้ภายในอีกอิลิเมนต์หนึ่ง โดยมีลักษณะเหมือนโครงสร้างแบบซ้อนชั้น (Nested Structure) ซึ่งทั้งสองอิลิเมนต์ต้องใช้แท็กเปิดและแท็กปิดให้ถูกลำดับ และไม่ซ้อนข้ามกัน

ลักษณะตัวอย่างของอิลิเมนต์ที่ซ้อนกันแสดงดังภาพที่ 5.4 ซึ่งประกอบด้วย อิลิเมนต์หลักคืออิลิเมนต์ 1 มีขอบเขตการพิจารณาตั้งแต่ <แท็ก1> ไปจนถึง </แท็ก1> โดยที่จะมองสิ่งต่าง ๆ ที่อยู่ระหว่างแท็กทั้งคู่นี้ว่าเป็นเนื้อหาของอิลิเมนต์ 1



ภาพที่ 5.4 อิลิเมนต์ที่ซ้อนกัน

เมื่อพิจารณาภายในเนื้อหาของอิลิเมนต์ 1 ก็จะพบว่าประกอบด้วย 2 อิลิเมนต์ คือ อิลิเมนต์ 1.1 และ อิลิเมนต์ 1.2 ตามลำดับ โดยอิลิเมนต์ทั้งสองนี้มีความสัมพันธ์เป็นอิลิเมนต์ลูก (Child Element) ของอิลิเมนต์ 1

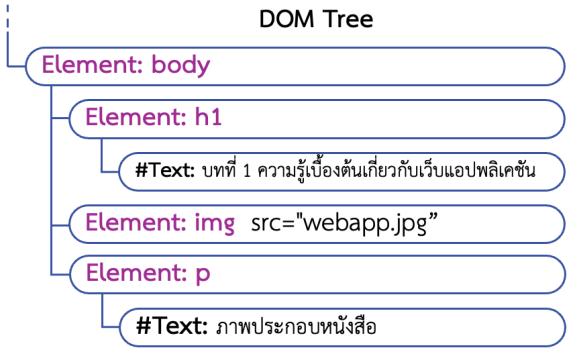
ในทางกลับกันอิลิเมนต์ 1 จะเรียกว่าเป็นอิลิเมนต์แม่ของอิลิเมนต์ 1.1 และ 1.2 ส่วนความสัมพันธ์ระหว่างอิลิเมนต์ 1.1 และ 1.2 เป็นอิลิเมนต์พี่น้อง (Sibling Element)

ตัวอย่างที่ 8 เป็นตัวอย่างของการเขียนภาษา HTML แสดงให้เห็นการสร้างอิลิเมนต์แม่โดยใช้แท็ก <body> และ </body> เพื่อกำกับขอบเขตของเนื้อหาหลักภายในหน้าเว็บ ซึ่งภายในอิลิเมนต์แม่ดังกล่าวประกอบด้วยอิลิเมนต์ลูกจำนวน 3 อิลิเมนต์ ได้แก่ h1, img และ p ที่แต่ละแท็กมีบทบาทเฉพาะในการแสดงผลเนื้อหาบนหน้าเว็บ

ตัวอย่างที่ 8

```
<body>
    <h1>บทที่ 5</h1>
    <h2>ภาษา HTML</h2>
    <img  src="html.png"  alt="รูปภาพ HTML">
</body>
```

DOM Tree



จากรหัสคำสั่งในตัวอย่างที่ 8 กลไกการแสดงผลของเว็บเบราว์เซอร์จะทำการสร้างอิลิเมนต์เป็นโครงสร้างต้นไม้ DOM Tree ซึ่งประกอบด้วยอิลิเมนต์ลูก 3 รายการภายใต้อิลิเมนต์ body เรียงตามลำดับดังนี้

- อิลิเมนต์ h1 มีเนื้อหาเป็น บทที่ 1 ความรู้เบื้องต้นเกี่ยวกับเว็บแอปพลิเคชัน
- อิลิเมนต์ img ซึ่งเป็นอิลิเมนต์ที่ไม่มีเนื้อหาและมีแอตทริบิวต์ src=“webapp.jpg” สำหรับแสดงภาพชื่อ webapp.jpg
- อิลิเมนต์ p มีเนื้อหาเป็น ภาพประกอบหนังสือ

ทั้งสามอิลิเมนต์ข้างต้นถือเป็นอิลิเมนต์พี่น้องที่อยู่ในลำดับเดียวกันภายใต้อิลิเมนต์แม่คือ body ซึ่งแสดงให้เห็นโครงสร้าง DOM Tree ที่จะนำไปใช้ในการแสดงผลในหน้าเว็บเบราว์เซอร์

3. ค่าเริ่มต้นการแสดงผลของอิลิเมนต์

ค่าเริ่มต้นการแสดงผลของอิลิเมนต์ เป็นลักษณะพฤติกรรมที่เว็บเบราว์เซอร์กำหนดให้อิลิเมนต์แต่ละประเภทแสดงผลบนหน้าเว็บโดยอัตโนมัติโดยไม่ต้องใช้ CSS ซึ่งมีผลต่อการจัดตำแหน่งและการเรียงตัวของเนื้อหาในเอกสาร HTML โดยสามารถแบ่งค่าเริ่มต้นของการแสดงผลออกเป็น 2 ประเภทหลัก ได้แก่ อิลิเมนต์แบบ Block-Level และอิลิเมนต์แบบ Inline

3.1 อิลิเมนต์แบบ Block-Level

อิลิเมนต์แบบ Block-Level จะเริ่มต้นแสดงผลในบรรทัดใหม่โดยอัตโนมัติ และเบราว์เซอร์จะเพิ่มระยะห่าง (margin) ด้านบนและด้านล่างของอิลิเมนต์โดยปริยาย อีกทั้งอิลิเมนต์ประเภทนี้ยังมีลักษณะการแสดงผลที่กินพื้นที่เต็มความกว้างขององค์ประกอบแม่ ซึ่งหมายความว่า อิลิเมนต์อื่นจะไม่สามารถอยู่เคียงกันในแนวนอนได้ในระดับเดียวกัน ตัวอย่างที่พบบ่อยของอิลิเมนต์ประเภทนี้ เช่น อิลิเมนต์ div เป็นอิลิเมนต์ที่ใช้ในการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา อิลิเมนต์ p เป็นอิลิเมนต์ที่ใช้ในการกำหนดย่อหน้า และอิลิเมนต์ table เป็นอิลิเมนต์ที่ใช้ในการสร้างตาราง เป็นต้น

3.2 อิลิเมนต์แบบ Inline

อิลิเมนต์แบบ Inline จะไม่เริ่มต้นในบรรทัดใหม่ และมีลักษณะการแสดงผลที่ใช้พื้นที่เฉพาะส่วนของเนื้อหาภายในอิลิเมนต์เท่านั้น กล่าวคือ จะใช้พื้นที่ในแนวนอนตามความยาวของเนื้อหาโดยไม่ขยายเต็มความกว้างขององค์ประกอบแม่ อิลิเมนต์ประเภทนี้จึงสามารถแสดงผลต่อเนื่องในบรรทัดเดียวกับอิลิเมนต์อื่นที่เป็นอิลิเมนต์แบบ Inline เหมือนกันได้ เช่น การวางช่องกรอกข้อมูลไว้ข้างป้ายข้อความ หรือการวางรูปภาพหลายๆ ภาพต่อเนื่องกัน

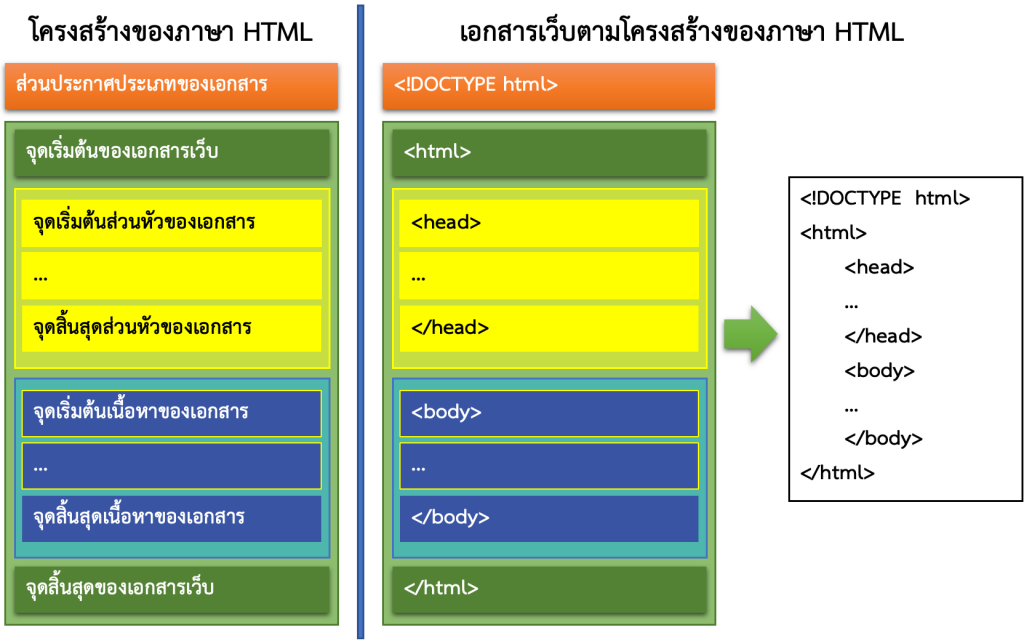
ตัวอย่างอิลิเมนต์แบบ Inline ที่พบบ่อย เช่น อิลิเมนต์ span เป็นอิลิเมนต์ที่ใช้ในการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา อิลิเมนต์ a เป็นอิลิเมนต์ที่ใช้ในการสร้างลิงก์ และอิลิเมนต์ img เป็นอิลิเมนต์ที่ใช้ในการแสดงรูปภาพ เป็นต้น อิลิเมนต์เหล่านี้มักใช้ในการจัดการเนื้อหาเฉพาะจุด

สิ่งที่ควรคำนึงในการใช้อิลิเมนต์แบบ Inline คือ อิลิเมนต์แบบ Inline ไม่สามารถมีอิลิเมนต์แบบ Block-Level เป็นอิลิเมนต์ลูกได้ ทั้งนี้เพื่อรักษาความถูกต้องของโครงสร้าง HTML และหลีกเลี่ยงปัญหาในการแสดงผลบนเว็บเบราว์เซอร์

4. โครงสร้างของภาษา HTML

ในทุกๆ ภาษาคอมพิวเตอร์จะมีการกำหนดโครงสร้างของภาษาที่แตกต่างกันออกไป เช่นเดียวกับภาษา HTML ที่ได้กำหนดรูปแบบโครงสร้างของเอกสารไว้อย่างชัดเจน เพื่อให้เว็บเบราว์เซอร์สามารถตีความและแสดงผลเนื้อหาได้อย่างถูกต้องตามมาตรฐาน โครงสร้างที่ถูกต้องยังช่วยให้การพัฒนาเว็บแอปพลิเคชันมีความเป็นมาตรฐานและง่ายต่อการบำรุงรักษา รวมถึงรองรับการทำงานร่วมกับเทคโนโลยีอื่นๆ เช่น CSS และ JavaScript ได้

โครงสร้างของภาษา HTML ประกอบด้วย 1) ส่วนประกาศชนิดของเอกสาร 2) จุดเริ่มต้นและจุดสิ้นสุดของเอกสารเว็บ 3) จุดเริ่มต้นและจุดสิ้นสุดส่วนหัวของเอกสาร และ 4) จุดเริ่มต้นและจุดสิ้นสุดเนื้อหาของเอกสาร แสดงดังภาพที่ 5.5



ภาพที่ 5.5 โครงสร้างภาษา HTML

4.1 ส่วนประกาศประเภทของเอกสาร

ส่วนประกาศประเภทของเอกสาร (Document Type Declaration) เป็นส่วนกำหนดข้อมูลที่แจ้งให้เว็บเบราว์เซอร์ทราบว่าเอกสารที่กำลังอ่านอยู่นั้นเป็นเอกสารเว็บ ที่เขียนตามมาตรฐานของภาษา HTML รุ่นใด ซึ่งมีผลต่อการตีความและแสดงผลเอกสารบนเว็บเบราว์เซอร์อย่างถูกต้อง ส่วนประกาศประเภทของเอกสารจะต้องอยู่ในบรรทัดแรกสุดของเอกสารเว็บเสมอ โดยรูปแบบส่วนประกาศประเภทของเอกสารตามมาตรฐาน HTML Living Standard มีดังนี้

รูปแบบ

```
<!DOCTYPE  html>
```

จากรูปแบบของส่วนประกาศประเภทของเอกสารข้างต้น ส่วนประกาศนี้ไม่จัดว่าเป็นแท็ก แต่เป็นข้อความที่ใช้ให้ข้อมูลแก่เว็บเบราว์เซอร์เพื่อระบุประเภทของเอกสารที่ต้องการให้เบราว์เซอร์ตีความโดยใช้คำสำคัญคำว่า DOCTYPE สำหรับการแจ้งข้อมูลแก่เว็บเบราว์เซอร์ ส่วนคำว่า html เป็นการระบุว่าเอกสารที่เว็บเบราว์เซอร์กำลังอ่านอยู่นี้สร้างขึ้นตามมาตรฐาน HTML Living Standard

การประกาศ <!DOCTYPE html> ไม่เพียงเป็นการระบุชนิดของเอกสารว่าเป็น HTML ตามมาตรฐาน HTML Living Standard แต่ยังมีบทบาทสำคัญในการกำหนดว่าเว็บเบราว์เซอร์จะตีความและแสดงผลเอกสารในโหมด No-Quirks Mode โดยกลไกของเว็บเบราว์เซอร์มีโหมดการตีความและแสดงผล 3 โหมด (Mozilla Developer Network, n.d.b; WHATWG, 2024) ดังนี้

4.1.1 Quirks Mode

Quirks Mode เป็นโหมดการทำงานที่เว็บเบราว์เซอร์ใช้ในการตีความและแสดงผลเอกสารเว็บโดยไม่ปฏิบัติตามข้อกำหนดของมาตรฐานที่เกี่ยวข้องอย่างเคร่งครัด ทั้งนี้เพื่อให้สามารถแสดงผลเว็บไซต์รุ่นเก่าที่ไม่ได้เขียนตามมาตรฐานของภาษา HTML อย่างถูกต้อง การจัดวางโครงร่างหน้าเว็บในโหมดนี้จะเลียนแบบพฤติกรรมของเว็บเบราว์เซอร์รุ่นเก่า เช่น Netscape Navigator 4 และ Internet Explorer 5 เพื่อรองรับเว็บไซต์ที่ถูกพัฒนาขึ้นในช่วงก่อนการยอมรับมาตรฐานเว็บ

4.1.2 No-Quirks Mode

No-Quirks Mode หรือเดิมถูกเรียกว่า Standards Mode เป็นโหมดการทำงานที่เว็บเบราว์เซอร์ตีความและแสดงผลเอกสารเว็บ ตามมาตรฐานที่กำหนดโดยองค์กร W3C และ WHATWG อย่างเคร่งครัด เป็นโหมดที่ควรใช้ในการพัฒนาเว็บแอปพลิเคชันยุคใหม่ เนื่องจากให้ผลลัพธ์ที่มีความถูกต้องตามมาตรฐานของภาษา HTML ช่วยให้การแสดงผลในเว็บเบราว์เซอร์ที่แตกต่างกันมีความถูกต้อง และสามารถทำงานร่วมกับเทคโนโลยีใหม่ ๆ ได้อย่างมีประสิทธิภาพ

4.1.3 Limited-Quirks Mode

Limited-Quirks Mode หรือเดิมถูกเรียกว่า Almost-Standards Mode เป็นโหมดที่มีลักษณะคล้ายกับ No-Quirks Mode หรือโหมดมาตรฐานเกือบทั้งหมด แต่มีการยกเว้นในบางกรณีเล็กน้อย เช่น การแสดงผลของภาพภายในเซลล์ของตาราง โดยการทำงานของเว็บเบราว์เซอร์ในโหมดนี้จะประมวลผลในลักษณะเดียวกับที่พบใน Quirks Mode ซึ่งมีพฤติกรรมการทำงานที่สอดคล้องกับเบราว์เซอร์เวอร์ชันเก่า เช่น Internet Explorer 7 และเวอร์ชันก่อนหน้า พฤติกรรมการทำงานดังกล่าวมีความสำคัญต่อการรักษารูปแบบการจัดหน้าเว็บที่อาศัยการจัดวางภาพในตารางแบบแยกส่วน (Sliced-images-in-tables layouts) ซึ่งเป็นเทคนิคที่นิยมใช้ในเว็บไซต์ยุคก่อน การที่โหมดนี้รองรับการจัดวางลักษณะดังกล่าวทำให้หน้าเว็บเหล่านี้ยังสามารถแสดงผลได้อย่างถูกต้องในเว็บเบราว์เซอร์สมัยใหม่ โหมดนี้ได้รับการรองรับในเว็บเบราว์เซอร์หลายค่ายซึ่งสะท้อนให้เห็นถึงความพยายามของผู้พัฒนาเบราว์เซอร์ในการรักษาความเข้ากันได้ย้อนหลัง (Backward Compatibility) ในขณะเดียวกันก็ยังคงรักษาแนวทางตามมาตรฐานเว็บร่วมด้วย

การระบุ DOCTYPE ที่ถูกต้องจะทำให้เว็บเบราว์เซอร์เลือกแสดงผลเอกสารในโหมด No-Quirks Mode ในทางตรงกันข้าม หากไม่มีการระบุ DOCTYPE หรือระบุในรูปแบบที่ไม่ถูกต้อง เบราว์เซอร์อาจเลือกใช้ Quirks Mode หรือ Limited-Quirks Mode แทน ซึ่งอาจทำให้การแสดงผลไม่ถูกต้องตามมาตรฐาน

4.2 จุดเริ่มต้นและจุดสิ้นสุดของเอกสารเว็บ

หลังจากส่วนประกาศประเภทของเอกสารแล้ว ส่วนต่อไปเป็นการกำหนดของเขตของเอกสารเว็บ โดยจะมีจุดเริ่มต้นและจุดสิ้นสุดที่ใช้บ่งบอกถึงขอบเขตของเอกสารซึ่งจะใช้แท็กต่อไปนี้ในการกำหนดจุดเริ่มต้นและจุดสิ้นสุดดังนี้

จุดเริ่มต้น	<html>	จุดสิ้นสุด	</html>

จุดเริ่มต้นและจุดสิ้นสุดที่ใช้แท็กเป็นตัวกำหนดขอบเขตของเอกสารเว็บ จะมีการกำหนดขอบเขตเพียงหนึ่งชุดหรือหนึ่งองค์ประกอบเท่านั้น (องค์ประกอบหมายถึงตำแหน่งของแท็กเริ่มต้นไปจนถึงแท็กสิ้นสุด) ซึ่งระหว่างจุดเริ่มต้นและจุดสิ้นสุดจะประกอบไปด้วยองค์ประกอบหลักของเอกสารเว็บ อีกสององค์ประกอบ คือ องค์ประกอบ head และองค์ประกอบ body ซึ่งจะได้กล่าวถึงในหัวข้อถัดไป

จากแนวปฏิบัติที่ดีในการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของเอกสารเว็บ ควรมีการระบุแอตทริบิวต์ภายในแท็กเปิด <html> เพื่อบอกให้เว็บเบราว์เซอร์และเครื่องมือค้นหาเข้าใจได้ว่าเนื้อหาของหน้าเว็บเป็นภาษาใด โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

<html lang="รหัสภาษา">

ตัวอย่างที่ 9

<html lang="th">

ตัวอย่างที่ 10

<html lang="en">

จากรูปแบบของการระบุคุณลักษณะภาษาที่ใช้ในเนื้อหาด้วยแท็ก html ส่วนที่เป็นรหัสภาษาจะต้องกำหนดตามมาตรฐาน BCP 47 เช่น การระบุว่าเนื้อหาในเอกสารเป็นภาษาไทยจะกำหนดเป็น <html lang="th"> ดังตัวอย่างที่ 9 และเนื้อหาที่เป็นภาษาอังกฤษจะกำหนดเป็น <html lang="en"> ดังตัวอย่างที่ 10 เป็นต้น

นอกจากนี้ การระบุภาษาของเนื้อหาในเอกสารเว็บอย่างถูกต้องยังมีความสำคัญในด้านคุณลักษณะการเข้าถึง (Accessibility) ของเว็บ โดยจะช่วยให้โปรแกรมช่วยอ่านหน้าจอสามารถเลือกเสียงและสำเนียงในการอ่านที่ถูกต้องตามภาษาของเนื้อหา หากไม่มีการกำหนด lang เครื่องมือช่วยอ่านอาจใช้ภาษาที่ตั้งไว้ในระบบปฏิบัติการ ซึ่งอาจทำให้เกิดความคลาดเคลื่อนในการรับรู้ของผู้ใช้งาน

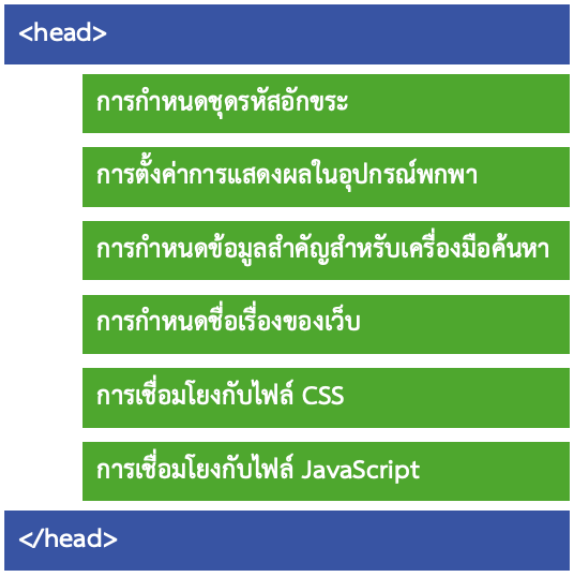
4.3 จุดเริ่มต้นและจุดสิ้นสุดส่วนหัวของเอกสาร

องค์ประกอบแรกที่อยู่ภายในองค์ประกอบ html คือองค์ประกอบที่ใช้กำหนดขอบเขตส่วนหัวของเอกสาร ซึ่งใช้สำหรับกำหนดข้อมูลเบื้องต้นที่เกี่ยวข้องกับหน้าเว็บ เช่น การกำหนดชุดรหัสอักขระ ชื่อเรื่อง ลิงก์ที่เชื่อมโยงไปยังไฟล์ภายนอกเพื่อนำมาใช้ร่วมกับเอกสารเว็บ คำอธิบายเอกสาร คำอธิบายสำหรับเครื่องมือค้นหา เป็นต้น รวมทั้งคำสั่งพิเศษบางประการที่เว็บเบราว์เซอร์ต้องรับทราบก่อนเริ่มการแสดงผลเนื้อหา ส่วนหัวของเอกสารเว็บ มีจุดเริ่มต้นและจุดสิ้นสุดดังนี้

จุดเริ่มต้น	<head>	จุดสิ้นสุด	</head>

องค์ประกอบ head จะไม่แสดงผลโดยตรงให้ผู้ใช้เห็นในเว็บเบราว์เซอร์ แต่จะเป็นข้อมูลประกอบที่สำคัญของเอกสารเว็บ ซึ่งมีผลต่อทั้งการแสดงผล การค้นหาข้อมูล และการตีความโดยระบบต่าง ๆ เช่น เครื่องมือ

ค้นหา (Search Engines) หรือบริการบนเว็บ โดยองค์ประกอบที่สำคัญที่ส่งผลต่อการพัฒนาเว็บแอปพลิเคชันซึ่งควรกำหนดในส่วนหัวของเอกสาร แสดงดังภาพที่ 5.6



ภาพที่ 5.6 องค์ประกอบสำคัญที่ควรกำหนดในส่วนหัวของเอกสาร

4.3.1 การกำหนดชุดรหัสอักขระ

การกำหนดชุดรหัสอักขระของเอกสาร (Character Encoding) เป็นการบ่งบอกเว็บเบราว์เซอร์ให้ทราบว่าจะต้องใช้รหัสอักขระแบบใดในการแสดงผลเนื้อหาภายในเอกสารเว็บ โดยการกำหนดชุดรหัสอักขระนี้จะกำหนดเป็นคุณสมบัติชื่อ charset ของแท็ก <meta> ซึ่งเป็นแท็กที่ไม่ต้องการจุดสิ้นสุดของแท็กหรือแท็กปิด โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

<meta charset="รหัสอักขระ">

ตัวอย่างที่ 11

```
<meta charset="UTF-8">
```

ตัวอย่างที่ 12

```
<meta charset="ISO-8859">
```

จากรูปแบบของการกำหนดชุดรหัสอักขระที่ใช้ในการแสดงผลข้างต้น ส่วนที่เป็นรหัสอักขระสามารถกำหนดรหัสอักขระที่ใช้ เช่น UTF-8, UTF-16, Windows-1252 และ ISO-8859 เป็นต้น จากตัวอย่างที่

10 เป็นการกำหนดชุดอักขระเป็น UTF-8 และตัวอย่างที่ 12 กำหนดเป็น ISO-8859 ตามลำดับ อย่างไรก็ตาม เพื่อให้เอกสารเว็บสอดคล้องกับมาตรฐาน HTML Living Standard จึงมีข้อแนะนำที่เกี่ยวข้องดังนี้ (Griffin, 2022)

- ในเอกสารเว็บมีแท็ก <meta> ที่มีแอตทริบิวต์ charset เพียงหนึ่งรายการเท่านั้น
- ค่า charset ที่ใช้ควรเป็น UTF-8 ตามคำแนะนำของมาตรฐาน HTML5 ซึ่งรองรับอักขระได้หลายภาษา และช่วยหลีกเลี่ยงปัญหาในการส่งข้อมูลไปยังเครื่องผู้ให้บริการเว็บผ่านฟอร์ม รวมทั้งการเข้ารหัส URL
- แท็ก <meta charset="utf-8"> ต้องอยู่ภายในส่วน <head> ของเอกสาร และควรอยู่ในลำดับต้น ๆ ของไฟล์เพื่อให้กลไกการประมวลผลของเว็บเบราว์เซอร์สามารถตรวจพบการเข้ารหัสอักขระภายใน 1,024 ไบต์แรกของเอกสารเว็บ เนื่องจากเว็บเบราว์เซอร์จะทำการประมวลผลเอกสารบรรทัดแรก หากไม่พบการกำหนดรหัสอักขระภายในช่วงดังกล่าว อาจทำให้การแสดงผลอักขระผิดเพี้ยน โดยเฉพาะภาษาไทยหรือภาษาที่มีอักขระพิเศษ

4.3.2 การตั้งค่าหน้าต่างมุมมอง

การแสดงผลของเว็บแอปพลิเคชันในยุคปัจจุบันไม่ได้จำกัดอยู่เฉพาะบนหน้าจอคอมพิวเตอร์เท่านั้น หากแต่ยังต้องรองรับการแสดงผลบนอุปกรณ์หลากหลายประเภท ไม่ว่าจะเป็นแท็บเล็ตหรือโทรศัพท์มือถือ ซึ่งมีขนาดหน้าจอแตกต่างกันอย่างมาก ด้วยเหตุนี้ การกำหนดพฤติกรรมของการแสดงผลให้เหมาะสมกับขนาดหน้าจอของผู้ใช้แต่ละคนจึงเป็นสิ่งสำคัญ โดยเฉพาะอย่างยิ่งการควบคุมขนาดของหน้าต่างมุมมอง (Viewport) ซึ่งหมายถึงพื้นที่ที่มองเห็นหน้าเว็บได้ในหน้าจอของอุปกรณ์หนึ่ง ๆ

ในอดีต หน้าเว็บส่วนใหญ่ได้รับการออกแบบโดยมีขนาดคงที่ และมักเหมาะสมกับจอภาพของเครื่องคอมพิวเตอร์ อย่างไรก็ตาม เมื่อมีการใช้งานอุปกรณ์พกพาในการเข้าถึงเว็บมากขึ้น การแสดงผลแบบคงที่ดังกล่าวจึงไม่สามารถรองรับการใช้งานได้ดีพอ โดยเว็บเบราว์เซอร์มักใช้วิธีลดขนาดของหน้าเว็บทั้งหน้าให้พอดีกับหน้าจอ ซึ่งถึงแม้จะช่วยให้เห็นเนื้อหาได้ครบ แต่กลับทำให้ตัวอักษรเล็ก อ่านยาก และส่งผลต่อประสบการณ์ของผู้ใช้

HTML Living Standard ได้กำหนดมาตรฐานในการกำหนดขอบเขตการแสดงผลบนอุปกรณ์พกพาคือการใช้แท็ก <meta> ร่วมกับแอตทริบิวต์ name="viewport" เพื่อกำหนดคุณลักษณะของหน้าต่างมุมมอง โดยขนาดของหน้าต่างมุมมองจะแตกต่างกันตามชนิดของอุปกรณ์ อย่างเช่น โทรศัพท์มือถือ และแท็บเล็ตมักมีขนาดหน้าต่างมุมมองที่เล็กกว่าคอมพิวเตอร์อย่างชัดเจน การกำหนดค่าหน้าต่างมุมมองอย่างเหมาะสมผ่านแท็ก <meta name="viewport"> จึงเป็นกลไกสำคัญที่ช่วยให้เว็บแอปพลิเคชันสามารถควบคุมลักษณะการแสดงผลให้เหมาะสมกับขนาดของหน้าจอแต่ละชนิด อันเป็นแนวทางพื้นฐานของการออกแบบเว็บแบบตอบสนอง (Responsive Web Design) ที่ทันสมัยในปัจจุบัน โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

```
<meta  name="viewport"  content="ค่ากำหนดคุณลักษณะของหน้าต่างมุมมอง">
```

จากรูปแบบของการกำหนดค่าคุณลักษณะของหน้าต่างมุมมองข้างต้น ค่าของแอตทริบิวต์ content เป็นค่ากำหนดคุณลักษณะของหน้าต่างมุมมอง ซึ่งใช้สำหรับระบุว่าหน้าเว็บควรแสดงผลอย่างไรบนอุปกรณ์ของผู้ใช้ โดยเฉพาะอย่างยิ่งในกรณีของอุปกรณ์พกพาที่มีขนาดหน้าจอแตกต่างกัน เช่น width, initial-scale, minimum-scale และ maximum-scale เป็นต้น ค่าของแอตทริบิวต์ content ประกอบด้วยคู่ของชื่อคุณลักษณะและค่าที่กำหนด โดยใช้เครื่องหมายจุลภาคแบ่ง เช่น width=device-width, initial-scale=1.0 ซึ่งสามารถเพิ่มเติมคุณลักษณะอื่น เช่น minimum-scale, maximum-scale และ user-scalable ได้ตามความเหมาะสมของการออกแบบเว็บแอปพลิเคชันแต่ละประเภท

ตัวอย่างที่ 13

```
<meta  name="viewport"  content="width=device-width, initial-scale=1.0">
```

ตัวอย่างที่ 13 เป็นการตั้งค่าหน้าต่างมุมมองให้มีความกว้างเท่ากับอุปกรณ์ซึ่งกำหนดจาก width=device-width และกำหนดระดับการขยายเริ่มต้น (Initial scale) เป็น 1.0 ซึ่งกำหนดจาก initial-scale=1.0 หมายถึง ให้แสดงผลหน้าเว็บในขนาดปกติที่ไม่ถูกย่อหรือขยาย

ตัวอย่างที่ 14

```
<meta  name="viewport"  content="width=320, initial-scale=1.0">
```

ตัวอย่างที่ 14 เป็นการกำหนดความกว้างของหน้าต่างมุมมองแบบคงที่ที่ 320 พิกเซล และกำหนดระดับการขยายเริ่มต้นในขนาดปกติ ซึ่งอาจเหมาะกับอุปกรณ์พกพารุ่นเก่าที่มีความกว้างจอภาพจำกัด

การกำหนดค่าหน้าต่างมุมมอง ที่เหมาะสมช่วยให้การแสดงผลในอุปกรณ์พกพาเป็นไปอย่างมีประสิทธิภาพ และหลีกเลี่ยงปัญหาการเลื่อนหน้าจอในแนวนอน เพื่อแสดงเนื้อหาของเว็บที่มีความกว้างเกินขนาดหน้าจอ ทำให้ส่งผลเสียต่อประสบการณ์ของผู้ใช้ซึ่งผู้พัฒนาเว็บแอปพลิเคชันไม่ควรมองข้ามในการพัฒนาเว็บที่รองรับอุปกรณ์พกพาในปัจจุบัน นอกจากนี้ยังมีแนวทางปฏิบัติเพิ่มเติมที่ควรคำนึงถึง ดังนี้ (W3Schools, n.d.)

- ไม่ควรใช้ส่วนประกอบของเนื้อหาที่มีความกว้างคงที่ขนาดใหญ่กว่าขนาดของหน้าจออุปกรณ์ เพราะจะทำให้ผู้ใช้ต้องเลื่อนหน้าจอในแนวนอน
- ไม่ควรให้การแสดงผลของหน้าเว็บพึ่งพาความกว้างของหน้าจอเพียงขนาดเดียว ควรออกแบบให้ยืดหยุ่นต่ออุปกรณ์ที่หลากหลาย
- ควรใช้คำสั่ง CSS media queries เพื่อปรับแต่งรูปแบบการแสดงผลให้เหมาะสมกับหน้าจอแต่ละขนาด เช่น การใช้ width: 100% แทนค่าคงที่ เป็นต้น

4.3.3 การกำหนดข้อมูลสำคัญสำหรับเครื่องมือค้นหา

การพัฒนาเว็บแอปพลิเคชันโดยทั่วไปมักมุ่งเน้นที่การทำงานของระบบและประสบการณ์ของผู้ใช้งานเป็นหลัก อย่างไรก็ตาม ในหลายกรณีโดยเฉพาะในหน้าเว็บที่เป็นสาธารณะ เช่น หน้าแนะนำระบบ หรือหน้าประชาสัมพันธ์ระบบ อาจมีความจำเป็นที่จะต้องสนับสนุนการค้นหาหน้าเว็บนั้น ๆ โดยเครื่องมือค้นหา เช่น Google หรือ Bing เป็นต้น เพื่อให้ผู้ใช้งานสามารถเข้าถึงเว็บแอปพลิเคชันได้โดยง่ายผ่านการค้นหา การกำหนดข้อมูลสำคัญเพื่อสนับสนุนเครื่องมือค้นหาจึงเป็นแนวปฏิบัติที่ควรให้ความสำคัญในส่วน <head> ของเอกสาร HTML เช่นกัน

แท็กที่ใช้ในการกำหนดข้อมูลสนับสนุนเครื่องมือค้นหาคือแท็ก <meta> รวมทั้งใช้แอตทริบิวต์ name และ content ในการระบุคำอธิบายเกี่ยวกับเนื้อหาของเว็บเพจและคำสำคัญที่เกี่ยวข้องตามลำดับ โดยมีรูปแบบดังนี้

รูปแบบ

```
<meta  name="ชื่อคุณลักษณะ"  content="ค่าที่ต้องการกำหนดให้คุณลักษณะ">
```

จากรูปแบบการกำหนดข้อมูลสนับสนุนเครื่องมือค้นหาข้างต้น ชื่อคุณลักษณะที่ใช้กำหนดข้อมูลสนับสนุนต่าง ๆ แสดงดังตารางที่ 5-1

ตารางที่ 5-1 ชื่อคุณลักษณะที่ใช้ในการกำหนดข้อมูลสนับสนุนเครื่องมือค้นหา

ชื่อคุณลักษณะ	คำอธิบาย
application-name	ใช้สำหรับระบุชื่อของเว็บแอปพลิเคชัน
author	ใช้สำหรับระบุชื่อของผู้เขียนเว็บ ผู้พัฒนาหรือองค์กรเจ้าของระบบ
generator	ใช้สำหรับระบุชื่อแอปพลิเคชันที่สร้างหน้าเว็บนี้ขึ้นมาซึ่งเป็นหน้าเว็บที่เกิดจากการสร้างโดยอัตโนมัติ
keywords	ใช้สำหรับระบุคำสำคัญที่เกี่ยวข้องกับเนื้อหาของหน้าเว็บ โดยคั่นแต่ละคำด้วยเครื่องหมายจุลภาค (,)
description	ใช้เพื่อให้คำอธิบายสั้น ๆ ของเนื้อหาทั้งหน้า โดยเครื่องมือค้นหาจะนำไปแสดงร่วมกับผลลัพธ์การค้นหา

ตัวอย่างของการกำหนดข้อมูลสนับสนุนเครื่องมือค้นหา โดยกำหนดให้ชื่อเว็บแอปพลิเคชันเป็นระบบจองห้องพัก โดยมีผู้พัฒนาเป็นวิชาเอกเทคโนโลยีธุรกิจดิจิทัล ในการค้นหาหน้าเว็บนี้จะใช้คำค้นสองคำ คือ ระบบจองห้องพัก และเว็บแอปพลิเคชันห้องพัก รวมทั้งมีคำอธิบายหน้าเว็บเป็น ระบบจองห้องพักออนไลน์ที่ใช้งานง่าย สามารถนำไปเขียนในโครงสร้างเอกสารเว็บดังตัวอย่างที่ 15

ตัวอย่างที่ 15

```
<meta name="application-name"  content="ระบบจองห้องพัก" >
<meta name="author"  content="วิชาเอกเทคโนโลยีธุรกิจดิจิทัล " >
<meta name="description"  content="ระบบจองห้องพักออนไลน์ที่ใช้งานง่าย" >
<meta name="keywords"  content="ระบบจองห้องพัก, เว็บแอปพลิเคชันห้องพัก" >
```

แม้ว่าแท็กที่เกี่ยวข้องกับเครื่องมือค้นหาอาจไม่ถือเป็นองค์ประกอบที่จำเป็นในเชิงเทคนิคสำหรับการทำงานของเว็บแอปพลิเคชันโดยตรง แต่อย่างไรก็ตาม หากเว็บแอปพลิเคชันมีหน้าสาธารณะหรือเนื้อหาที่ต้องการให้ผู้ใช้งานทั่วไปสามารถค้นพบผ่านเครื่องมือค้นหาได้ การกำหนดข้อมูลอย่างเหมาะสมในส่วนนี้จึงเป็นแนวปฏิบัติที่ควรพิจารณาร่วมด้วย เพื่อเสริมสร้างโอกาสในการเข้าถึงเนื้อหา และส่งเสริมภาพลักษณ์ของระบบให้ดูน่าเชื่อถือยิ่งขึ้น ในทางกลับกัน หากเว็บแอปพลิเคชันถูกออกแบบมาเพื่อใช้งานภายในองค์กร ซึ่งมีการตั้งค่าความปลอดภัยเพื่อป้องกันไม่ให้เครื่องมือค้นหาเข้าถึงข้อมูลได้ หรือมีการกำหนดการยืนยันตัวตนก่อนเข้าถึงระบบ ก็อาจละเว้นการกำหนดข้อมูลในส่วนที่เกี่ยวข้องกับเครื่องมือค้นหาได้โดยไม่ส่งผลต่อการทำงานของระบบ

4.3.4 การกำหนดชื่อเรื่องของเว็บ

ชื่อเรื่องของเว็บ (Page Title) เป็นองค์ประกอบพื้นฐานของเอกสารเว็บ ซึ่งใช้ในการระบุหัวข้อหรือชื่อของหน้าเว็บนั้น ๆ โดยจะแสดงผลที่แถบชื่อของเบราว์เซอร์ (Browser Tab) และเป็นข้อมูลสำคัญที่เครื่องมือค้นหาใช้ในการจัดแสดงผลการค้นหา ชื่อเรื่องที่กำหนดไว้อย่างเหมาะสมช่วยให้ผู้ใช้สามารถเข้าใจวัตถุประสงค์ของหน้าเว็บได้อย่างรวดเร็ว ซึ่งช่วยส่งเสริมประสบการณ์ของผู้ใช้โดยรวม เช่น ระบบจองห้องพัก-จัดการข้อมูลห้องพัก เป็นการบอกให้ผู้ใช้ทราบว่าหน้าเว็บที่ใช้อยู่ในขณะนี้เป็นการจัดการข้อมูลห้องพัก เป็นต้น

การกำหนดชื่อเรื่องจะใช้แท็ก <title> ซึ่งเป็นองค์ประกอบที่อยู่ภายในส่วน <head> ของเอกสารเว็บ ในแต่ละหน้าเว็บจะมีองค์ประกอบ title เพียงหนึ่งองค์ประกอบเท่านั้น และควรมีความยาวที่เหมาะสม ไม่ยาวหรือสั้นจนเกินไป โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

<title>ชื่อเรื่องของหน้าเว็บ</title>

จากรูปแบบของการกำหนดชื่อเรื่องของหน้าเว็บ จะกำหนดเป็นข้อความล้วน (Text-Only) ไม่สามารถมีองค์ประกอบซ้อนอยู่ภายใน ซึ่งจะเขียนอยู่ระหว่างแท็กเปิด <title> และแท็กปิด </title>

ตัวอย่างที่ 16

```
<title>ระบบจองห้องพัก</title>
```

ตัวอย่างที่ 16 แสดงถึงการกำหนดชื่อเรื่องเป็นคำว่า ระบบจองห้องพัก ซึ่งเหมาะกับการใช้เป็นชื่อเรื่องของหน้าแรกของระบบ

ตัวอย่างที่ 17

```
<title>ระบบจองห้องพัก - จัดการข้อมูลห้องพัก</title>
```

ตัวอย่างที่ 17 เป็นการแสดงชื่อของเว็บแอปพลิเคชันและบริบทของหน้าเว็บที่ใช้งานอยู่โดยเป็นหน้าเว็บสำหรับการจัดการข้อมูลห้องพัก

4.3.5 การเชื่อมโยงกับไฟล์ CSS

ในการพัฒนาเว็บแอปพลิเคชัน การจัดรูปแบบการแสดงผลของหน้าเว็บเป็นอีกองค์ประกอบที่สำคัญควบคู่กับโครงสร้างของเนื้อหา ซึ่งการจัดรูปแบบดังกล่าวจะใช้ภาษา CSS เพื่อควบคุมลักษณะการแสดงผลขององค์ประกอบ HTML ต่าง ๆ เช่น สี พื้นหลัง ตัวอักษร ระยะห่าง ขนาด และการจัดตำแหน่งของเนื้อหา เป็นต้น

การใช้งาน CSS สามารถเขียนไว้ในเอกสารเว็บเองหรือการสร้างเป็นไฟล์แยกออกจากเอกสารเว็บ ซึ่งวิธีหลังเป็นแนวทางที่ช่วยให้การจัดการรหัสคำสั่งเป็นระเบียบ และสามารถนำไฟล์ CSS ไปใช้ร่วมกับหน้าเว็บอื่น ๆ ได้ ทั้งนี้การแยกไฟล์ CSS ออกจากเอกสารเว็บจำเป็นต้องมีการระบุการเชื่อมโยงไปยังไฟล์ CSS นั้นๆ ด้วยแท็ก <link> ซึ่งเป็นแท็กที่ไม่มีแท็กปิด โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

```
<link  rel="stylesheet"  href="ชื่อไฟล์.css">
```

จากรูปแบบการเชื่อมโยงกับไฟล์ CSS ด้วยการใช้แท็ก <link> จะประกอบด้วยแอตทริบิวต์ rel (Relation: rel) ใช้สำหรับการกำหนดความสัมพันธ์ ซึ่งมีค่าเป็น stylesheet เพื่อเป็นการสื่อความหมายว่าไฟล์ที่เชื่อมโยงนั้นมีความสัมพันธ์กับเอกสารเว็บในการเป็นไฟล์สำหรับการจัดรูปแบบการแสดงผล และแอตทริบิวต์ href (Hypertext Reference: href) ใช้สำหรับระบุที่อยู่ของไฟล์ CSS ที่ต้องการเชื่อมโยง

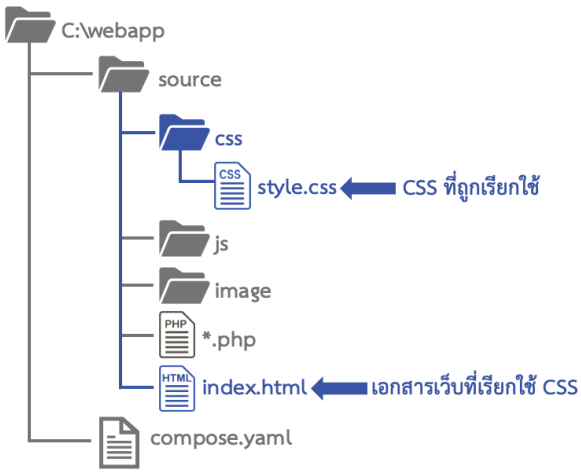
ตัวอย่างที่ 18

```
<link  rel="stylesheet"  href="style.css">
```

ตัวอย่างที่ 19

```
<link  rel="stylesheet"  href="css/style.css">
```

ตัวอย่างที่ 18 แสดงการเชื่อมโยงกับไฟล์ชื่อ style.css ซึ่งอยู่ในโฟลเดอร์เดียวกันกับไฟล์เอกสารเว็บที่เรียกใช้งาน CSS ส่วนตัวอย่างที่ 19 แสดงการเชื่อมโยงกับไฟล์ชื่อ style.css ซึ่งจัดเก็บอยู่ในโฟลเดอร์ชื่อ css โดยไฟล์เอกสารเว็บที่เรียกใช้งานจะอยู่ในตำแหน่งเดียวกับโฟลเดอร์ css ดังแสดงในภาพที่ 5.7



ภาพที่ 5.7 ตัวอย่างการอ้างอิงไฟล์ CSS ที่จัดเก็บในโฟลเดอร์

ตัวอย่างที่ 20

```
<link
    rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css">
```

ตัวอย่างที่ 20 แสดงการเชื่อมโยงกับไฟล์ชื่อ bootstrap.min.css ที่ระบุตำแหน่งโดยใช้ URL จากตัวอย่างเป็น URL ของเครือข่ายผู้ให้บริการเนื้อหาที่กระจายอยู่ตามภูมิภาคต่าง ๆ ทั่วโลก (Content Delivery Network: CDN) ซึ่งการเชื่อมโยงกับไฟล์ในลักษณะนี้เป็นแนวทางการใช้ไฟล์ CSS ที่ผู้พัฒนาไม่จำเป็นต้องมีไฟล์ในเครื่องผู้ให้บริการของตนเอง แต่ใช้ความสามารถของเครือข่าย CDN ที่มีกระจายอยู่ในภูมิภาคต่าง ๆ โดยเนื้อหาจะถูกโหลดจากเครื่องผู้ให้บริการที่อยู่ใกล้กับผู้เรียกใช้หน้าเว็บมากที่สุด ทำให้สามารถเพิ่มความเร็วในการโหลด ลดภาระของเครื่องผู้ให้บริการหลัก และรองรับผู้ใช้งานจำนวนมากได้อย่างมีประสิทธิภาพ

4.3.6 การเชื่อมโยงกับไฟล์ JavaScript

ในการพัฒนาเว็บแอปพลิเคชัน การจัดการกับเหตุการณ์ต่าง ๆ ที่ผู้ใช้กระทำบนหน้าเว็บ เช่น การคลิกปุ่ม การกรอกข้อมูล หรือการโหลดหน้าเว็บ รวมทั้งการตรวจสอบข้อมูลที่ผู้ใช้ป้อนในแบบฟอร์มก่อนส่งไปยังเครื่องผู้ให้บริการ จำเป็นต้องมีการใช้ภาษา JavaScript ในกระบวนการเหล่านี้ โดยทั่วไป การใช้ JavaScript สามารถแทรกคำสั่งไว้ในเอกสารเว็บโดยตรง หรือแยกออกเป็นไฟล์ภายนอกเช่นกัน การเชื่อมโยงกับไฟล์ JavaScript ภายนอกจะใช้องค์ประกอบ script โดยระบุไฟล์ผ่านแอตทริบิวต์ src ดังรูปแบบและตัวอย่างดังต่อไปนี้

รูปแบบ

```
<script  src="ชื่อไฟล์.js"> </script>
```

จากรูปแบบข้างต้น องค์ประกอบ script เป็นองค์ประกอบที่มีการกำหนดจุดเริ่มต้นด้วยแท็กเปิด <script> และแท็กปิด </script> แต่จะไม่มีเนื้อหาระหว่างแท็กเปิดและปิดนี้ และใช้แอตทริบิวต์ src (source) เพื่อระบุที่อยู่ของไฟล์ JavaScript

ตัวอย่างที่ 21

```
<script  src="script.js"> </script>
```

ตัวอย่างที่ 21 แสดงการเชื่อมโยงไฟล์ JavaScript ชื่อ script.js ซึ่งอาจอยู่ในโฟลเดอร์เดียวกันกับเอกสารเว็บที่เรียกใช้งาน

ตัวอย่างที่ 22

```
<script  src="js/script.js"> </script>
```

ตัวอย่างที่ 22 แสดงการเชื่อมโยงไฟล์ JavaScript ชื่อ script.js ซึ่งจัดเก็บอยู่ในโฟลเดอร์ชื่อ js โดยไฟล์เอกสารเว็บที่เรียกใช้งานจะอยู่ในตำแหน่งเดียวกับโฟลเดอร์ js

ตัวอย่างที่ 23

```
<script  src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.min.js">
</script>
```

ตัวอย่างที่ 23 แสดงการเชื่อมโยงกับไฟล์ชื่อ bootstrap.min.js ที่ระบุตำแหน่งโดยใช้ URL ซึ่งเป็นไฟล์ที่มีให้บริการใน CDN ลักษณะเดียวกันกับการใช้งาน CSS

นอกจากนี้ แนวปฏิบัติทั่วไปในการพัฒนาเว็บแอปพลิเคชันที่การทำงานของ JavaScript ไม่เกี่ยวข้องกับกระบวนเริ่มต้นการทำงาน ควรทำการเชื่อมโยงไฟล์ JavaScript ไว้ในตำแหน่งก่อนปิดแท็ก </body> ของเอกสารเว็บ เพื่อให้หน้าเว็บโหลดองค์ประกอบทั้งหมดของเอกสารเว็บเสร็จสิ้นก่อน จากนั้นจึงโหลด JavaScript อันเป็นการช่วยเพิ่มประสิทธิภาพในการแสดงผล และลดความเสี่ยงของข้อผิดพลาดจากการทำงานของสคริปต์ที่อาจมีการอ้างถึงองค์ประกอบในเอกสารเว็บ

จากการกำหนดส่วนหัวของเอกสารเว็บดังที่ได้กล่าวมาแล้วข้างต้น เมื่อนำส่วนการกำหนดต่าง ๆ ประกอบด้วยการกำหนดชุดรหัสอักขระ การตั้งค่าการหน้าต่างมุมมอง การกำหนดข้อมูลสำคัญสำหรับเครื่องมือค้นหา การกำหนดชื่อเรื่องของเว็บ การเชื่อมโยงกับไฟล์ CSS และ JavaScript สามารถเขียนเป็นภาพรวมของส่วนหัวของเอกสารเว็บได้ดังตัวอย่างที่ 24

ตัวอย่างที่ 24

```
<head>
    <meta  charset="UTF-8">
    <meta  name="viewport"  content="width=device-width, initial-scale=1.0">
    <meta name="application-name" content="ระบบจองห้องพัก" >
    <meta name="author"  content="วิชาเอกเทคโนโลยีธุรกิจดิจิทัล " >
    <meta name="description". content="ระบบจองห้องพักออนไลน์ที่ใช้งานง่าย" >
    <meta name="keywords"  content="ระบบจองห้องพัก, เว็บแอปพลิเคชันห้องพัก">
    <title>ระบบจองห้องพัก</title>
    <link  rel="stylesheet"  href="css/style.css">
    <script  src="js/script.js"> </script>
</head>
```

4.4 จุดเริ่มต้นและจุดสิ้นสุดเนื้อหาของเอกสาร

หลังจากการกำหนดส่วนหัวของเอกสาร HTML แล้ว ส่วนต่อไปเป็นการกำหนดส่วนเนื้อหาของเอกสาร ซึ่งหมายถึงองค์ประกอบต่าง ๆ ที่จะแสดงผลต่อผู้ใช้งานในเว็บเบราว์เซอร์ เช่น ข้อความ รูปภาพ ปุ่ม แบบฟอร์ม ตาราง หรือวิดีโอ เป็นต้น การกำหนดส่วนเนื้อหาของเอกสารเว็บจะใช้แท็ก <body> สำหรับกำหนดจุดเริ่มต้น และแท็ก </body> สำหรับกำหนดจุดสิ้นสุดของเนื้อหา โดยเนื้อหาทั้งหมดที่ต้องการแสดงผลบนหน้าเว็บจะต้องอยู่ภายในอิลิเมนต์นี้

จุดเริ่มต้น	<body>	จุดสิ้นสุด	</body>

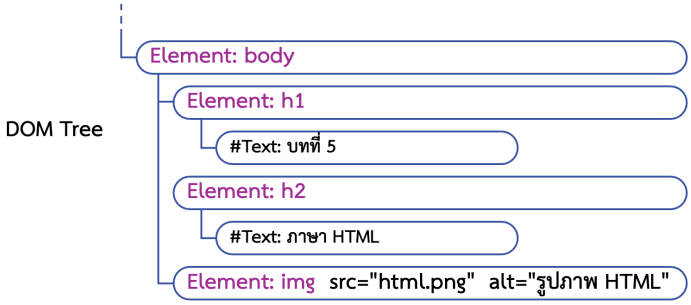
ภายในอิลิเมนต์ body ผู้พัฒนาสามารถใช้คำสั่งกำกับต่าง ๆ ของ HTML ได้อย่างอิสระ โดยสามารถแทรกแท็กเพื่อแสดงหัวข้อ ย่อหน้า รูปภาพ ลิงก์ ปุ่ม หรือแม้กระทั่งอิลิเมนต์ที่มีโครงสร้างซับซ้อน เช่น ตาราง รายการ และแบบฟอร์ม ซึ่งอิลิเมนต์เหล่านี้จะถูกกลไกการแสดงผลของเว็บเบราว์เซอร์แปลงให้เป็นโครงสร้างต้นไม้ DOM Tree เพื่อใช้ในการแสดงผลที่เหมาะสมกับบริบทของเนื้อหานั้น ๆ ตัวอย่างของส่วนเนื้อหาที่จะปรากฏในเว็บเบราว์เซอร์แสดงดังตัวอย่างที่ 25

ตัวอย่างที่ 25

```
<!DOCTYPE  html>
<html>
    <head>
        ...
```

```
    </head>
    <body>
        <h1>บทที่ 5</h1>
        <h2>ภาษา HTML</h2>
        <img src="html.png" alt="รูปภาพ HTML">
    </body>
</html>
```

DOM Tree



ผลลัพธ์



จากตัวอย่างรหัสคำสั่งในภาษา HTML ที่แสดงในตัวอย่างที่ 25 เป็นการนำส่วนเนื้อหามาเขียนตามโครงสร้างของภาษา HTML โดยกลไกการแสดงผลของเว็บเบราว์เซอร์จะทำการสร้างอิลิเมนต์ body ให้เป็นอิลิเมนต์แม่ของอิลิเมนต์ h1, h2 และ img ซึ่งนำมาสู่การแสดงผลลัพธ์ในหน้าเว็บเบราว์เซอร์

5. การเขียนบรรทัดหมายเหตุ

การเขียนบรรทัดหมายเหตุในภาษา HTML (HTML Comment Tag) เป็นแนวปฏิบัติที่สำคัญในการพัฒนาเว็บแอปพลิเคชันโดยใช้สำหรับอธิบายรหัสคำสั่ง ช่วยให้ผู้อื่นรวมทั้งผู้พัฒนาเองสามารถเข้าใจโครงสร้างและตรรกะของรหัสคำสั่งได้ง่ายขึ้นในภายหลัง นอกจากนี้ยังสามารถใช้สำหรับยกเว้นรหัสคำสั่งจากการประมวลผลในระหว่างการทดสอบ โดยเว็บเบราว์เซอร์จะไม่ประมวลผลรหัสที่อยู่ภายในบรรทัดหมายเหตุ ซึ่งจะไม่ส่งผลต่อผลลัพธ์ที่แสดงในเว็บเบราว์เซอร์ เนื่องจากบรรทัดหมายเหตุจะไม่ถูกประมวลผลในการแสดงผลหน้าเว็บ รูปแบบการเขียนบรรทัดหมายเหตุจะมีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดดังนี้

จุดเริ่มต้น	<!--	จุดสิ้นสุด	-->

การเขียนบรรทัดหมายเหตุในภาษา HTML จะใช้เครื่องหมาย <!-- เพื่อเริ่มต้น และใช้ --> เพื่อสิ้นสุดข้อความหมายเหตุ ข้อความที่อยู่ระหว่างเครื่องหมายทั้งสองจะไม่ถูกรวมอยู่ในการแสดงผลของเว็บเบราว์เซอร์ ทั้งนี้สามารถเขียนได้ทั้งแบบบรรทัดเดียวหรือหลายบรรทัดดังตัวอย่างที่ 26

ตัวอย่างที่ 26

```
1  <!-- นี่คือบรรทัดหมายเหตุแบบบรรทัดเดียว -->
2  <!-- <p>ยินดีต้อนรับสู่เว็บแอปพลิเคชัน</p> -->
3  <!--
4      นี่คือบรรทัดหมายเหตุแบบหลายบรรทัด
5      คำอธิบายประกอบ
6  -->
7  <img src="webapp.jpg" alt="รูปภาพหนังสือเว็บแอปพลิเคชัน">
```

จากตัวอย่างที่ 26 ในบรรทัดที่ 1 เป็นการเขียนบรรทัดหมายเหตุแบบบรรทัดเดียวที่ใช้เขียนข้อความอธิบายแก่ผู้พัฒนา ในบรรทัดที่ 2 เป็นการเขียนบรรทัดหมายเหตุเพื่อยกเว้นการประมวลผลคำสั่งกำกับ p และในบรรทัดที่ 3 – 6 เป็นการเขียนบรรทัดหมายเหตุแบบหลายบรรทัด ทั้งนี้จากตัวอย่างที่ 26 เว็บเบราว์เซอร์จะประมวลผลและแสดงผลเฉพาะบรรทัดที่ 7 เท่านั้นซึ่งเป็นคำสั่ง <img> ที่อยู่นอกช่วงของบรรทัดหมายเหตุ

6. สรุปท้ายบท

ภาษา HTML ทำหน้าที่กำหนดโครงสร้างของเอกสารเว็บ โดยใช้แท็กเป็นกลไกหลักในการระบุความหมายและบทบาทของเนื้อหาแต่ละส่วนในหน้าเว็บให้แสดงผลได้อย่างเป็นระบบ รองรับการตีความโดยเว็บเบราว์เซอร์และสามารถเข้าถึงได้ผ่านอุปกรณ์ที่หลากหลาย

กลไกการกำกับในภาษา HTML ประกอบด้วยแท็ก แอตทริบิวต์ และอิลิเมนต์ ซึ่งแต่ละส่วนมีหน้าที่ร่วมกันในการควบคุมลักษณะของข้อมูล โดยแท็กทำหน้าที่กำกับเนื้อหา แอตทริบิวต์ให้ข้อมูลเพิ่มเติม และอิลิเมนต์คือผลรวมของทั้งสองส่วนที่ใช้กำหนดบทบาทของข้อมูลในเอกสารเว็บ

ค่าเริ่มต้นของการแสดงผลของอิลิเมนต์ใน HTML แบ่งออกเป็นสองประเภทหลัก ได้แก่ อิลิเมนต์แบบ Block-Level และแบบ Inline โดย Block-Level จะแสดงผลในบรรทัดใหม่และกินพื้นที่เต็มแนวกว้างขององค์ประกอบแม่ ส่วน Inline จะแสดงต่อเนื่องกันในบรรทัดเดียวและใช้พื้นที่เท่าที่เนื้อหาภายในกำหนด ความเข้าใจในพฤติกรรมเหล่านี้ช่วยให้สามารถจัดโครงสร้างเนื้อหาได้อย่างเหมาะสม

โครงสร้างเอกสาร HTML ที่ถูกต้องประกอบด้วยส่วนประกาศประเภทของเอกสาร จุดเริ่มต้นและจุดสิ้นสุดของเอกสาร ส่วนหัวของเอกสารซึ่งประกอบด้วยข้อมูลเมตา และส่วนเนื้อหาที่แสดงผลจริงในหน้าเว็บ การกำหนดโครงสร้างเหล่านี้อย่างครบถ้วนช่วยให้เอกสารมีความถูกต้องตามมาตรฐานและสามารถปรับใช้กับการแสดงผลบนอุปกรณ์ต่าง ๆ ได้อย่างมีประสิทธิภาพ

การใช้บรรทัดหมายเหตุใน HTML เป็นแนวทางที่ช่วยให้สามารถอธิบายส่วนต่าง ๆ ของรหัสคำสั่งได้อย่างชัดเจน ส่งผลให้การอ่าน การพัฒนา และการปรับปรุงระบบในระยะยาวทำได้อย่างเป็นระบบโดยไม่กระทบต่อการแสดงผลของหน้าเว็บ

7. กิจกรรมท้ายบท

7.1 จงอธิบายความหมายและบทบาทของภาษา HTML ในการพัฒนาเว็บแอปพลิเคชัน

7.2 แท็กและแอตทริบิวต์ในภาษา HTML คืออะไร? มีความแตกต่างกันอย่างไร? และอิลิเมนต์เกิดขึ้นได้อย่างไร? จงอธิบายพร้อมยกตัวอย่างประกอบ

7.3 ค่าเริ่มต้นการแสดงผลของอิลิเมนต์แบ่งได้ 2 ประเภท มีอะไรบ้าง? พร้อมทั้งอธิบายลักษณะการแสดงผลในแต่ละประเภทและยกตัวอย่างแท็กในแต่ละประเภท

7.4 จงเขียนโครงสร้างของภาษา HTML ที่ใช้ในการสร้างเอกสารเว็บตามมาตรฐาน HTML Living Standard ให้ถูกต้อง โดยมีรายละเอียดดังนี้

7.4.1 กำหนดรหัสอักขระที่ใช้ในเว็บเป็น UTF-8 และตั้งค่าหน้าต่างมุมมองให้มีความกว้างเท่ากับอุปกรณ์ (Device Width) และกำหนดระดับการขยายเริ่มต้น (Initial scale) เป็น 1.0

7.4.2 กำหนดข้อมูลสนับสนุนเครื่องมือค้นหา ดังนี้

- กำหนดให้ชื่อเว็บแอปพลิเคชัน (Application Name) เป็น ระบบถามตอบ
- ระบุคำสำคัญที่เกี่ยวข้องกับเนื้อหาในหน้าเว็บ (Keywords) จำนวน 3 คำ คือ Web App, ระบบเว็บ และการเขียนโปรแกรมบนเว็บ

7.4.3 กำหนดชื่อเรื่องของเว็บ (Page Title) เป็น Web Application

7.5 จากโครงสร้างภาษา HTML ในข้อ 7.4 จงวาดภาพ DOM Tree เพื่อแสดงความสัมพันธ์ของอิลิเมนต์ต่าง ๆ

7.6 บรรทัดหมายเหตุ (Comment) ในภาษา HTML คืออะไร? มีรูปแบบการเขียนและประโยชน์อย่างไร?



แท็ก HTML ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน

บทที่ 6 มุ่งนำเสนอแท็ก HTML ที่สำคัญและพบบ่อยในการพัฒนาเว็บแอปพลิเคชัน โดยเน้นแท็กที่เกี่ยวข้องกับการแสดงผลและการโต้ตอบกับผู้ใช้ผ่านเว็บเบราว์เซอร์ พร้อมจัดกลุ่มแท็กตามลักษณะการใช้งานเพื่อความเข้าใจที่เป็นระบบ หัวข้อหลักในบทนี้ประกอบด้วย

1. แท็กสำหรับการแสดงรูปภาพ
2. แท็กสำหรับการสร้างลิงก์
3. แท็กสำหรับการกำหนดรายการ
4. แท็กสำหรับการกำหนดหัวเรื่องและข้อความ
5. แท็กสำหรับการสร้างตาราง
6. แท็กสำหรับการสร้างฟอร์มรับข้อมูล
7. แท็กสำหรับการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา
8. แท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมาย
9. สรุปท้ายบท
10. กิจกรรมท้ายบท

1. แท็กสำหรับการแสดงรูปภาพ

แท็กสำหรับการแสดงรูปภาพ (Image Tag) จะใช้แท็ก <img> ซึ่งเป็นแท็กเดี่ยวที่มีหน้าที่ฝังภาพจากแหล่งภายนอกลงในหน้าเว็บ โดยไม่ใช่การนำไฟล์รูปภาพมาบรรจุในเอกสารเว็บโดยตรง แต่เป็นการสร้างพื้นที่ให้เบราว์เซอร์ดึงรูปภาพจากแหล่งที่อยู่ที่ระบุไว้มาแสดงผล แท็กนี้มักถูกใช้เพื่อแสดงโลโก้ ภาพประกอบเนื้อหา หรือภาพอินโฟกราฟิกต่าง ๆ เพื่อเพิ่มความเข้าใจและความน่าสนใจให้กับเนื้อหาในเว็บแอปพลิเคชัน รูปแบบของแท็ก <img> มีดังนี้

รูปแบบ

```
<img  src="ไฟล์รูปภาพ" alt="ข้อความแทนรูปภาพ">
```

ในการใช้งานแท็ก <img> จำเป็นต้องระบุแอตทริบิวต์ src ซึ่งใช้ระบุเส้นทางที่อยู่ (Path) ของรูปภาพ และควรกำหนดแอตทริบิวต์ alt ซึ่งมีหน้าที่แสดงข้อความแทนรูปภาพในกรณีที่ไม่สามารถแสดงภาพได้ เช่น เครือข่ายขัดข้อง หรือลิงก์ภาพเสีย นอกจากนี้ alt ยังเป็น แอตทริบิวต์ที่จำเป็นต่อการเข้าถึง เนื่องจากเครื่องอ่านหน้าจอจะอ่านข้อความในแอตทริบิวต์นี้ให้ผู้ใช้งานที่มีความบกพร่องทางการมองเห็นรับรู้ข้อความแทนรูปภาพนั้นได้ ในกรณีที่รูปภาพใช้ในการตกแต่งเท่านั้นไม่ได้มีส่วนที่เกี่ยวข้องกับเนื้อหา สามารถกำหนดค่า alt="" เพื่อไม่ให้โปรแกรมอ่านหน้าจออ่านข้อความที่ไม่จำเป็น ตัวอย่างของการแสดงรูปภาพในหน้าเว็บมีดังนี้

ตัวอย่างที่ 1

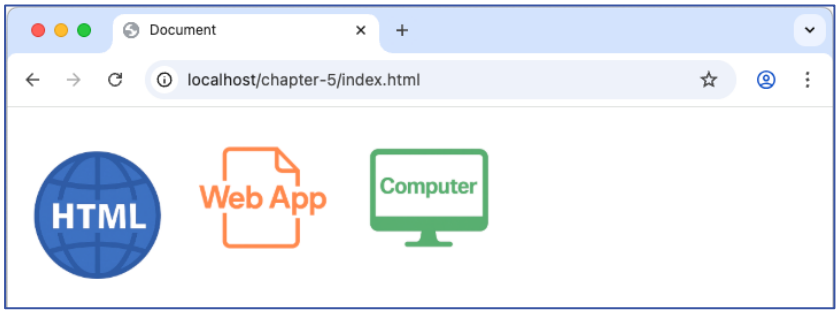
```
1  <img src="html.png" alt="ภาพโลโก้ HTML">
2  <img src="image/webapp.png" alt="ภาพโลโก้ Web Application">
3  <img src="https://www.ajnok.com/images/computer.png" alt="ภาพโลโก้ Computer">
```

จากตัวอย่างที่ 1 ในบรรทัดที่ 1 เป็นการแสดงรูปภาพชื่อ html.png ซึ่งอยู่ในตำแหน่งเดียวกันกับไฟล์เอกสารเว็บที่เรียกใช้รูปภาพ โดยมีการกำหนดข้อความแสดงแทนรูปภาพเป็น *ภาพโลโก้ HTML*

บรรทัดที่ 2 เป็นการแสดงรูปภาพชื่อ webapp.png จากโฟลเดอร์ชื่อ images ซึ่งโฟลเดอร์นี้อยู่ในตำแหน่งเดียวกันกับไฟล์เอกสารเว็บที่เรียกใช้รูปภาพ และใช้ข้อความแสดงแทนรูปภาพเป็น *ภาพโลโก้ Web Application*

บรรทัดที่ 3 เป็นการแสดงรูปภาพจากเครื่องผู้ให้บริการเว็บที่มี URL เป็น https://www.ajnok.com โดยแสดงรูปภาพชื่อ computer.png จากโฟลเดอร์ images และใช้ข้อความแสดงแทนรูปภาพเป็น *ภาพโลโก้ Computer* ผลลัพธ์ที่ได้ในเว็บเบราว์เซอร์แสดงดังนี้

ผลลัพธ์ของตัวอย่างที่ 1



จากผลลัพธ์ของตัวอย่างที่ 1 แสดงให้เห็นถึงการแสดงรูปภาพจากแหล่งต่าง ๆ โดยค่าเริ่มต้นของการแสดงผลของแท็ก img จะมีลักษณะแบบ Inline กล่าวคือ หากพื้นที่ในบรรทัดที่แสดงรูปภาพมีมากพอ รูปภาพจะมีการแสดงผลต่อเนื่องในบรรทัดเดียวกับข้อความหรืออิลิเมนต์รอบข้างที่มีลักษณะแบบ Inline เช่นกัน ในการกำหนดขนาดของรูปภาพสามารถใช้แอตทริบิวต์ width และ height ในแท็ก <img> หรือใช้การกำหนดขนาดด้วย CSS ซึ่งจะกล่าวถึงในบทต่อไป

แนวปฏิบัติที่ดีในการแสดงรูปภาพในหน้าเว็บคือการจัดทำรูปภาพในขนาดและความละเอียดที่เหมาะสมในการแสดงผลโดยไม่ต้องมาปรับในส่วนของเอกสารเว็บอีก นอกจากนี้ชนิดของไฟล์รูปภาพที่เว็บเบราว์เซอร์ส่วนใหญ่สนับสนุนประกอบด้วย PNG, JPEG, SVG, WebP, GIF, APNG และ AVIF ซึ่งแต่ละชนิดมีจุดเด่นที่ต่างกัน เช่น PNG รองรับความโปร่งใส WebP มีขนาดไฟล์เล็กและประสิทธิภาพสูง ส่วน AVIF เป็นฟอร์แมตใหม่ที่เน้นการบีบอัดแต่ยังรักษาคุณภาพของภาพได้ดี เป็นต้น ทั้งนี้ การเลือกชนิดของรูปภาพที่เหมาะสมควรพิจารณาจากความสมดุลระหว่างคุณภาพของภาพและประสิทธิภาพในการโหลดหน้าเว็บ

2. แท็กสำหรับการสร้างลิงก์

แท็กสำหรับการสร้างลิงก์ (Anchor Tag) จะใช้แท็ก <a> ซึ่งเป็นแท็กแบบคู่ที่มีหน้าที่สร้างการเชื่อมโยง (Hyperlink) ไปยังหน้าเว็บอื่นๆ ทั้งที่อยู่ในตำแหน่งที่เก็บทรัพยากรในการให้บริการเว็บที่เดียวกันหรืออยู่ในเครื่องผู้ให้บริการเว็บอื่น รวมทั้งการเชื่อมโยงไปยังส่วนต่าง ๆ ของเอกสารเว็บหน้าเดียวกัน (Anchor) การดาวน์โหลดไฟล์ การส่งอีเมลล์ และการโทรศัพท์ รูปแบบของแท็กสำหรับการสร้างลิงก์มีดังนี้

รูปแบบที่ 1 **ใช้ในการเชื่อมโยงไปยังตำแหน่งในหน้าเว็บเดียวกัน**

ข้อความหรืออิลิเมนต์

ตัวอย่างที่ 2

```
...  ...
15   <a href="#topic-10">ไปยังหัวข้อที่ 10</a>
```

```
...  ...
90   <h2 id="topic-10">หัวข้อเรื่องที่ 10</h2>
```

รูปแบบการใช้งานแท็ก <a> ในรูปแบบที่ 1 ใช้สำหรับการเลื่อนหน้าเว็บไปยังตำแหน่งเฉพาะภายในเอกสาร HTML เดียวกัน โดยกำหนด href เป็น # ตามด้วยชื่อของ id ของอิลิเมนต์ปลายทาง ซึ่งเมื่อผู้ใช้คลิกที่ข้อความลิงก์ เว็บเบราว์เซอร์จะเลื่อน (Scroll) ไปยังตำแหน่งของอิลิเมนต์ที่มี id ตรงกับที่ระบุใน href

จากตัวอย่างที่ 2 ข้อความ *ไปยังหัวข้อที่ 10* มีการแสดงผลในในส่วนต้นของหน้าเว็บเนื่องจากเขียนอยู่ในบรรทัดที่ 10 ของเอกสารเว็บ และถูกกำหนดให้เป็นลิงก์เชื่อมโยงไปยังอิลิเมนต์ที่มีแอตทริบิวต์ id เป็น topic-10 (href="#topic-10") ซึ่งเป็น id ของอิลิเมนต์ h2 (หัวข้อเรื่องที่ 10) โดยอิลิเมนต์ดังกล่าวอาจมีการแสดงผลในส่วนท้ายๆ ของหน้าเว็บเนื่องจากถูกเขียนไว้ในบรรทัดที่ 90 ของเอกสารเว็บ เมื่อมีการคลิกข้อความที่เป็นลิงก์แล้ว เว็บเบราว์เซอร์จะเลื่อนลงไปแสดงผลที่ *หัวข้อเรื่องที่ 10* โดยอัตโนมัติ ช่วยให้ผู้ใช้สามารถเข้าถึงเนื้อหาที่สนใจได้อย่างรวดเร็ว

รูปแบบที่ 2 **ใช้ในการเชื่อมโยงไปยัง URL ที่กำหนด**

ข้อความหรืออิลิเมนต์

ตัวอย่างที่ 3

```
<a href="https://www.ajnok.com">เว็บ AJNOK</a>
```

รูปแบบการใช้งานแท็ก <a> ในรูปแบบที่ 2 ใช้สำหรับทำข้อความหรืออิลิเมนต์ที่เป็นเนื้อหาของแท็ก <a> ให้เป็นลิงก์เชื่อมโยงไปยังแหล่งข้อมูลภายนอกเว็บไซต์ โดยระบุ URL ให้กับแอตทริบิวต์ href ของแท็ก <a>

จากตัวอย่างที่ 3 ข้อความ *เว็บ AJNOK* ถูกกำหนดให้เป็นลิงก์เพื่อเชื่อมโยงไปยัง https://www.ajnok.com

รูปแบบที่ 3 **ใช้ในการเชื่อมโยงไปยังเอกสารเว็บที่อยู่ในเครื่องผู้ให้บริการเว็บเดียวกัน**

ข้อความหรืออิลิเมนต์

ตัวอย่างที่ 4

```
1  <a href="schedule.html">ตารางสอน</a>
2  <a href="course/webapp.html">วิชาการพัฒนาเว็บแอปพลิเคชัน</a>
```

การใช้งานแท็ก <a> ในรูปแบบที่ 3 ใช้สำหรับทำข้อความหรืออิลิเมนต์ที่เป็นเนื้อหาของแท็ก <a> ให้เป็นลิงก์เชื่อมโยงไปยังไฟล์เอกสารเว็บที่อยู่ในเครื่องผู้ให้บริการเว็บเดียวกัน โดยจะอ้างอิงตำแหน่งของไฟล์เอกสารเว็บปลายทางแบบสัมพันธ์ (Relative Path) ตำแหน่งของเอกสารเว็บต้นทางที่ลิงก์นี้ตั้งอยู่

จากตัวอย่างที่ 4 บรรทัดที่ 1 มีการกำหนดข้อความ *ตารางสอน* ให้เป็นลิงก์เพื่อเชื่อมโยงไปยังไฟล์เอกสารเว็บชื่อ schedule.html ที่อยู่ในโฟลเดอร์เดียวกันกับเอกสารเว็บต้นทาง ส่วนบรรทัดที่ 2 เป็นการทำให้ข้อความ

วิชาการพัฒนาเว็บแอปพลิเคชัน เป็นลิงก์เชื่อมโยงไปยังไฟล์เอกสารเว็บชื่อ webapp.html ซึ่งถูกจัดเก็บไว้ในโฟลเดอร์ชื่อ course

รูปแบบที่ 4 **ใช้ในการเชื่อมโยงเพื่อส่งอีเมล**

ข้อความหรืออิลิเมนต์

ตัวอย่างที่ 5

```
<a href="mailto:webapp@ajnokbook.com">ส่งเมลถึงผู้สอน</a>
```

รูปแบบการใช้งานแท็ก <a> ในรูปแบบที่ 4 ใช้สำหรับทำข้อความหรืออิลิเมนต์ที่เป็นเนื้อหาของแท็ก <a> ให้เป็นลิงก์เชื่อมโยงเพื่อส่งอีเมล โดยเมื่อผู้ใช้คลิกที่ลิงก์ เว็บเบราว์เซอร์จะเรียกโปรแกรมรับส่งอีเมล เช่น Outlook หรือ Mail ที่ติดตั้งไว้ในเครื่องผู้ใช้ขึ้นมา พร้อมระบุช่องผู้รับล่วงหน้าด้วยอีเมลที่ระบุในค่า mailto: ภายในแอตทริบิวต์ href ของแท็ก <a> โดยอัตโนมัติ

จากตัวอย่างที่ 5 ข้อความ *ส่งเมลถึงผู้สอน* ถูกกำหนดให้เป็นลิงก์อีเมล โดยระบุที่อยู่อีเมลเป็น webapp@ajnokbook.com ซึ่งเมื่อคลิกแล้ว โปรแกรมอีเมลของผู้ใช้จะเปิดหน้าต่างเขียนจดหมายขึ้นมาใหม่ และระบุที่อยู่อีเมลของผู้รับโดยอัตโนมัติในช่องส่งถึงของโปรแกรมอีเมล

รูปแบบที่ 5 **ใช้ในการเชื่อมโยงเพื่อโทรออก**

ข้อความหรืออิลิเมนต์

ตัวอย่างที่ 6

```
<a href="tel:0888888888">ติดต่อทางโทรศัพท์</a>
```

รูปแบบการใช้งานแท็ก <a> ในรูปแบบที่ 5 ใช้สำหรับสร้างลิงก์ที่เมื่อผู้ใช้คลิกแล้วจะเป็นการโทรออกไปยังหมายเลขโทรศัพท์ที่กำหนดไว้ โดยลิงก์ดังกล่าวจะสามารถทำงานได้บนอุปกรณ์ที่รองรับการโทรออก เช่น สมาร์ตโฟน หรืออุปกรณ์ที่ติดตั้งโปรแกรมโทรศัพท์ผ่านเครือข่ายอินเทอร์เน็ต (VoIP)

จากตัวอย่างที่ 6 ข้อความ *ติดต่อทางโทรศัพท์* ถูกกำหนดให้เป็นลิงก์ที่เชื่อมโยงไปยังหมายเลขโทรศัพท์ 0888888888 โดยใช้ค่า tel: นำหน้าหมายเลขในแอตทริบิวต์ href ของแท็ก <a> เมื่อผู้ใช้แตะที่ลิงก์นี้จากสมาร์ตโฟน ระบบจะเปิดหน้าต่างโทรศัพท์โดยอัตโนมัติและเตรียมโทรออกไปยังหมายเลขที่ระบุ ซึ่งเป็นแนวทางที่นิยมใช้สำหรับเว็บไซต์ที่ต้องการให้ผู้ใช้งานติดต่อสอบถามได้สะดวกผ่านอุปกรณ์สมาร์ตโฟน

รูปแบบที่ 6

ข้อความหรืออิลิเมนต์

ตัวอย่างที่ 7

```
<a href="webapp.pdf"  download>ดาวน์โหลดหนังสือ</a>
```

รูปแบบการใช้งานแท็ก <a> ในรูปแบบที่ 6 ใช้สำหรับสร้างลิงก์ให้กับข้อความหรืออิลิเมนต์ที่เมื่อผู้ใช้คลิกแล้วจะทำการดาวน์โหลดไฟล์จาก URL หรือไฟล์ที่ระบุไว้ในแอตทริบิวต์ href แทนที่จะเปิดแสดงผลในเว็บเบราว์เซอร์เหมือนพฤติกรรมปกติของไฟล์บางประเภท เช่น PDF หรือรูปภาพ เป็นต้น

ในตัวอย่างที่ 7 ข้อความ *ดาวน์โหลดหนังสือ* ถูกกำหนดให้เป็นลิงก์ที่เชื่อมโยงไปยังไฟล์ชื่อ webapp.pdf ซึ่งอยู่ในโฟลเดอร์เดียวกันกับไฟล์เอกสารเว็บที่เรียกใช้งาน และด้วยการเพิ่มแอตทริบิวต์ download เบราว์เซอร์จะสั่งให้ดาวน์โหลดไฟล์ webapp.pdf โดยตรงไปยังอุปกรณ์ของผู้ใช้เมื่อมีการคลิกที่ลิงก์ดังกล่าว

3. แท็กสำหรับการกำหนดรายการ

แท็กสำหรับการกำหนดรายการ (List Tags) เป็นกลุ่มแท็กที่ใช้ในการจัดระเบียบข้อมูลให้อยู่ในรูปของรายการ (List) เพื่อให้ผู้อ่านสามารถทำความเข้าใจเนื้อหาที่แบ่งเป็นข้อๆ ได้สะดวกมากยิ่งขึ้น โดยเฉพาะในกรณีที่ต้องการแสดงขั้นตอน กระบวนการ หรือกลุ่มของข้อมูลที่มีลักษณะเหมือนกันหลายรายการ ซึ่งเป็นโครงสร้างที่พบได้ทั่วไปในการพัฒนาเว็บแอปพลิเคชัน เช่น รายการสินค้า รายการบริการ รายการเมนู หรือรายการคำอธิบาย เป็นต้น แท็กสำหรับการกำหนดรายการที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชันประกอบด้วย แท็กสำหรับรายการแบบไม่มีลำดับ และแท็กสำหรับรายการแบบมีลำดับ

3.1 แท็กสำหรับรายการแบบไม่มีลำดับ

แท็กสำหรับรายการแบบไม่มีลำดับ (Unordered List: UL) จะใช้แท็ก <ul> ซึ่งเป็นแท็กแบบคู่ที่ทำหน้าที่จัดกลุ่มข้อมูลให้อยู่ในรูปของรายการที่ไม่จำเป็นต้องมีลำดับก่อนหลัง โดยรายการแต่ละรายการ (List Item: LI) จะกำกับด้วยแท็ก <li> ซึ่งมีความสัมพันธ์เป็นอิลิเมนต์ลูกของอิลิเมนต์ ul แท็กสำหรับรายการแบบไม่มีลำดับนี้เหมาะกับกรณีที่ลำดับของรายการไม่มีนัยสำคัญเชิงเปรียบเทียบของลำดับรายการ เช่น การแสดงหมวดหมู่สินค้า รายการฟีเจอร์ของระบบ หรือเมนูนำทาง เป็นต้น ลักษณะการแสดงผลของ <ul> และ <li> มีค่าเริ่มต้นการแสดงผลเป็นแบบ Block-Level กล่าวคือ จะแสดงรายการในบรรทัดใหม่โดยอัตโนมัติ และแต่ละรายการมักจะแสดงสัญลักษณ์นำหน้า เช่น จุดกลม วงกลม หรือสี่เหลี่ยม ซึ่งสามารถกำหนดเพิ่มเติมได้ผ่าน CSS โดยใช้คุณสมบัติ list-style-type รูปแบบและตัวอย่างการใช้งานแท็กสำหรับรายการที่ไม่มีลำดับมีดังนี้

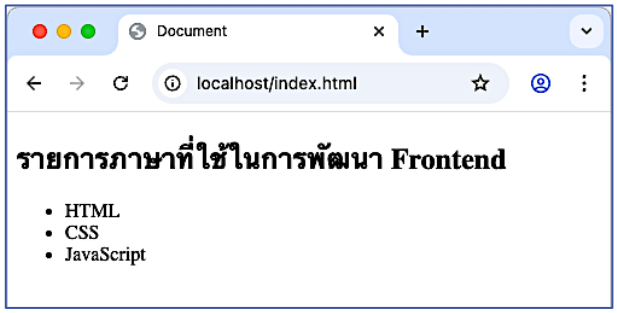
รูปแบบ

```
<ul>
    <li>ข้อความหรืออิลิเมนต์</li>
    <li>ข้อความหรืออิลิเมนต์</li>
    ...
    <li>ข้อความหรืออิลิเมนต์</li>
</ul>
```

ตัวอย่างที่ 8

```
1  <h2>รายการภาษาที่ใช้ในการพัฒนา Frontend</h2>
2  <ul>
3      <li>HTML</li>
4      <li>CSS</li>
5      <li>JavaScript</li>
6  </ul>
```

ผลลัพธ์ของตัวอย่างที่ 8



จากตัวอย่างที่ 8 บรรทัดที่ 2-6 เป็นการกำหนดเนื้อหาให้มีการแสดงผลในลักษณะรายการที่ไม่มีลำดับ ซึ่งกำหนดโดยให้อิลิเมนต์ ul เป็นอิลิเมนต์แม่โดยที่มีอิลิเมนต์ li เป็นอิลิเมนต์ลูก ประกอบด้วย รายการ HTML CSS และ JavaScript ตามลำดับ เมื่อแสดงผลบนเว็บเบราว์เซอร์ รายการแต่ละรายการจะถูกจัดเรียงในแนวตั้ง และแสดงสัญลักษณ์นำหน้าตามค่าเริ่มต้นตามผลลัพธ์ที่แสดงด้านล่างนี้ ทั้งนี้สามารถปรับลักษณะของสัญลักษณ์ด้วย CSS ได้ในภายหลัง

3.2 แท็กสำหรับรายการแบบมีลำดับ

แท็กสำหรับรายการแบบมีลำดับ (Ordered List: OL) จะมีการใช้แท็ก <ol> ซึ่งเป็นแท็กแบบคู่ที่ใช้ร่วมกับแท็ก <li> เพื่อแสดงรายการเนื้อหาที่มีลำดับความสำคัญหรือแสดงขั้นตอนตามลำดับ เช่น ลำดับการดำเนินการ ลำดับเหตุการณ์ หรือขั้นตอนการกรอกแบบฟอร์ม โดยค่าเริ่มต้นของรายการที่แสดงจะเป็นตัวเลขเรียงลำดับในแนวตั้ง แต่สามารถปรับเปลี่ยนรูปแบบการแสดงลำดับเป็นตัวอักษรหรือตัวเลขโรมันได้ผ่าน CSS ได้เช่นกัน ทั้งนี้ มีค่าเริ่มต้นเป็น Block-Level Element เช่นเดียวกับ <ul> รูปแบบและตัวอย่างการใช้งานแท็กสำหรับการกำหนดรายการแบบมีลำดับนี้จะคล้ายกับการกำหนดรายการแบบไม่มีลำดับดังนี้

รูปแบบ

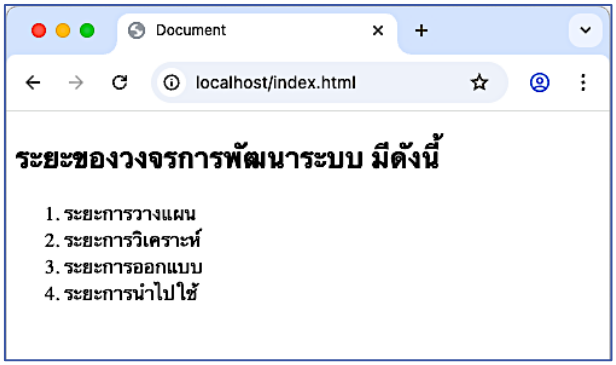
```
<ol>
    <li>ข้อความหรืออิลิเมนต์</li>
```

```
    <li>ข้อความหรืออิลิเมนต์</li>
    ...
    <li>ข้อความหรืออิลิเมนต์</li>
</ol>
```

ตัวอย่างที่ 9

```
1   <h2>ระยะของวงจรการพัฒนาระบบ มีดังนี้ </h2>
2   <ol>
3       <li>ระยะการวางแผน</li>
4       <li>ระยะการวิเคราะห์</li>
5       <li>ระยะการออกแบบ</li>
6       <li>ระยะการนำไปใช้</li>
7   </ol>
```

ผลลัพธ์ของตัวอย่างที่ 9



จากตัวอย่างที่ 9 บรรทัดที่ 2–7 เป็นการแสดงรายการแบบมีลำดับโดยใช้แท็ก <ol> ครอบคลุมรายการที่กำหนดด้วยแท็ก <li> เพื่อแสดงลำดับของระยะต่าง ๆ ในการพัฒนาระบบ ซึ่งเมื่อแสดงผลในเว็บเบราว์เซอร์ รายการเหล่านี้จะเรียงลำดับเป็นตัวเลขอัตโนมัติดังผลลัพธ์ที่แสดงด้านล่างนี้ โดยลำดับมีความหมายต่อเนื้อหา ถ้าสลับลำดับจะทำให้ความหมายเปลี่ยนไป ต่างจากแท็ก <ul> ที่ลำดับไม่มีนัยสำคัญ

4. แท็กสำหรับการกำหนดหัวเรื่องและข้อความ

แท็กที่ใช้ในการกำหนดหัวเรื่องและข้อความเป็นแท็กที่มีเป้าประสงค์เพื่อจัดระเบียบและแสดงเนื้อหาข้อความในหน้าเว็บให้มีโครงสร้างชัดเจน เข้าใจง่าย และส่งเสริมการสื่อสารสาระสำคัญให้กับผู้ใช้งาน โดยเฉพาะอย่างยิ่งในการจัดลำดับความสำคัญของเนื้อหา เช่น การแสดงหัวเรื่องหลัก หัวเรื่องย่อย และข้อความแต่ละย่อหน้า ตลอดจนการแทรกบรรทัดใหม่หรือการคั่นเนื้อหาด้วยเส้นคั่น ทั้งนี้ แท็กในกลุ่มนี้ยังส่งผลต่อการเข้าถึงของเครื่องมือค้นหา

(SEO) และช่วยให้เครื่องมือช่วยอ่านหน้าจอสามารถตีความโครงสร้างของเอกสารได้อย่างถูกต้องอีกด้วย แท็กที่ใช้ในการกำหนดหัวเรื่องและข้อความที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน ประกอบด้วย แท็กสำหรับกำหนดระดับของหัวเรื่อง แท็กสำหรับกำหนดย่อหน้า แท็กสำหรับการขึ้นบรรทัดใหม่ และแท็กสำหรับการแสดงเส้นคั่น

4.1 แท็กสำหรับกำหนดระดับของหัวเรื่อง

แท็กสำหรับกำหนดระดับของหัวเรื่อง (Heading Tags) เป็นแท็กที่มีจุดมุ่งหมายในการจัดลำดับความสำคัญของหัวเรื่องในหน้าเว็บตามระดับความสำคัญ ในภาษา HTML มีการแบ่งความสำคัญของหัวเรื่องออกเป็น 6 ระดับ โดยใช้แท็ก <h1> ถึง <h6> ซึ่งแท็ก <h1> เป็นหัวเรื่องที่มีระดับความสำคัญสูงที่สุด เช่น ชื่อบท หรือชื่อเรื่องหลักของหน้าเว็บนั้นๆ เป็นต้น ส่วนแท็ก <h6> เป็นหัวเรื่องที่มีระดับความสำคัญน้อยที่สุดในหน้าเว็บนั้น แท็กสำหรับกำหนดระดับของหัวเรื่องจะเขียนไว้ในอิลิเมนต์ body โดยมีรูปแบบดังนี้

รูปแบบ

```
<h1>ชื่อหัวเรื่องความสำคัญระดับ 1</h1>
<h2>ชื่อหัวเรื่องความสำคัญระดับ 2</h2>
<h3>ชื่อหัวเรื่องความสำคัญระดับ 3</h3>
<h4>ชื่อหัวเรื่องความสำคัญระดับ 4</h4>
<h5>ชื่อหัวเรื่องความสำคัญระดับ 5</h5>
<h6>ชื่อหัวเรื่องความสำคัญระดับ 6</h6>
```

การใช้งานแท็กกำหนดระดับของหัวเรื่องจะใช้แท็กคู่ โดยมีแท็กเปิดและแท็กปิดเพื่อระบุจุดเริ่มต้นและจุดสิ้นสุดของข้อความหัวเรื่อง เช่น <h1>ชื่อหัวเรื่องความสำคัญระดับ 1</h1> โดยนักพัฒนาควรใช้ลำดับของแท็กตามโครงสร้างของเนื้อหา เช่น ไม่ควรใช้ <h3> ก่อน <h2> เพื่อให้โครงสร้างเว็บมีความสมเหตุสมผลและถูกต้องตามมาตรฐาน นอกจากนี้มีแนวปฏิบัติที่ดีในการใช้แท็กกำหนดระดับของหัวเรื่อง (Mozilla Developer Network, 2024) ดังนี้

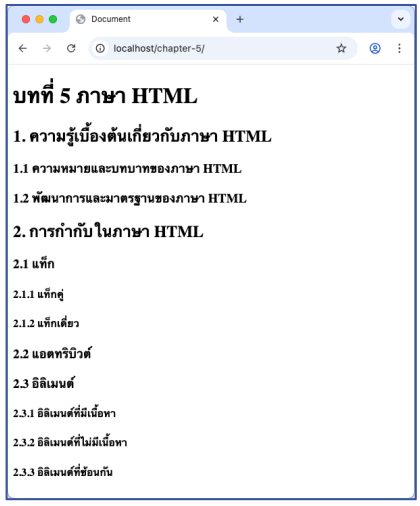
- ใช้ <h1> เพียงหนึ่งรายการต่อหนึ่งหน้าเว็บ เพื่อแสดงหัวเรื่องหลัก
- หลีกเลี่ยงการข้ามลำดับ เช่น ไม่ควรใช้ <h3> ก่อนการใช้ <h2> ยกเว้นในกรณีที่มีการเรียงลำดับหัวเรื่องและต้องการเริ่มหัวเรื่องใหม่ลักษณะดังตัวอย่างที่ 27
- หลีกเลี่ยงการใช้ <h1> ซ้ำกันในหน้าเว็บเดียวกันโดยไม่จำเป็น
- แท็กกำหนดระดับของหัวเรื่องนี้มีจุดมุ่งหมายสำหรับการจัดโครงสร้างของเนื้อหา ไม่ได้มีจุดมุ่งหมายเพื่อการกำหนดขนาดของตัวอักษร

ตัวอย่างที่ 10

```
1 <h1>บทที่ 5 ภาษา HTML</h1>
```

```
2   <h2>1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML</h2>
3   <h3>1.1 ความหมายและบทบาทของภาษา HTML</h3>
4   <h3>1.2 พัฒนาการและมาตรฐานของภาษา HTML </h3>
5   <h2>2. การกำกับในภาษา HTML</h2>
6   <h3>2.1 แท็ก</h3>
7   <h4>2.1.1 แท็กคู่</h4>
8   <h4>2.1.2 แท็กเดี่ยว</h4>
9   <h3>2.2 แอตทริบิวต์</h3>
10  <h3>2.3 อิลิเมนต์</h3>
11  <h4>2.3.1 อิลิเมนต์ที่มีเนื้อหา</h4>
12  <h4>2.3.2 อิลิเมนต์ที่ไม่มีเนื้อหา</h4>
13  <h4>2.3.3 อิลิเมนต์ที่ซ้อนกัน</h4>
```

ผลลัพธ์ของตัวอย่างที่ 10



จากตัวอย่างที่ 10 เป็นการใช้แท็ก <h1> ถึง <h4> เพื่อจัดลำดับหัวเรื่องของเอกสาร HTML อย่างเป็นระบบ โดยมีการกำหนดหัวเรื่องหลักของหน้าเว็บเป็น *บทที่ 5 ภาษา HTML* ซึ่งใช้แท็ก <h1> ในการกำกับข้อความให้เป็นหัวเรื่องระดับสำคัญที่สุดในเอกสาร จากนั้นจึงใช้ <h2> สำหรับข้อความที่เป็นหัวเรื่องรอง ประกอบด้วย *1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML* และ *2. การกำกับในภาษา HTML* ซึ่งเป็นหัวเรื่องระดับรองในบทนั้น ส่วนหัวเรื่องย่อยภายใต้หัวเรื่องรองจะใช้ <h3> และหัวเรื่องย่อยลงไปอีกชั้นหนึ่งจะใช้ <h4> ตามลำดับ โครงสร้างเช่นนี้แสดงให้เห็นถึงการจัดระดับหัวเรื่องอย่างชัดเจน ทั้งในแง่การแสดงผลให้ผู้ใช้มองเห็นและเครื่องมือช่วยอ่านหน้าจอที่สามารถวิเคราะห์ลำดับความสำคัญของเนื้อหาได้ถูกต้องตามลำดับ

ผลลัพธ์ของการใช้แท็ก <h1> ถึง <h4> ในการจัดระดับหัวเรื่องในเอกสารเว็บตามตัวอย่างที่ 11 โดยแต่ละแท็กจะมีลักษณะการแสดงผลที่เว็บเบราว์เซอร์กำหนดไว้ล่วงหน้า (Default Style) เช่น ขนาดตัวอักษร ความหนา และระยะห่างของแต่ละหัวเรื่อง โดยอิลิเมนต์ h1 จะมีการแสดงผลด้วยตัวอักษรขนาดใหญ่ที่สุดและลดหลั่นลงมาจนถึงอิลิเมนต์ h6 ซึ่งมีขนาดตัวอักษรเล็กที่สุด ทั้งนี้การแสดงผลเหล่านี้สามารถปรับเปลี่ยนได้ด้วยการใช้คำสั่ง CSS เช่น font-size หรือ font-weight เพื่อให้เหมาะสมกับการออกแบบของเว็บแอปพลิเคชัน นอกจากนี้ ค่าเริ่มต้นในการแสดงผลของอิลิเมนต์ h1 ถึง h6 จะถูกจัดให้อยู่ในลักษณะ Block-level หมายความว่า แต่ละหัวเรื่องจะเริ่มแสดงผลที่บรรทัดใหม่เสมอ และยึดครองความกว้างทั้งหมดของพื้นที่ที่บรรจุอยู่ (Container) จากตัวอย่างที่ 10 อิลิเมนต์หัวเรื่องต่าง ๆ จะอยู่ในพื้นที่การแสดงผลของอิลิเมนต์ body ซึ่งก็คือพื้นที่การแสดงผลของเว็บเบราว์เซอร์

4.2 แท็กสำหรับกำหนดย่อหน้า

การกำหนดข้อความให้แสดงผลในลักษณะของย่อหน้าจะใช้แท็ก <p> ในการกำหนด โดยข้อความแต่ละย่อหน้าจะถูกแสดงแยกออกจากกันด้วยระยะห่างแนวตั้ง และเริ่มต้นที่บรรทัดใหม่โดยอัตโนมัติ แท็กสำหรับกำหนดย่อหน้านี้จะถูกเขียนไว้ในอิลิเมนต์ body โดยมีรูปแบบดังนี้

รูปแบบ

```
<p>ข้อความของย่อหน้า</p>
```

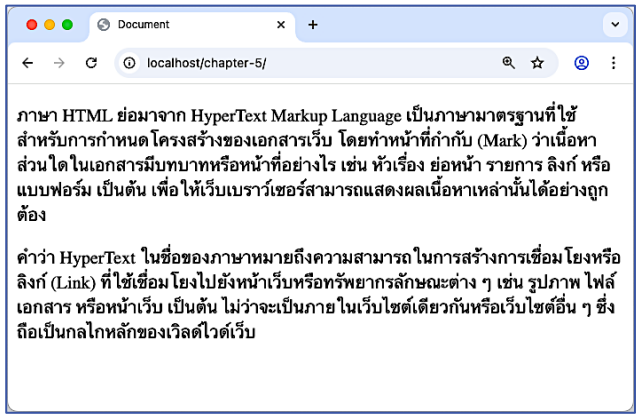
การใช้งานแท็กกำหนดย่อหน้าจะใช้แท็กคู่ โดยมีแท็กเปิดและแท็กปิดเพื่อระบุจุดเริ่มต้นและจุดสิ้นสุดของย่อหน้า ค่าเริ่มต้นในการแสดงผลของอิลิเมนต์ p เป็นแบบ Block-level เช่นเดียวกันอิลิเมนต์ h1-h6 ตัวอย่างของการกำหนดข้อความให้เป็นย่อหน้าแสดงดังตัวอย่างที่ 11

ตัวอย่างที่ 11

```
1  <p>
2  ภาษา HTML ย่อมาจาก HyperText Markup Language เป็นภาษามาตรฐานที่ใช้สำหรับการกำหนดโครงสร้างของเอกสารเว็บ โดยทำหน้าที่กำกับ (Mark) ว่าเนื้อหาส่วนใดในเอกสารมีบทบาทหรือหน้าที่อย่างไร เช่น หัวเรื่อง ย่อหน้า รายการ ลิงก์ หรือแบบฟอร์ม เป็นต้น เพื่อให้เว็บเบราว์เซอร์สามารถแสดงผลเนื้อหาเหล่านั้นได้อย่างถูกต้อง
3  </p>
4  <p>
5  คำว่า HyperText ในชื่อของภาษาหมายถึงความสามารถในการสร้างการเชื่อมโยงหรือลิงก์ (Link) ที่ใช้เชื่อมโยงไปยังหน้าเว็บหรือทรัพยากรลักษณะต่าง ๆ เช่น รูปภาพ ไฟล์เอกสาร หรือหน้าเว็บ เป็นต้น ไม่ว่าจะเป็นภายในเว็บไซต์เดียวกันหรือเว็บไซต์อื่น ๆ ซึ่งถือเป็นกลไกหลักของเวิลด์ไวด์เว็บ
6  </p>
```

จากตัวอย่างที่ 11 มีการกำหนดข้อความให้เป็นย่อหน้าจำนวน 2 ย่อหน้า โดยย่อหน้าแรกมีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของย่อหน้าที่บรรทัดที่ 1 และ 3 ตามลำดับ ส่วนย่อหน้าที่ 2 มีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของย่อหน้าที่บรรทัดที่ 4 และ 6 ตามลำดับ

ผลลัพธ์ของตัวอย่างที่ 11



ผลลัพธ์การแสดงผลของตัวอย่างที่ 11 จะแสดงในลักษณะแยกย่อหน้ากันอย่างชัดเจน โดยเริ่มบรรทัดใหม่และมีช่องว่างระหว่างกันตามค่าเริ่มต้นของเว็บเบราว์เซอร์ ทั้งนี้นักพัฒนาสามารถใช้ CSS ปรับระยะห่างหรือรูปแบบของย่อหน้าเพิ่มเติมได้ตามต้องการ พื้นที่การแสดงผลของแต่ละย่อหน้าจะขึ้นอยู่กับขนาดของพื้นที่ที่เป็นอิลิเมนต์แม่ ซึ่งในที่นี้อิลิเมนต์ body เป็นอิลิเมนต์แม่ของอิลิเมนต์ p ทั้งสอง ทำให้พื้นที่การแสดงผลคือความกว้างของหน้าต่างเว็บเบราว์เซอร์

4.3 แท็กสำหรับการขึ้นบรรทัดใหม่

ตามปกติแล้วเนื้อหาที่ไม่ได้มีการกำกับการแสดงผลใด ๆ จะแสดงต่อเนื่องกันไปตามขนาดความกว้างของเว็บเบราว์เซอร์ หากต้องการกำหนดจุดแสดงผลในบรรทัดใหม่ (Line Break) โดยไม่ต้องสร้างเป็นย่อหน้าใหม่ จะใช้แท็ก
 ซึ่งเป็นแท็กเดี่ยว ไม่ต้องมีแท็กปิดเพื่อกำหนดจุดสิ้นสุดของการขึ้นบรรทัดใหม่ โดยมีรูปแบบและตัวอย่างการใช้งานดังนี้

รูปแบบ

**ข้อความหรือเนื้อหาลักษณะอื่นก่อนขึ้นบรรทัดใหม่
ข้อความหรือเนื้อหาบรรทัดถัดไป**

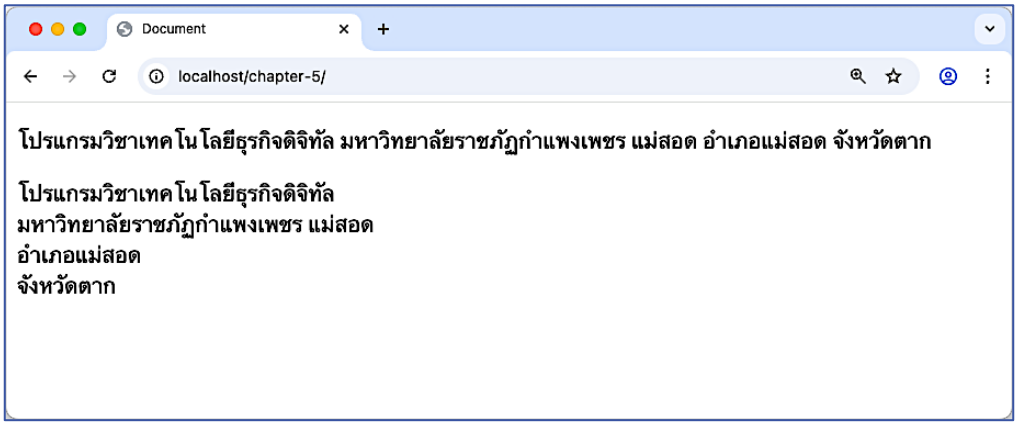
ตัวอย่างที่ 12

```
1  <p>
2  โปรแกรมวิชาเทคโนโลยีธุรกิจดิจิทัล
3  มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด
```

```
4   อำเภอแม่สอด
5   จังหวัดตาก
6   </p>
7   <p>
8   โปรแกรมวิชาเทคโนโลยีธุรกิจดิจิทัล<br>มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด<br>อำเภอแม่สอด
    <br>จังหวัดตาก
9   </p>
```

จากตัวอย่างที่ 12 เป็นการเปรียบเทียบการแสดงผลในสองย่อหน้า ย่อหน้าแรก (บรรทัดที่ 1-6) เป็นการแยกข้อความแต่ละบรรทัดด้วยการกดปุ่ม Enter ในการเขียนรหัสคำสั่งในไฟล์เอกสารเว็บซึ่งไม่ได้มีการใช้แท็ก
 ส่วนย่อหน้าที่สอง (บรรทัดที่ 7-9 เป็นการเขียนข้อความแบบเรียงต่อกันและใช้แท็ก
 เพื่อกำหนดจุดขึ้นบรรทัดใหม่ ผลลัพธ์การแสดงผลในเว็บเบราว์เซอร์มีลักษณะดังนี้

ผลลัพธ์ของตัวอย่างที่ 12



ผลลัพธ์ที่แสดงในเว็บเบราว์เซอร์ตามตัวอย่างที่ 12 แสดงให้เห็นว่า การกดปุ่ม Enter เพื่อขึ้นบรรทัดใหม่ในการเขียนรหัสคำสั่งภายในไฟล์เอกสารเว็บจะไม่มีผลต่อการแสดงผลในเว็บเบราว์เซอร์ เนื่องจากเว็บเบราว์เซอร์จะตีความข้อความทั้งหมดภายในอิลิเมนต์ p ให้เป็นข้อความติดต่อกันในบรรทัดเดียวโดยไม่สนใจการขึ้นบรรทัดที่เกิดจากการกด Enter ในรหัสคำสั่ง หากต้องการให้ข้อความขึ้นบรรทัดใหม่ในตำแหน่งที่ต้องการจะต้องใช้แท็ก
 เพื่อระบุจุดขึ้นบรรทัดใหม่อย่างชัดเจนดังที่แสดงในย่อหน้าที่สอง ซึ่งผลลัพธ์ที่ได้จะทำให้ข้อความแสดงผลในรูปแบบที่แต่ละบรรทัดเริ่มต้นใหม่ตามตำแหน่งที่มี
 ปรากฏอยู่ในรหัสคำสั่ง

4.4 แท็กสำหรับการแสดงเส้นคั่น

แท็กสำหรับการแสดงเส้นคั่น (Horizontal Rule) เป็นแท็กที่ใช้สำหรับแสดงเส้นคั่นแนวนอนบนหน้าเว็บ โดยทั่วไปมักใช้เพื่อแบ่งเนื้อหาออกเป็นส่วน ๆ อย่างชัดเจน เช่น ระหว่างหัวข้อ ระหว่างบท หรือระหว่างย่อหน้าสำคัญ เส้นคั่นนี้ไม่มีเนื้อหาในการแสดงผลจึงมีลักษณะเป็นแท็กเดี่ยวมีรูปแบบ ตัวอย่างและผลลัพธ์ดังต่อไปนี้

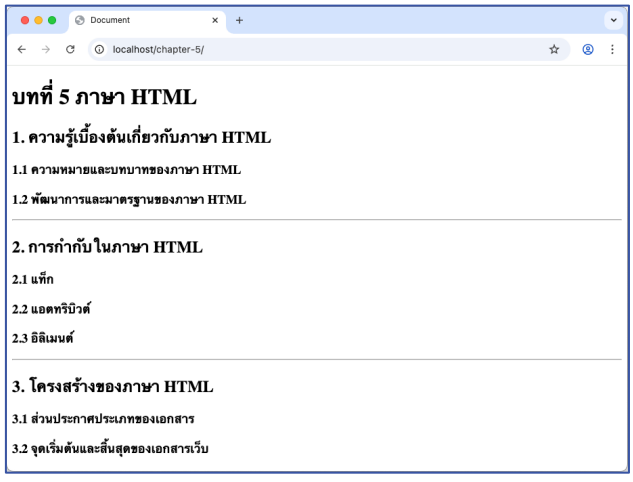
รูปแบบ

```
<hr>
```

ตัวอย่างที่ 13

```
1  <h1>บทที่ 5 ภาษา HTML</h1>
2  <h2>1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML</h2>
3  <h3>1.1 ความหมายและบทบาทของภาษา HTML</h3>
4  <h3>1.2 พัฒนาการและมาตรฐานของภาษา HTML </h3>
5  <hr>
6  <h2>2. การกำกับในภาษา HTML</h2>
7  <h3>2.1 แท็ก</h3>
8  <h3>2.2 แอตทริบิวต์</h3>
9  <h3>2.3 อิลิเมนต์</h3>
10 <hr>
11 <h2>3. โครงสร้างของภาษา HTML</h2>
12 <h3>3.1 ส่วนประกาศประเภทของเอกสาร</h3>
13 <h3>3.2 จุดเริ่มต้นและสิ้นสุดของสเอกสารเว็บ</h3>
```

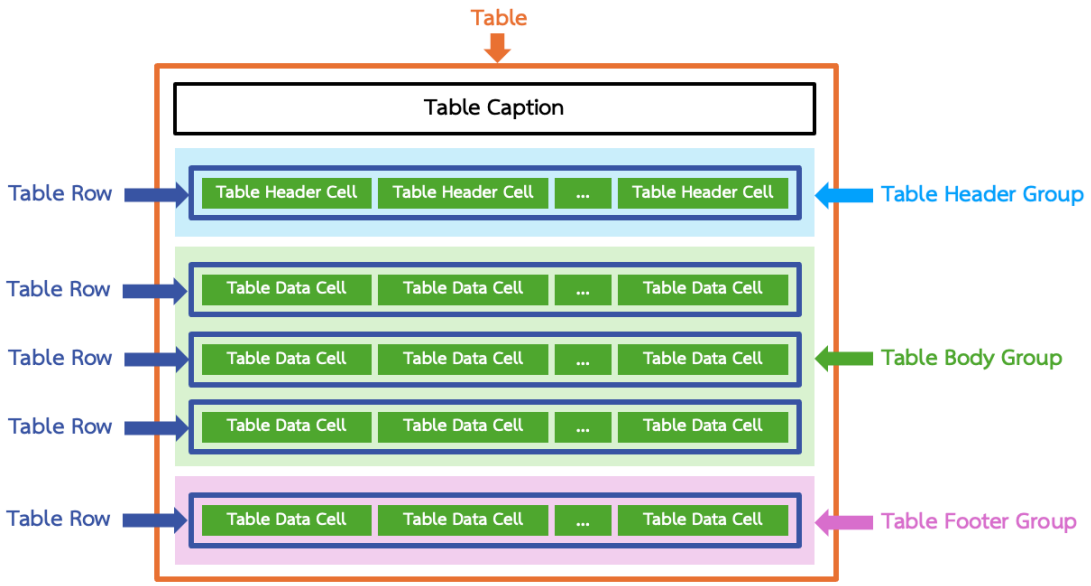
ผลลัพธ์ของตัวอย่างที่ 13



จากตัวอย่างที่ 13 มีการใช้แท็ก <hr> จำนวน 2 ครั้งคือในบรรทัดที่ 5 และ 10 เพื่อแบ่งเนื้อหาออกเป็น 3 ส่วนหลัก ได้แก่ ความรู้เบื้องต้นเกี่ยวกับภาษา HTML การกำกับในภาษา HTML และโครงสร้างของภาษา HTML เส้นคั่นที่ปรากฏจะช่วยให้ผู้ใช้งานสามารถแยกเนื้อหาแต่ละกลุ่มได้ง่ายขึ้นทางสายตา ส่งผลให้การอ่านมีความต่อเนื่องและแบ่งตอนของเนื้อหาอย่างเป็นระเบียบ ทั้งนี้ hr เป็นอิลิเมนต์แบบ Block-level ที่แสดงผลในบรรทัดใหม่และแสดงเต็มความกว้างของอิลิเมนต์แม่ ในการปรับลักษณะของเส้นสามารถทำได้ด้วย CSS เช่น ความหนา สี หรือความยาวของเส้นคั่น

5. แท็กสำหรับการสร้างตาราง

แท็กสำหรับการสร้างตารางเป็นกลุ่มแท็กที่ใช้ในการจัดข้อมูลในลักษณะตาราง (Table) ซึ่งเหมาะสำหรับการแสดงผลข้อมูลที่มีความสัมพันธ์ระหว่างแถวและคอลัมน์ เช่น ตารางแสดงรายการสินค้า ตารางแสดงผลลัพธ์จากฐานข้อมูล หรือสรุปข้อมูลในรูปแบบที่ต้องการการเปรียบเทียบ เป็นต้น องค์ประกอบของโครงสร้างตารางในภาษา HTML แสดงดังภาพที่ 6.1



ภาพที่ 6.1 องค์ประกอบของโครงสร้างตาราง

5.1 องค์ประกอบของโครงสร้างตาราง

ตารางในภาษา HTML ประกอบด้วยกลุ่มแท็กต่าง ๆ ที่มีหน้าที่กำหนดโครงสร้างของข้อมูลในรูปแบบแถวและคอลัมน์ โดยแต่ละแท็กจะมีบทบาทเฉพาะเพื่อจัดระเบียบและให้ความหมายกับข้อมูลที่แสดงผลในลักษณะตาราง สำหรับโครงสร้างของตารางสามารถจำแนกองค์ประกอบหลักได้ 8 ส่วน ตามภาพที่ 6.1 ประกอบด้วย แท็กกำหนดตาราง แท็กกำหนดคำอธิบายตาราง แท็กกำหนดแถว แท็กกำหนดเซลล์หัวข้อ แท็ก

กำหนดเซลล์ข้อมูล แท็กกำหนดกลุ่มส่วนหัวของตาราง แท็กกำหนดกลุ่มข้อมูลของตาราง แท็กกำหนดกลุ่มส่วนท้ายของตาราง

5.1.1 แท็กกำหนดตาราง

แท็กกำหนดตารางจะใช้แท็ก <table> มีลักษณะเป็นแท็กคู่ที่ต้องมีการกำหนดจุดเริ่มต้นและจุดสิ้นสุดของตาราง รวมทั้งมีค่าเริ่มต้นของการแสดงผลเป็นแบบ Block-Level รูปแบบของแท็กกำหนดตารางมีดังนี้

รูปแบบ

```
<table>รายการแท็กสำหรับการสร้างองค์ประกอบส่วนอื่นๆ ของตาราง</table>
```

5.1.2 แท็กกำหนดคำอธิบายตาราง

แท็กกำหนดคำอธิบายตารางจะใช้แท็ก <caption> มีลักษณะเป็นแท็กคู่ใช้สำหรับระบุคำอธิบายหรือชื่อของตาราง โดยมีวัตถุประสงค์เพื่ออธิบายเนื้อหาหรือความหมายโดยรวมของข้อมูลที่แสดงอยู่ภายในตารางนั้น ๆ อิลิเมนต์ caption จะต้องเป็นอิลิเมนต์ลูกของอิลิเมนต์ <table> และจะต้องปรากฏก่อนอิลิเมนต์อื่น ๆ ภายในอิลิเมนต์ table เสมอ ค่าเริ่มต้นของการแสดงผลในเว็บเบราว์เซอร์ คำอธิบายของตารางที่อยู่ภายในแท็ก <caption> จะปรากฏอยู่เหนือ โดยแท็กกำหนดคำอธิบายตารางนี้อาจละไว้ไม่ต้องกำหนดให้กับตารางก็ได้ แต่การใช้แท็กนี้ช่วยให้ตารางมีความสมบูรณ์ทางโครงสร้าง และช่วยเพิ่มประสิทธิภาพในการเข้าถึงสำหรับผู้ใช้ที่ใช้เทคโนโลยีช่วยเหลือ รูปแบบของแท็กกำหนดคำอธิบายตารางมีดังนี้

รูปแบบ

```
<caption>คำอธิบายหรือตัวอย่างชื่อของตาราง</caption>
```

5.1.3 แท็กกำหนดแถว

แท็กกำหนดแถว (Table Row: TR) จะใช้แท็ก <tr> มีลักษณะเป็นแท็กคู่ที่ใช้สำหรับกำหนดแถวของตาราง โดยจะเป็นอิลิเมนต์ลูกของอิลิเมนต์ table หรืออิลิเมนต์กลุ่มส่วนหัว กลุ่มส่วนข้อมูล หรือกลุ่มส่วนท้ายของตารางก็ได้ รูปแบบของแท็กกำหนดแถวมีดังนี้

รูปแบบ

```
<tr>รายการแท็กสำหรับกำหนดเซลล์หัวข้อหรือเซลล์ข้อมูล</tr>
```

5.1.4 แท็กกำหนดเซลล์หัวข้อ

แท็กกำหนดเซลล์หัวข้อ (Table Header Cell: TH) จะใช้แท็ก <th> ซึ่งเป็นแท็กคู่ที่ใช้สำหรับกำหนดเซลล์ในตารางที่มีลักษณะเป็นหัวข้อของแถวหรือคอลัมน์ โดยค่าที่แสดงในเซลล์นี้มักจะเป็นคำอธิบายหรือชื่อหัวข้อของข้อมูลที่อยู่ในแถวหรือคอลัมน์เดียวกัน ค่าเริ่มต้นของการแสดงผลโดยทั่วไปเว็บเบราว์เซอร์จะกำหนด

ลักษณะการแสดงผลของเซลล์หัวข้อให้มีตัวอักษรหนา และมีการจัดตำแหน่งตรงกลางโดยอัตโนมัติ ทั้งนี้อิลิเมนต์ th จะต้องเป็นอิลิเมนต์ลูกของอิลิเมนต์ tr รูปแบบของแท็กกำหนดเซลล์หัวข้อมีดังนี้

รูปแบบ

```
<th>ชื่อหัวข้อของข้อมูล</th>
```

5.1.5 แท็กกำหนดเซลล์ข้อมูล

แท็กกำหนดเซลล์ข้อมูล (Table Data Cell: TD) จะใช้อิลิเมนต์ <td> ซึ่งเป็นแท็กคู่ที่ใช้สำหรับกำหนดเซลล์ที่ใช้เก็บข้อมูลทั่วไปภายในตาราง โดยค่าที่อยู่ภายในอิลิเมนต์นี้อาจเป็นข้อความ ตัวเลข รูปภาพ หรืออิลิเมนต์อื่น ๆ ที่สามารถวางอยู่ในเซลล์ข้อมูลได้ ทั้งนี้อิลิเมนต์ td จะต้องเป็นอิลิเมนต์ลูกของอิลิเมนต์ tr เช่นเดียวกับอิลิเมนต์ th ค่าเริ่มต้นของการแสดงผลนั้นจะเป็นการจัดข้อความชิดซ้ายและไม่เป็นตัวหนา ซึ่งแตกต่างจากอิลิเมนต์ th ที่เน้นหัวข้อของข้อมูล รูปแบบของแท็กกำหนดเซลล์ข้อมูลมีดังนี้

รูปแบบ

```
<td>ข้อมูลภายในเซลล์</td>
```

5.1.6 แท็กกำหนดกลุ่มส่วนหัวของตาราง

แท็กกำหนดกลุ่มส่วนหัวของตาราง (Table Head Group: THEAD) จะใช้แท็ก <thead> ซึ่งเป็นแท็กคู่ที่ใช้ในการจัดกลุ่มของแถวที่มีบทบาทเป็นส่วนหัวของตาราง โดยมักจะเป็นแถวที่ประกอบด้วยอิลิเมนต์ th เพื่อแสดงหัวข้อของแต่ละคอลัมน์ อิลิเมนต์ thead จะเป็นอิลิเมนต์ลูกของอิลิเมนต์ table และทำหน้าที่ครอบอิลิเมนต์ tr ที่เป็นแถวของเซลล์หัวข้อของตาราง ทั้งนี้แถวที่อยู่ในอิลิเมนต์ thead สามารถมีมากกว่าหนึ่งแถวได้ รูปแบบของแท็กกำหนดเซลล์ข้อมูลมีดังนี้

รูปแบบ

```
<thead>
    <tr>
        <th>หัวข้อคอลัมน์ที่ 1</th>
        <th>หัวข้อคอลัมน์ที่ 2</th>
        ...
        </tr>
</thead>
```

การใช้แท็ก <thead> ช่วยให้สามารถจัดการลักษณะการแสดงผลของส่วนหัวของตารางได้ง่ายขึ้นผ่าน CSS และมีประโยชน์อย่างยิ่งในการแสดงผลตารางขนาดใหญ่ที่มีการแบ่งหน้าหรือมีแถวข้อมูลเป็นจำนวน

มาก โดยสามารถกำหนดให้ส่วนหัวแสดงในตำแหน่งคงที่อยู่ด้านบนของแถวต่าง ๆ ตลอดเวลา ทั้งนี้แท็ก <thead> เป็นทางเลือกในการใช้งานซึ่งอาจไม่ต้องใช้ก็ได้ แต่ถือเป็นแนวทางปฏิบัติที่ดีในการจัดระเบียบข้อมูล

5.1.7 แท็กกำหนดกลุ่มข้อมูลของตาราง

แท็กกำหนดกลุ่มข้อมูลของตาราง (Table Body Group: TBODY) จะใช้แท็ก <tbody> ซึ่งเป็นแท็กคู่ที่ใช้ในการจัดกลุ่มของแถวข้อมูลหลักของตาราง โดยแถวที่อยู่ภายในอิลิเมนต์ tbody จะเป็นข้อมูลที่สัมพันธ์กับหัวตาราง และแสดงเนื้อหาหลักของตารางในแต่ละแถว อิลิเมนต์ tbody จะเป็นอิลิเมนต์ลูกของอิลิเมนต์ table และมักใช้ร่วมกับอิลิเมนต์ thead และ tfoot เพื่อแบ่งโครงสร้างของตารางอย่างเป็นระบบ

รูปแบบ

```
<tbody>
    <tr><td>ข้อมูลแถวที่ 1 เซลล์ที่ 1</td><td>ข้อมูลแถวที่ 1 เซลล์ที่ 2</td></tr>
    <tr><td>ข้อมูลแถวที่ 2 เซลล์ที่ 1</td><td>ข้อมูลแถวที่ 2 เซลล์ที่ 2</td></tr>
    ...
</tbody>
```

การใช้งานแท็ก <tbody> แม้จะไม่จำเป็นต้องระบุในกรณีของตารางขนาดเล็ก แต่การจัดกลุ่มด้วยแท็กนี้จะช่วยให้สามารถควบคุมการแสดงผลเฉพาะส่วนข้อมูล เช่น การกำหนดรูปแบบของแถวข้อมูลด้วย CSS การแสดงผลแบบแบ่งหน้า หรือการสลับสีของแถว (Zebra Striping) ได้ง่ายและเป็นระบบมากขึ้น นอกจากนี้ในทางเทคนิค เว็บเบราว์เซอร์จะสร้างอิลิเมนต์ tbody โดยอัตโนมัติหากไม่ได้มีการระบุไว้

5.1.8 แท็กกำหนดกลุ่มส่วนท้ายของตาราง

แท็กกำหนดกลุ่มส่วนท้ายของตาราง (Table Foot Group: TFOOT) จะใช้แท็ก tfoot ซึ่งเป็นแท็กแบบคู่สำหรับจัดกลุ่มแถวที่ใช้เป็นส่วนสรุปหรือปิดท้ายของข้อมูลในตาราง เช่น แสดงผลรวมของข้อมูลในแต่ละคอลัมน์ หมายเหตุท้ายตาราง หรือค่ารวมเชิงสถิติต่าง ๆ อิลิเมนต์ tfoot เป็นอิลิเมนต์ลูกของอิลิเมนต์ table และทำหน้าที่ครอบอิลิเมนต์ tr ที่เป็นแถวของเซลล์ข้อมูลในส่วนท้าย โดยตำแหน่งในการเขียนสามารถอยู่ก่อนหรือหลังอิลิเมนต์ tbody ก็ได้ แต่ในการแสดงผลเว็บเบราว์เซอร์จะแสดง tfoot ไว้หลังข้อมูลใน tbody เสมอ รูปแบบของแท็กกำหนดกลุ่มส่วนท้ายของตารางมีดังนี้

รูปแบบ

```
<tfoot>
    <tr><td>ข้อมูลแถวที่ 1 เซลล์ที่ 1</td><td>ข้อมูลแถวที่ 1 เซลล์ที่ 2</td></tr>
    <tr><td>ข้อมูลแถวที่ 2 เซลล์ที่ 1</td><td>ข้อมูลแถวที่ 2 เซลล์ที่ 2</td></tr>
    ...
</tfoot>
```

การใช้ tfoot มีประโยชน์อย่างมากเมื่อต้องจัดรูปแบบตารางด้วย CSS โดยสามารถกำหนดลักษณะการแสดงผลเฉพาะส่วนท้ายได้อย่างยืดหยุ่น เช่น การเน้นสีแถวสรุป การแยกเส้นขอบ หรือการกำหนดขนาดฟอนต์เฉพาะกลุ่มข้อมูลในส่วนท้าย อีกทั้งยังช่วยให้องค์ประกอบของตารางมีความสมบูรณ์และชัดเจนในเชิงโครงสร้างเอกสาร

5.2 ตัวอย่างการสร้างตาราง

การสร้างตารางในภาษา HTML สามารถใช้แท็กต่าง ๆ ที่กล่าวถึงในหัวข้อก่อนหน้าเพื่อนิยามโครงสร้างของตารางให้ครบถ้วน โดยในตัวอย่างนี้จะแสดงการสร้างตารางที่มีองค์ประกอบครบตามภาพที่ 6.1 ซึ่งประกอบด้วยส่วนหัวของตาราง ส่วนข้อมูล และส่วนท้ายของตาราง รวมถึงมีการระบุหัวข้อของตาราง (caption) เพื่อให้ตารางมีความสมบูรณ์ในเชิงโครงสร้าง ดังนี้

ตัวอย่างที่ 14

```
1  <table>
2      <caption>รายการสินค้าในระบบคลัง</caption>
3      <thead>
4          <tr>
5              <th>รหัสสินค้า</th>
6              <th>ชื่อสินค้า</th>
7              <th>ประเภท</th>
8              <th>จำนวนคงเหลือ</th>
9          </tr>
10     </thead>
11     <tbody>
12         <tr>
13             <td>P001</td>
14             <td>ปากกาลูกลื่น</td>
15             <td>เครื่องเขียน</td>
16             <td>120</td>
17         </tr>
18         <tr>
19             <td>P002</td>
20             <td>สมุดบันทึก A5</td>
21             <td>เครื่องเขียน</td>
```

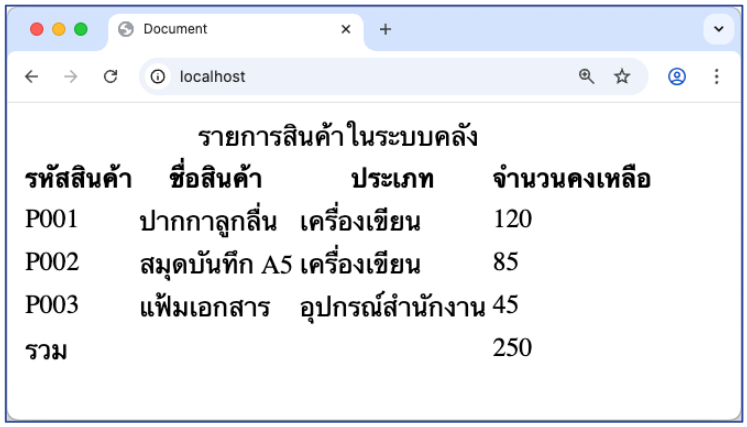
```
22              <td>85</td>
23          </tr>
24          <tr>
25              <td>P003</td>
26              <td>แฟ้มเอกสาร</td>
27              <td>อุปกรณ์สำนักงาน</td>
28              <td>45</td>
29          </tr>
30      </tbody>
31      <tfoot>
32          <tr>
33              <td>รวม</td>
34              <td> </td>
35              <td> </td>
36              <td>250</td>
37          </tr>
38      </tfoot>
39  </table>
```

จากตัวอย่างที่ 14 ใช้แท็ก <table> เพื่อสร้างตารางโดยมีจุดเริ่มต้นที่บรรทัดที่ 1 และจุดสิ้นสุดของตารางที่บรรทัดที่ 39 มีการกำหนดชื่อตารางเป็น *รายการสินค้าในระบบคลัง* ซึ่งระบุผ่านแท็ก <caption> ถือเป็นอิลิเมนต์ลูกอันดับแรกของอิลิเมนต์ table โดยมีการจัดกลุ่มตารางออกเป็น 3 กลุ่ม ดังนี้

- กลุ่มส่วนหัวของตารางด้วยแท็ก มีการกำหนดด้วยการใช้แท็ก<thead> ดังแสดงในบรรทัดที่ 3-10 โดยกลุ่มนี้มีการสร้างแถวส่วนหัวของตารางจำนวน 1 แถวด้วยแท็ก <tr> ดังแสดงในบรรทัดที่ 4-9 และภายในแถวมีการแบ่งออกเป็น 4 เซลล์หรือ 4 คอลัมน์ดังแสดงในบรรทัดที่ 5-8 ซึ่งทั้ง 4 เซลล์นี้จะใช้แท็ก <th> เพื่อกำหนดให้เป็นเซลล์หัวข้อ ประกอบด้วย รหัสสินค้า ชื่อสินค้า ประเภท และจำนวนคงเหลือ
- กลุ่มส่วนข้อมูลของตาราง มีการกำหนดด้วยการใช้แท็ก tbody ดังแสดงในบรรทัดที่ 11-30 ส่วนข้อมูลของตารางนี้มีแถวข้อมูลจำนวน 3 แถวโดยใช้แท็ก <tr> ดังแสดงในบรรทัดที่ 12-17, 18-23 และ 24-29 ตามลำดับ ในแต่ละแถวมีการสร้างเซลล์ข้อมูลด้วยแท็ก <td> แถวละ 4 เซลล์ซึ่งมีจำนวนเท่ากับเซลล์หัวข้อตาราง ในแต่ละแถวข้อมูลนี้เป็นการแสดงข้อมูลของสินค้าแต่ละรายการ
- กลุ่มส่วนท้ายของตาราง มีการกำหนดด้วยการใช้แท็ก <tfoot> เพื่อสรุปข้อมูลจำนวนคงเหลือ ดังแสดงในบรรทัดที่ 31-38 ส่วนท้ายของตารางมีแถวข้อมูลจำนวน 1 แถวโดยใช้แท็ก <tr> ดังแสดงในบรรทัดที่

32-37 ในแถวประกอบด้วยเซลล์ข้อมูลจำนวน 4 เซลล์ที่สร้างด้วยแท็ก <td> ดังแสดงในบรรทัดที่ 33-36 ใช้สำหรับแสดงผลรวมของจำนวนสินค้าในคอลัมน์สุดท้าย ทั้งนี้ในเซลล์ที่ 2 และ 3 ไม่มีเนื้อหาอยู่ในเซลล์ (บรรทัดที่ 34 และ 35)

ผลลัพธ์ของตัวอย่างที่ 14



ผลลัพธ์การแสดงผลในเว็บเบราว์เซอร์ของตัวอย่างที่ 14 ข้อมูลรายการสินค้าถูกจัดระเบียบในรูปแบบของตารางตามลำดับโครงสร้างตารางในภาษา HTML อย่างไรก็ตาม เส้นของตารางยังไม่ปรากฏในการแสดงผล เนื่องจากแท็กต่าง ๆ ในภาษา HTML จะเป็นเพียงการกำหนดโครงสร้างข้อมูล ส่วนลักษณะการแสดงผล เช่น เส้นขอบ สี ความกว้าง และการจัดตำแหน่งนั้นจะต้องใช้การกำหนดรูปแบบด้วย CSS เพิ่มเติม ทั้งนี้ ค่าเริ่มต้นการแสดงผลของเว็บเบราว์เซอร์จะกำหนดความกว้างของคอลัมน์โดยอิงจากความกว้างของเนื้อหาภายในเซลล์ กล่าวคือ คอลัมน์ใดมีเซลล์ที่มีข้อความยาวที่สุด ความกว้างของคอลัมน์นั้นจะถูกปรับตามเซลล์ดังกล่าวโดยอัตโนมัติ (Content-based Width)

6. แท็กสำหรับการสร้างฟอร์มรับข้อมูล

แท็กสำหรับการสร้างฟอร์มรับข้อมูล เป็นกลุ่มแท็กที่มีความสำคัญในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากทำหน้าที่เป็นช่องทางในการรับข้อมูลจากผู้ใช้งาน (User Input) แล้วส่งไปยังเครื่องผู้ให้บริการเว็บเพื่อทำการประมวลผล เช่น การลงทะเบียน การเข้าสู่ระบบ การกรอกแบบสอบถาม หรือการส่งคำสั่งซื้อสินค้า เป็นต้น โดยโครงสร้างของฟอร์มจะประกอบด้วยแท็กหลากหลายที่ทำหน้าที่ต่างกัน เช่น แท็กกำหนดฟอร์ม ป้ายกำกับ กล่องข้อความ กล่องรหัสผ่าน ปุ่มเลือก และปุ่มส่งข้อมูล เป็นต้น การเลือกใช้องค์ประกอบของฟอร์มให้เหมาะสมและสื่อสารได้อย่างชัดเจนจะช่วยเพิ่มประสิทธิภาพในการรับข้อมูล และยกระดับประสบการณ์ของผู้ใช้งานในระบบเว็บแอปพลิเคชัน

6.1 แท็กกำหนดขอบเขตของฟอร์ม

แท็กกำหนดขอบเขตของฟอร์มจะใช้แท็ก <form> ซึ่งเป็นแท็กคู่ที่มีบทบาทสำคัญที่สุดในการสร้างฟอร์ม โดยทำหน้าที่เป็นกรอบหรือขอบเขตที่ครอบคลุมอิลิเมนต์อื่น ๆ ที่เกี่ยวข้องกับการรับข้อมูลจากผู้ใช้ เช่น กล่องข้อความ ปุ่มเลือก และปุ่มส่งข้อมูล เป็นต้น ฟอร์มที่ไม่มีการครอบด้วยแท็ก form อย่างถูกต้อง จะไม่สามารถส่งข้อมูลไปยังปลายทางที่ต้องการได้ แท็ก <form> มีค่าเริ่มต้นการแสดงผลเป็นแบบ Block-Level โดยรูปแบบของแท็ก <form> มีดังนี้

รูปแบบ

```
<form action="URL หรือชื่อไฟล์สำหรับรับข้อมูล" method="GET หรือ POST">
    อิลิเมนต์สำหรับรับข้อมูลต่าง ๆ
</form>
```

จากรูปแบบของแท็ก <form> มีแอตทริบิวต์สำคัญที่จะต้องกำหนดจำนวน 2 รายการ ประกอบด้วย action และ method

6.1.1 แอตทริบิวต์ action

แอตทริบิวต์ action ใช้สำหรับระบุปลายทางที่ข้อมูลจากฟอร์มจะถูกส่งไปเมื่อผู้ใช้กดปุ่มส่งข้อมูล โดยค่าของ action อาจอยู่ในรูปแบบของ URL เช่น https://webapp.com/process หรือเป็นชื่อไฟล์ภายในเครื่องผู้ให้บริการเว็บเดียวกัน เช่น process.php หรือ controllers/submit.php เป็นต้น ข้อมูลที่ส่งจากแบบฟอร์มจะถูกนำไปประมวลผลโดยปลายทางดังกล่าว ซึ่งมักเป็นภาษาประมวลผลฝั่งผู้ให้บริการ เช่น PHP หรือ Node.js เป็นต้น

6.1.2 แอตทริบิวต์ method

แอตทริบิวต์ method ใช้สำหรับกำหนดวิธีการส่งข้อมูลจากฟอร์มไปยังปลายทางที่ระบุใน action ตามมาตรฐานของภาษ HTML ค่าที่รองรับมีหลายค่า เช่น GET POST PUT และDELETE แต่ในแบบฟอร์มทั่วไปจะใช้เพียง GET และ POST เท่านั้น ซึ่งแต่ละค่าจะส่งผลต่อรูปแบบการส่งข้อมูล ความปลอดภัย ขนาดข้อมูลที่สามารถส่งได้ และพฤติกรรมของเว็บเบราว์เซอร์ ดังนี้

1) การส่งข้อมูลด้วยวิธี GET

การส่งข้อมูลด้วยวิธี GET จะเป็นลักษณะที่ข้อมูลจากฟอร์มจะถูกแนบไว้ท้าย URL ในรูปแบบของพารามิเตอร์ เช่น app.com/save.php?name=Nok&age=25 จากตัวอย่างนี้เป็นการส่งค่าจำนวน 2 ค่าไปยังไฟล์ save.php ที่อยู่ใน app.com โดยมีข้อมูลที่ส่งคือชื่อ name มีค่าเป็น Nok และอายุ age มีค่าเป็น 25 การส่งข้อมูลด้วยวิธีนี้จะทำให้ข้อมูลที่กรอกสามารถมองเห็นได้จากแถบที่อยู่ของเว็บเบราว์เซอร์ (Address Bar) และข้อมูลจะถูกบันทึกไว้ในประวัติการเข้าถึง (Browser History) อีกทั้งยังสามารถบุ๊กมาร์กผลลัพธ์ได้ จึงเหมาะสำหรับการใช้ในแบบฟอร์มค้นหา หรือการส่งข้อมูลที่ไม่เป็นความลับ อย่างไรก็ตาม วิธีนี้มีข้อจำกัดด้านความ

ปลอดภัยเนื่องจากข้อมูลจะเปิดเผยใน URL และข้อจำกัดด้านความยาวของข้อมูลที่สามารถส่งได้ ซึ่งโดยทั่วไปไม่ควรเกิน 2,048 ตัวอักษร จึงไม่เหมาะสำหรับใช้ส่งข้อมูลที่เป็นความลับ เช่น รหัสผ่าน หรือข้อมูลส่วนบุคคล เป็นต้น

2) การส่งข้อมูลด้วยวิธี POST

การส่งข้อมูลด้วยวิธี POST จะเป็นลักษณะที่ข้อมูลที่กรอกในฟอร์มจะถูกส่งไปยังปลายทางที่ระบุใน action ผ่านส่วนเนื้อหาของคำขอ HTTP (HTTP Request Body) โดยจะไม่แสดงใน URL ของเว็บเบราว์เซอร์ ดังนั้น ผู้ใช้จะไม่สามารถเห็นข้อมูลที่ส่งได้จากแถบที่อยู่ ส่งผลให้มีความปลอดภัยมากกว่าวิธี GET โดยเฉพาะอย่างยิ่งสำหรับการส่งข้อมูลที่เป็นความลับ เช่น รหัสผ่าน หรือข้อมูลส่วนบุคคล อีกทั้งยังไม่มีข้อจำกัดด้านขนาดของข้อมูลที่สามารถส่งได้ ทำให้เหมาะกับการส่งข้อมูลจำนวนมากหรือไฟล์ที่มีขนาดใหญ่ อย่างไรก็ตาม วิธีนี้ไม่สามารถทำการบุ๊กมาร์กผลลัพธ์ของแบบฟอร์มได้ และข้อมูลที่ส่งจะไม่ถูกบันทึกในประวัติของเบราว์เซอร์ เนื่องจากข้อมูลไม่ปรากฏใน URL โดยรวมแล้ว POST จึงเหมาะสำหรับการส่งข้อมูลที่ต้องการความเป็นส่วนตัวและความถูกต้องในการประมวลผล แม้ว่าการส่งข้อมูลด้วยวิธี POST จะไม่แสดงข้อมูลใน URL แต่ก็ยังควรกำหนดให้ใช้งานร่วมกับ HTTPS เสมอเพื่อป้องกันการดักข้อมูลระหว่างทาง

6.2 แท็กสำหรับรับข้อความสั้น

แท็กสำหรับรับข้อความสั้น จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยวที่ใช้สำหรับสร้างช่องกรอกข้อความเพียง 1 บรรทัด เช่น ชื่อผู้ใช้ ชื่อสินค้า เป็นต้น มีแอตทริบิวต์ที่สำคัญที่ต้องระบุเพื่อกำหนดช่องกรอกข้อความลักษณะนี้คือ type โดยกำหนดให้มีค่าเป็น text และแอตทริบิวต์ name ที่ใช้ระบุชื่อเขตข้อมูลที่จะส่งไปยังปลายทาง มีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="text" name="ชื่อเขตข้อมูล">
```

6.3 แท็กสำหรับรับรหัสผ่าน

แท็กสำหรับรับรหัสผ่าน จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยว เหมือนกับการรับข้อความสั้น แต่จะแสดงผลในลักษณะซ่อนข้อความที่พิมพ์ไว้ เช่น แสดงเป็นดอกจันหรือจุด เพื่อปกป้องข้อมูลที่เป็นความลับ เช่น รหัสผ่าน มีแอตทริบิวต์สำคัญคือ type โดยกำหนดค่าเป็น password และ name สำหรับระบุชื่อเขตข้อมูลที่จะส่งไปยังปลายทาง มีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="password" name="ชื่อเขตข้อมูล">
```

6.4 แท็กสำหรับรับตัวเลข

แท็กสำหรับรับตัวเลข จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยวที่ใช้สำหรับสร้างช่องกรอกข้อมูลที่ต้องเป็นตัวเลขเท่านั้น เช่น จำนวนสินค้า ราคาสินค้า หรืออายุ เป็นต้น โดยสามารถกำหนดขอบเขตของค่าตัวเลขต่ำสุดและค่าตัวเลขสูงสุดที่ยอมรับได้ รวมทั้งการกำหนดอัตราการเพิ่มและลดจำนวนด้วยแอตทริบิวต์ min max และ step ตามลำดับ มีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="number" name="ชื่อเขตข้อมูล" min="ค่าต่ำสุด" max="ค่าสูงสุด" step="ค่าเพิ่ม/ลด">
```

6.5 แท็กสำหรับรับอีเมล

แท็กสำหรับรับอีเมล จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยวที่ใช้สำหรับรับข้อมูลในรูปแบบของที่อยู่อีเมล โดยเว็บเบราว์เซอร์จะทำการตรวจสอบรูปแบบของข้อมูลว่าเป็นอีเมลที่ถูกต้องก่อนการส่งข้อมูล มีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="email" name="ชื่อเขตข้อมูล">
```

6.6 แท็กสำหรับรับวันที่

แท็กสำหรับรับวันที่ จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยวที่ใช้สำหรับสร้างช่องกรอกวันที่ โดยเบราว์เซอร์จะเปลี่ยนช่องให้เป็นปฏิทินเพื่อให้ผู้ใช้สามารถเลือกวันได้ง่าย มีรูปแบบในการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="date" name="ชื่อเขตข้อมูล">
```

6.7 แท็กสำหรับการเลือกรายการเดียว

แท็กสำหรับการเลือกรายการเดียว (Radio Button) จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยว โดยกำหนด type="radio" และตั้งค่า name ให้เป็นชื่อเขตข้อมูลเหมือนกันสำหรับชุดตัวเลือกเดียวกัน เพื่อให้ผู้ใช้สามารถเลือกได้เพียงหนึ่งตัวเลือกเท่านั้น โดยค่าที่เลือกซึ่งกำหนดด้วยแอตทริบิวต์ value จะถูกส่งไปยังปลายทาง รูปแบบและตัวอย่างในการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="radio" name="ชื่อเขตข้อมูล" value="ค่าที่ 1"> ข้อความแสดงค่าที่ 1
<input type="radio" name="ชื่อเขตข้อมูล" value="ค่าที่ 2"> ข้อความแสดงค่าที่ 2
...
```

<input type="radio" name="ชื่อเขตข้อมูล" value="ค่าที่ n"> ข้อความแสดงค่าที่ n

ตัวอย่างที่ 15

```
1  <input type="radio" name="gender" value="male"> ชาย
2  <input type="radio" name="gender" value="female"> หญิง
```

จากตัวอย่างที่ 15 เป็นการสร้างช่องกรอกข้อมูลแบบเลือกรายการเดียว โดยเป็นการสร้างตัวเลือกให้ผู้ใช้ได้เลือกจำนวน 2 ตัวเลือกคือ ชาย และ หญิง จะสังเกตได้ว่าทั้งสองอิลิเมนต์มีการกำหนดชื่อเขตข้อมูลเหมือนกันเพื่อให้ผู้ใช้เลือกรายการอย่างใดอย่างหนึ่งได้เพียงรายการเดียว นอกจากนี้ข้อความตัวเลือกที่ปรากฏให้ผู้ใช้เห็นในเว็บเบราว์เซอร์จะเป็นคำว่าชายและหญิง แต่ค่าที่ส่งไปยังปลายทางจะเป็น male และ female ตามที่กำหนดในแอตทริบิวต์ value

6.8 แท็กสำหรับการเลือกหลายรายการ

แท็กสำหรับการเลือกหลายรายการ (Checkbox) จะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยว โดยกำหนด type="checkbox" ใช้ในกรณีที่ผู้ใช้สามารถเลือกตัวเลือกได้มากกว่าหนึ่ง เช่น ความสนใจหรือทักษะความสามารถทางภาษา เป็นต้น การกำหนดแอตทริวบิวต์ name ของแต่ละตัวเลือกจะตั้งชื่อแตกต่างกันในแต่ละตัวเลือก นอกจากนี้ เฉพาะตัวเลือกที่ถูกเลือกเท่านั้นที่จะส่งไปยังปลายทาง รูปแบบและตัวอย่างในการใช้งานเบื้องต้นดังนี้

รูปแบบ

<input type="checkbox" name="ชื่อเขตข้อมูล 1" value="ค่าที่ 1"> ข้อความแสดงค่าที่ 1
<input type="checkbox" name="ชื่อเขตข้อมูล 2" value="ค่าที่ 2"> ข้อความแสดงค่าที่ 2
...
<input type="checkbox" name="ชื่อเขตข้อมูล 3" value="ค่าที่ n"> ข้อความแสดงค่าที่ n

ตัวอย่างที่ 16

```
1  <input type="checkbox" name="skill_1 " value="html"> HTML
2  <input type="checkbox" name="skill_2" value="css"> CSS
3  <input type="checkbox" name="skill_3" value="javascript"> JavaScript
```

จากตัวอย่างที่ 16 เป็นการสร้างช่องกรอกข้อมูลแบบเลือกหลายรายการ โดยเป็นการสร้างตัวเลือกให้ผู้ใช้ได้เลือกจำนวน 3 ตัวเลือกคือ HTML CSS และ JavaScript ทั้งสามอิลิเมนต์จะมีการกำหนดแอตทริบิวต์ name ที่แตกต่างกัน

6.9 แท็กสำหรับการกรอกข้อความหลายบรรทัด

แท็กสำหรับการกรอกข้อความหลายบรรทัด (Textarea) จะใช้แท็ก <textarea> ซึ่งเป็นแท็กคู่ที่มีค่าเริ่มต้นการแสดงผลเป็นแบบ Block-Level ใช้สำหรับรับข้อความที่มีความยาวหรือมีหลายบรรทัด เช่น ข้อเสนอแนะหรือคำอธิบาย โดยสามารถกำหนดขนาดได้ด้วยแอตทริบิวต์ rows และ cols มีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<textarea name="ชื่อเขตข้อมูล" rows="ความสูง" cols="ความกว้าง"></textarea>
```

6.10 แท็กสำหรับการเลือกจากรายการที่กำหนดไว้

แท็กสำหรับการเลือกจากรายการที่กำหนดไว้ (Dropdown List) จะใช้แท็ก <select> ร่วมกับแท็ก <option> ซึ่งทั้งสองเป็นแท็กคู่ โดยแท็ก <select> มีค่าเริ่มต้นการแสดงผลเป็นแบบ Inline โดยทั่วไปแล้วใช้สำหรับแสดงรายการให้ผู้ใช้เลือก 1 รายการ มีรูปแบบและตัวอย่างการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<select name="ชื่อเขตข้อมูล">
    <option value="ค่าของตัวเลือกที่ 1">ข้อความแสดงค่าตัวเลือกที่ 1</option>
    <option value="ค่าของตัวเลือกที่ 2">ข้อความแสดงค่าตัวเลือกที่ 2</option>
    ...
    <option value="ค่าของตัวเลือกที่ n">ข้อความแสดงค่าตัวเลือกที่ n</option>
</select>
```

ตัวอย่างที่ 17

```
1  <select name="province">
2      <option value="Bangkok">กรุงเทพฯ</option>
3      <option value="Chiangmai">เชียงใหม่</option>
4      <option value="Tak">ตาก</option>
5  </select>
```

6.11 แท็กสำหรับปุ่มส่งข้อมูล (Submit Button)

แท็กสำหรับปุ่มส่งข้อมูลจะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยวที่มีค่าเริ่มต้นการแสดงผลเป็นแบบ Inline โดยกำหนด type="submit" ใช้สำหรับให้ผู้ใช้คลิกเพื่อส่งข้อมูลทั้งหมดที่อยู่ในฟอร์มไปยังปลายทางที่ระบุไว้ในแอตทริบิวต์ action มีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
<input type="submit" value="ข้อความแสดงบนปุ่ม">
```

6.12 แท็กสำหรับปุ่มรีเซ็ตข้อมูลในฟอร์ม

แท็กสำหรับปุ่มรีเซ็ตข้อมูลในฟอร์มจะใช้แท็ก <input> ซึ่งเป็นแท็กเดี่ยวที่มีค่าเริ่มต้นการแสดงผลเป็นแบบ Inline โดยกำหนด type="reset" เพื่อสร้างปุ่มที่เมื่อผู้ใช้คลิก จะล้างค่าทั้งหมดในฟอร์มและคืนค่าให้กับอิลิเมนต์ต่าง ๆ เป็นค่าตั้งต้นตามที่กำหนดไว้ หากไม่ได้ระบุค่าเริ่มต้นไว้ ข้อมูลในช่องกรอกจะกลายเป็นค่าว่าง รูปแบบของแท็กมีดังนี้

รูปแบบ

```
<input type="submit" value="ข้อความแสดงบนปุ่ม">
```

7. แท็กสำหรับการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา

แท็กที่ใช้ในการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา (Non-semantic Grouping Tags) เป็นแท็กที่ใช้รวมเนื้อหาหรืออิลิเมนต์หลายรายการให้อยู่ในกลุ่มเดียวกัน เพื่อความสะดวกในการจัดการด้วย CSS และการควบคุมพฤติกรรมผ่าน JavaScript เช่น การจัดกลุ่มข้อความ รูปภาพ หรืออิลิเมนต์อื่น ๆ เพื่อให้สามารถกำหนดรูปแบบหรือจัดการพฤติกรรมร่วมกันได้ง่ายขึ้น แท็กเหล่านี้ไม่มีความหมายในเชิงเนื้อหา กล่าวคือ ไม่ได้ระบุบทบาทหรือบริบทของเนื้อหาภายในกลุ่ม แต่ทำหน้าที่เสมือนภาชนะสำหรับจัดกลุ่มเท่านั้น โดยจะไม่มีผลต่อการแสดงผลจนกว่าจะมีการกำหนดสไตล์เพิ่มเติมผ่าน CSS แท็กที่พบบ่อย ได้แก่ แท็กสำหรับการจัดกลุ่มแบบ Block และแท็กสำหรับการจัดกลุ่มแบบ Inline

7.1 แท็กสำหรับการจัดกลุ่มเนื้อหาแบบ Block

แท็กสำหรับการจัดกลุ่มเนื้อหาแบบ Block เป็นแท็กสำหรับจัดกลุ่มเนื้อหาหรืออิลิเมนต์ (Division Element) ที่มีค่าเริ่มต้นของการแสดงผลของกลุ่มเป็นแบบ Block-Level จุดประสงค์หลักของแท็กนี้เพื่อรวมกลุ่มอิลิเมนต์หลายรายการที่เกี่ยวข้องกันให้อยู่ในโครงสร้างเดียวกันโดยไม่ระบุบทบาทหรือบริบทของกลุ่ม เช่น การรวมข้อความ ปุ่ม หรือรูปภาพ เพื่อความสะดวกในการจัดรูปแบบด้วย CSS หรือควบคุมพฤติกรรมด้วย JavaScript แท็กนี้ไม่ส่งผลต่อการแสดงผลจนกว่าจะมีการกำหนดสไตล์เพิ่มเติม มักถูกใช้เมื่อไม่มีแท็กเชิงความหมายอื่นที่เหมาะสม เช่น <section>, <article> หรือ <nav> เป็นต้น แท็กสำหรับการจัดกลุ่มเนื้อหาแบบ Block จะใช้แท็ก ซึ่งเป็นแท็กคู่ที่ต้องมีการระบุจุดเริ่มต้นและจุดสิ้นสุดของกลุ่ม โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

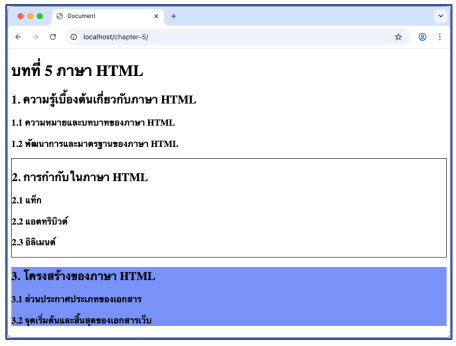
```
<div>เนื้อหาที่ต้องการจัดกลุ่ม เช่น ข้อความ รูปภาพ ลิงก์</div>
```

ตัวอย่างที่ 18

```
1  <h1>บทที่ 5 ภาษา HTML</h1>
2  หัวข้อเนื้อหาของบทที่ 5 มีดังนี้
3  <div>
4      <h2>1. ความรู้เบื้องต้นเกี่ยวกับภาษา HTML</h2>
5      <h3>1.1 ความหมายและบทบาทของภาษา HTML</h3>
6      <h3>1.2 พัฒนาการและมาตรฐานของภาษา HTML</h3>
7  </div>
8  <div style="border: 1px solid black;">
9      <h2>2. การกำกับในภาษา HTML</h2>
10     <h3>2.1 แท็ก</h3>
11     <h3>2.2 แอตทริบิวต์</h3>
12     <h3>2.3 อิลิเมนต์</h3>
13 </div>
14 <div style="background-color: #7892f8;">
15     <h2>3. โครงสร้างของภาษา HTML</h2>
16     <h3>3.1 ส่วนประกาศประเภทของเอกสาร</h3>
17     <h3>3.2 จุดเริ่มต้นและสิ้นสุดของเอกสารเว็บ</h3>
18 </div>
```

ในตัวอย่างที่ 18 มีการใช้แท็ก เพื่อจัดกลุ่มอิลิเมนต์แต่ละชุดให้เป็นกลุ่มเดียวกันจำนวน 3 กลุ่ม ได้แก่ กลุ่มอิลิเมนต์ของความรู้เบื้องต้นเกี่ยวกับภาษา HTML (บรรทัดที่ 3–7) กลุ่มอิลิเมนต์ของการกำกับในภาษา HTML (บรรทัดที่ 8–13) และกลุ่มอิลิเมนต์ของโครงสร้างของภาษา HTML (บรรทัดที่ 14–18)

ผลลัพธ์ของตัวอย่างที่ 18



ผลลัพธ์การแสดงผลของแต่ละกลุ่มเนื้อหาที่แสดงในเว็บเบราว์เซอร์ตามตัวอย่างที่ 18 จะเป็นลักษณะ Block-Level โดยสังเกตจากเนื้อหาในบรรทัดที่ 2 จะแสดงคนละบรรทัดกับกลุ่มที่ 1 รวมทั้งแต่ละกลุ่มจะมีการขึ้นบรรทัดใหม่ก่อนก่อนแสดงผล

กลุ่มเนื้อหาที่ 2 และ 3 มีการกำหนดรูปแบบการแสดงผลเพิ่มเติมด้วย CSS โดยใช้แอตทริบิวต์ style ภายในแท็ก ซึ่งในกลุ่มที่ 2 ได้กำหนดลักษณะเส้นขอบเป็นเส้นตรงสีดำ ความหนา 1 พิกเซล (border: 1px solid black) ส่วนกลุ่มที่ 3 กำหนดสีพื้นหลังเป็นโทนสีน้ำเงินอ่อนโดยใช้รหัสสี #7892f8 (background-color: #7892f8) ทั้งนี้กลุ่มที่ 1 ไม่ได้มีการจัดรูปแบบการแสดงผล ผลลัพธ์ที่ได้จึงไม่มีสิ่งใดที่เปลี่ยนแปลงไปโดยจะแสดงผลตามค่าเริ่มต้นของแต่ละเว็บเบราว์เซอร์ดังตัวอย่างด้านล่างนี้ อนึ่ง รายละเอียดเกี่ยวกับการกำหนดลักษณะด้วย CSS จะได้นำเสนออย่างละเอียดในบทถัดไป

7.2 แท็กสำหรับการจัดกลุ่มแบบ Inline

แท็กสำหรับการจัดกลุ่มแบบ Inline เป็นแท็กที่ใช้สำหรับจัดกลุ่มเนื้อหาภายในบรรทัดเดียวกันโดยไม่แสดงผลในลักษณะขึ้นบรรทัดใหม่ ซึ่งเหมาะสำหรับการควบคุมลักษณะการแสดงผลหรือการประมวลผลด้วย CSS และ JavaScript เฉพาะจุด เช่น การปรับสีข้อความ การทำให้ตัวหนา หรือการใส่เอฟเฟกต์กับข้อความบางช่วง แท็กนี้เหมาะสำหรับการควบคุมเฉพาะเนื้อหาภายในบรรทัดหรือย่อหน้าเดียวกัน แท็กสำหรับการจัดกลุ่มแบบ Inline จะใช้แท็ก ซึ่งจัดเป็นแท็กคู่ที่ไม่มีความหมายในเชิงเนื้อหาเช่นเดียวกับ แต่ต่างกันตรงที่ เป็นแท็กแบบ inline-level ขณะที่ เป็น block-level รูปแบบและตัวอย่างการใช้แท็ก มีดังนี้

รูปแบบ

```
<span>เนื้อหาที่ต้องการจัดกลุ่ม</span>
```

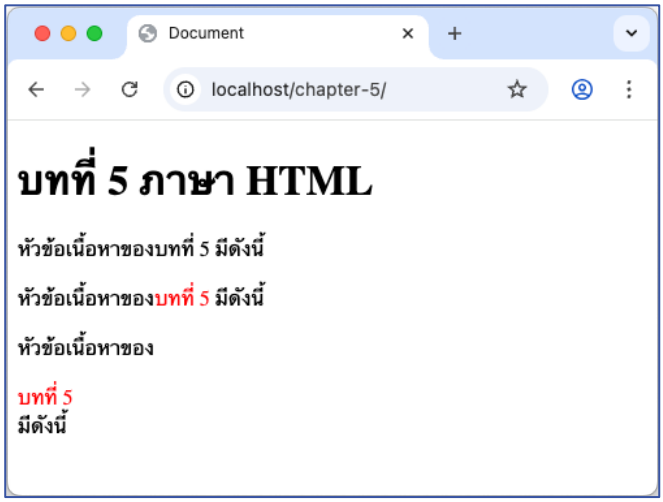
ตัวอย่างที่ 19

```
1  <h1>บทที่ 5 ภาษา HTML</h1>
2  <p>หัวข้อเนื้อหาของ<span>บทที่ 5 </span> มีดังนี้</p>
3  <p>หัวข้อเนื้อหาของ<span style="color:red; border: 1px solid black;">บทที่ 5 </span> มี
   ดังนี้
   </p>
4  <p>หัวข้อเนื้อหาของ<div style="color:red; border: 1px solid black;">บทที่ 5 </div> มี
   ดังนี้</p>
```

ในตัวอย่างที่ 19 มีการเปรียบเทียบการใช้แท็ก และ เพื่อจัดกลุ่มข้อความ *บทที่ 5* ภายในย่อหน้าเดียวกัน โดยในบรรทัดที่ 2 เป็นการใช้แท็ก แบบไม่กำหนดรูปแบบเพิ่มเติมซึ่งไม่มีผลต่อการแสดงผล เนื่องจาก เป็นอิลิเมนต์ที่มีค่าเริ่มต้นของการแสดงผลเป็นแบบ Inline-Level กล่าวคือ มีการ

แสดงผลต่อเนื่องในบรรทัดเดียวกับข้อความรอบข้างและจะมีขอบเขตของกลุ่มถึงแท็กปิด เท่านั้น ส่วนในบรรทัดที่ 3 มีการใช้ เช่นเดียวกันแต่เพิ่มการกำหนดรูปแบบด้วย CSS ผ่านแอตทริบิวต์ style ทำให้ข้อความ *บทที่ 5* แสดงเป็นสีแดง (color:red;) พร้อมทั้งมีเส้นขอบล้อมรอบ (border: 1px solid black;) โดยยังคงอยู่ในแนวเดียวกับข้อความทั้งหมดไม่เกิดการเว้นบรรทัดใหม่ แสดงดังผลลัพธ์ของตัวอย่างที่ 19

ผลลัพธ์ของตัวอย่างที่ 19



สำหรับบรรทัดที่ 4 ของตัวอย่างที่ 19 มีการใช้แท็ก ซึ่งเป็นอิลิเมนต์แบบ Block-level ในการจัดกลุ่มข้อความเดียวกันพร้อมกับระบุลักษณะการแสดงผลด้วย CSS เช่นกัน โดยผลลัพธ์ที่ได้จะแตกต่างอย่างชัดเจน กล่าวคือมีการขึ้นบรรทัดใหม่ทั้งก่อนและหลังข้อความ *บทที่ 5* ตามพฤติกรรมของอิลิเมนต์แบบ Block-level ซึ่งแสดงให้เห็นถึงข้อแตกต่างของแท็กทั้งสองประเภทนี้อย่างชัดเจน อนึ่งการใช้ แทรกภายใน <p> ในกรณีตัวอย่างนี้ใช้เพื่อนำเสนอความแตกต่างของ Block-Level และ Inline-Level เท่านั้น ซึ่งในทางปฏิบัติควรหลีกเลี่ยงการจัดโครงสร้างเนื้อหาในลักษณะนี้

8. แท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมาย

แท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมาย (Semantic Grouping Tags) เป็นแท็กที่ใช้ในการจัดระเบียบโครงสร้างเนื้อหาในหน้าเว็บโดยระบุความหมายเฉพาะของแต่ละกลุ่มอย่างชัดเจน เช่น ส่วนหัวของเว็บ เมนูนำทาง เนื้อหาหลัก ส่วนท้าย หรือเนื้อหาด้านข้างของเว็บ เป็นต้น การจัดกลุ่มเช่นนี้ช่วยให้ผู้อ่านรหัสคำสั่งเข้าใจได้ง่ายขึ้นว่ากลุ่มเนื้อหานั้นมีบทบาทหน้าที่อะไร อีกทั้งยังสนับสนุนการเข้าถึงเนื้อหาผ่านเทคโนโลยีช่วยเหลือ เช่น โปรแกรมอ่านหน้าจอ และช่วยให้เครื่องมือค้นหาวิเคราะห์โครงสร้างของเอกสารได้ถูกต้อง

แท็กกลุ่มนี้มีลักษณะการใช้งานคล้ายกับแท็ก ซึ่งนิยมใช้ในการจัดกลุ่มเนื้อหาโดยไม่มีความหมายเชิงบริบท อย่างไรก็ตาม แท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมายจะให้ข้อมูลเชิงโครงสร้างแก่เอกสาร HTML โดยตรง และช่วยเพิ่มความเข้าใจในบริบทของเนื้อหานั้น ๆ ได้อย่างชัดเจน ทั้งนี้ แท็กในกลุ่มนี้ล้วนเป็นแท็กคู่และมีค่าเริ่มต้นของการแสดงผลเป็นแบบ Block-Level เช่นเดียวกับแท็ก กล่าวคือ จะเริ่มแสดงผลที่บรรทัดใหม่และครอบคลุมความกว้างของพื้นที่แสดงผลของอิลิเมนต์แม่ ซึ่งช่วยให้สามารถจัดเรียงเป็นองค์ประกอบของหน้าเว็บในแนวตั้งได้อย่างเป็นระบบ และเอื้อต่อการปรับแต่งเพิ่มเติมด้วย CSS ในภายหลัง

8.1 การจัดกลุ่มเนื้อหาโดยการแบ่งส่วน

การนำแท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมายมาใช้มีวัตถุประสงค์หลักเพื่อกำหนดพื้นที่ของกลุ่มเนื้อหาตามบทบาทในหน้าเว็บอย่างชัดเจน ซึ่งนำไปสู่ความเป็นระเบียบในการพัฒนาเว็บแอปพลิเคชัน และเอื้อต่อการจัดตำแหน่งผ่าน CSS ดังที่แสดงในภาพที่ 6.2 ซึ่งเป็นตัวอย่างของการจัดกลุ่มเนื้อหาโดยแบ่งออกเป็น 7 ส่วนตามบริบท ได้แก่ ส่วนหัวของเว็บ ส่วนเมนูนำทาง ส่วนเนื้อหาหลัก ส่วนบทความ ส่วนย่อย ส่วนด้านข้าง และส่วนท้ายของเว็บ โดยบางกลุ่มจะมีการเขียนเป็นอิลิเมนต์ที่ซ้อนกัน เช่น ส่วนบทความและส่วนด้านข้างจะเป็นอิลิเมนต์ลูกของส่วนเนื้อหาหลัก และส่วนย่อยหลายๆ ส่วนจะเป็นอิลิเมนต์ลูกของส่วนบทความ เป็นต้น



ภาพที่ 6.2 ตัวอย่างการจัดกลุ่มเนื้อหาเป็นส่วนต่าง ๆ

จากภาพที่ 6.2 เป็นตัวอย่างการจัดกลุ่มเนื้อหาด้วยแท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมายซึ่งอาจแตกต่างกันออกไปขึ้นอยู่กับการออกแบบของผู้พัฒนา แต่ยังคงต้องคำนึงวัตถุประสงค์ของแต่ละส่วนดังนี้

8.1.1 ส่วนหัวของเว็บ

ส่วนหัวของเว็บ (Page Header Group) ใช้แท็ก <header> ซึ่งเป็นแท็กแบบคู่เพื่อระบุส่วนบนสุดของหน้าเว็บ ซึ่งมักใช้แสดงโลโก้ ชื่อเว็บไซต์ คำโปรย โดยเป็นพื้นที่แรกที่ผู้ใช้เห็นเมื่อเข้าสู่เว็บไซต์ ทำหน้าที่สร้างความประทับใจแรกเริ่มและสะท้อนอัตลักษณ์ของเว็บไซต์ โดยทั่วไปมักมีองค์ประกอบอย่างเช่นแท็กหัวเรื่อง

รูปโลโก้ ชื่อหน่วยงานหรือชื่อผู้เขียน ทั้งนี้แท็ก <header> สามารถนำไปใช้เป็นอิลิเมนต์ลูกของอิลิเมนต์อื่น เช่น article หรือ section เพื่อระบุหัวเรื่องของเนื้อหาย่อย อย่างไรก็ตาม ควรหลีกเลี่ยงการวางแท็ก <header> ซ้อนอยู่ภายในแท็ก <footer> และ <address> แต่ถ้าหากอิลิเมนต์ header เป็นอิลิเมนต์ลักษณะดังภาพที่ 6.2 จะต้องไม่ปรากฏอิลิเมนต์ header ในระดับเดียวกันซ้ำอีก เพื่อรักษาความถูกต้องของโครงสร้างเอกสารตามมาตรฐาน รูปแบบของการใช้แท็ก <header> เพื่อกำหนดส่วนหัวของเว็บไซต์มีดังนี้

รูปแบบ

```
<header>เนื้อหาส่วนหัวของเว็บไซต์</header>
```

8.1.2 ส่วนเมนูนำทาง

ส่วนเมนูนำทาง (Navigation) จะใช้แท็ก <nav> ซึ่งเป็นแท็กแบบคู่ที่ใช้ระบุส่วนของเนื้อหาที่มีวัตถุประสงค์เพื่อจัดกลุ่มลิงก์ที่ใช้สำหรับนำทางไปยังหน้าหรือส่วนต่าง ๆ ของเว็บไซต์ โดยลิงก์เหล่านี้อาจเชื่อมโยงไปยังส่วนอื่นภายในเอกสารเดียวกันหรือเอกสารอื่น ๆ ก็ได้ แท็ก <nav> เป็นหนึ่งในแท็กเชิงความหมายที่ช่วยให้ผู้อ่านรหัสคำสั่งสามารถเข้าใจวัตถุประสงค์ของกลุ่มเนื้อหานั้นได้ทันที รวมทั้งสนับสนุนการเข้าถึงผ่านเทคโนโลยีช่วยเหลือ เช่น โปรแกรมอ่านหน้าจอ ซึ่งสามารถรับรู้ได้ว่าส่วนใดของเอกสารเป็นส่วนของเมนูนำทาง และสามารถละเว้นหรือจัดการแสดงผลให้ผู้ใช้ที่มีข้อจำกัดในการมองเห็นได้เหมาะสม ในหน้าเว็บหนึ่งหน้าสามารถมีส่วนเมนูนำทางได้หลายส่วน เช่น ส่วนเมนูนำทางหลักของเว็บไซต์ ส่วนเมนูนำทางไปยังเนื้อหาส่วนต่าง ๆ ในหน้าเว็บเป็นต้น รูปแบบของการใช้แท็ก <nav> เพื่อกำหนดส่วนเมนูนำทางมีดังนี้

รูปแบบ

```
<nav>รายการลิงก์เชื่อมโยง</nav>
```

8.1.3 ส่วนเนื้อหาหลัก

ส่วนเนื้อหาหลักของหน้าเว็บ (Main Content Group) จะใช้แท็ก <main> ซึ่งเป็นแท็กคู่เพื่อระบุพื้นที่แสดงเนื้อหาหลักหรือสาระสำคัญของหน้าที่เกี่ยวข้องโดยตรงกับฟังก์ชันหลักของเว็บแอปพลิเคชันในหน้าเว็บนั้น ๆ เช่น เนื้อหาของบทความหรือข้อมูลบริการในเว็บไซต์ ฟอร์มกรอกข้อมูล เป็นต้น ทั้งนี้ควรใช้แท็ก <main> เพียงหนึ่งครั้งต่อหนึ่งหน้าเว็บเท่านั้น และไม่ควรวางไว้ภายในแท็กอื่นที่ใช้ระบุส่วนต่าง ๆ ของหน้าเว็บ เช่น <header> <footer> <nav> <aside> หรือ <article> เพื่อให้โครงสร้างเอกสารมีความถูกต้องตามมาตรฐาน HTML อย่างไรก็ตาม ผู้พัฒนาสามารถแบ่งส่วนเนื้อหาหลักออกเป็นส่วนย่อย เช่น ส่วนบทความ ส่วนเนื้อหาย่อย และส่วนด้านข้าง ได้ตามลักษณะการจัดวางของหน้าเว็บ ดังที่แสดงในภาพที่ 6.2 รูปแบบของการใช้แท็ก <main> เพื่อกำหนดส่วนเนื้อหาหลักมีดังนี้

รูปแบบ

```
<main>เนื้อหาหลักของหน้าเว็บ ส่วนบทความ ส่วนเนื้อหาย่อยหรือส่วนด้านข้าง</main>
```

8.1.4 ส่วนบทความ

ส่วนบทความ (Article Group) จะใช้แท็ก <article> ซึ่งเป็นแท็กแบบคู่เพื่อระบุเนื้อหาที่สามารถแยกนำไปเผยแพร่ซ้ำได้อย่างอิสระจากหน้าเว็บหลัก ตัวอย่างของเนื้อหาประเภทนี้ได้แก่ บทความข่าว บทความในบล็อก ความคิดเห็นของผู้ใช้ เป็นต้น ซึ่งสามารถแสดงหรือจัดส่งให้ผู้อ่านภายนอกเว็บไซต์ได้โดยไม่จำเป็นต้องพึ่งพาส่วนอื่นของหน้าเว็บนั้น

ในหน้าเว็บหนึ่งสามารถมีอิลิเมนต์ article ได้หลายรายการ โดยแต่ละอิลิเมนต์ควรมีหัวเรื่องเพื่อให้สามารถแยกแยะเนื้อหาแต่ละบทความได้อย่างชัดเจน นอกจากนี้ ยังสามารถซ้อนอิลิเมนต์ article กันได้ เช่น ความคิดเห็นของผู้ใช้ที่อยู่ภายในบทความหลัก ซึ่งถือเป็นบทความย่อยที่สัมพันธ์กับเนื้อหาหลัก

การใช้แท็ก <article> จึงเหมาะสำหรับการจัดกลุ่มเนื้อหาที่มีความเป็นอิสระ และสามารถแยกนำไปจัดวางหรือแสดงผลในบริบทอื่นได้อย่างสมบูรณ์ ซึ่งเป็นประโยชน์อย่างมากในการพัฒนาเว็บแอปพลิเคชันที่มีลักษณะเนื้อหาเป็นรายการ เช่น บทความ ข่าว หรือความเห็นของผู้ใช้ รูปแบบของการใช้แท็ก <article> เพื่อกำหนดส่วนบทความมีดังนี้

รูปแบบ

```
<article>เนื้อหาที่สามารถแสดงผลหรือแยกเผยแพร่ได้</article>
```

8.1.5 ส่วนเนื้อหาย่อย

ส่วนเนื้อหาย่อย (Section Group) ใช้แท็ก <section> ซึ่งเป็นแท็กแบบคู่ที่ใช้สำหรับจัดกลุ่มเนื้อหาย่อยจากส่วนหลัก เช่น ส่วนหัวข้อย่อยภายในบทความเรื่องหนึ่งๆ หรือกลุ่มผลการค้นหา โดยเนื้อหาภายในอิลิเมนต์ section ควรมีการกำหนดหัวเรื่องเพื่อบอกให้ชัดเจนว่าเนื้อหานั้นเกี่ยวข้องกับเรื่องใด ทั้งนี้ แท็ก <section> ถือเป็นแท็กเชิงความหมายที่ใช้ในกรณีที่ไม่มีแทกเฉพาะอื่นที่เหมาะสมกว่า เช่น หากเป็นบทความให้ใช้ <article> หรือหากเป็นกลุ่มลิงก์นำทางควรใช้ <nav> ข้อควรระวังในการใช้แท็ก <section> คือไม่ควรใช้เพียงเพื่อจัดวางหรือจัดรูปแบบด้วย CSS เท่านั้น หากไม่มีหัวเรื่องหรือความชัดเจนในบทบาทของกลุ่มเนื้อหานั้น ควรพิจารณาใช้แท็ก แทน รูปแบบของการใช้แท็ก <section> เพื่อกำหนดส่วนเนื้อหาย่อยมีดังนี้

รูปแบบ

```
<section>หัวเรื่องและเนื้อหาย่อย</section>
```

8.1.6 ส่วนด้านข้าง

ส่วนด้านข้าง (Aside Section) ใช้แท็ก <aside> ซึ่งเป็นแท็กแบบคู่ที่ใช้สำหรับจัดกลุ่มเนื้อหาที่เกี่ยวข้องในทางอ้อมกับเนื้อหาหลักของหน้าเว็บ เช่น ลิงก์บทความที่เกี่ยวข้อง กล่องข้อความเพิ่มเติม โฆษณา คำคมหรือข้อมูลเสริมต่าง ๆ ซึ่งเนื้อหาในส่วนนี้มักถูกจัดวางให้อยู่ด้านข้างของพื้นที่แสดงเนื้อหาหลัก เช่น บริเวณ

ด้านซ้ายหรือด้านขวาของเนื้อหาหลัก เป็นต้น โดยมีจุดประสงค์เพื่อเสริมความเข้าใจหรือเพิ่มข้อมูลประกอบให้กับผู้ใช้งาน รูปแบบของการใช้แท็ก <aside> เพื่อกำหนดส่วนบทความมีดังนี้

รูปแบบ

<aside>เนื้อหาที่เกี่ยวข้อง เช่น บทความแนะนำ ลิงก์เพิ่มเติม หรือโฆษณา</aside>

8.1.7 ส่วนท้ายของเว็บ

ส่วนท้ายของเว็บ (Page Footer) ใช้แท็ก <footer> ซึ่งเป็นแท็กแบบคู่ที่ใช้สำหรับกำหนดกลุ่มเนื้อหาที่ปรากฏอยู่ตอนท้ายของหน้าเว็บหรือท้ายของอิลิเมนต์อื่น เช่น article section หรือ main โดยทั่วไปส่วนนี้มักประกอบด้วยข้อมูลลิขสิทธิ์ ลิงก์เชื่อมโยงไปยังหน้าอื่น ๆ ที่เกี่ยวข้อง เช่น ข้อกำหนดการใช้งาน นโยบายความเป็นส่วนตัว หรือลิงก์เครือข่ายสังคมออนไลน์ ตลอดจนข้อมูลของผู้พัฒนาเว็บไซต์ โดยในแต่ละหน้าเว็บสามารถมี <footer> ได้หลายตำแหน่ง และเมื่อใช้ภายในอิลิเมนต์ body โดยตรง ถือว่าเป็นส่วนท้ายของหน้าเว็บทั้งหมด ในขณะที่หากใช้ภายในอิลิเมนต์ article หรือ section จะถือว่าเป็นส่วนท้ายเฉพาะกลุ่มเนื้อหานั้น ๆ รูปแบบของการใช้แท็ก <footer> เพื่อกำหนดส่วนท้ายของเว็บมีดังนี้

รูปแบบ

<footer>เนื้อหาส่วนท้ายของเว็บหรือส่วนท้ายขององค์ประกอบอื่นๆ</footer>

8.2 ตัวอย่างการจัดกลุ่มเนื้อหาโดยการแบ่งส่วนตามบริบท

จากที่ได้กล่าวถึงการจัดกลุ่มเนื้อหาเชิงความหมายโดยการแบ่งออกเป็นส่วนต่าง ๆ ตามบริบท ข้างต้น ตัวอย่างต่อไปนี้แสดงให้เห็นถึงการใช้แท็กเชิงความหมายในการจัดวางโครงสร้างให้มีลักษณะดังภาพที่ 6.2

ตัวอย่างที่ 20 **ไฟล์ index.html**

```
1   <!-- ส่วนหัวของเว็บ -->
2   <header>
3       <img src="html.png" alt="">
4       <h1>การพัฒนาเว็บแอปพลิเคชัน</h1>
5   </header>
6   <!—ส่วนนำทาง -->
7   <nav>
8       <a href="index.html">หน้าแรก</a> |
9       <a href="chapter-1.html">บทที่ 1</a> |
10      <a href="chapter-2.html">บทที่ 2</a> |
11      <a href="chapter-3.html">บทที่ 3</a>
```

```
12      <a href="chapter-4.html">บทที่ 4</a> |
13  </nav>
14  <!-- ส่วนเนื้อหาหลัก -->
15  <main>
16      <!-- ส่วนบทความ -->
17      <article>
18          <!-- ส่วนย่อยลำดับที่ 1 -->
19          <section>
20              <h2>ความรู้เบื้องต้นเกี่ยวกับภาษา HTML</h2>
21              <p>
22                  ภาษา HTML ย่อมาจาก HyperText Markup Language เป็นภาษามาตรฐานที่ใช้
23                  สำหรับการกำหนดโครงสร้างของเอกสารเว็บ โดยทำหน้าที่กำกับ (Mark) ว่าเนื้อหา
24                  ส่วนใดในเอกสารมีบทบาทหรือหน้าที่อย่างไร... <a href="#">อ่านต่อ >>> </a>
25              </p>
26          </section>
27          <!-- ส่วนย่อยลำดับที่ 2 -->
28          <section>
29              <h2>ความรู้เบื้องต้นเกี่ยวกับภาษา HTML</h2>
30              <p>
31                  แท็ก (Tag) คือ คำสั่งกำกับที่ใช้ในการควบคุมหรือระบุลักษณะของเนื้อหาใน
32                  เอกสาร HTML โดยคำสั่งกำกับจะถูกเขียนระหว่างเครื่องหมาย ... <a href="#">
33                  อ่านต่อ >>></a>
34              </p>
35          </section>
36      </article>
37      <!—ส่วนด้านข้าง -->
38      <aside>
39          <h2>ดัชนีคำศัพท์</h2>
40              <ul>
41                  <li><a href="#">Backend</a></li>
42                  <li><a href="#">Container Technology</a></li>
43                  <li><a href="#">Frontend</a></li>
```

```
44                  <li><a href="#">Web Application</a></li>
45              </ul>
46      </aside>
47  </main>
48  <!-- ส่วนท้ายของเว็บ -->
49  <footer>
50      <h3> &copy; พ.ศ. 2568 โดย AJNOK.COM</h3>
51  </footer>
```

จากตัวอย่างที่ 20 เป็นรหัสคำสั่งภาษา HTML ที่เขียนในอิลิเมนต์ body ของเอกสารเว็บ โดยมีการใช้แท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมาย ดังนี้

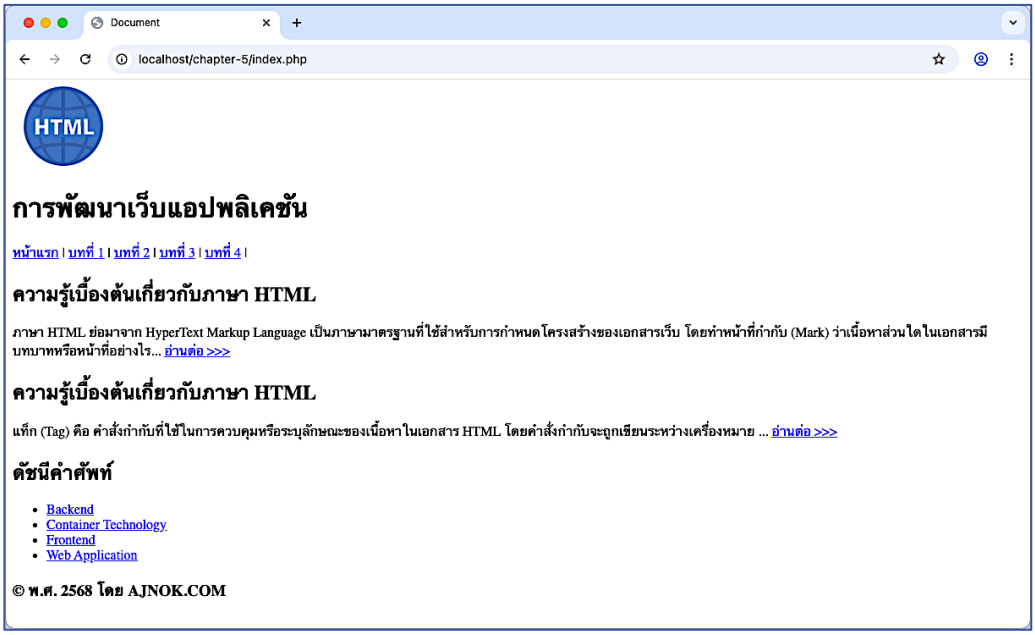
- บรรทัดที่ 2-5 เป็นการกำหนดส่วนหัวของเว็บโดยใช้แท็ก <header> โดยภายในประกอบด้วยอิลิเมนต์การแสดงรูปภาพและการกำหนดหัวเรื่องระดับ 1
- บรรทัดที่ 7-13 เป็นการกำหนดส่วนเมนูนำทางโดยใช้แท็ก <nav> ซึ่งใช้เป็นเมนูนำทางหลักของหน้าเว็บไปยังเว็บเพจอื่นๆ โดยภายในประกอบด้วยลิงก์ที่นำไปสู่ไฟล์เอกสารเว็บของเนื้อหาหน้าแรก และเนื้อหาบทที่ 1-4
- บรรทัดที่ 15-47 เป็นการกำหนดส่วนเนื้อหาหลักโดยใช้แท็ก <main> ซึ่งภายในประกอบด้วยอิลิเมนต์ลูกจำนวน 2 อิลิเมนต์ คืออิลิเมนต์ส่วนบทความ ที่กำหนดด้วยแท็ก <article> แสดงในบรรทัดที่ 17-36 และอิลิเมนต์ส่วนด้านข้างที่กำหนดด้วยแท็ก <aside> แสดงในบรรทัดที่ 38-46 นอกจากนี้ภายในอิลิเมนต์ส่วนบทความก็ยังมีการจัดกลุ่มเนื้อหาออกเป็น 2 กลุ่มโดยใช้แท็ก <section> แสดงในบรรทัดที่ 19-26 และ 28-35 ตามลำดับ
- บรรทัดที่ 49-51 เป็นการกำหนดส่วนท้ายของเว็บโดยใช้แท็ก <footer>

ผลลัพธ์ของตัวอย่างที่ 20 จะเห็นได้ว่าการจัดกลุ่มเนื้อหาด้วยแท็กเชิงความหมายโดยแบ่งเป็นส่วนต่าง ๆ ทำให้เว็บเบราว์เซอร์สามารถเรียงลำดับการแสดงผลของเนื้อหาได้อย่างเป็นระบบ เริ่มจากส่วนหัวของเว็บที่ประกอบด้วยโลโก้และชื่อเว็บไซต์ ถัดมาเป็นแถบเมนูนำทาง จากนั้นเป็นเนื้อหาหลักซึ่งประกอบด้วยบทความ 2 ตอนตามลำดับของ section และตามด้วยเนื้อหาด้านข้างที่อยู่ใน aside โดยแสดงผลถัดจากบทความตามลำดับของรหัสคำสั่ง สุดท้ายจึงเป็นส่วนท้ายของเว็บที่แสดงข้อความลิขสิทธิ์

ลักษณะการแสดงผลตามตัวอย่างผลลัพธ์นี้ ยังคงเป็นเพียงการแสดงในแนวตั้งเรียงต่อกันตามลำดับที่ปรากฏในรหัสคำสั่ง ซึ่งเป็นผลจากค่าเริ่มต้นของการแสดงผลแบบ Block-Level ของแท็กสำหรับการจัดกลุ่มในส่วนต่าง ๆ ทั้งนี้ผลลัพธ์ยังไม่มีการแบ่งพื้นที่หรือจัดตำแหน่งอย่างชัดเจน เช่น การแสดงบทความอยู่ทางซ้ายและแถบด้านข้างอยู่ทางขวา ซึ่งเป็นข้อจำกัดของการใช้ภาษา HTML เพียงอย่างเดียว โดยหากต้องการให้มีการจัดวาง

องค์ประกอบของเว็บอย่างเป็นรูปแบบ เช่น แบบคอลัมน์ หรือมีการกำหนดขนาดพื้นที่ ต้องใช้ CSS มาช่วยในการจัดรูปแบบและตกแต่งเพิ่มเติมซึ่งจะได้กล่าวถึงในบทต่อไป

ผลลัพธ์ของตัวอย่างที่ 20



9. สรุปท้ายบท

แท็ก HTML ที่ใช้ในการพัฒนาเว็บแอปพลิเคชันมีบทบาทสำคัญในการกำหนดโครงสร้างและองค์ประกอบที่แสดงผลในหน้าเว็บ โดยแบ่งตามลักษณะการใช้งาน เช่น แสดงภาพ แสดงข้อความ แสดงลิงก์ การจัดกลุ่มเนื้อหา การสร้างตาราง และการสร้างฟอร์มสำหรับรับข้อมูลจากผู้ใช้ ซึ่งแต่ละแท็กมีคุณลักษณะเฉพาะและมีค่าการแสดงผลเริ่มต้นแตกต่างกัน รวมถึงแอตทริบิวต์สำคัญที่ต้องระบุเพื่อให้สามารถทำงานได้ถูกต้องตามวัตถุประสงค์ของเว็บแอปพลิเคชัน

การใช้แท็กเชิงความหมายแทนแท็กทั่วไป เช่น div ช่วยให้โครงสร้างของเอกสาร HTML มีความเข้าใจง่ายและเป็นมิตรต่อเครื่องมือค้นหา รวมถึงการเข้าถึงผ่านเทคโนโลยีช่วยเหลือ ในขณะที่การจัดกลุ่มเนื้อหาผ่านแท็กแบบไม่มีความหมายเชิงเนื้อหา เช่น div และ span ก็ยังคงมีบทบาทในการแบ่งกลุ่มเนื้อหาสำหรับการจัดการด้วย CSS หรือ JavaScript

ฟอร์มสำหรับรับข้อมูลจากผู้ใช้ถือเป็นองค์ประกอบสำคัญของเว็บแอปพลิเคชัน โดยมีแท็กที่เกี่ยวข้องหลายประเภท ทั้งช่องกรอกข้อมูล ปุ่มกด และกลไกควบคุมการส่งข้อมูลไปยังฝั่งผู้ให้บริการ การกำหนดค่าแอตทริบิวต์

action และ method อย่างถูกต้อง รวมถึงการเลือกประเภทของอินพุตให้เหมาะสมกับลักษณะของข้อมูลที่ต้องการรับ จะช่วยให้การส่งข้อมูลมีความปลอดภัยและสอดคล้องกับพฤติกรรมของผู้ใช้งาน

การเข้าใจและใช้แท็ก HTML อย่างถูกต้องถือเป็นรากฐานสำคัญของการพัฒนาเว็บแอปพลิเคชัน ช่วยให้สามารถจัดการโครงสร้างเนื้อหา รับข้อมูลจากผู้ใช้ และควบคุมลำดับการแสดงผลได้อย่างมีประสิทธิภาพ ซึ่งเป็นจุดเริ่มต้นสำคัญในการประยุกต์ใช้ CSS เพื่อควบคุมรูปแบบ และ JavaScript เพื่อจัดการพฤติกรรมการทำงานของเว็บแอปพลิเคชันในระดับที่ซับซ้อนยิ่งขึ้น

10. กิจกรรมท้ายบท

10.1 จงอธิบายหน้าที่ของแท็ก HTML ตามกลุ่มแท็กต่อไปนี้ พร้อมยกตัวอย่างแท็กในแต่ละกลุ่มอย่างน้อยกลุ่มละ 1 แท็ก

10.1.1 แท็กสำหรับแสดงรูปภาพ

10.1.2 แท็กสำหรับสร้างลิงก์

10.1.3 แท็กสำหรับกำหนดรายการ

10.1.4 แท็กสำหรับกำหนดหัวเรื่องและข้อความ

10.2 จงเลือกแท็ก HTML ที่เหมาะสมสำหรับสถานการณ์ต่อไปนี้ พร้อมอธิบายเหตุผลพอสังเขป:

10.2.1 การแสดงภาพประกอบสินค้า

10.2.2 การเชื่อมโยงไปยังหน้าเว็บอื่น

10.2.3 การแสดงรายการเมนูอาหาร

10.2.4 การแสดงข้อมูลตารางเปรียบราคา

10.2.5 การรับข้อมูลอีเมลจากผู้ใช้

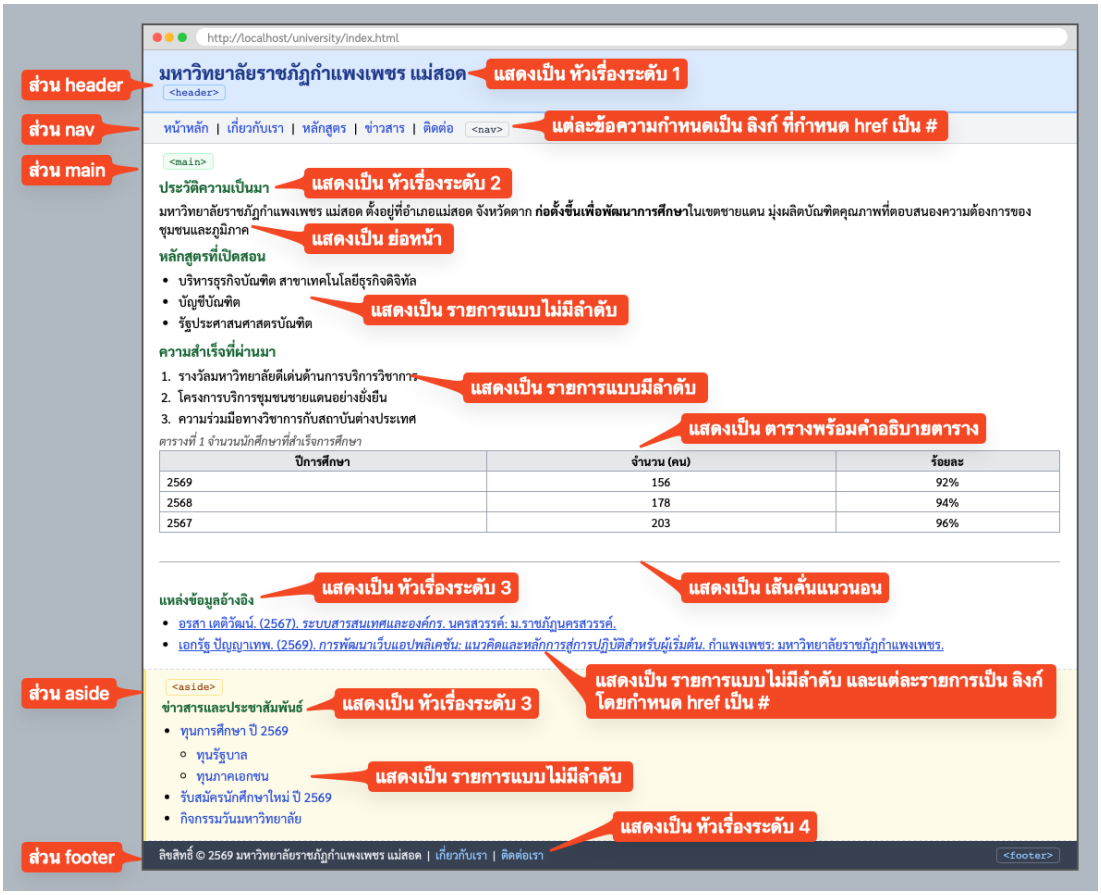
10.2.6 การจัดกลุ่มส่วนเนื้อหาหลักของหน้าเว็บ

10.3 จงเปรียบเทียบความแตกต่างระหว่างการใช้แท็กสำหรับการจัดกลุ่มเนื้อหาที่ไม่มีความหมายเชิงเนื้อหา (เช่น div และ span) กับแท็กสำหรับการจัดกลุ่มเนื้อหาเชิงความหมาย (เช่น header, nav, main, section, article, aside และ footer) พร้อมยกตัวอย่างสถานการณ์ที่ควรใช้แท็กแต่ละกลุ่ม

10.4 จงเขียนรหัส HTML เพื่อสร้างตารางแสดงรายการสินค้าอย่างน้อย 3 แถว โดยใช้โครงสร้างของตารางให้ครบถ้วน ได้แก่ ส่วนหัวของตาราง ส่วนเนื้อหาของตาราง และเซลล์ข้อมูลในแต่ละแถว พร้อมอธิบายหน้าที่ของแท็กที่ใช้ในตารางพอสังเขป

10.5 จงเขียนหน้าเว็บโดยใช้ภาษา HTML บันทึกไฟล์ชื่อ member.html เพื่อสร้างฟอร์มที่มีช่องกรอกข้อมูลชื่อ สกุล และหมายเลขโทรศัพท์ รวมทั้งช่องตัวเลือกข้อมูลระดับการศึกษา ปุ่มส่ง และปุ่มรีเซ็ต โดยกำหนดให้ฟอร์มส่งข้อมูลไปยังไฟล์ชื่อ record.php และใช้วิธีการส่งข้อมูลแบบ GET

10.6 จากภาพผลลัพธ์ของหน้าเว็บต่อไปนี้ จงเขียนหน้าเว็บโดยใช้ภาษา HTML โดยตั้งชื่อไฟล์เป็น index.html ทั้งนี้ยังไม่ต้องทำการกำหนดรูปแบบสีโดยใช้ CSS



ภาพที่ 6.3 ตัวอย่างการจัดกลุ่มเนื้อหาและบทบาทของเนื้อหาในเอกสารเว็บ

10.6.1 กำหนดโครงสร้างเอกสาร HTML ให้ครบถ้วนตามรูปแบบพื้นฐานของเอกสารเว็บ ได้แก่ การประกาศชนิดเอกสาร ส่วน html ส่วน head และส่วน body พร้อมกำหนดชื่อหน้าเว็บให้เหมาะสมกับเนื้อหา

10.6.2 จัดกลุ่มส่วนประกอบหลักของหน้าเว็บด้วยแท็กเชิงความหมาย ได้แก่ header, nav, main, aside และ footer เพื่อแสดงให้เห็นบทบาทของเนื้อหาแต่ละส่วนอย่างชัดเจนตามภาพประกอบ

10.6.3 ในส่วน header ให้แสดงชื่อเว็บไซต์หรือชื่อหน่วยงานเป็นหัวเรื่องระดับที่ 1 โดยใช้ข้อความว่า “มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด” และให้เพิ่มข้อความภาษาอังกฤษขนาดเล็กใต้ชื่อหน่วยงานตามความเหมาะสม

10.6.4 ในส่วน nav ให้สร้างรายการเชื่อมโยงเมนูนำทางอย่างน้อย 5 รายการ ได้แก่ หน้าแรก, เกี่ยวกับเรา, หลักสูตร, ข่าวสาร และ ติดต่อ โดยกำหนดค่า href ของลิงก์เป็น # เพื่อใช้เป็นตัวอย่างการสร้างลิงก์ภายในหน้าเว็บ

10.6.5 ในส่วน main ให้สร้างหัวเรื่องระดับที่ 2 สำหรับหัวข้อ “ประวัติความเป็นมา” และเขียนย่อหน้าสั้น ๆ เพื่อแสดงเนื้อหาเกี่ยวกับความเป็นมาของหน่วยงานหรือเว็บไซต์

10.6.6 ภายในส่วน main ให้สร้างหัวเรื่องระดับที่ 3 สำหรับหัวข้อ “หลักการที่เปิดสอน” แล้วแสดงรายการหลักสูตรในรูปแบบรายการแบบไม่มีลำดับ โดยมีรายการอย่างน้อย 3 รายการ เช่น บริหารธุรกิจบัณฑิต เทคโนโลยีธุรกิจดิจิทัล และรัฐประศาสนศาสตรบัณฑิต

10.6.7 สร้างหัวเรื่องระดับที่ 3 สำหรับหัวข้อ “ความสำเร็จที่ผ่านมา” แล้วแสดงรายการข้อมูลในรูปแบบรายการแบบมีลำดับ โดยมีข้อมูลอย่างน้อย 3 รายการ

10.6.8 สร้างตารางแสดงข้อมูลจำนวนนักศึกษาหรือข้อมูลสถิติที่เกี่ยวข้อง โดยตารางต้องมีคอลัมน์อย่างน้อย 3 คอลัมน์ ได้แก่ ปีการศึกษา, จำนวน (คน) และ ร้อยละ และมีข้อมูลอย่างน้อย 3 แถว พร้อมกำหนดโครงสร้างตารางให้ครบถ้วน ได้แก่ ส่วนหัวของตาราง แถวของตาราง และเซลล์ข้อมูล

10.6.9 หลังตาราง ให้เพิ่มส่วน “แหล่งข้อมูลอ้างอิง” โดยใช้หัวเรื่องระดับที่ 3 และแสดงรายการลิงก์อ้างอิงอย่างน้อย 2 รายการ ในรูปแบบรายการแบบไม่มีลำดับ โดยกำหนดค่า href ของแต่ละลิงก์เป็น #

10.6.10 ในส่วน aside ให้สร้างหัวเรื่องระดับที่ 3 จำนวนอย่างน้อย 2 หัวข้อ เช่น “ข่าวล่าสุด” และ “ข่าวประชาสัมพันธ์” แล้วแสดงรายการข่าวเป็นรายการแบบไม่มีลำดับ โดยแต่ละรายการให้กำหนดเป็นลิงก์ตัวอย่างด้วยค่า href="#"

10.6.11 ในส่วน footer ให้แสดงข้อความลิขสิทธิ์ของเว็บไซต์ และเพิ่มลิงก์อย่างน้อย 2 รายการ เช่น เกี่ยวกับเรา และ ติดต่อเรา โดยกำหนดค่า href เป็น #

10.6.12 ให้ตรวจสอบว่าเอกสาร HTML ที่สร้างขึ้นมีการเลือกใช้แท็กสอดคล้องกับบทบาทของเนื้อหา เช่น ใช้หัวเรื่องตามลำดับระดับความสำคัญ ใช้รายการแบบมีลำดับหรือไม่มีลำดับให้เหมาะสม ใช้ตารางสำหรับข้อมูลที่มีโครงสร้างเป็นแถวและคอลัมน์ และใช้แท็กเชิงความหมายในการจัดกลุ่มส่วนประกอบหลักของหน้าเว็บ



ภาษา CSS

บทที่ 7 มุ่งเน้นการนำเสนอหลักการของ CSS สำหรับการกำหนดลักษณะการแสดงผลของอิลิเมนต์ HTML เพื่อให้องค์ประกอบต่าง ๆ มีรูปแบบที่เหมาะสมกับการนำเสนอข้อมูล มีความน่าสนใจ และสามารถปรับเปลี่ยนให้สอดคล้องกับอุปกรณ์ที่ใช้แสดงผลได้ โดยเนื้อหาในบทนี้จะเริ่มจากการทำความเข้าใจโครงสร้างของคำสั่ง CSS ชนิดของการใช้งาน CSS และลำดับความสำคัญของการกำหนดคุณสมบัติ หัวข้อหลักในบทนี้ประกอบด้วย

1. ความรู้เบื้องต้นเกี่ยวกับ CSS
2. รูปแบบการกำหนดคุณสมบัติการแสดงผล
3. วิธีใช้ CSS ร่วมกับเอกสารเว็บ
4. สีและหน่วยวัด
5. สรุปท้ายบท
7. กิจกรรมท้ายบท

1. ความรู้เบื้องต้นเกี่ยวกับ CSS

การจัดรูปแบบการแสดงผลให้กับเนื้อหา HTML เป็นองค์ประกอบสำคัญของการพัฒนาเว็บแอปพลิเคชัน โดยเฉพาะอย่างยิ่งในส่วนการแสดงผลในเว็บเบราว์เซอร์ให้กับผู้ใช้ ซึ่งต้องมีการจัดตำแหน่ง กำหนดขนาด สี ขอบ ตัวอักษร และรูปแบบอื่น ๆ เพื่อให้การนำเสนอข้อมูลมีความสวยงาม ชัดเจน และเหมาะสมกับบริบทของการใช้งาน CSS เป็นภาษามาตรฐานที่ใช้ร่วมกับ HTML สำหรับควบคุมลักษณะการแสดงผลขององค์ประกอบต่าง ๆ บนหน้าเว็บ ไม่ว่าจะเป็นการแสดงผลบนหน้าจอ บนกระดาษ หรือสื่ออื่น ๆ โดยแยกหน้าที่ออกจาก HTML อย่างชัดเจน กล่าวคือ HTML ทำหน้าที่เป็นโครงสร้างของเนื้อหาดังที่ได้กล่าวมาแล้วในบทก่อนหน้า ส่วน CSS ทำหน้าที่กำหนดลักษณะของการแสดงผล

1.1 ความหมายและบทบาทของ CSS

จากคำอธิบายของ W3C (2025) ซึ่งเป็นผู้กำหนดมาตรฐานของ CSS ได้กล่าวไว้ว่า CSS ย่อมาจากคำว่า Cascading Style Sheets เป็นภาษาสำหรับเขียนชุดของกฎที่ใช้ควบคุมลักษณะการแสดงผลที่เรียกว่า Style Sheets ของเอกสารต้นทาง (Source Document) โดยไม่ได้เปลี่ยนแปลงความหมายเชิงโครงสร้างของเอกสารนั้น ๆ กล่าวคือ CSS ทำหน้าที่บรรยายการจัดรูปแบบขององค์ประกอบต่าง ๆ เช่น ขนาดตัวอักษร สี เส้นขอบ หรือช่องว่าง โดยอาศัยคำสั่งที่แยกออกจากภาษา HTML ซึ่งเป็นภาษาที่ใช้สร้างโครงสร้างเอกสาร ตัวอย่างเช่น สามารถกำหนดให้หัวเรื่องระดับ 1 แสดงเป็นตัวหนาสีฟ้า หรือให้ย่อหน้ามีระยะห่างเท่ากันโดยไม่ต้องแก้ไขโครงสร้าง HTML เดิมแต่อย่างใด เป็นต้น

การประมวลผลคำสั่ง CSS เพื่อนำไปแสดงผลในสื่อที่แตกต่างกัน เช่น หน้าจอ กระดาษ หรือเสียง จะดำเนินการโดยโปรแกรมที่เรียกว่า User Agent ซึ่งหมายถึงเว็บเบราว์เซอร์ เช่น Google Chrome, Mozilla Firefox หรือ Safari เป็นต้น ผู้ที่สร้างเอกสารเว็บและชุดของกฎที่ใช้ควบคุมลักษณะการแสดงผล เรียกว่า ผู้เขียน (Author) ส่วนผู้ใช้ (User) คือผู้ที่เข้าถึงเอกสารเว็บผ่านเบราว์เซอร์

CSS มีบทบาทสำคัญในการจัดรูปแบบการแสดงผลของเว็บ โดยทำหน้าที่แยกชั้นการนำเสนอ (Presentation Layer) ออกจากชั้นโครงสร้างเนื้อหา (Structure Layer) ที่รับผิดชอบโดยภาษา HTML เพื่อให้สามารถควบคุมการจัดวาง (Layout) และควบคุมลักษณะการแสดงผลขององค์ประกอบต่าง ๆ ในหน้าเว็บได้

บทบาทที่สำคัญอีกประการหนึ่งของ CSS คือการสร้างเอกลักษณ์และความสม่ำเสมอในการออกแบบหน้าเว็บ เช่น การกำหนดให้ปุ่มนำทางมีขนาด รูปทรง และสีเดียวกันในทุกหน้า หรือการจัดวางเนื้อหาตามแนวทางที่สอดคล้องกับหลักการออกแบบขององค์กร นอกจากนี้ยังสามารถกำหนดรูปแบบการแสดงผลแบบรวมศูนย์ (Centralized Styling) โดยการรวมกฎที่ใช้ควบคุมลักษณะการแสดงผลไว้ด้วยกันสำหรับควบคุมรูปแบบของเว็บหลายๆ หน้า ซึ่งช่วยลดความซ้ำซ้อนของการเขียนกฎและทำให้การดูแล ปรับปรุงทำได้ง่ายขึ้น และช่วยรักษาความสม่ำเสมอในการแสดงผลของเว็บไซต์ให้มีรูปแบบเดียวกันตลอด

นอกจากนี้ CSS ยังมีบทบาทสำคัญต่อการปรับปรุงประสบการณ์ผู้ใช้ และสนับสนุนการพัฒนาเว็บด้วยแนวทางการออกแบบหน้าเว็บแบบตอบสนอง (Responsive Design) เพื่อตอบสนองต่อการใช้งานในอุปกรณ์ที่มีขนาดหน้าจอที่แตกต่างกันอีกด้วย

1.2 พัฒนาการและมาตรฐานของ CSS

พัฒนาการและมาตรฐานของ CSS ที่กล่าวถึงโดย W3C (2025) ซึ่งเป็นองค์กรที่พัฒนาและกำหนดมาตรฐานของภาษา CSS ได้กล่าวว่า ภาษา CSS มีการพัฒนามาตั้งแต่ปี พ.ศ. 2539 เพื่อแก้ไขข้อจำกัดของ HTML ในการควบคุมลักษณะการแสดงผลของเอกสารเว็บ โดยในช่วงเริ่มต้นได้มีการเผยแพร่ CSS1 ซึ่งสามารถกำหนดลักษณะพื้นฐานของข้อความ เช่น สี ตัวหนา และการจัดตำแหน่ง

ถัดมาในปีพ.ศ. 2541 ได้มีการพัฒนา CSS2 ที่เพิ่มความสามารถด้านการจัดวางและการนำเสนออย่างมีโครงสร้างมากขึ้น เช่น การจัดวางแบบกล่อง (Box Model) และการกำหนดชั้นของการแสดงผล (Z-index) เป็นต้น

ในระยะแรกการพัฒนา CSS ถูกกำหนดในลักษณะเป็นรุ่นเวอร์ชัน เช่น CSS1, CSS2 และ CSS2.1 อย่างไรก็ตาม แนวทางดังกล่าวเริ่มมีข้อจำกัดต่อการรองรับเทคโนโลยีใหม่ ๆ ที่เกิดขึ้นอย่างต่อเนื่อง จึงนำไปสู่การเปลี่ยนแปลงรูปแบบการพัฒนาในเวลาต่อมา โดยตั้งแต่ปี พ.ศ. 2551 เป็นต้นมา W3C ได้กำหนดให้ CSS พัฒนาในรูปแบบโมดูล (CSS Modules) เปรียบเสมือนส่วนต่อขยายที่นำมาใช้แทนการรวมทุกความสามารถไว้ในเวอร์ชันเดียว ซึ่งทำให้สามารถแยกพัฒนาแต่ละคุณลักษณะของ CSS ได้อย่างอิสระและต่อเนื่อง เช่น CSS Flexbox, CSS Grid Layout, CSS Transitions และ CSS Animations โดยแต่ละโมดูลจะมีเวอร์ชันของตนเองและสามารถปรับปรุงได้อย่างยืดหยุ่น

จากแนวทางใหม่นี้ W3C จึงไม่ใช้ชื่อรุ่นเวอร์ชันแบบ CSS3 หรือ CSS4 อีกต่อไป แต่จะใช้ชื่อ CSS เพียงอย่างเดียว โดยหมายถึงชุดของโมดูลที่พัฒนาต่อเนื่องในลักษณะ Living Standard ซึ่งเป็นการปรับปรุงและเผยแพร่เป็นระยะอย่างต่อเนื่องโดยไม่ต้องรอการสรุปเป็นหมายเลขเวอร์ชันเหมือนในอดีตที่ผ่านมา เป็นลักษณะเดียวกันกับการกำหนดมาตรฐานของภาษา HTML ที่ใช้ชื่อมาตรฐานเป็น HTML Living Standard แนวทางนี้ทำให้ CSS สามารถตอบสนองต่อความเปลี่ยนแปลงของเทคโนโลยีเว็บและอุปกรณ์แสดงผลที่หลากหลายได้อย่างมีประสิทธิภาพ

มาตรฐาน CSS ได้รับการดูแลโดย W3C CSS Working Group ซึ่งเป็นคณะทำงานเฉพาะด้านที่เปิดรับข้อเสนอแนะจากนักพัฒนาและองค์กรต่าง ๆ โดยตรง กระบวนการนี้ทำให้ CSS ยังคงเป็นหนึ่งในภาษาหลักของเว็บแบบเปิด (Open Web Platform) ที่เว็บเบราว์เซอร์ให้การสนับสนุนอย่างกว้างขวาง และเป็นเครื่องมือสำคัญในการจัดรูปแบบเว็บแอปพลิเคชันให้มีความสวยงาม น่าใช้ และตอบสนองต่ออุปกรณ์หลากหลายประเภท

ด้วยพัฒนาการที่ยืดหยุ่นและการรองรับอย่างแพร่หลายจากเครื่องมือพัฒนาเว็บและเว็บเบราว์เซอร์ CSS จึงเป็นภาษามาตรฐานในการจัดรูปแบบการนำเสนอของเอกสารเว็บ และเป็นพื้นฐานสำคัญในการพัฒนาเว็บแอปพลิเคชัน อันนำไปสู่การสร้างประสบการณ์ที่ดีของผู้ใช้ ซึ่งเป็นปัจจัยหนึ่งของการส่งเสริมการยอมรับการใช้งานเว็บแอปพลิเคชัน

1.3 ประโยชน์ของ CSS

จากบทบาทของ CSS ในการพัฒนาเว็บและเว็บแอปพลิเคชัน สามารถสรุปประโยชน์ของ CSS ได้ดังนี้

1.3.1 แยกส่วนการจัดรูปแบบออกจากโครงสร้างเนื้อหา

CSS ช่วยให้สามารถแยกการกำหนดลักษณะการแสดงผลออกจากโครงสร้างของเอกสาร HTML ได้อย่างชัดเจน ซึ่งเอื้อต่อการจัดการและดูแลรูปแบบของเว็บไซต์โดยไม่กระทบต่อโครงสร้างข้อมูลหรือเนื้อหาเดิม

1.3.2 สร้างความสม่ำเสมอในการออกแบบ

สามารถกำหนดรูปแบบส่วนต่าง ๆ ของเว็บไซต์ให้มีลักษณะเดียวกัน เช่น ขนาดและสีของหัวข้อ รูปแบบของปุ่มนำทาง หรือช่องกรอกข้อมูล ส่งผลให้ผู้ใช้มีประสบการณ์ใช้งานที่ต่อเนื่อง

1.3.3 สนับสนุนการออกแบบหน้าเว็บแบบตอบสนอง

CSS รองรับการออกแบบที่ปรับเปลี่ยนรูปแบบการแสดงผลให้เหมาะสมกับอุปกรณ์ที่ใช้งาน เช่น โทรศัพท์มือถือ แท็บเล็ต หรือจอคอมพิวเตอร์ โดยใช้เทคนิคเช่น Media Queries ที่มีอยู่ใน CSS ซึ่งเป็นประโยชน์ต่อการพัฒนาเว็บแอปพลิเคชันเพื่อรองรับอุปกรณ์มีความหลากหลายของขนาดหน้าจอ

1.3.4 ปรับเปลี่ยนและปรับปรุงได้อย่างรวดเร็ว

การรวมกฎการจัดรูปแบบไว้ในไฟล์ CSS ภายนอก ช่วยให้สามารถปรับเปลี่ยนลักษณะการแสดงผลของเว็บไซต์ทั้งระบบได้จากจุดเดียว ลดเวลาในการแก้ไข และเพิ่มความสะดวกในการดูแลรักษา รวมทั้งทำให้เว็บแต่ละหน้ามีรูปแบบที่ความสอดคล้องกันทั้งหมด

1.3.5 ช่วยลดขนาดรหัสคำสั่ง HTML และเพิ่มความเร็วในการโหลดหน้าเว็บ

การแยกชุดกำหนดรูปแบบการแสดงผลออกจากโครงสร้างของเนื้อหาทำให้เอกสารเว็บมีความกระชับ อ่านง่าย และสามารถโหลดได้เร็วขึ้น โดยเฉพาะเมื่อเว็บเบราว์เซอร์มีการจัดเก็บ CSS ไว้ในหน่วยบันทึกข้อมูลของเว็บเบราว์เซอร์เอง

1.3.6 รองรับการเข้าถึงและใช้งานสำหรับผู้ใช้หลากหลายกลุ่ม

CSS สามารถปรับรูปแบบการแสดงผลให้เหมาะสมกับผู้ใช้ที่มีข้อจำกัดทางสายตาหรือความสามารถในการใช้งาน ผ่านการตั้งค่าแบบกำหนดเอง เช่น High-contrast Mode หรือ Custom Stylesheets ที่ใช้งานผ่านเบราว์เซอร์

2. รูปแบบการกำหนดคุณสมบัติการแสดงผล

การใช้งานภาษา CSS เพื่อจัดรูปแบบเอกสารเว็บจะอยู่บนพื้นฐานของการกำหนดคุณสมบัติการแสดงผลให้กับอิลิเมนต์ต่าง ๆ ของเอกสารเว็บ ทำได้โดยการกำหนดชื่อคุณสมบัติ (Property) ที่ต้องการพร้อมด้วยค่าของคุณสมบัติ (Value) ตามลำดับ โดยมีรูปแบบของการกำหนดคุณสมบัติดังนี้

รูปแบบที่ 1

ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
ชื่อคุณสมบัติที่ 2: ค่าของคุณสมบัติที่ 2;
...
ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;

รูปแบบที่ 2

ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1; ชื่อคุณสมบัติที่ 2: ค่าของคุณสมบัติที่ 2; ...;

จากรูปแบบของการกำหนดคุณสมบัติการแสดงผลในรูปแบบที่ 1 และ 2 การกำหนดค่าให้กับคุณสมบัติใดๆ จะใช้เครื่องหมาย : (Collon) เป็นเครื่องหมายของการกำหนดค่า ในอิลิเมนต์หนึ่งๆ สามารถกำหนดคุณสมบัติการแสดงผลได้หลายคุณสมบัติ โดยจะใช้เครื่องหมาย ; (Semi-Collon) เพื่อแสดงจุดสิ้นสุดของการกำหนดค่าให้แต่ละคุณสมบัติ ความแตกต่างของทั้งสองรูปแบบคือ แบบแรกเป็นการกำหนดคุณสมบัติการแสดงผลโดยเขียนแยกบรรทัดเพื่อความสะดวกในการอ่าน ส่วนรูปแบบที่ 2 เป็นการกำหนดคุณสมบัติหลายๆ คุณสมบัติโดยเขียนเรียงต่อกันเป็นบรรทัดเดียว ซึ่งการเขียนรูปแบบนี้จะใช้ในกรณีที่ต้องกำหนดคุณสมบัติลงในแท็กโดยตรง

รูปแบบของการกำหนดคุณสมบัติและค่าของ CSS ที่นำเสนอนี้ จะถูกนำไปใช้ในการกำหนดคุณสมบัติการแสดงผลเมื่อนำ CSS ไปใช้ร่วมกับเอกสารเว็บวิธีต่าง ๆ ซึ่งจะได้กล่าวถึงในลำดับถัดไป

3. วิธีใช้ CSS ร่วมกับเอกสารเว็บ

การใช้ CSS ร่วมกับเอกสารเว็บสามารถทำได้ 3 วิธี ได้แก่ การใช้ CSS แบบแทรกในแท็ก การใช้ CSS แบบภายใน และการใช้ CSS แบบภายนอก โดยทั้งสามวิธีก็จะมีรูปแบบการกำหนดคุณสมบัติการแสดงผลตามรูปแบบที่แสดงไว้ในหัวข้อที่ 2.1 ดังนี้

3.1 การใช้ CSS แบบแทรกในแท็ก

การใช้ CSS แบบแทรกในแท็ก (Inline CSS) เป็นวิธีการกำหนดลักษณะการแสดงผลของอิลิเมนต์ HTML โดยตรงภายในแท็กของอิลิเมนต์นั้น ๆ ผ่านแอตทริบิวต์ style ซึ่งระบุชื่อคุณสมบัติและค่าของคุณสมบัติที่ต้องการจัดรูปแบบไว้ภายในแท็ก ลักษณะการเขียนจะเหมาะกับการกำหนดรูปแบบเฉพาะจุดที่ไม่มีการใช้ซ้ำหลายแห่ง เช่น การกำหนดสีเฉพาะสำหรับข้อความคำเตือน หรือการปรับขนาดตัวอักษรในบางตำแหน่งให้แตกต่างจากส่วนอื่นของเอกสาร โดยมีรูปแบบและตัวอย่างการใช้งานดังนี้

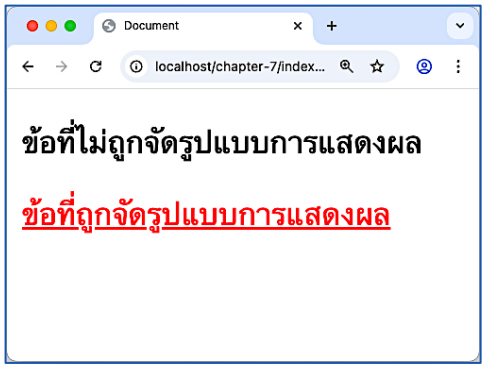
รูปแบบ

<ชื่อแท็ก style="ชื่อคุณสมบัติ: ค่าของคุณสมบัติ; ...;">

ตัวอย่างที่ 1

```
1  <p>ข้อที่ไม่ถูกจัดรูปแบบ</p>
2  <p style="color: red; text-decoration: underline;">ข้อที่ถูกจัดรูปแบบการแสดงผล</p>
```

ผลลัพธ์ของตัวอย่างที่ 1



จากตัวอย่างที่ 1 มีการสร้างอิลิเมนต์การแสดงเนื้อหาเป็นย่อหน้าจำนวน 2 อิลิเมนต์ด้วยแท็ก <p> โดยอิลิเมนต์แรก (บรรทัดที่ 1) ไม่มีการกำหนดคุณสมบัติการแสดงผลด้วย CSS ทำให้รูปแบบของการแสดงผลเป็นไปตามตามค่าเริ่มต้นของเว็บเบราว์เซอร์ ส่วนอิลิเมนต์ที่ 2 (บรรทัดที่ 2) มีการกำหนดคุณสมบัติการแสดงผลจำนวน 2 รายการคือ คุณสมบัติสีของตัวอักษร (color) กำหนดค่าเป็น red หมายถึงสีแดง และคุณสมบัติการตกแต่งข้อความ (text-decoration) กำหนดค่าเป็น underline หมายถึงการขีดเส้นใต้ คุณสมบัติที่กำหนดเหล่านี้จะส่งผลต่อเนื้อหาของอิลิเมนต์ซึ่งเป็นข้อความคำว่า *ข้อที่ถูกจัดรูปแบบการแสดงผล* ทำให้ข้อความดังกล่าวแสดงเป็นสีแดงและมีการขีดเส้นใต้

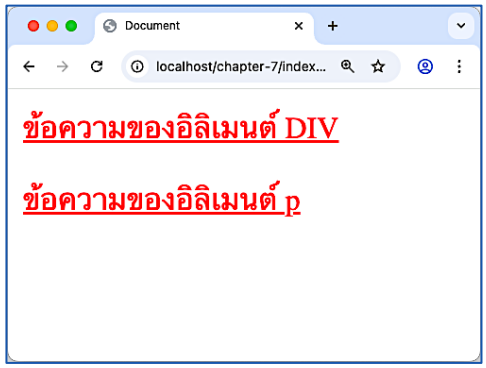
ตัวอย่างที่ 2

```
1  <div style="color: red; text-decoration: underline;">
2      ข้อความของอิลิเมนต์ DIV
3      <p>ข้อความของอิลิเมนต์ p</p>
4  </div>
```

จากตัวอย่างที่ 2 มีการสร้างอิลิเมนต์จัดกลุ่มเนื้อหาโดยใช้แท็ก (บรรทัดที่ 1-4) โดยภายในประกอบด้วยข้อความธรรมดาที่เป็นคำว่า *ข้อความของอิลิเมนต์ DIV* (บรรทัดที่ 2) และอิลิเมนต์ p ซึ่งทำหน้าที่แสดงเนื้อหาเป็นย่อหน้า ภายในอิลิเมนต์มีข้อความว่า *ข้อความของอิลิเมนต์ p* (บรรทัดที่ 3) ซึ่งกล่าวได้ว่า อิลิเมนต์ div เป็นอิลิเมนต์แม่ของทั้งข้อความธรรมดาและอิลิเมนต์ p โดยที่อิลิเมนต์ div มีการกำหนดคุณสมบัติการแสดงผลด้วย CSS แบบแทรกในแท็ก จำนวน 2 รายการ ประกอบด้วย คุณสมบัติสีของตัวอักษร (color) กำหนดค่า

เป็น red หมายถึงสีแดง และคุณสมบัติการตกแต่งข้อความ (text-decoration) กำหนดค่าเป็น underline หมายถึงการขีดเส้นใต้เช่นเดียวกับตัวอย่างก่อนหน้า

ผลลัพธ์ของตัวอย่างที่ 2

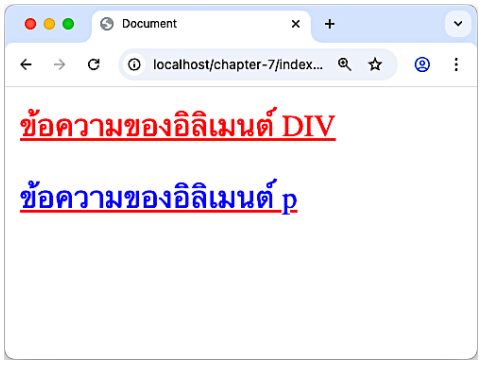


จากผลลัพธ์ของตัวอย่างที่ 2 ที่ปรากฏบนเบราว์เซอร์จะพบว่า ทั้งข้อความธรรมดาและข้อความที่เป็นเนื้อหาของอิลิเมนต์ p ต่างก็ได้รับการแสดงผลด้วยสีแดงและมีเส้นใต้ตามที่กำหนดในแท็ก ซึ่งเป็นผลมาจากคุณสมบัติการถ่ายทอด (Inheritance) ของ CSS ที่มีผลครอบคลุมไปยังอิลิเมนต์ลูกที่อยู่ภายในอิลิเมนต์แม่ทั้งหมด

ตัวอย่างที่ 3

```
1  <div style="color: red; text-decoration: underline;">
2      ข้อความของอิลิเมนต์ DIV
3      <p style="color: blue";>ข้อความของอิลิเมนต์ p</p>
4  </div>
```

ผลลัพธ์ของตัวอย่างที่ 3



จากตัวอย่างที่ 3 เป็นการสร้างอิลิเมนต์จัดกลุ่มเนื้อหาโดยใช้แท็ก ซึ่งภายในประกอบด้วยข้อความธรรมดาและอิลิเมนต์ <p> ที่แสดงผลเป็นย่อหน้า โดยมีความสัมพันธ์ระหว่างอิลิเมนต์แม่และอิลิเมนต์ลูกเช่นเดียวกับตัวอย่างก่อนหน้า ในอิลิเมนต์ div มีการกำหนดคุณสมบัติการแสดงผลด้วย CSS จำนวน 2 รายการ ได้แก่ สีตัวอักษรเป็นสีแดง (color: red;) และการขีดเส้นใต้ข้อความ (text-decoration: underline;) ซึ่งยังส่งผลต่อทั้งข้อความธรรมดาและอิลิเมนต์ลูกที่อยู่ภายใน

อย่างไรก็ตาม ในตัวอย่างนี้มีการกำหนดคุณสมบัติการแสดงผลเพิ่มเติมในอิลิเมนต์ p โดยระบุคุณสมบัติ color: blue; ในแท็ก <p> ซึ่งเป็นการทำให้ข้อความเป็นสีน้ำเงิน เพื่อเปรียบเทียบผลกระทบของการกำหนดค่าคุณสมบัติซ้ำในอิลิเมนต์ลูก จากผลลัพธ์ที่แสดงบนเบราว์เซอร์จะพบว่า ข้อความธรรมดาที่อยู่ในอิลิเมนต์ div ยังคงแสดงเป็นสีแดงและมีเส้นใต้ซึ่งเป็นผลจากคุณสมบัติของการถ่ายทอด ในขณะที่ข้อความภายในอิลิเมนต์ p แสดงเป็นสีน้ำเงินและมีเส้นใต้ แสดงให้เห็นว่า CSS หากอิลิเมนต์ลูกมีการกำหนดคุณสมบัติซ้ำ (Override) เว็บเบราว์เซอร์จะแสดงผลตามค่าที่กำหนดไว้ในอิลิเมนต์ลูก โดยยังคงรับการถ่ายทอดคุณสมบัติอื่นจากอิลิเมนต์แม่ตามปกติ

การใช้ CSS แบบแทรกในแท็กเหมาะสำหรับการปรับแต่งอิลิเมนต์เฉพาะจุดหรือกรณีที่ต้องการเปลี่ยนแปลงลักษณะการแสดงผลเพียงเล็กน้อย โดยคุณสมบัติที่กำหนดจะส่งผลต่ออิลิเมนต์นั้นโดยตรงทันที อย่างไรก็ตาม วิธีนี้ไม่เหมาะสำหรับการจัดรูปแบบอิลิเมนต์จำนวนมาก เนื่องจากจะทำให้รหัสภาษา HTML ขาดความเป็นระเบียบ ไม่สอดคล้องกับแนวทางการแยกส่วนโครงสร้างเนื้อหาออกจากการนำเสนอ และอาจก่อให้เกิดความซ้ำซ้อนหรือความยุ่งยากในการปรับปรุง หากมีอิลิเมนต์หลายรายการที่ต้องการรูปแบบเดียวกัน

3.2 การใช้ CSS แบบภายใน

การใช้ CSS แบบภายใน (Internal CSS) เป็นวิธีการเขียนคำสั่ง CSS รวมไว้ในไฟล์เอกสารเว็บเดียวกัน โดยวางคำสั่งทั้งหมดไว้ภายในแท็ก <style> ซึ่งต้องอยู่ภายในอิลิเมนต์ head ของเอกสารเว็บ วิธีนี้เหมาะสำหรับเว็บเพจขนาดเล็ก หรือกรณีที่ต้องการกำหนดรูปแบบเฉพาะในบางหน้าเว็บ โดยไม่ต้องสร้างไฟล์ CSS แยกออกจากไฟล์ HTML โดยมีรูปแบบดังนี้

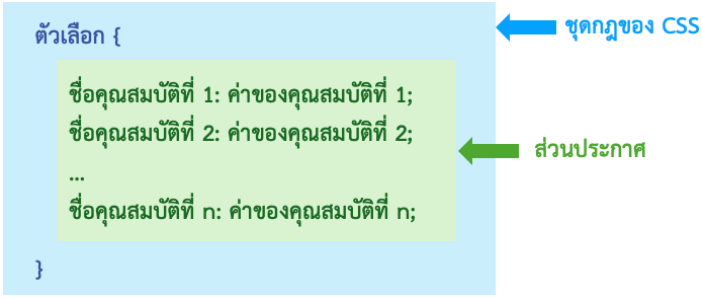
รูปแบบ

```
<head>
    <style>
        ตัวเลือก {
            ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
            ชื่อคุณสมบัติที่ 2: ค่าของคุณสมบัติที่ 2;
            ...
            ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
        }
```

```
    </style>
</head>
```

3.2.1 ชุดกฎของ CSS

หลังจากแยกคำสั่ง CSS ออกจากแท็ก HTML ไปไว้ในส่วน head ของเอกสารเว็บ สิ่งสำคัญที่ต้องพิจารณาคือการระบุอิลิเมนต์เป้าหมายที่ต้องการให้ได้รับผลจากคุณสมบัติการแสดงผลนั้น ๆ โดยการควบคุมการแสดงผลจะใช้ชุดกฎของ CSS (CSS Ruleset) เป็นโครงสร้างหลัก ซึ่งประกอบด้วยตัวเลือกและส่วนประกาศ ดังภาพที่ 7.1



ภาพที่ 7.1 ชุดกฎของ CSS

1) ตัวเลือก

ตัวเลือก (Selector) เป็นองค์ประกอบที่ใช้ในการระบุอิลิเมนต์เป้าหมายที่ต้องการให้จัดรูปแบบการแสดงผลตามที่กำหนดไว้ในส่วนประกาศของกฎ CSS โดยตัวเลือกพื้นฐานแบ่งออกเป็น 3 ชนิดหลัก ได้แก่ ตัวเลือกแบบแท็ก ตัวเลือกแบบคลาส และตัวเลือกแบบรหัสประจำอิลิเมนต์ เช่น p, .info, และ #main ตามลำดับ

- **ตัวเลือกแบบแท็ก**

ตัวเลือกแบบแท็ก (Tag Selector) เป็นการระบุอิลิเมนต์เป้าหมายโดยใช้ชื่อแท็กในภาษา HTML โดยตรง เช่น h1, p, div เป็นต้น เมื่อใช้ชื่อแท็กเป็นตัวเลือก รูปแบบการแสดงผลตามที่กำหนดไว้ในส่วนประกาศของกฎ CSS จะมีผลต่ออิลิเมนต์ทั้งหมดที่ใช้แท็กนั้นโดยทันที โดยไม่จำเป็นต้องกำหนดแอตทริบิวต์เพิ่มเติมแต่อย่างใด

- **ตัวเลือกแบบคลาส**

ตัวเลือกแบบคลาส (Class Selector) เป็นการระบุอิลิเมนต์เป้าหมายโดยใช้ค่าของแอตทริบิวต์ class ที่กำหนดไว้ในแท็กของภาษา HTML มาใช้เป็นตัวเลือกเช่น หากมีการกำหนดแอตทริบิวต์ class ให้กับแท็ก p เป็น ... ก็จะใช้ info เป็นชื่อของตัวเลือก เมื่อนำชื่อคลาสมาใช้เป็นตัวเลือกตามองค์ประกอบของชุดกฎ CSS จะต้องมีเครื่องหมาย . (Dot) นำหน้าชื่อตัวเลือก เช่น .info เป็นต้น ทั้งนี้การ

กำหนดชื่อคลาสให้กับแท็กต่าง ๆ เป็นการจัดกลุ่มแท็กเพื่อนำไปสู่การจัดรูปแบบที่มีคุณสมบัติการแสดงผลเหมือนกัน โดยชื่อคลาสในแต่ละแท็กสามารถตั้งชื่อซ้ำกันได้เพื่อแสดงถึงการเป็นกลุ่มแท็กที่จะมีคุณลักษณะการแสดงผลลักษณะเดียวกัน

- **ตัวเลือกแบบรหัสประจำอิลิเมนต์**

ตัวเลือกแบบรหัสประจำอิลิเมนต์ (ID Selector) เป็นการระบุอิลิเมนต์เป้าหมายโดยใช้ค่าของแอตทริบิวต์ id ที่กำหนดไว้ในแท็กของภาษา HTML โดยค่าของ id นี้ทำหน้าที่เป็นรหัสประจำอิลิเมนต์ซึ่งต้องไม่ซ้ำกันในเอกสารเว็บ เช่น หากมีการกำหนด id ให้กับแท็ก p เป็น ... จะถือว่า result เป็นรหัสประจำอิลิเมนต์ที่อิลิเมนต์อื่นไม่สามารถนำไปตั้งซ้ำได้อีก เมื่อนำรหัสประจำอิลิเมนต์มาใช้เป็นตัวเลือกในชุดกฎของ CSS จะต้องมีเครื่องหมาย # (Hash) นำหน้า เช่น #result เป็นต้น ทั้งนี้เนื่องจากรหัสประจำอิลิเมนต์ไม่สามารถซ้ำกันได้ในเอกสารเว็บหนึ่งๆ จึงเหมาะกับการควบคุมการแสดงผลของอิลิเมนต์เฉพาะรายที่มีความสำคัญหรือมีบทบาทเฉพาะเจาะจงในหน้าเว็บ

จากตัวเลือกชนิดต่าง ๆ ที่ได้กล่าวถึงข้างต้น สามารถสรุปเปรียบเทียบลักษณะการใช้งานร่วมกับแท็ก HTML ได้ดังตารางที่ 7-1 ซึ่งแสดงให้เห็นรูปแบบของการระบุตัวเลือกในแต่ละประเภท พร้อมตัวอย่างการใช้งานที่สอดคล้อง

ตารางที่ 7-1 เปรียบเทียบลักษณะการใช้ตัวเลือกของ CSS

ชนิดของตัวเลือก	ตัวอย่างแท็กในภาษา HTML	ตัวเลือกที่ใช้ใน CSS	ลักษณะการใช้งาน
Tag Selector	เนื้อหา	p	ใช้กับทุกแท็กที่ตรงกับตัวเลือก
Class Selector	เนื้อหา	.info	ใช้กับแท็กที่มี class เดียวกัน
ID Selector	เนื้อหา	#result	ใช้กับแท็กที่มี id เฉพาะเจาะจง

2) ส่วนประกาศ

ส่วนประกาศ (Declaration Block) เป็นองค์ประกอบที่ใช้ระบุรายการคุณลักษณะการแสดงผลให้กับอิลิเมนต์เป้าหมายที่ถูกเลือกโดยตัวเลือกซึ่งจะเขียนระหว่างเครื่องหมาย { และ } ซึ่งใช้แสดงจุดเริ่มต้นและจุดสิ้นสุดของชุดกฎนั้นๆ โดยภายในส่วนนี้จะประกอบด้วยชุดของคุณสมบัติและค่าของคุณสมบัตินั้น ๆ ซึ่งเขียนในรูปแบบ property: value; ตามที่ได้อธิบายไว้ในหัวข้อที่ 2.1 โดยแต่ละคุณสมบัติจะแยกจากกันด้วยเครื่องหมาย ; และสามารถเขียนต่อกันในหลายบรรทัดหรือบรรทัดเดียว ทั้งนี้ค่าที่กำหนดในส่วนประกาศจะเป็นตัวควบคุมลักษณะการแสดงผลของอิลิเมนต์ตามที่กำหนดไว้ในชุดกฎของ CSS อย่างเป็นระบบ

3.2.2 รูปแบบการเขียนชุดกฎสำหรับตัวเลือกแบบแท็ก

การใช้ตัวเลือกแบบแท็กเป็นวิธีที่ตรงไปตรงมาและเป็นพื้นฐานในการกำหนดรูปแบบการแสดงผลให้กับอิลิเมนต์ในเอกสารเว็บโดยไม่ต้องกำหนดแอตทริบิวต์เพิ่มเติมใดๆ ในแท็ก HTML เพียงแค่ระบุชื่อของแท็กที่ต้องการควบคุม เช่น p, h1, div เป็นต้น ตัวเลือกชนิดนี้จะมีผลกับทุกอิลิเมนต์ในหน้าเว็บที่ตรงกับชื่อแท็กที่กำหนดไว้ รูปแบบและตัวอย่างของการเขียนชุดกฎสำหรับตัวเลือกแบบแท็กเพื่อใช้งานร่วมกับเอกสารเว็บ มีดังนี้

รูปแบบ

```
ชื่อแท็ก {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
    ...
    ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}
```

ตัวอย่างที่ 4

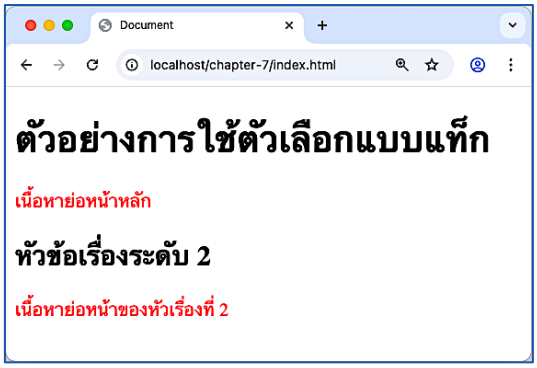
```
1   <head>
2       <style>
3           p {
4            color: red;
5           }
6       </style>
7   </head>
8   <body>
9       <h1>ตัวอย่างการใช้ตัวเลือกแบบแท็ก</h1>
10      <p>เนื้อหาย่อหน้าหลัก</p>
11      <div>
12          <h2>หัวข้อเรื่องรัดบ 2</h2>
13          <p>เนื้อหาย่อหน้าของหัวเรื่องที่ 2</p>
14      </div>
15  </body>
```

จากตัวอย่างที่ 4 มีการกำหนดชุดกฎของ CSS ไว้ภายในแท็ก <style> ซึ่งอยู่ภายในอิลิเมนต์ head ของเอกสาร HTML โดยชุดกฎที่ใช้กำหนดลักษณะการแสดงผลมีเพียงหนึ่งชุด คือการใช้ตัวเลือก

แบบแท็กโดยใช้ p เป็นตัวเลือก ซึ่งหมายถึงให้นำกฎที่ระบุในส่วนประกาศไปใช้กับอิลิเมนต์ p ทุกอิลิเมนต์ในเอกสาร

ในส่วนประกาศของชุดกฎดังกล่าวมีการกำหนดคุณสมบัติการแสดงผลเพียงรายการเดียว คือ color: red; ซึ่งมีผลให้ข้อความภายในอิลิเมนต์ p ทั้งหมดแสดงผลเป็นตัวอักษรสีแดง โดยไม่ส่งผลใด ๆ กับอิลิเมนต์อื่น เช่น h1, h2 หรือ div เนื่องจากไม่ได้ระบุให้เป็นตัวเลือกในกฎ CSS

ผลลัพธ์ของตัวอย่างที่ 4



จากผลลัพธ์ของตัวอย่างที่ 4 ที่ปรากฏบนเว็บเบราว์เซอร์ จะพบว่าข้อความ *เนื้อหาย่อหน้าหลัก* และ *เนื้อหาย่อหน้าของหัวเรื่องที่ 2* ซึ่งอยู่ในอิลิเมนต์ p (บรรทัดที่ 10 และ 13 ตามลำดับ) แสดงผลเป็นสีแดงตามคุณสมบัติที่กำหนดไว้ในส่วนประกาศของชุดกฎ CSS โดยไม่ขึ้นกับตำแหน่งของอิลิเมนต์ในโครงสร้าง DOM Tree ขณะที่ลิเมนต์อื่น เช่น h1 และ h2 ยังคงแสดงผลตามค่าเริ่มต้นของเว็บเบราว์เซอร์เนื่องจากไม่ได้ถูกกำหนดให้เป็นอิลิเมนต์เป้าหมายของตัวเลือกในชุดกฎดังกล่าว

3.2.3 รูปแบบการเขียนชุดกฎสำหรับตัวเลือกแบบคลาส

การเขียนชุดกฎของ CSS โดยใช้ตัวเลือกแบบคลาสเป็นวิธีที่ช่วยให้สามารถกำหนดรูปแบบการแสดงผลให้อิลิเมนต์หลายรายการได้พร้อมกัน โดยไม่จำเป็นต้องอ้างอิงตามชื่อแท็ก HTML วิธีนี้เหมาะสำหรับการจัดกลุ่มอิลิเมนต์ที่ต้องการให้แสดงผลในลักษณะเดียวกัน แม้จะเป็นคนละประเภทของแท็กก็ตาม โดยอาศัยการกำหนดแอตทริบิวต์ class ภายในแท็ก HTML ซึ่งจะต้องใช้ชื่อเดียวกัน และในชุดกฎ CSS จะใช้ชื่อคลาสนั้นโดยมีเครื่องหมายจุด (.) นำหน้าเพื่อระบุตัวเลือก รูปแบบและตัวอย่างของการเขียนชุดกฎสำหรับตัวเลือกแบบคลาสเพื่อใช้งานร่วมกับเอกสารเว็บ มีดังนี้

รูปแบบ

```
.ชื่อคลาส {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
```

...

ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;

}

ตัวอย่างที่ 5

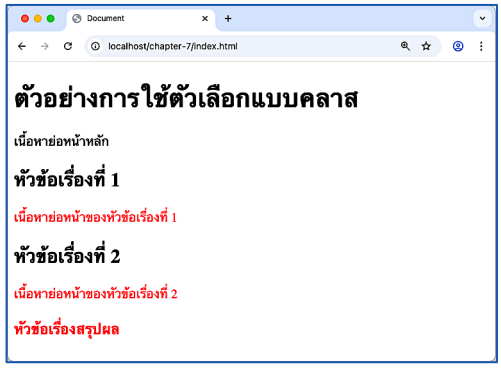
```
1   <head>
2       <style>
3           .result {
4           color: red;
5           }
6       </style>
7   </head>
8   <body>
9       <h1>ตัวอย่างการใช้ตัวเลือกแบบคลาส</h1>
10      <p class="intro">เนื้อหาย่อหน้าหลัก</p>
11      <div>
12          <h2>หัวข้อเรื่องที่ 1</h2>
13          <p class="result">เนื้อหาย่อหน้าของหัวข้อเรื่องที่ 1</p>
14      </div>
15      <div>
16          <h2>หัวข้อเรื่องที่ 2</h2>
17          <p class="result">เนื้อหาย่อหน้าของหัวข้อเรื่องที่ 2</p>
18      </div>
19      <h3 class="result">หัวข้อเรื่องสรุปผล</h3>
20  </body>
```

จากตัวอย่างที่ 5 มีการกำหนดชุดกฎของ CSS ไว้ภายในแท็ก <style> ซึ่งอยู่ภายในอิลิเมนต์ head ของเอกสาร HTML โดยชุดกฎที่ใช้กำหนดลักษณะการแสดงผลมีเพียงหนึ่งชุดคือการใช้ตัวเลือกแบบคลาสโดยใช้ .result เป็นตัวเลือก ซึ่งหมายถึงให้นำกฎที่ระบุในส่วนประกาศไปใช้กับอิลิเมนต์ใดๆ ที่มีการกำหนด class="result" ในส่วนประกาศของชุดกฎ CSS มีการกำหนดคุณสมบัติการแสดงผลเพียงรายการเดียว คือ color: red; เป็นการทำให้ตัวอักษรเป็นสีแดง

ผลลัพธ์ของตัวอย่างที่ 5 ที่ปรากฏบนเว็บเบราว์เซอร์ จะพบว่า ข้อความที่อยู่ภายในอิลิเมนต์ที่มีคลาสเป็น result จะแสดงผลเป็นตัวอักษรสีแดงทั้งหมด ขณะที่อิลิเมนต์อื่น ๆ เช่น h1, h2 และ p ซึ่งไม่ได้การ

กำหนด class เป็น result ยังคงแสดงผลด้วยลักษณะมาตรฐานของเว็บเบราว์เซอร์ เนื่องจากไม่ได้ถูกกำหนดให้เป็นอิลิเมนต์เป้าหมายของตัวเลือกในชุดกฎ CSS ดังกล่าว

ผลลัพธ์ของตัวอย่างที่ 5



3.2.4 รูปแบบการเขียนชุดกฎสำหรับตัวเลือกแบบรหัสประจำอิลิเมนต์

การเขียนชุดกฎของ CSS โดยใช้ตัวเลือกแบบรหัสประจำอิลิเมนต์ เป็นวิธีที่ใช้สำหรับกำหนดรูปแบบเฉพาะเจาะจงให้กับอิลิเมนต์ใดอิลิเมนต์หนึ่งในเอกสารเว็บซึ่งอาศัยการกำหนดแอตทริบิวต์ id ให้กับแท็ก HTML เพื่อทำหน้าที่เป็นรหัสเฉพาะที่ไม่สามารถซ้ำกันได้ในแต่ละหน้าเว็บ โดยในชุดกฎ CSS จะใช้ชื่อ id ดังกล่าวเป็นตัวเลือก พร้อมเครื่องหมาย # นำหน้าเพื่อระบุว่าเป็นตัวเลือกแบบรหัสประจำอิลิเมนต์ รูปแบบและตัวอย่างของการเขียนชุดกฎสำหรับตัวเลือกชนิดดังกล่าวเพื่อใช้งานร่วมกับเอกสารเว็บ มีดังนี้

รูปแบบ

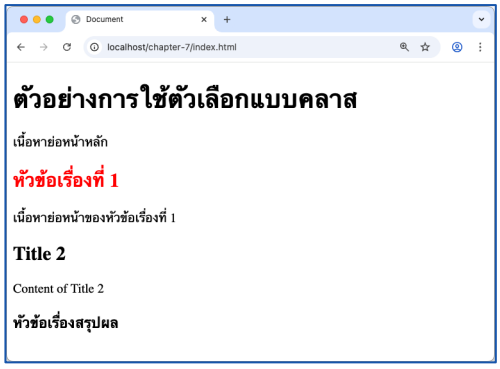
```
.ชื่อคลาส {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
    ...
    ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}
```

ตัวอย่างที่ 6

```
1  <head>
2      <style>
3          #thai {
4           color: red;
5          }
```

```
6       </style>
7   </head>
8   <body>
9       <h1 id="chapter">ตัวอย่างการใช้ตัวเลือกแบบคลาส</h1>
10      <p>เนื้อหาย่อหน้าหลัก</p>
11      <div>
12          <h2 id="thai">หัวข้อเรื่องที่ 1</h2>
13          <p>เนื้อหาย่อหน้าของหัวข้อเรื่องที่ 1</p>
14      </div>
15      <div>
16          <h2 id="eng">Title 2</h2>
17          <p>Content of Title 2</p>
18      </div>
19      <h3 class="result">หัวข้อเรื่องสรุปผล</h3>
20  </body>
```

ผลลัพธ์ของตัวอย่างที่ 6



จากตัวอย่างที่ 6 มีการกำหนดชุดกฎของ CSS ไว้ภายในแท็ก <style> ซึ่งอยู่ภายในอิลิเมนต์ head ของเอกสาร HTML โดยใช้ตัวเลือกแบบรหัสประจำอิลิเมนต์ คือ #thai เป็นตัวเลือก หมายความว่า ชุดกฎ CSS จะมีผลต่ออิลิเมนต์ที่มีค่า id เท่ากับ thai เท่านั้น ซึ่งในตัวอย่างนี้ อิลิเมนต์ h2 ในบรรทัดที่ 12 มีการกำหนด id เป็น thai ทั้งนี้อิลิเมนต์อื่นๆ ในเอกสารเว็บนี้จะไม่สามารถตั้ง id เป็น thai ได้อีก เนื่องจากรหัสประจำอิลิเมนต์จะต้องไม่ซ้ำกันในเอกสารเว็บ ในส่วนประกาศของชุดกฎ CSS ดังกล่าว มีการกำหนดคุณสมบัติการแสดงผลเพียงรายการเดียว คือ color: red; ซึ่งเป็นการกำหนดให้อิลิเมนต์เป้าหมายแสดงผลเป็นตัวอักษรสีแดง

ผลลัพธ์ของตัวอย่างที่ 6 ที่ปรากฏบนเว็บเบราว์เซอร์ จะพบว่าเฉพาะข้อความในอิลิเมนต์ที่มี id="thai" เท่านั้นที่แสดงผลเป็นตัวอักษรสีแดง คือข้อความ *หัวข้อเรื่องที่ 1* ซึ่งอยู่ภายในอิลิเมนต์ h2 เนื่องจากในชุดกฎ CSS ได้กำหนด #thai เป็นตัวเลือก และกำหนดคุณสมบัติ color: red; ไว้ในส่วนประกาศ โดยตัวเลือก #thai นี้จะส่งผลเฉพาะอิลิเมนต์ที่มีค่า id ตรงกับชื่อที่ระบุเท่านั้น ขณะที่อิลิเมนต์อื่น เช่น h1 ที่มี id="chapter", h2 ที่มี id="eng" และ h3 ที่มี class="result" จะยังคงแสดงผลด้วยค่ามาตรฐานของเว็บเบราว์เซอร์ หรือได้รับผลจากกฎอื่นที่ไม่ได้เกี่ยวข้องกับตัวเลือก #thai ดังกล่าว

3.2.5 รูปแบบการเขียนชุดกฎสำหรับตัวเลือกแบบผสม

รูปแบบการเขียนชุดกฎสำหรับตัวเลือกแบบผสม (Combination Selector) เป็นการเขียนตัวเลือกที่ประกอบด้วยตัวเลือกหลายชนิดร่วมกัน เพื่อระบุอิลิเมนต์เป้าหมายได้อย่างเจาะจงและมีเงื่อนไขมากขึ้น โดยสามารถใช้งานได้หลากหลายรูปแบบ เช่น การเลือกอิลิเมนต์ที่อยู่ภายในอิลิเมนต์อื่น การเลือกอิลิเมนต์ที่มี class และ tag ร่วมกัน หรือการเลือกอิลิเมนต์ที่มีคุณสมบัติหลายประการพร้อมกัน ซึ่งช่วยให้สามารถควบคุมการแสดงผลได้อย่างมีประสิทธิภาพและยืดหยุ่น โดยมีรูปแบบที่ใช้กันทั่วไป ดังนี้

1) ตัวเลือกแบบกลุ่ม

ตัวเลือกแบบกลุ่ม (Group Selector) เป็นรูปแบบของการรวมตัวเลือกหลายตัวไว้ในชุดกฎเดียวกัน โดยใช้เครื่องหมาย , (Comma) เพื่อแบ่งระหว่างตัวเลือกแต่ละตัว จุดมุ่งหมายของการใช้ตัวเลือกแบบกลุ่มคือการกำหนดชุดของคุณสมบัติการแสดงผลที่อยู่ในส่วนประกาศเดียวกันให้กับอิลิเมนต์หลายชนิดพร้อมกัน โดยไม่จำเป็นต้องเขียนชุดกฎ CSS แยกกันสำหรับแต่ละอิลิเมนต์ ซึ่งช่วยลดความซ้ำซ้อนของรหัสคำสั่ง และทำให้สามารถจัดการดูแลได้อย่างเป็นระบบ รูปแบบและตัวอย่างการสร้างชุดกฎ CSS โดยใช้ตัวเลือกแบบกลุ่มมีดังนี้

รูปแบบ

```
ตัวเลือก_1, ตัวเลือก_2, ..., ตัวเลือก_n {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
    ...
    ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}
```

ตัวอย่างที่ 7

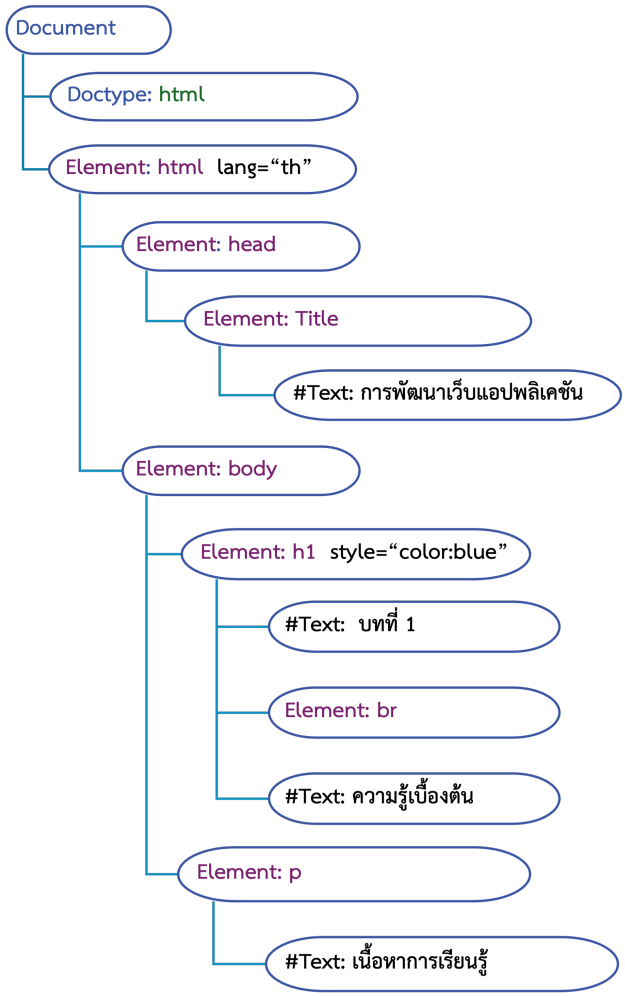
```
1  h1, h2, p, .ans {
2      color: blue;
3  }
```

จากตัวอย่างที่ 7 เป็นการสร้างชุดกฎของ CSS โดยใช้ตัวเลือกแบบกลุ่ม ซึ่งประกอบด้วยตัวเลือกแท็ก ได้แก่ h1, h2, และ p รวมถึงตัวเลือกแบบคลาสคือ .ans ชุดกฎนี้มีการกำหนดคุณสมบัติ color:

blue; ในส่วนประกาศ ซึ่งหมายความว่า อิลิเมนต์ทั้งหมดที่อยู่ภายใต้ตัวเลือกเหล่านี้ไม่ว่าจะเป็นแท็ก h1, h2, p หรืออิลิเมนต์ใด ๆ ที่มีการกำหนดคลาสเป็น ans จะได้รับการแสดงผลเป็นข้อความสีฟ้า

2) ตัวเลือกแบบลูกหลาน

ตัวเลือกแบบลูกหลาน (Descendant Selector) เป็นรูปแบบของการระบุอิลิเมนต์เป้าหมายที่พิจารณาในรูปแบบของความสัมพันธ์เชิงลำดับชั้นของอิลิเมนต์ในโครงสร้าง DOM Tree ที่เกิดจากการเขียนแท็กต่าง ๆ ตามโครงสร้างของภาษา HTML ดังที่ได้กล่าวถึงในบทที่ 1 และบทที่ 5 ทั้งนี้ก่อนที่จะนำเสนอถึงตัวเลือกแบบลูกหลาน จะนำเสนอความสัมพันธ์ระหว่างอิลิเมนต์ในโครงสร้าง DOM Tree โดยใช้ภาพที่ 7.2 ประกอบการอธิบายพอสังเขปดังนี้



ภาพที่ 7.2 ตัวอย่างของ DOM Tree ที่นำมาใช้พิจารณาความสัมพันธ์เชิงลำดับชั้นของอิลิเมนต์

- **อิลิเมนต์บรรพบุรุษ**

อิลิเมนต์บรรพบุรุษ (Ancestor Element) หมายถึงอิลิเมนต์ใด ๆ ที่อยู่เหนืออิลิเมนต์เป้าหมายในโครงสร้าง DOM Tree ไม่ว่าจะมีความลึกของระดับชั้นมากน้อยเพียงใด ยกตัวอย่างจากภาพที่ 7.2 อิลิเมนต์ html ถือเป็นบรรพบุรุษของทุกอิลิเมนต์ในเอกสาร เนื่องจากอยู่ในตำแหน่งบนสุดของโครงสร้าง ในขณะที่อิลิเมนต์ body เป็นบรรพบุรุษของ h1, br และ p แต่ไม่ถือเป็นบรรพบุรุษของ html, head หรือ title เนื่องจากไม่ได้ครอบอิลิเมนต์เหล่านั้นไว้ในสายลำดับชั้นเดียวกัน กล่าวอีกนัยหนึ่ง หากพิจารณาอิลิเมนต์ p เป็นอิลิเมนต์เป้าหมาย อิลิเมนต์ body และ html จะถือเป็นบรรพบุรุษของ p เนื่องจากอยู่ในเส้นทางลำดับชั้นเดียวกันและอยู่ในตำแหน่งที่สูงกว่า

- **อิลิเมนต์แม่**

อิลิเมนต์แม่ (Parent Element) หมายถึงอิลิเมนต์ที่อยู่เหนืออิลิเมนต์เป้าหมายโดยตรงในโครงสร้าง DOM Tree หนึ่งลำดับชั้น หรือกล่าวอีกนัยหนึ่งคือ อิลิเมนต์ที่มีอิลิเมนต์เป้าหมายเป็นลูกของตนเองโดยตรงหนึ่งระดับ ยกตัวอย่างจากภาพที่ 7.2 หากอิลิเมนต์ p เป็นอิลิเมนต์เป้าหมายในการจัดรูปแบบ อิลิเมนต์ body จะถือเป็นอิลิเมนต์แม่ของ p เนื่องจากอยู่ในระดับชั้นที่สูงกว่าเพียงหนึ่งระดับและครอบครอง p โดยตรง ซึ่งแตกต่างจากอิลิเมนต์ html ที่แม้จะอยู่เหนือ p เช่นกัน แต่เนื่องจากมีระยะชั้นมากกว่าหนึ่งระดับ จึงจัดเป็นอิลิเมนต์บรรพบุรุษ ไม่ใช่อิลิเมนต์แม่ตามนิยามของความสัมพันธ์เชิงโครงสร้างภายใน DOM Tree

- **อิลิเมนต์ลูก**

อิลิเมนต์ลูก (Child Element) หมายถึงอิลิเมนต์ที่อยู่ถัดจากอิลิเมนต์แม่ลงมาหนึ่งลำดับชั้นในโครงสร้าง DOM Tree หรือกล่าวอีกนัยหนึ่ง คืออิลิเมนต์ที่ถูกครอบโดยอิลิเมนต์แม่โดยตรง ยกตัวอย่างจากภาพที่ 7.2 หากกำหนดให้ body เป็นอิลิเมนต์แม่ จะพบว่าอิลิเมนต์ h1 และ p เป็นอิลิเมนต์ลูกของ body เนื่องจากถูกครอบโดยอิลิเมนต์ body โดยตรง

- **อิลิเมนต์ลูกหลาน**

อิลิเมนต์ลูกหลาน (Descendant Element) หมายถึงอิลิเมนต์ที่อยู่ภายใต้การครอบของอิลิเมนต์บรรพบุรุษไม่ว่าจะอยู่ห่างกันกี่ระดับชั้นในโครงสร้าง DOM Tree ก็ตาม กล่าวคือ อิลิเมนต์ลูกหลานเป็นอิลิเมนต์ที่อยู่ในสายลำดับของอิลิเมนต์แม่หรือบรรพ-บุรุษโดยอาจเป็นลูกโดยตรงหรืออยู่ในชั้นลึกถัดไป เช่น หลาน หรือเหลน เป็นต้น ยกตัวอย่างจากภาพที่ 7.2 หากอิลิเมนต์ html เป็นอิลิเมนต์บรรพบุรุษ จะพบว่าอิลิเมนต์ head, title, body, h1, br และ p เป็นอิลิเมนต์ลูกหลานของ html ทั้งหมด ในอีกกรณีหนึ่ง หากอิลิเมนต์ body เป็นอิลิเมนต์แม่ อิลิเมนต์ h1, br และ p จะเป็นอิลิเมนต์ลูกหลานในขณะที่ title ไม่ถือเป็นลูกหลานของ body เพราะอยู่ในโครงสร้างสายอื่น

- **อิลิเมนต์พี่น้อง**

อิลิเมนต์พี่น้อง (Sibling Element) หมายถึงอิลิเมนต์ที่อยู่ในระดับชั้นเดียวกันในโครงสร้าง DOM Tree และมีอิลิเมนต์แม่ (Parent Element) ร่วมกัน กล่าวคือ อิลิเมนต์พี่น้องคืออิลิเมนต์ที่ถูกครอบโดยอิลิเมนต์แม่เดียวกันแต่เป็นคนละอิลิเมนต์กัน เช่น ตัวอย่างจากภาพที่ 7.2 หากพิจารณาอิลิเมนต์ head

ซึ่งมีอิลิเมนต์ลูกคือ title จะพบว่า title ไม่มีพี่น้อง ในขณะที่อิลิเมนต์ body ซึ่งมีลูก 2 อิลิเมนต์ คือ h1และ p ทั้งสองอิลิเมนต์นี้จะถือว่าเป็นอิลิเมนต์พี่น้องกันทั้งหมดเพราะมี body เป็นอิลิเมนต์แม่เดียวกัน

จากความสัมพันธ์ระหว่างอิลิเมนต์ตามโครงสร้าง DOM Tree ข้างต้น รูปแบบและตัวอย่างการสร้างชุดกฎ CSS โดยใช้ตัวเลือกแบบลูกหลานมีดังนี้

รูปแบบ

ตัวเลือกบรรพบุรุษ ตัวเลือกลูกหลาน ... {
ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
...
ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}

ตัวอย่างที่ 8

```
1   div  p {
2           color: blue;
3   }
```

จากตัวอย่างที่ 8 เป็นการสร้างชุดกฎของ CSS โดยใช้ตัวเลือกแบบลูกหลาน ซึ่งกำหนดให้อิลิเมนต์ p ทุกอิลิเมนต์ที่มีบรรพบุรุษเป็นอิลิเมนต์ div จะได้รับคุณสมบัติการแสดงผลตามที่ระบุไว้ในส่วนประกาศคือแสดงข้อความด้วยสีน้ำเงิน (color: blue;) อย่างไรก็ตาม อิลิเมนต์ p ไม่มีบรรพบุรุษเป็น div จะไม่ถูกนำมาพิจารณาโดยกฎนี้ แม้จะเป็นแท็ก p เช่นเดียวกัน เนื่องจากไม่ได้เป็นลูกหลานของ div ตามโครงสร้างของ DOM Tree

ตัวอย่างที่ 9

```
1   div  section h2 {
2           color: blue;
3   }
```

จากตัวอย่างที่ 9 เป็นการสร้างชุดกฎของ CSS โดยใช้ตัวเลือกแบบลูกหลาน ซึ่งกำหนดให้อิลิเมนต์ h2 ทุกอิลิเมนต์ที่มีบรรพบุรุษเป็นอิลิเมนต์ section และอิลิเมนต์ section นี้ต้องมีอิลิเมนต์บรรพบุรุษเป็น div ด้วยจึงจะได้รับคุณสมบัติการแสดงผลตามที่ระบุไว้ในส่วนประกาศคือแสดงข้อความด้วยสีน้ำเงิน (color: blue;)

3) ตัวเลือกแบบลูกโดยตรง

ตัวเลือกแบบลูกโดยตรง (Child Selector) เป็นรูปแบบของการระบุอิลิเมนต์เป้าหมายที่อิงจากความสัมพันธ์เชิงลำดับชั้นในโครงสร้าง DOM Tree เช่นเดียวกับตัวเลือกแบบลูกหลานแต่มีความจำเพาะมากกว่า กล่าวคือ ตัวเลือกนี้จะมีผลเฉพาะอิลิเมนต์ที่เป็น ลูกโดยตรงของอิลิเมนต์แม่เท่านั้น โดยใช้เครื่องหมาย > เป็นตัวเชื่อมระหว่างตัวเลือกทั้งสอง รูปแบบและตัวอย่างการสร้างชุดกฎ CSS โดยใช้ตัวเลือกแบบลูกโดยตรงมีดังนี้

รูปแบบ

```
ตัวเลือกระดับแม่ > ตัวเลือกระดับลูก {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
    ...
    ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}
```

ตัวอย่างที่ 10

```
1  .result > p {
2      color: blue;
3  }
```

จากตัวอย่างที่ 10 เป็นการสร้างชุดกฎของ CSS โดยใช้ตัวเลือกแบบลูกโดยตรง ซึ่งกำหนดให้อิลิเมนต์ p ทุกอิลิเมนต์ที่เป็นลูกโดยตรงของอิลิเมนต์ใด ๆ ที่มีคลาสเป็น result จะได้รับคุณสมบัติการแสดงผลตามที่ระบุไว้ในส่วนประกาศ คือแสดงข้อความด้วยสีน้ำเงิน (color: blue;) ทั้งนี้ ตัวเลือกแบบลูกโดยตรงจะไม่มีผลกับอิลิเมนต์ p ที่อยู่ลึกลงไปมากกว่าหนึ่งระดับภายในอิลิเมนต์ที่มีคลาส result ซึ่งแตกต่างจากตัวเลือกแบบลูกหลานที่ครอบคลุมถึงอิลิเมนต์ลูกระดับลึกทั้งหมด

4) ตัวเลือกแบบร่วม

ตัวเลือกแบบร่วม (Compound Selector) เป็นรูปแบบของการรวมตัวเลือกหลายลักษณะไว้ด้วยกันโดยไม่ใช้เครื่องหมายเว้นวรรคหรือเครื่องหมายอื่นขั้นกลาง เช่น การระบุชื่อแท็กพร้อมกับชื่อคลาส หรือชื่อแท็กพร้อมกับรหัสประจำอิลิเมนต์ในรูปแบบที่กระชับยิ่งขึ้น ซึ่งช่วยให้สามารถระบุอิลิเมนต์เป้าหมายได้อย่างเฉพาะเจาะจงมากยิ่งขึ้น โดยตัวเลือกแบบร่วมจะเลือกเฉพาะอิลิเมนต์ที่มีคุณสมบัติตรงกับทุกเงื่อนไขที่ระบุไว้เท่านั้น รูปแบบและตัวอย่างการสร้างชุดกฎ CSS โดยใช้ตัวเลือกแบบร่วมมีดังนี้

รูปแบบที่ 1

```
ชื่อแท็ก.ชื่อคลาส {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
    ...
```

ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;

}

ตัวอย่างที่ 11

```
1   p.result {
2       color: blue;
3   }
```

ในตัวอย่างที่ 11 เป็นการใช้ตัวเลือกแบบร่วมระหว่างชื่อแท็ก p และชื่อคลาส result โดยไม่มีการเว้นวรรคหรือเครื่องหมายใด ๆ คั่นกลาง หมายความว่า เฉพาะอิลิเมนต์ที่เป็นแท็ก p และมีการกำหนด class=“result” เท่านั้น ที่จะได้รับการกำหนดคุณสมบัติการแสดงผลตามที่ระบุในส่วนประกาศคือ color: blue; อิลิเมนต์อื่น ๆ ที่มีเพียงชื่อแท็ก p แต่ไม่มีคลาส result หรือมีคลาส result แต่ไม่ได้ใช้แท็ก p จะ ไม่ได้รับผลจากกฎ CSS ดังกล่าว เนื่องจากไม่ตรงตามเงื่อนไขของตัวเลือกแบบร่วมทุกประการ

รูปแบบที่ 2

ชื่อคลาส_1.ชื่อคลาส_2 {

ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;

...

ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;

}

ตัวอย่างที่ 12

```
1   .result.important {
2       color: blue;
3   }
```

ในตัวอย่างที่ 12 เป็นการสร้างชุดกฎ CSS โดยใช้ตัวเลือกแบบร่วมระหว่างคลาส result และคลาส important ซึ่งหมายความว่า เฉพาะอิลิเมนต์ที่กำหนดทั้ง class=“result” และ class=“important” พร้อมกัน เท่านั้น จึงจะได้รับคุณสมบัติการแสดงผลที่ระบุไว้ในส่วนประกาศของชุดกฎนี้ หากอิลิเมนต์ใดมีเพียงคลาสใดคลาสหนึ่ง เช่น มีเฉพาะ class="result" หรือ class="important" แต่ไม่ครบทั้งสองคลาส ก็จะไม่ถูกเลือกโดยชุดกฎนี้ ดังนั้นการเขียนตัวเลือกในลักษณะนี้จึงเหมาะสำหรับการควบคุมลักษณะการแสดงผลที่ต้องการความเฉพาะเจาะจงยิ่งขึ้น

5) ตัวเลือกแบบสถานะจำเพาะ

ตัวเลือกแบบสถานะจำเพาะ (Pseudo-class Selector) เป็นรูปแบบของตัวเลือกที่ใช้ระบุสถานะจำเพาะของอิลิเมนต์ เช่น สถานะเมื่อผู้ใช้เลื่อนเมาส์ผ่านลิงก์ (hover) สถานะของลิงก์เคยถูกคลิกเพื่อเข้า

เยี่ยมชม (visited) เป็นต้น โดยจะใช้เครื่องหมาย : (Colon) นำหน้าชื่อสถานะ ตัวเลือกแบบนี้ช่วยให้สามารถจัดรูปแบบอิลิเมนต์ตามสถานการณ์การโต้ตอบของผู้ใช้หรือสถานะของอิลิเมนต์นั้น ๆ ได้อย่างยืดหยุ่น รูปแบบและตัวอย่างการสร้างชุดกฎ CSS โดยใช้ตัวเลือกแบบสถานะจำเพาะมีดังนี้

รูปแบบ

```
ตัวเลือก:ชื่อสถานะจำเพาะ {
    ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
    ...
    ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}
```

ตัวอย่างที่ 13

```
1   a {
2       color: red;
3       text-decoration: none;
4   }
5   a:hover {
6       color: blue;
7   }
8   a:visited {
9       color: green;
10  }
```

จากตัวอย่างที่ 13 เป็นการสร้างชุดกฎของ CSS จำนวน 3 ชุด โดยกำหนดคุณสมบัติการแสดงผลให้กับแท็ก <a> ซึ่งเป็นอิลิเมนต์สำหรับการสร้างลิงก์เชื่อมโยงภายในเอกสารเว็บ โดยใช้ ตัวเลือกแบบสถานะจำเพาะ (Pseudo-class Selector) เพื่อแสดงรูปแบบที่แตกต่างกันตามสถานะของอิลิเมนต์ ดังนี้

- ชุดกฎแรก (บรรทัดที่ 1-4) เป็นการกำหนดรูปแบบพื้นฐานของลิงก์ทุกตัวให้แสดงด้วยตัวอักษรสีแดง และไม่มีเส้นใต้ โดยไม่มีเงื่อนไขเกี่ยวกับสถานะของผู้ใช้
- ชุดกฎที่สอง (บรรทัดที่ 5-7) จะมีผลเมื่อผู้ใช้เลื่อนเมาส์ไปวางเหนือลิงก์ (สถานะ hover) ตัวอักษรของลิงก์จะเปลี่ยนเป็นสีน้ำเงิน โดยลักษณะการแสดงผลนี้จะใช้แทนกฎก่อนหน้าในขณะที่เกิดสถานะดังกล่าว

- ชุดกฎที่สาม (บรรทัดที่ 8-10) จะมีผลในกรณีที่ผู้ใช้เคยคลิกลิงก์นั้นไปแล้ว (สถานะ visited) เบราว์เซอร์จะจำประวัติดังกล่าวไว้ และแสดงลิงก์นั้นด้วยตัวอักษรสีเขียว เพื่อให้ผู้ใช้แยกแยะได้ว่าลิงก์ดังกล่าวเคยเยี่ยมชมมาแล้ว

จากตัวอย่างที่ 12 ผลลัพธ์ที่แสดงบนเบราว์เซอร์จะเปลี่ยนแปลงตามสถานะของลิงก์ในแต่ละช่วงเวลา โดยอิลิเมนต์ <a> เดียวกันอาจแสดงสีที่แตกต่างกันตามการโต้ตอบของผู้ใช้ เช่น ก่อนคลิก หลังคลิก หรือเมื่อชี้ด้วยเมาส์ ทั้งนี้ การใช้ตัวเลือกแบบสถานะจำเพาะ ช่วยเพิ่มความยืดหยุ่นในการกำหนดรูปแบบและสร้างประสบการณ์การใช้งานที่ชัดเจนยิ่งขึ้นให้แก่ผู้ใช้ ตัวอย่างของสถานะจำเพาะที่พบบ่อยแสดงดังตารางที่ 7-2

ตารางที่ 7-2 สถานะจำเพาะที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน

ชื่อสถานะ	คำอธิบาย
:hover	สถานะที่เกิดขึ้นเมื่อผู้ใช้เลื่อนเมาส์เหนืออิลิเมนต์
:focus	สถานะที่เกิดขึ้นเมื่ออิลิเมนต์ได้รับโฟกัส เช่น จากการคลิกหรือใช้แป้นพิมพ์
:visited	สถานะที่ใช้กับลิงก์ที่ผู้ใช้เคยคลิกเยี่ยมชมแล้ว
:checked	สถานะที่เกิดขึ้นเมื่อ checkbox หรือ radio button ถูกเลือก
:first-child	สถานะที่ใช้กับอิลิเมนต์ลูกตัวแรกของอิลิเมนต์แม่
:nth-child(n)	สถานะที่ใช้กับอิลิเมนต์ลูกที่อยู่ในลำดับที่ n

6) ตัวเลือกแบบองค์ประกอบเสมือน

ตัวเลือกแบบองค์ประกอบเสมือน (Pseudo-element Selector) เป็นรูปแบบของตัวเลือกที่ใช้เพื่อเข้าถึงและจัดรูปแบบส่วนหนึ่งของอิลิเมนต์ โดยไม่ได้เป็นอิลิเมนต์ที่ปรากฏอยู่จริงในโครงสร้างของ HTML แต่เป็นเสมือนองค์ประกอบที่ถูกแทรกหรือเลือกเฉพาะบางส่วนของอิลิเมนต์ที่มีอยู่ ตัวเลือกแบบนี้จะใช้เครื่องหมาย :: (Double Colon) นำหน้าชื่อองค์ประกอบเสมือน เพื่อให้แตกต่างจากตัวเลือกแบบสถานะจำเพาะซึ่งใช้ : เพียงตัวเดียว

ประโยชน์ของการใช้ตัวเลือกแบบองค์ประกอบเสมือน คือ ช่วยให้สามารถแทรกเนื้อหาเพิ่ม หรือจัดการส่วนใดส่วนหนึ่งของข้อความภายในอิลิเมนต์ได้โดยไม่ต้องเพิ่มแท็ก HTML เพิ่มเติม เช่น การใส่คำขึ้นต้นหรือคำต่อท้ายข้อความ การเลือกเฉพาะตัวอักษรแรกในย่อหน้า หรือการจัดรูปแบบบรรทัดแรก เป็นต้น รูปแบบและตัวอย่างตัวเลือกแบบองค์ประกอบเสมือนมีดังนี้

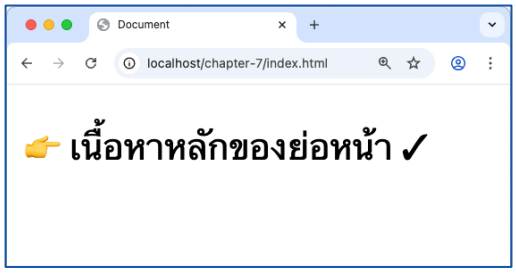
รูปแบบ

```
ตัวเลือก::ชื่อองค์ประกอบเสมือน {
        ชื่อคุณสมบัติที่ 1: ค่าของคุณสมบัติที่ 1;
        ...
        ชื่อคุณสมบัติที่ n: ค่าของคุณสมบัติที่ n;
}
```

ตัวอย่างที่ 14

```
1   <head>
2       <style>
3           p::before {
4           content: "👉 ";
5           }
6           p::after {
7           content: " ✓";
8           }
9       </style>
10  </head>
11  <body>
12      <p>เนื้อหาหลักของย่อหน้า</p>
13  </body>
```

ผลลัพธ์ของตัวอย่างที่ 14



จากตัวอย่างที่ 14 เป็นการกำหนดตัวเลือกแบบองค์ประกอบเสมือน สองตัวเลือกคือ ::before และ ::after ให้กับอิลิเมนต์ p ซึ่งใช้สำหรับแทรกเนื้อหาเพิ่มเติมก่อนและหลังเนื้อหาของอิลิเมนต์ที่เป็นเป้าหมาย โดย ::before จะเพิ่มอักขระลูกศร 👉 ไว้ก่อนข้อความของย่อหน้า ส่วน ::after จะเพิ่มเครื่องหมาย ✓ ต่อท้ายข้อความในอิลิเมนต์ p ทุกตัว ผลลัพธ์ที่แสดงในเว็บเบราว์เซอร์ของตัวอย่างที่ 14 เป็นดังนี้

การกำหนดตัวเลือกแบบองค์ประกอบเสมือนจำเป็นต้องใช้คุณสมบัติ content: เพื่อระบุข้อความหรือสัญลักษณ์ที่ต้องการแทรก ซึ่งเป็นคุณสมบัติ content นี้ใช้ได้กับตัวเลือกแบบองค์ประกอบเสมือนเท่านั้น องค์ประกอบเสมือนที่พบบ่อยแสดงดังตารางที่ 7-3

ตารางที่ 7-3 องค์ประกอบเสมือนที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน

ชื่อองค์ประกอบเสมือน	คำอธิบาย
::before	แทรกเนื้อหาหรือสัญลักษณ์ไว้ก่อนเนื้อหาหลักของอิลิเมนต์
::after	แทรกเนื้อหาหรือสัญลักษณ์ไว้หลังเนื้อหาหลักของอิลิเมนต์
::first-line	เลือกเฉพาะบรรทัดแรกของเนื้อหาภายในอิลิเมนต์
::first-letter	เลือกเฉพาะอักษรตัวแรกของเนื้อหาภายในอิลิเมนต์

3.3 การใช้ CSS แบบภายนอก

การใช้ CSS แบบภายนอก (External CSS) เป็นแนวทางที่เหมาะสมสำหรับการพัฒนาเว็บแอปพลิเคชันที่มีหลายหน้า หรือมีความต้องการในการควบคุมรูปแบบของอิลิเมนต์จำนวนมากในลักษณะที่สามารถดูแลและจัดการได้อย่างเป็นระบบ โดยหลักการของการใช้ CSS แบบภายนอกคือการแยกส่วนการกำหนดรูปแบบการแสดงผลออกจากโครงสร้างเอกสารเว็บ ซึ่งส่งผลให้รหัสคำสั่งในเอกสารเว็บเป็นระเบียบ ง่ายต่อการดูแลรักษา และสามารถปรับปรุงเปลี่ยนแปลงรูปแบบโดยไม่กระทบต่อโครงสร้างเนื้อหาเดิม การใช้ CSS แบบภายนอกมีขั้นตอนดังต่อไปนี้

การใช้ CSS แบบภายนอกยังสนับสนุนแนวคิดสำคัญในการพัฒนาเว็บ คือ แนวคิดการแยกความรับผิดชอบ (Separation of Concerns) ซึ่งเป็นหลักการออกแบบระบบที่มุ่งเน้นให้แต่ละส่วนของระบบทำหน้าที่เฉพาะด้านอย่างชัดเจน โดยในกรณีของการพัฒนาเว็บเพจ หมายถึงการแยก โครงสร้างและเนื้อหา ที่เขียนด้วยภาษา HTML ออกจาก รูปแบบการแสดงผล ที่กำหนดด้วยภาษา CSS โดยไม่ผสมรวมอยู่ในที่เดียวกัน

การแยกหน้าที่ดังกล่าวช่วยให้การเขียนรหัสคำสั่งมีความเป็นระเบียบ ง่ายต่อการทำความเข้าใจ และสามารถปรับเปลี่ยนส่วนใดส่วนหนึ่งของระบบได้โดยไม่กระทบต่อส่วนอื่น เช่น สามารถแก้ไขการจัดรูปแบบในไฟล์ CSS เพียงแห่งเดียว โดยไม่ต้องเข้าไปปรับแต่งในเอกสารเว็บหลายหน้าที่ใช้รูปแบบเดียวกัน ซึ่งเป็นประโยชน์อย่างยิ่งต่อการพัฒนาและดูแลรักษาเว็บแอปพลิเคชันในระยะยาวให้มีความยืดหยุ่น และสามารถขยายต่อได้อย่างมีประสิทธิภาพ การใช้ CSS แบบภายนอกมีขั้นตอนการใช้งานดังนี้

ขั้นตอนที่ 1) สร้างไฟล์ใหม่ที่มีนามสกุล .css เช่น style.css เพื่อบันทึกชุดกฎของ CSS ไว้ในไฟล์ที่สร้างใหม่นี้ โดยภายในไฟล์ CSS นี้จะต้องประกอบด้วยชุดกฎของ CSS เท่านั้น ไม่ต้องมีแท็กใดๆ ของภาษา HTML

ขั้นตอนที่ 2) เขียนชุดกฎของ CSS ที่ต้องการในไฟล์ที่ได้สร้างในขั้นตอนที่ 1 โดยใช้รูปแบบการเขียนเช่นเดียวกับที่ได้อธิบายไว้ในหัวข้อเกี่ยวกับ CSS แบบภายในก่อนหน้านี้

ขั้นตอนที่3) เชื่อมโยงไฟล์ CSS เข้ากับไฟล์เอกสารเว็บ ซึ่งการเชื่อมโยงนี้จะทำในไฟล์เอกสารเว็บโดยใช้แท็ก <link> กำหนดไว้ในอิลิเมนต์ head ดังที่ได้กล่าวมาแล้วในบทที่ 5 โดยมีรูปแบบดังนี้

รูปแบบ

```
<link rel="stylesheet" href="ชื่อไฟล์.css">
```

การเชื่อมโยงในลักษณะนี้จะทำให้เว็บเบราว์เซอร์สามารถโหลดคำสั่ง CSS จากไฟล์ภายนอกและนำมาปรับใช้กับอิลิเมนต์ในหน้าเว็บโดยอัตโนมัติ ซึ่งช่วยลดความซ้ำซ้อนของรหัสคำสั่งและส่งเสริมให้สามารถเปลี่ยนแปลงรูปแบบการแสดงผลได้ง่าย โดยไม่ต้องแก้ไข HTML หลายหน้า

4. สีและหน่วยวัด

การกำหนดลักษณะการแสดงผลด้วย CSS ไม่ได้มีเพียงแค่การเลือกอิลิเมนต์เป้าหมายเท่านั้น แต่ยังต้องอาศัยการกำหนดค่าคุณสมบัติที่เหมาะสมเพื่อควบคุมลักษณะต่าง ๆ ของอิลิเมนต์ ซึ่งหนึ่งในกลุ่มคุณสมบัติที่ใช้บ่อยในการพัฒนาเว็บแอปพลิเคชันคือสีและหน่วยวัดที่ใช้กำหนดขนาด พื้นที่ และระยะห่างในหน้าเว็บ

4.1 การกำหนดค่าสีใน CSS

การกำหนดสีในภาษา CSS เป็นกระบวนการแปลงแนวคิดเชิงนามธรรมของสีให้กลายเป็นข้อมูลดิจิทัลที่คอมพิวเตอร์สามารถเข้าใจและประมวลผลได้ โดยใช้แบบจำลองสี (Color Model) ซึ่งเป็นระบบทางคณิตศาสตร์ที่ใช้ค่าตัวเลขแทนองค์ประกอบของแม่สีที่ใช้แสดงผลในจอคอมพิวเตอร์ CSS รองรับแบบจำลองสีในหลายมาตรฐาน เช่น RGB (Red-Green-Blue), HSL (Hue-Saturation-Lightness), HWB (Hue-Whiteness-Blackness) และ Lab (CIELAB) เป็นต้น (Mozilla Developer Network, 2024) สำหรับการพัฒนาเว็บแอปพลิเคชัน การกำหนดค่าสีมักใช้แบบจำลอง RGB เป็นหลัก ซึ่งสามารถกำหนดได้ 4 วิธี ดังนี้

4.1.1 การกำหนดค่าสีโดยใช้ชื่อสี

การกำหนดค่าสีโดยใช้ชื่อสี (Color Name) เป็นวิธีการที่ง่ายและสะดวกในการเลือกใช้ค่าสีโดยผู้พัฒนาสามารถระบุชื่อของสีเป็นภาษาอังกฤษตามที่มาตรฐาน CSS กำหนดไว้ได้โดยตรง เช่น red, blue, black, white และ gray เป็นต้น วิธีนี้ไม่จำเป็นต้องคำนวณหรือกำหนดค่าของแม่สีในรูปแบบตัวเลขเหมือนในวิธีการอื่น จึงเหมาะกับการใช้งานเบื้องต้นหรือการทดลองออกแบบหน้าเว็บอย่างรวดเร็ว ชื่อสีที่สามารถใช้ใน CSS มีมากกว่า 160 ชื่อ ซึ่งได้รับการกำหนดไว้ล่วงหน้าตามมาตรฐานของ W3C และสามารถใช้งานร่วมกับคุณสมบัติที่เกี่ยวข้องกับสี เช่น color, background-color, border-color และอื่น ๆ ได้อย่างหลากหลาย

ตัวอย่างที่ 15

```
1  a {
2      color: red;
3  }
4  h1
5  {
```

```
6       border-color: orange;
7   }
```

จากตัวอย่างที่ 15 เป็นการกำหนดค่าสีให้กับคุณสมบัติการแสดงผลของอิลิเมนต์ต่าง ๆ โดยทุกอิลิเมนต์ a จะแสดงสีของข้อความเป็นสีแดง (red) และทุกอิลิเมนต์ h1 มีการกำหนดสีของเส้นขอบรอบอิลิเมนต์เป็นสีส้ม (orange) เป็นต้น

4.1.2 การกำหนดค่าสีโดยใช้ฟังก์ชัน rgb( )

การกำหนดค่าสีโดยใช้ฟังก์ชัน rgb( ) เป็นวิธีที่อาศัยการระบุค่าความเข้มของแม่สี ประกอบด้วยสีแดง (Red: R) สีเขียว (Green: G) และสีน้ำเงิน (Blue: B) ซึ่งเป็นแม่สีที่ใช้ในการแสดงผลบนจอภาพตามแบบจำลองสี RGB โดยกำหนดค่าเป็นตัวเลขจำนวนเต็มในระบบเลขฐานสิบ (Decimal) ซึ่งเป็นระบบตัวเลขที่ใช้กันทั่วไปใน จากนั้นเว็บเบราว์เซอร์จะนำค่าความเข้มของแต่ละแม่สีที่ระบุในฟังก์ชันไปประมวลผลในการรวมแม่สีเพื่อแสดงผลบนจอภาพ รูปแบบของฟังก์ชัน RGB มีลักษณะดังนี้

รูปแบบ

rgb(ระดับความเข้มของสีแดง, ระดับความเข้มของสีเขียว, ระดับความเข้มของสีน้ำเงิน)

จากรูปแบบการกำหนดค่าสีโดยใช้ฟังก์ชัน RGB ข้างต้น แต่ละค่าที่ระบุในฟังก์ชันจะอยู่ในช่วงตั้งแต่ 0 ถึง 255 ซึ่งใช้แทนระดับความเข้มของแม่สีแต่ละสี โดยค่า 0 หมายถึงไม่มีความเข้มเลย (ไม่ปรากฏสีดังกล่าว) และค่า 255 หมายถึงความเข้มสูงสุด (ปรากฏสีดังกล่าวเต็มที่) เช่น rgb(0, 0, 0) หมายถึงสีดำสนิท (ไม่มีแม่สีใดปรากฏ) ส่วน rgb(255, 255, 255) หมายถึงสีขาว (ประกอบด้วยแม่สีทั้งสามในระดับสูงสุด) การกำหนดค่าสีด้วยฟังก์ชัน RGB จึงเหมาะสำหรับผู้ที่ต้องการควบคุมความเข้มของแต่ละแม่สีอย่างละเอียด และสามารถประยุกต์ใช้ในการปรับระดับสีต่าง ๆ ได้อย่างหลากหลาย ทั้งในเชิงการออกแบบและการสร้างความน่าสนใจให้กับเว็บแอปพลิเคชัน

ตัวอย่างที่ 16

```
1   a {
2       color: rgb(255, 0 ,0);
3   }
```

จากตัวอย่างที่ 16 เป็นการกำหนดค่าสีให้กับคุณสมบัติการแสดงผลของอิลิเมนต์ a โดยใช้ฟังก์ชัน rgb( ) ซึ่งมีการกำหนดค่าความเข้มของสีแดงไว้ที่ระดับสูงสุด (255) ขณะที่แม่สีเขียวและน้ำเงินถูกกำหนดให้มีค่าความเข้มเป็น 0 ทั้งคู่ ส่งผลให้ข้อความของอิลิเมนต์ a แสดงผลเป็นสีแดงบนเว็บเบราว์เซอร์

4.1.3 การกำหนดค่าสีโดยใช้ฟังก์ชัน rgba( )

การกำหนดค่าสีโดยใช้ฟังก์ชัน rgba( ) เป็นรูปแบบการกำหนดค่าสีที่พัฒนาต่อจากฟังก์ชัน RGB โดยเพิ่มองค์ประกอบที่สี่คือ ค่าความโปร่งใส (Transparency) หรือที่เรียกว่า ค่าอัลฟา (Alpha) เข้ามาเป็นพารามิเตอร์เพิ่มเติม ทำให้สามารถควบคุมระดับความโปร่งใสของสีที่แสดงผลบนหน้าเว็บได้ โดยมีรูปแบบการใช้ฟังก์ชันดังนี้

รูปแบบ

rgba(ค่าความเข้มของสีแดง, ค่าความเข้มของสีเขียว, ค่าความเข้มของสีน้ำเงิน, ค่าความโปร่งใส)

รูปแบบของฟังก์ชัน rgba( ) ใช้แม่สีหลักตามแบบจำลองสี RGB ได้แก่ สีแดง (Red) สีเขียว (Green) และสีน้ำเงิน (Blue) โดยใช้ค่าความเข้มของสีในช่วง 0 ถึง 255 ตามระบบเลขฐานสิบเช่นเดียวกับฟังก์ชัน rgb( ) แต่จะเพิ่มค่าพารามิเตอร์ที่สี่ของฟังก์ชันซึ่งเป็นค่าความโปร่งใส ที่อยู่ในช่วง 0.0 ถึง 1.0 โดยค่า 0.0 หมายถึงโปร่งใส (มองทะลุไปยังอิลิเมนต์ที่อยู่ด้านล่าง) และค่า 1.0 หมายถึงทึบแสง (ไม่โปร่งใส ไม่สามารถมองเห็นอิลิเมนต์ที่อยู่ด้านล่าง) การใช้ฟังก์ชัน rgba() เหมาะสำหรับการออกแบบส่วนติดต่อผู้ใช้ (UI) ที่มีการซ้อนทับขององค์ประกอบ เช่น กล่องข้อความโปร่งใสที่อยู่เหนือภาพพื้นหลัง หรือการกำหนดเงา เอฟเฟกต์แสง และองค์ประกอบที่ต้องการการมองเห็นแบบบางส่วน

ตัวอย่างที่ 17

```
1  div {
2      background-color: rgba(0, 0, 0, 0.3);
3  }
```

จากตัวอย่างที่ 17 เป็นการกำหนดสีพื้นหลังของอิลิเมนต์ div โดยใช้ฟังก์ชัน rgba( ) ซึ่งกำหนดให้มีค่าแม่สีแดง เขียว และน้ำเงิน เท่ากับ 0 ทั้งหมด ทำให้ได้พื้นหลังของอิลิเมนต์เป็นสีดำ และกำหนดค่าความโปร่งใส (Alpha) เท่ากับ 0.3 ส่งผลให้พื้นหลังของอิลิเมนต์ div แสดงเป็นสีดำที่โปร่งใสบางส่วน สามารถมองเห็นเนื้อหาหรือภาพที่อยู่ด้านล่างได้ ทำให้เกิดผลลัพธ์ที่ดูนุ่มนวลและกลมกลืนในการออกแบบหน้าเว็บ

4.1.4 การกำหนดค่าสีโดยใช้เลขฐานสิบหก

การกำหนดค่าสีโดยใช้เลขฐานสิบหก (Hexadecimal Color Code) เป็นวิธีระบุค่าความเข้มของแม่สีตามแบบจำลองสี RGB ประกอบด้วย สีแดง สีเขียว และสีน้ำเงิน เช่นเดียวกับฟังก์ชัน rgb( ) แต่จะใช้ระบบตัวเลขฐานสิบหก (Hexadecimal) แทนค่าความเข้มของแม่สีหลักทั้งสาม โดยมีรูปแบบดังนี้

รูปแบบ

#RRGGBB

จากรูปแบบการกำหนดค่าสีโดยใช้เลขฐานสิบหกข้างต้น ตัวอักษร RR GG และ BB ใช้แทนค่าความเข้มของสีแดง สีเขียวและสีน้ำเงินตามลำดับ โดยค่าทั้งสามเป็นตัวเลขในระบบเลขฐานสิบหกจำนวน 2 หลัก ซึ่งแปลงมาจากค่าความเข้มของแม่สีในระบบเลขฐานสิบในช่วง 0-255 ซึ่งมีค่าเท่ากับ 00-FF ในระบบเลขฐานสิบหก กล่าวคือ ค่า 00 หมายถึงไม่มีความเข้มของสี ส่วนค่า FF มีค่าความเข้มของสีในระดับสูงสุด (เท่ากับ 255 ในระบบเลขฐานสิบ) เช่น #FF0000 หมายถึงสีแดง (R=255, G=0, B=0) หรือ #000000 หมายถึงสีดำ (R=0, G=0, B=0) และ #FFFFFF หมายถึงสีขาว (R=255, G=255, B=255) เป็นต้น

ในบางกรณี หากแต่ละคู่ของรหัสสีมีค่าซ้ำกัน เช่น #FF, #AA หรือ #00 สามารถย่อรหัสสีให้เหลือเพียง 3 หลักได้ โดยใช้รูปแบบ #RGB เช่น #F00 ย่อมาจาก #FF0000 ซึ่งแสดงสีแดง หรือ #0F0 ย่อมาจาก #00FF00 ซึ่งแสดงสีเขียวสด วิธีการนี้ช่วยให้การเขียนรหัสคำสั่งสั้นและอ่านง่ายขึ้น แต่ควรระมัดระวังในกรณีที่ต้องการค่าที่มีความละเอียดสูง

ตัวอย่างที่ 18

```
1  a {
2      color: #FF0000;
3  }
4  h1 {
5      border-color: #FFA500;
6  }
```

จากตัวอย่างที่ 18 เป็นการกำหนดค่าสีให้กับคุณสมบัติการแสดงผลของอิลิเมนต์โดยใช้เลขฐานสิบหก อิลิเมนต์ a มีการกำหนดข้อความให้แสดงเป็นสีแดง (#FF0000) ซึ่งเทียบเท่ากับ rgb(255, 0, 0) ส่วนอิลิเมนต์ h1 มีการกำหนดสีของเส้นขอบเป็นสีส้ม (#FFA500) ซึ่งเป็นสีที่ประกอบด้วยแม่สีแดงระดับสูงสุด (FF), สีเขียวระดับกลาง (A5) และไม่มีสีน้ำเงิน (00)

4.2 การใช้หน่วยวัดใน CSS

หน่วยวัด (Units) ใน CSS มีบทบาทสำคัญในการกำหนดขนาดขององค์ประกอบต่าง ๆ บนหน้าเว็บ เช่น ความกว้าง ความสูง ระยะห่าง ช่องไฟ ขนาดตัวอักษร หรือเส้นขอบ เป็นต้น โดยช่วยให้สามารถควบคุมลักษณะการแสดงผลได้อย่างเหมาะสมกับการใช้งานในบริบทต่าง ๆ ไม่ว่าจะเป็นจอคอมพิวเตอร์ แท็บเล็ต หรืออุปกรณ์พกพา หน่วยวัดใน CSS แบ่งออกได้เป็น 2 ประเภทหลัก ได้แก่ หน่วยวัดสัมบูรณ์ (Absolute units) และหน่วยวัดสัมพัทธ์ (Relative units) โดยแต่ละประเภทมีลักษณะการใช้งานและความเหมาะสมที่แตกต่างกัน ดังนี้

4.2.1 หน่วยวัดสัมบูรณ์

หน่วยวัดสัมบูรณ์ (Absolute Units) เป็นหน่วยวัดที่มีขนาดคงที่แน่นอน ไม่ขึ้นอยู่กับบริบทของอุปกรณ์หรือองค์ประกอบอื่นในหน้าเว็บ เช่น ความละเอียดหน้าจอหรือขนาดแบบอักษรของผู้ใช้ เหมาะกับ

การออกแบบที่ต้องการความเที่ยงตรงสูง เช่น การจัดวางองค์ประกอบเฉพาะจุด หรือสื่อสิ่งพิมพ์ เป็นต้น ตัวอย่างหน่วยวัดสัมบูรณ์แสดงดังตารางที่ 7-4

ตารางที่ 7-4 ตัวอย่างหน่วยวัดสัมบูรณ์ใน CSS

หน่วย	ความหมายของหน่วย	ลักษณะการใช้งาน
px	พิกเซล (Pixel)	ใช้กำหนดขนาดทั่วไปขององค์ประกอบในเว็บ
cm	เซนติเมตร (Centimeter)	ใช้กับงานที่ต้องการมาตราวัดจริง
mm	มิลลิเมตร (Millimeter)	ใช้กำหนดขนาดในหน่วยเล็กที่แม่นยำ
in	นิ้ว (Inch)	นิยมในงานพิมพ์หรืองานกราฟิก
pt	พอยต์ (Point) = 1/72 นิ้ว	ใช้กำหนดขนาดตัวอักษรแบบสิ่งพิมพ์
pc	พีกา (Pica) = 12 พอยต์	ใช้ในงานจัดหน้าเอกสาร เช่นนิตยสาร หนังสือ

ตัวอย่างที่ 19

	CSS	HTML
1	<style>	
2	.box-px {	ความกว้าง 200px
3	width: 200px;	
4	height: 100px;	
5	background-color: orange;	
6	}	
7	</style>	

จากตัวอย่างที่ 19 อิลิเมนต์ div ถูกกำหนดให้มีความกว้างเป็น 200 พิกเซล และสูง 100 พิกเซลซึ่งเป็นค่าคงที่ไม่ขึ้นอยู่กับขนาดของหน้าจอหรือองค์ประกอบแม่ ไม่ว่าจะเปิดหน้าเว็บในจอขนาดเล็กหรือใหญ่ อิลิเมนต์ div จะมีความกว้างและความสูงเท่าเดิมทุกครั้ง โดย 1 พิกเซลหมายถึง 1 หน่วยจุดของหน้าจอแสดงผล

การใช้งานลักษณะนี้เหมาะสำหรับองค์ประกอบที่ต้องการความแม่นยำในขนาด เช่น ปุ่มขนาดคงที่ โลโก้ หรือช่องกรอกข้อมูลที่ไม่ต้องปรับตามขนาดอุปกรณ์ อย่างไรก็ตาม หากใช้หน่วย px มากเกินไปในการออกแบบ อาจทำให้การแสดงผลไม่เหมาะสมกับหน้าจอขนาดเล็ก เช่น โทรศัพท์มือถือ หรือแท็บเล็ต เพราะขนาดจะไม่ปรับเปลี่ยนตามหน้าจอ

4.2.2 หน่วยวัดสัมพัทธ์

หน่วยวัดสัมพัทธ์ (Relative Units) เป็นหน่วยวัดที่มีค่าอ้างอิงกับองค์ประกอบแวดล้อม เช่น องค์ประกอบแม่ ขนาดของหน้าจอหรือหน้าต่างแสดงผล (View Port) และขนาดตัวอักษรเริ่มต้น (Root Font

Size) เป็นต้น หน่วยวัดสัมพันธ์ช่วยให้องค์ประกอบมีความยืดหยุ่นตามอุปกรณ์หรือขนาดจอที่แตกต่างกัน เหมาะสำหรับการออกแบบเว็บแบบตอบสนอง ซึ่งสามารถปรับเปลี่ยนขนาดให้เหมาะสมกับอุปกรณ์หลากหลายประเภท ตัวอย่างหน่วยวัดสัมบูรณ์แสดงดังตารางที่ 7-5

ตารางที่ 7-5 ตัวอย่างหน่วยวัดสัมพัทธ์ใน CSS

หน่วย	ความหมายของหน่วย	ลักษณะการใช้งาน
%	ร้อยละขององค์ประกอบแม่	ใช้กำหนดความกว้างหรือระยะห่างแบบยืดหยุ่น
em	อิงกับขนาดตัวอักษรขององค์ประกอบแม่	เหมาะกับการกำหนดขนาดฟอนต์หรือช่องว่างแบบสัมพันธ์
rem	อิงกับขนาดตัวอักษรของ <html>	ใช้กำหนดขนาดทั่วทั้งเว็บให้มีความสม่ำเสมอ
vw	ร้อยละของความกว้างของหน้าต่างแสดงผล	เหมาะกับการกำหนดเลย์เอาต์ที่ปรับตามขนาดหน้าจอ
vh	ร้อยละของความสูงของหน้าต่างแสดงผล	ใช้กำหนดความสูงที่สัมพันธ์กับขนาดจอแบบพลวัต
vmin	ร้อยละของด้านที่สั้นที่สุดของหน้าจอ	เหมาะกับการคงอัตราส่วนในองค์ประกอบที่แสดงบนหลายอุปกรณ์
vmax	ร้อยละของด้านที่ยาวที่สุดของหน้าจอ	ใช้ในการออกแบบที่ต้องการให้ขนาดองค์ประกอบสัมพันธ์กับหน้าจอใหญ่

ตัวอย่างที่ 20

	CSS	HTML
1	<style>	
2	.box-vw {	ความกว้าง 50% ของ viewport
3	width: 50vw;	
4	height: 100px;	
5	background-color: orange;	
6	}	
7	</style>	

จากตัวอย่างที่ 20 อิลิเมนต์ div ถูกกำหนดให้มีความกว้างเป็น 50vw ซึ่งย่อมาจาก 50% ของความกว้างของหน้าจอแสดงผลโดยไม่ขึ้นกับองค์ประกอบแม่เหมือนหน่วย % และไม่ยึดค่าคงที่เหมือนหน่วยวัด px

หน่วย vw เป็นหนึ่งในหน่วยที่สำคัญในการออกแบบเว็บแบบตอบสนอง เพราะช่วยให้องค์ประกอบสามารถปรับขนาดได้อัตโนมัติตามความกว้างของหน้าจอผู้ใช้ เช่น ถ้าเปิดเว็บบนหน้าจอขนาด 1200px อิลิเมนต์นี้จะแสดงผลที่ 600px โดยอัตโนมัติ และหากย่อหน้าจอเหลือ 800px กล่องจะแสดงผลที่ 400px แทนโดยไม่ต้องเขียนรหัสคำสั่งเพิ่มเติม เหมาะสำหรับการสร้างโครงร่างเว็บที่ต้องการการแสดงผลแบบคงสัดส่วนสัมพันธ์กับขนาดพื้นที่การแสดงผล

5. สรุปท้ายบท

Cascading Style Sheets (CSS) เป็นภาษามาตรฐานสำหรับควบคุมลักษณะการแสดงผลขององค์ประกอบ HTML โดยแยกส่วนการจัดรูปแบบ (Presentation Layer) ออกจากส่วนโครงสร้าง (Structure Layer) ของเอกสาร ทำให้การออกแบบเว็บเป็นระบบและง่ายต่อการปรับปรุง CSS ช่วยสร้างการนำเสนอข้อมูลที่สวยงามสอดคล้องกัน และรองรับการใช้งานกับอุปกรณ์ที่หลากหลายตาม

CSS ได้พัฒนาจาก CSS1 และ CSS2 ซึ่งเน้นการจัดรูปแบบพื้นฐาน สู่ CSS Modules เช่น Flexbox และ Grid Layout เพื่อรองรับเทคโนโลยีใหม่ โดยมาตรฐานปัจจุบันเป็นแบบ Living Standard ที่ปรับปรุงต่อเนื่อง การใช้งาน CSS ทำได้หลายรูปแบบ ได้แก่ Inline CSS, Internal CSS และ External CSS สามารถใช้ตัวเลือก (Selector) เช่น Tag, Class และ ID เพื่อกำหนดรูปแบบได้อย่างยืดหยุ่น

การกำหนดค่าสีใน CSS มีทั้งการใช้ชื่อสี (Color Name), แบบจำลองสี RGB และ RGBA รวมถึงรหัสสีแบบเลขฐานสิบหก (Hexadecimal) สำหรับการกำหนดขนาดและสัดส่วนขององค์ประกอบ CSS รองรับหน่วยวัด Absolute Units เช่น px และ Relative Units เช่น %, em, vw เพื่อให้การจัดวางมีความยืดหยุ่นตามอุปกรณ์ต่าง ๆ ซึ่งเป็นพื้นฐานสำคัญในการพัฒนาเว็บแอปพลิเคชันที่ทันสมัย

6. กิจกรรมท้ายบท

6.1 จงอธิบายความหมายและบทบาทของ CSS ในการพัฒนาเว็บแอปพลิเคชัน พร้อมอธิบายว่าแนวทาง CSS Modules ที่กำหนดโดย W3C แตกต่างจากการกำหนดเวอร์ชันแบบเดิม (CSS1, CSS2) อย่างไร?

6.2 โครงสร้างของคำสั่ง CSS ประกอบด้วยส่วนประกอบใดบ้าง? อธิบายความหมายและหน้าที่ของแต่ละส่วนพอสังเขป

6.3 การแทรกคำสั่ง CSS สามารถทำได้กี่วิธี? แต่ละวิธีมีลักษณะอย่างไร? และเหมาะสมกับการใช้งานในกรณีใด? พร้อมระบุลำดับความสำคัญของแต่ละวิธีเมื่อมีการกำหนดคุณสมบัติซ้ำซ้อนกัน

6.4 อธิบายหน้าที่ของ Selector ชนิดต่าง ๆ ดังต่อไปนี้ พร้อมยกตัวอย่างการใช้งานร่วมกับอิลิเมนต์ของเอกสารเว็บ

6.4.1 Type Selector

6.4.2 Class Selector

6.4.3 ID Selector

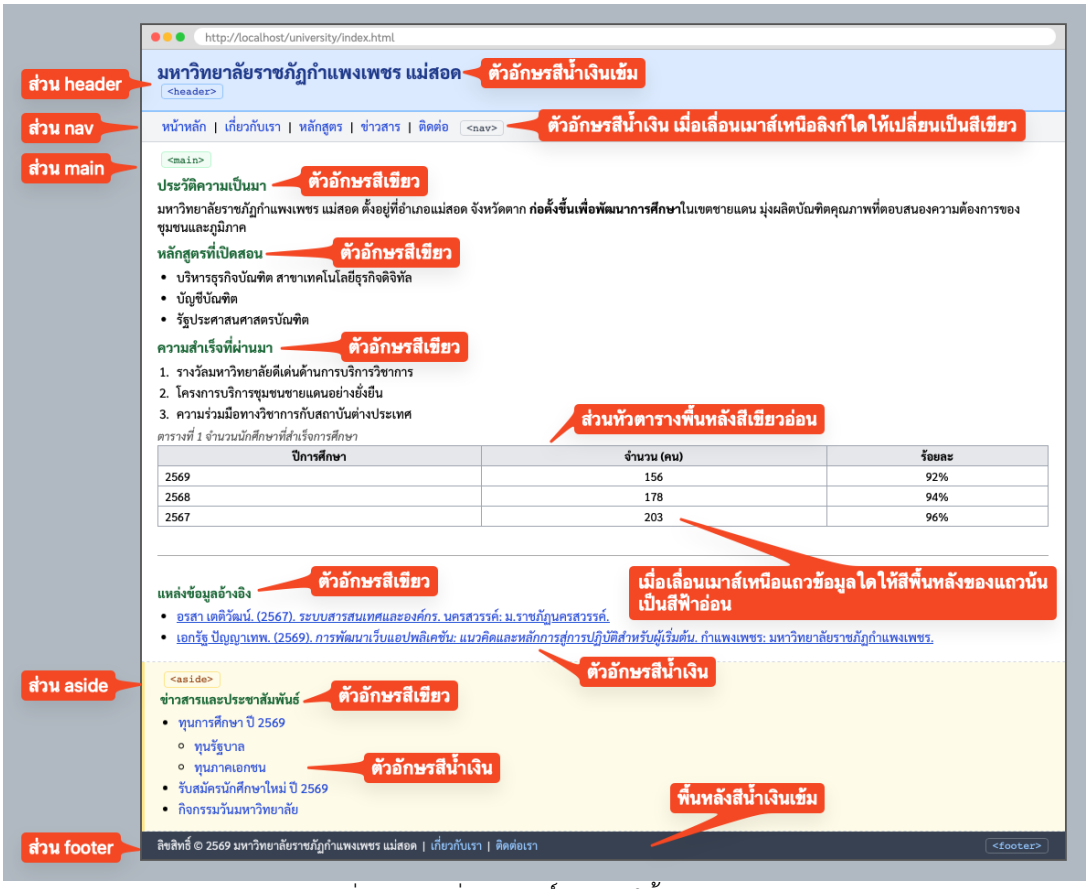
6.4.4 Descendant Selector

6.4.5 Pseudo-class Selector

6.5 หน่วยวัดใน CSS มีแบบใดบ้าง? จงเปรียบเทียบความแตกต่างระหว่างหน่วยวัดแบบสัมบูรณ์และสัมพัทธ์ และหน่วยใดที่เหมาะสมสำหรับการพัฒนาเว็บที่ตอบสนองต่ออุปกรณ์ (Responsive Design)

6.6 การกำหนดสีใน CSS มีรูปแบบใดบ้าง? จงอธิบายความแตกต่างระหว่างการใช้ชื่อสี, รหัส Hex และรูปแบบ RGB พร้อมยกตัวอย่างประกอบ

6.7 จากโครงหน้าเว็บที่สร้างขึ้นจากบทก่อนหน้า จงสร้างไฟล์ CSS ภายนอกชื่อ style.css และเชื่อมโยงกับไฟล์ HTML เพื่อกำหนดลักษณะการแสดงผลของหน้าเว็บให้ใกล้เคียงกับตัวอย่างดังภาพที่ 7.3 โดยมุ่งเน้นการใช้ Selector การกำหนดสี หน่วยวัด และ Pseudo-class Selector ตามที่ศึกษาในบทนี้ โดยมีข้อกำหนดดังนี้



ภาพที่ 7.3 ตัวอย่างผลลัพธ์ของการใช้ CSS ภายนอก

6.7.1 กำหนดส่วนหัวของหน้าเว็บ (header) ให้มี id เป็น site-header และใช้ ID Selector เพื่อกำหนดสีพื้นหลังเป็นสีฟ้าอ่อน ส่วนหัวเรื่องหลัก (h1) ให้ใช้ Type Selector เพื่อกำหนดสีข้อความเป็นสีน้ำเงินเข้ม โดยกำหนดสีด้วยรหัส Hex

6.7.2 กำหนดส่วนนำทาง (nav) ให้มี class เป็น main-menu และใช้ Class Selector เพื่อกำหนดสีพื้นหลังเป็นสีเทาอ่อน พร้อมกำหนด padding เป็น 5px และใช้ Pseudo-class Selector กับลิงก์ทุกลิงก์ เพื่อเปลี่ยนสีข้อความหรือสีพื้นหลังเมื่อผู้ใช้เลื่อนตัวชี้เมาส์ไปวางบนลิงก์

6.7.3 กำหนดพื้นที่ส่วนเนื้อหาหลัก (main) ให้มี class เป็น content และส่วนด้านข้าง (aside) ให้มี id เป็น sidebar จากนั้นใช้ Class Selector และ ID Selector เพื่อกำหนดสีพื้นหลังลักษณะดังภาพ พร้อมกำหนด padding ของทั้งสองส่วนเป็น 5px

6.7.4 กำหนดตารางให้มีความกว้าง 100% และกำหนด padding ของอิลิเมนต์ th และ td เป็น 5px โดยให้หัวตารางมีสีพื้นหลังเป็นสีเทาอ่อนและสีข้อความเป็นสีดำ

6.7.5 ใช้ Pseudo-class Selector เช่น tr:nth-child(even) หรือ tr:nth-child(odd) เพื่อกำหนดสีพื้นหลังของแถวข้อมูลในตารางแบบสลับแถว โดยกำหนดสีด้วยรหัส Hex หรือรูปแบบ RGB

6.7.6 กำหนดส่วนท้ายของหน้าเว็บ (footer) ให้มี class เป็น footer และใช้ Class Selector เพื่อกำหนดสีพื้นหลังเป็นสีเข้ม สีข้อความเป็นสีขาว และกำหนด padding เป็น 5px

6.7.7 ทดลองเปลี่ยนความกว้างของตารางเป็น 50% จากนั้นปรับลดและขยายขนาดของเว็บเบราว์เซอร์แล้วสังเกตผลการเปลี่ยนแปลงของตาราง



การกำหนดลักษณะการแสดงผลและการสร้างองค์ประกอบของเว็บแอปพลิเคชันด้วย CSS

บทที่ 8 มุ่งเน้นการกำหนดลักษณะการแสดงผลของอิลิเมนต์ HTML ด้วยการกำหนดคุณสมบัติต่าง ๆ ด้วย CSS อีกทั้งนำเสนอหลักการของ Flexbox และ CSS Grid ซึ่งเป็นเทคนิคสำหรับการจัดวางโครงร่างหน้าเว็บที่มีความยืดหยุ่นสูง เหมาะสำหรับการออกแบบเว็บที่ตอบสนองต่ออุปกรณ์และขนาดหน้าจอที่หลากหลาย พร้อมทั้งแนะนำการใช้งาน CSS Framework เพื่ออำนวยความสะดวกแก่ผู้พัฒนา หัวข้อหลักในบทนี้ประกอบด้วย

1. คุณสมบัติ CSS พื้นฐานในการจัดรูปแบบอิลิเมนต์
2. หลักการพื้นฐานของการจัดวางโครงร่างด้วย CSS Flexbox และ CSS Grid
3. CSS Framework
4. การใช้งาน Bootstrap สำหรับสร้างองค์ประกอบในเว็บแอปพลิเคชัน
5. สรุปท้ายบท
6. กิจกรรมท้ายบท

1. คุณสมบัติ CSS พื้นฐานในการจัดรูปแบบอิลิเมนต์

การกำหนดคุณสมบัติการแสดงผลของอิลิเมนต์ด้วยภาษา CSS ถือเป็นส่วนสำคัญในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากมีบทบาทโดยตรงต่อรูปลักษณ์ ความสวยงาม การจัดวางองค์ประกอบ และประสบการณ์ของผู้ใช้งาน โดยเฉพาะในการพัฒนาเว็บที่ต้องรองรับอุปกรณ์หลากหลายประเภท การเลือกใช้คุณสมบัติ CSS อย่างเหมาะสมจะช่วยให้สามารถควบคุมลักษณะการแสดงผลของอิลิเมนต์ HTML ได้อย่างมีประสิทธิภาพและยืดหยุ่น หัวข้อนี้มุ่งเน้นการนำเสนอคุณสมบัติ CSS ที่พบบ่อยและมีความสำคัญในการพัฒนาเว็บแอปพลิเคชัน ดังนี้

1.1 คุณสมบัติในการควบคุมขนาดของอิลิเมนต์

การควบคุมขนาดของอิลิเมนต์ใน CSS (Size Property) เป็นพื้นฐานสำคัญในการจัดการพื้นที่การแสดงผลของเว็บแอปพลิเคชัน คุณสมบัติเหล่านี้ช่วยให้ผู้พัฒนาสามารถกำหนดขอบเขตขององค์ประกอบต่าง ๆ ให้เหมาะสมกับบริบทการใช้งาน ประกอบด้วย การกำหนดขนาดคงที่ ขนาดขั้นต่ำ และขนาดสูงสุด โดยสามารถใช้หน่วยวัดพื้นฐานได้ทั้งแบบสัมบูรณ์และสัมพัทธ์ เช่น px, %, em และ rem เป็นต้น คุณสมบัติในการควบคุมขนาดของอิลิเมนต์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชันแสดงดังตารางที่ 8-1

ตารางที่ 8-1 รายการคุณสมบัติในการควบคุมขนาดของอิลิเมนต์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
width	กำหนดความกว้างของอิลิเมนต์	auto หรือค่าตามหน่วยวัดขนาด
height	กำหนดความสูงของอิลิเมนต์	auto หรือค่าตามหน่วยวัดขนาด
min-width	กำหนดความกว้างต่ำสุดของอิลิเมนต์ เพื่อป้องกันไม่ให้ความกว้างน้อยกว่าที่กำหนด	ค่าตามหน่วยวัดขนาด
max-width	กำหนดความกว้างสูงสุดของอิลิเมนต์ เพื่อป้องกันไม่ให้ความกว้างมากกว่าที่กำหนด	ค่าตามหน่วยวัดขนาด
min-height	กำหนดความสูงต่ำสุดของอิลิเมนต์ เพื่อป้องกันไม่ให้ความสูงน้อยกว่าที่กำหนด	ค่าตามหน่วยวัดขนาด
max-height	กำหนดความสูงสูงสุดของอิลิเมนต์ เพื่อป้องกันไม่ให้ความสูงมากกว่าที่กำหนด	ค่าตามหน่วยวัดขนาด

จากคุณสมบัติในการควบคุมขนาดของอิลิเมนต์ที่แสดงในตารางที่ 8-1 สามารถนำมาใช้ร่วมกันเพื่อกำหนดขอบเขตการแสดงผลของอิลิเมนต์ในบริบทต่าง ๆ ได้อย่างยืดหยุ่น โดยตัวอย่างต่อไปนี้แสดงการใช้งานคุณสมบัติกับอิลิเมนต์ div สองรายการ

ตัวอย่างที่ 1

	CSS	HTML
1	.box1 {	อิลิเมนต์ที่ 1
2	width: 250px;	อิลิเมนต์ที่ 2
3	height: 150px;	
4	background-color: lightblue;	
5	}	
6	.box2 {	
7	width: 60%;	
8	min-width: 200px;	
9	max-width: 400px;	
10	height: 150px;	
11	background-color: lightgreen;	
12	}	

จากตัวอย่างที่ 1 โครงสร้าง HTML ประกอบด้วยอิลิเมนต์ div จำนวน 2 รายการ โดยมีการกำหนดแอตทริบิวต์ class="box1" และ class="box2" ตามลำดับ เพื่อใช้เป็นเป้าหมายของตัวเลือกแบบคลาสในภาษา CSS

- ตัวเลือก .box1 ถูกใช้เพื่อกำหนดคุณสมบัติให้กับอิลิเมนต์ div แรก โดยระบุความกว้างที่ 250px และความสูงที่ 150px รวมถึงกำหนดสีพื้นหลังให้เป็นสีฟ้าอ่อน
- ตัวเลือก .box2 ถูกใช้กับอิลิเมนต์ div ที่สอง ซึ่งกำหนดความกว้างแบบสัมพัทธ์ไว้ที่ 60% ของพื้นที่อิลิเมนต์แม่ พร้อมกำหนดความกว้างต่ำสุด (min-width) ที่ 200px และความกว้างสูงสุด (max-width) ที่ 400px ขณะที่ความสูงถูกตั้งไว้เท่ากันคือ 150px และกำหนดสีพื้นหลังเป็นสีเขียวอ่อน

เมื่อแสดงผลบนเว็บเบราว์เซอร์ อิลิเมนต์ div แรกจะแสดงผลตามขนาดที่กำหนดแบบคงที่ ส่วนอิลิเมนต์ div ที่สองจะแสดงผลด้วยความกว้างที่ปรับเปลี่ยนตามสัดส่วน 60% ของอิลิเมนต์แม่ แต่จะไม่ต่ำกว่า 200px และไม่เกิน 400px แม้มีการปรับขนาดหน้าต่างเบราว์เซอร์ ซึ่งเป็นผลจากการใช้คุณสมบัติ min-width และ max-width ที่ช่วยจำกัดขอบเขตการขยายหรือย่อของอิลิเมนต์

1.2 คุณสมบัติในการจัดการเส้นขอบและระยะห่างของอิลิเมนต์

การจัดการจัดการเส้นขอบและระยะห่างของอิลิเมนต์ในภาษา CSS เป็นองค์ประกอบสำคัญของแบบจำลองกล่อง ซึ่งได้อธิบายไว้ในบทที่ 1 โดยองค์ประกอบของกล่องนอกเหนือจากส่วนเนื้อหาของอิลิเมนต์แล้ว การจัดการองค์ประกอบอื่นตามแบบจำลองกล่องก็เป็นส่วนสำคัญต่อการจัดวางอิลิเมนต์ ประกอบด้วย เส้นขอบ ระยะห่างภายใน และระยะห่างภายนอก

1.2.1 คุณสมบัติในการจัดการเส้นขอบของอิลิเมนต์

คุณสมบัติในการจัดการเส้นขอบของอิลิเมนต์ใน CSS ใช้เพื่อควบคุมการแสดงผลของขอบในแต่ละด้านของอิลิเมนต์ โดยสามารถกำหนดลักษณะของเส้นขอบในแต่ละด้านแยกกัน หรือใช้คำสั่งแบบย่อ (Shorthand) เพื่อกำหนดลักษณะร่วมกันทั้ง 4 ด้านได้ในคำสั่งเดียว รายละเอียดของคุณสมบัติที่พบบ่อยแสดงดังตารางที่ 8-2

ตารางที่ 8-2 รายการคุณสมบัติที่เกี่ยวข้องกับการจัดการเส้นขอบของอิลิเมนต์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
border-width	กำหนดความหนาของเส้นขอบทั้ง 4 ด้าน	ชื่อความหนาหรือค่าตามหน่วยวัด
border-style	กำหนดรูปแบบของเส้นขอบทั้ง 4 ด้าน	none หรือชื่อรูปแบบเส้นขอบ
border-color	กำหนดสีของเส้นขอบทั้ง 4 ด้าน	ค่าตามการกำหนดค่าสี
border-top	กำหนดลักษณะของเส้นขอบด้านบน	ความหนา รูปแบบ และสี
border-right	กำหนดลักษณะของเส้นขอบด้านขวา	ความหนา รูปแบบ และสี
border-bottom	กำหนดลักษณะของเส้นขอบด้านล่าง	ความหนา รูปแบบ และสี
border-left	กำหนดลักษณะของเส้นขอบด้านซ้าย	ความหนา รูปแบบ และสี
border	กำหนดลักษณะของเส้นขอบทั้ง 4 ด้านในคำสั่งเดียว	ความหนา รูปแบบ และสี

คุณสมบัติที่แสดงในตารางที่ 8-2 สามารถนำมาใช้เพื่อควบคุมลักษณะของเส้นขอบในแต่ละด้านของอิลิเมนต์ได้อย่างหลากหลาย ไม่ว่าจะเป็นการกำหนดสี รูปแบบ หรือความหนาของเส้นขอบ ทั้งแบบเจาะจงเฉพาะด้านและแบบรวมทุกด้านในคำสั่งเดียว ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 2

	CSS	HTML
1	.box1 {	
2	border-top: 3px dashed #ff6600;	กล่องที่ใช้กำหนดเส้นขอบแยกแต่
3	border-right: 2px dotted #0066cc;	ละด้าน
4	border-bottom: 4px solid #009966;	
5	border-left: 2px double #cc0000;	
6	}	กล่องที่ใช้กำหนดเส้นขอบทั้ง 4 ด้าน
7	.box2 {	ในคำสั่งเดียว
8	border: 5px solid #333333;	
9	}	

จากตัวอย่างที่ 2 มีการกำหนดอิลิเมนต์ div จำนวน 2 รายการ โดยอิลิเมนต์แรกมีการกำหนดแอตทริบิวต์ class="box1" และอิลิเมนต์ที่สองมีการกำหนด class="box2" ตามลำดับ ในส่วนของ CSS มีการใช้ตัวเลือกแบบคลาส เพื่อควบคุมคุณสมบัติเส้นขอบที่แสดงผลในแต่ละอิลิเมนต์

- อิลิเมนต์ที่มีคลาสเป็น box1 ใช้การกำหนดเส้นขอบแบบแยกด้านด้วยคุณสมบัติ border-top, border-right, border-bottom และ border-left (บรรทัดที่ 2-5) ซึ่งแต่ละคุณสมบัติมีการกำหนดค่า 3 รายการ โดยการกำหนดค่าต้องเรียงตามลำดับให้ถูกต้อง ได้แก่ ความหนา ชื่อรูปแบบเส้น และค่าสีตามลำดับ ซึ่งแต่ละค่าจะใช้การเว้นวรรคเพื่อแยกค่าแต่ละค่า
- อิลิเมนต์ที่มีคลาส box2 ใช้การกำหนดเส้นขอบในรูปแบบคำสั่งย่อ ด้วยคุณสมบัติ border (บรรทัดที่ 8) ซึ่งจะส่งผลกับเส้นขอบทั้ง 4 ด้านผ่านการกำหนดค่าในคุณสมบัติเดียว โดยการกำหนดคุณสมบัติ border จะต้องเรียงลำดับค่าตามลำดับให้ถูกต้อง ได้แก่ ความหนา ชื่อรูปแบบเส้น และค่าสีตามลำดับ โดยกำหนดความหนาขนาด 5px ใช้รูปแบบเส้นแบบทึบ (solid) และสีเทาเข้ม (#333333)

ผลลัพธ์ที่แสดงบนเว็บเบราว์เซอร์ จะพบว่าอิลิเมนต์ที่มีคลาสเป็น box1 จะแสดงเส้นขอบแต่ละด้านด้วยลักษณะที่แตกต่างกันตามที่กำหนดไว้ในชุดกฎของ CSS ในขณะที่ box2 จะแสดงเส้นขอบเหมือนกันทุกด้าน เนื่องจากใช้คำสั่งย่อที่จะส่งผลไปยังขอบทุกด้านโดยไม่ต้องแยกคำสั่ง

1.2.2 คุณสมบัติในการจัดการระยะห่างภายในของอิลิเมนต์

การกำหนดระยะห่างภายในของอิลิเมนต์ (Padding) เป็นการเว้นพื้นที่ระหว่างขอบด้านในของอิลิเมนต์กับเนื้อหาภายใน เช่น ข้อความหรือภาพ โดยใช้คุณสมบัติ padding ของภาษา CSS ซึ่งสามารถกำหนดได้แบบแยกแต่ละด้าน หรือกำหนดแบบย่อในคำสั่งเดียว ทั้งนี้ การกำหนดค่าดังกล่าวจะส่งผลต่อขนาดโดยรวมของอิลิเมนต์ตามแบบจำลองกล่องที่อธิบายไว้ในบทที่ 1 คุณสมบัติที่ใช้ในการจัดการระยะห่างภายในของอิลิเมนต์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน แสดงดังตารางที่ 8-3

ตารางที่ 8-3 รายการคุณสมบัติที่ใช้ในการจัดการระยะห่างภายในของอิลิเมนต์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
padding-top	กำหนดระยะห่างภายในด้านบนของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
padding-right	กำหนดระยะห่างภายในด้านขวาของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
padding-bottom	กำหนดระยะห่างภายในด้านล่างของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
padding-left	กำหนดระยะห่างภายในด้านซ้ายของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
padding	กำหนดระยะห่างภายในทุกด้านในคำสั่งเดียว โดยเรียงค่าตามด้านบน–ขวา–ล่าง–ซ้าย	ค่าตามหน่วยวัดระยะ

คุณสมบัติที่แสดงในตารางที่ 8.3 สามารถนำมาใช้เพื่อควบคุมระยะห่างภายในแต่ละด้านของอิลิเมนต์ ทั้งแบบกำหนดแยกแต่ละด้านหรือแบบกำหนดพร้อมกันในคำสั่งเดียวได้อย่างยืดหยุ่น ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 3

	CSS	HTML
1	.box1 {	
2	padding-top: 10px;	ระยะห่างภายในแบบกำหนดแยก
3	padding-right: 20px;	แต่ละด้าน
4	padding-bottom: 15px;	
5	padding-left: 25px;	
6	background-color: #e6f2ff;	ระยะห่างภายในแบบย่อในคำสั่ง เดียว
7	}	
8	.box2 {	
9	padding: 10px 20px 15px 25px;	
10	background-color: #d9f9d9;	
11	}	

จากตัวอย่างที่ 3 มีการกำหนดอิลิเมนต์ div จำนวน 2 รายการ โดยอิลิเมนต์แรกมีการกำหนดแอตทริบิวต์ class="box1" และอิลิเมนต์ที่สองมีการกำหนด class="box2" ในส่วนของ CSS มีการใช้ตัวเลือกแบบคลาสเพื่อควบคุมคุณสมบัติ padding ที่แสดงผลในแต่ละอิลิเมนต์

- อิลิเมนต์ที่มีคลาสเป็น box1 ใช้การกำหนดระยะห่างภายในแบบแยกด้านด้วยคุณสมบัติ padding-top, padding-right, padding-bottom และ padding-left (บรรทัดที่ 2–5) ซึ่งจะเว้นระยะห่างภายในอิลิเมนต์แต่ละด้านตามค่าที่ระบุ
- อิลิเมนต์ที่มีคลาสเป็น box2 ใช้คำสั่งย่อ padding (บรรทัดที่ 9) โดยระบุค่าทั้ง 4 ด้านเรียงตามลำดับด้านบน ด้านขวา ด้านล่าง และด้านซ้าย ตามลำดับ

ผลลัพธ์ที่แสดงบนเว็บเบราว์เซอร์จะพบว่าอิลิเมนต์ box1 และ box2 แสดงผลระยะห่างภายในเท่ากันทุกด้าน โดยพื้นที่ว่างรอบข้อความภายในอิลิเมนต์จะแสดงตามค่าที่กำหนด ทำให้เนื้อหาไม่ชิดขอบของอิลิเมนต์มากเกินไป ช่วยให้อ่านง่ายและเป็นระเบียบ

1.2.3 คุณสมบัติในการจัดการระยะห่างภายนอกของอิลิเมนต์

การกำหนดระยะห่างภายนอกของอิลิเมนต์ (Margin) เป็นการเว้นระยะว่างภายนอกขอบของอิลิเมนต์โดยใช้คุณสมบัติ margin เพื่อแยกตำแหน่งระหว่างอิลิเมนต์แต่ละรายการในหน้าเว็บ โดยสามารถกำหนดได้ทั้งแบบแยกแต่ละด้านและแบบรวมทุกด้านในคำสั่งเดียว ทั้งนี้ยังสามารถใช้ค่า auto เพื่อให้เว็บเบราว์เซอร์คำนวณระยะห่างโดยอัตโนมัติ คุณสมบัติที่ใช้ในการจัดการระยะห่างภายนอกของอิลิเมนต์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน แสดงดังตารางที่ 8-4

ตารางที่ 8-4 รายการคุณสมบัติในการจัดการระยะห่างภายนอกของอิลิเมนต์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
margin-top	กำหนดระยะห่างภายนอกด้านบนของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
margin-right	กำหนดระยะห่างภายนอกด้านขวาของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
margin-bottom	กำหนดระยะห่างภายนอกด้านล่างของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
margin-left	กำหนดระยะห่างภายนอกด้านซ้ายของอิลิเมนต์	ค่าตามหน่วยวัดระยะ
margin	กำหนดระยะห่างภายนอกทุกด้านในคำสั่งเดียว โดยเรียงค่าตาม บน–ขวา–ล่าง–ซ้าย	ค่าตามหน่วยวัดระยะ

คุณสมบัติที่แสดงในตารางที่ 8.4 สามารถนำมาใช้ควบคุมตำแหน่งและระยะห่างระหว่างอิลิเมนต์ต่าง ๆ ได้ ทั้งแบบเจาะจงด้านและแบบคำสั่งย่อ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 4

	CSS	HTML
1	.box1 {	
2	margin-top: 10px;	ระยะห่างภายนอกแบบกำหนดแยกแต่
3	margin-right: 20px;	ละด้าน
4	margin-bottom: 15px;	
5	margin-left: 25px;	
6	background-color: #fff0cc;	ระยะห่างภายนอกแบบกำหนดรวมโดย
7	}	ใช้คุณสมบัติเดียว
8	.box2 {	
9	margin: 10px 20px 15px 25px;	
10	background-color: #ffe0e0;	
11	}	

จากตัวอย่างที่ 4 มีการกำหนดอิลิเมนต์ div จำนวน 2 รายการ โดยอิลิเมนต์แรกมีการกำหนดแอตทริบิวต์ class="box1" และอิลิเมนต์ที่สองมีการกำหนด class="box2" ในส่วนของ CSS มีการใช้ตัวเลือกแบบคลาสเพื่อควบคุมคุณสมบัติ margin ดังนี้

- อิลิเมนต์ที่มีคลาสเป็น box1 ใช้การกำหนดระยะห่างภายนอกแบบแยกแต่ละด้านด้วย margin-top, margin-right, margin-bottom และ margin-left (บรรทัดที่ 2–5) ซึ่งจะส่งผลให้แต่ละด้านมีค่าระยะห่างตามที่ระบุ

- อิลิเมนต์ที่มีคลาส box2 ใช้คำสั่งย่อ margin (บรรทัดที่ 9) โดยระบุค่าทั้ง 4 ด้านเรียงตามลำดับบน–ขวา–ล่าง–ซ้าย

ผลลัพธ์ที่แสดงบนเว็บเบราว์เซอร์จะพบว่าอิลิเมนต์ทั้งสองรายการแสดงตำแหน่งห่างจากอิลิเมนต์อื่น ๆ หรือขอบของอิลิเมนต์แม่ตามค่าที่กำหนดในแต่ละด้าน

1.3 คุณสมบัติในการจัดตำแหน่งข้อความและอิลิเมนต์

การจัดตำแหน่งข้อความและอิลิเมนต์ในภาษา CSS เป็นส่วนสำคัญในการควบคุมลักษณะการแสดงผลให้ถูกต้อง เป็นระเบียบ และตรงกับวัตถุประสงค์ของการออกแบบ โดยเฉพาะในบริบทของเว็บแอปพลิเคชันที่ต้องรองรับข้อมูลหลากหลายประเภท ทั้งข้อความ ตัวเลข ป้ายกำกับ หรือปุ่มกดต่าง ๆ การจัดตำแหน่งที่ดีจะช่วยให้ผู้ใช้เข้าใจโครงสร้างข้อมูลและโต้ตอบกับระบบได้อย่างสะดวก คุณสมบัติที่ใช้ในการจัดตำแหน่งและลักษณะของข้อความ รวมถึงอิลิเมนต์ที่เกี่ยวข้องกับข้อความ แสดงดังตารางที่ 8-5

ตารางที่ 8-5 รายการคุณสมบัติในการจัดตำแหน่งข้อความและอิลิเมนต์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
direction	กำหนดทิศทางของข้อความ	ltr, rtl
letter-spacing	ปรับระยะห่างระหว่างตัวอักษร	ค่าตามหน่วยวัดระยะ
line-height	กำหนดความสูงของบรรทัด	normal, ค่าตามหน่วยวัดระยะ
text-align	จัดตำแหน่งข้อความในแนวนอนภายในอิลิเมนต์	left, right, center, justify
text-decoration	กำหนดลักษณะการตกแต่งข้อความ เช่น ขีดเส้นใต้ ขีดทับ	none, underline, line-through, overline
text-transform	กำหนดลักษณะการเปลี่ยนรูปแบบตัวอักษร เช่น ตัวพิมพ์ใหญ่ พิมพ์เล็ก	none, capitalize, uppercase, lowercase
vertical-align	จัดตำแหน่งในแนวตั้งภายในอิลิเมนต์แบบ Inline	ชื่อการจัดตำแหน่ง
white-space	ควบคุมการแสดงผลของช่องว่างและการขึ้นบรรทัดอัตโนมัติ	ชื่อควบคุมการแสดงผล
word-spacing	ปรับระยะห่างระหว่างคำในข้อความ	ค่าตามหน่วยวัดระยะ

คุณสมบัติที่แสดงในตารางที่ 8-5 สามารถนำมาใช้เพื่อควบคุมลักษณะการจัดตำแหน่งข้อความภายในอิลิเมนต์และรูปแบบของข้อความ เช่น การแปลงรูปแบบตัวอักษร การเว้นระยะระหว่างบรรทัด และการเว้นระยะระหว่างอักษร โดยสามารถใช้งานร่วมกันเพื่อเสริมประสบการณ์ของผู้ใช้งานผ่านการนำเสนอเนื้อหาที่อ่านง่ายและสอดคล้องกับโครงร่างของเว็บแอปพลิเคชัน ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 5

	CSS	HTML
1	.text-sample1 {	
2	text-align: justify;	ภาษาไทยเป็นภาษาที่มีวรรณยุกต์และ
3	text-indent: 40px;	ความละเอียดอ่อนในการใช้คำ การจัด
4	line-height: 1.8;	ตำแหน่งข้อความให้เหมาะสมกับบริบท
5	}	ของเนื้อหาเป็นสิ่งสำคัญที่ช่วยให้ผู้ใช้งาน
6	.text-sample2 {	สามารถอ่านและทำความเข้าใจข้อมูลได้ดี
7	text-transform: uppercase;	ขึ้น
8	}	
		web application คือระบบที่ทำงานผ่าน
		เว็บเบราว์เซอร์ โดยสามารถโต้ตอบกับ
		ผู้ใช้ได้แบบทันที

จากตัวอย่างที่ 5 มีการกำหนดอิลิเมนต์ div จำนวน 2 รายการ โดยมีการกำหนดแอตทริบิวต์ class="text-sample1" สำหรับอิลิเมนต์แรก และ class="text-sample2" สำหรับอิลิเมนต์ที่สองตามลำดับ ส่วนของ CSS มีการใช้ตัวเลือกแบบคลาสเพื่อควบคุมคุณสมบัติของข้อความในแต่ละอิลิเมนต์ ดังนี้

- อิลิเมนต์ที่มีคลาส text-sample1 ถูกกำหนดให้มีการจัดแนวข้อความแบบเต็มบรรทัด (justify) โดยใช้คุณสมบัติ text-align: justify; (บรรทัดที่ 2) พร้อมทั้งมีการจัดระยะเยื้องของข้อความบรรทัดแรก text-indent: 40px; (บรรทัดที่ 3) และกำหนดระยะห่างระหว่างบรรทัดด้วย line-height: 1.8; (บรรทัดที่ 4) เพื่อให้ข้อความอ่านง่ายขึ้น
- อิลิเมนต์ที่มีคลาส text-sample2 ถูกกำหนดให้แปลงข้อความทั้งหมดในย่อหน้าให้เป็นตัวพิมพ์ใหญ่โดยใช้คุณสมบัติ text-transform: uppercase; (บรรทัดที่ 7) ทั้งนี้ผลของคุณสมบัติ text-transform จะมีผลต่อข้อความที่เป็นภาษาอังกฤษ

ผลลัพธ์ที่แสดงผลในเว็บเบราว์เซอร์ในส่วนย่อหน้าที่มีคลาสเป็น text-sample1 จะจัดข้อความให้กระจายเต็มบรรทัดด้านซ้ายและขวา โดยบรรทัดแรกของย่อหน้าจะถูกเยื้องไปทางขวา 40 พิกเซล และมีระยะห่างระหว่างบรรทัดที่กว้างขึ้นจากค่าปกติ ในส่วนย่อหน้าที่เป็นคลาส text-sample2 จะแสดงข้อความภาษาอังกฤษทุกคำด้วยตัวพิมพ์ใหญ่ทั้งหมด ซึ่งเหมาะกับการเน้นข้อความหรือใช้ในหัวข้อที่ต้องการความโดดเด่น

1.4 คุณสมบัติในการกำหนดแบบตัวอักษร

การกำหนดแบบอักษรของข้อความใน CSS สามารถกำหนดลักษณะต่าง ๆ เช่น ชนิด ขนาด ความหนา และความเอียงของฟอนต์ ตลอดจนใช้รูปแบบคำสั่งแบบย่อเพื่อความกระชับในการเขียนรหัสคำสั่ง CSS คุณสมบัติที่เกี่ยวข้องกับกำหนดแบบตัวอักษรที่พบบ่อย แสดงดังตารางที่ 8-6

ตารางที่ 8-6 คุณสมบัติที่เกี่ยวข้องกับการกำหนดแบบตัวอักษร

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
font-family	กำหนดแบบอักษรที่ใช้แสดงข้อความ	ชื่อแบบอักษร
font-size	กำหนดขนาดของตัวอักษร	ค่าตามหน่วยวัดขนาด
font-style	กำหนดลักษณะการเอียงของตัวอักษร	normal, italic, oblique
font-weight	กำหนดน้ำหนักหรือลักษณะความหนาของตัวอักษร	normal, bold, 100–900
font	กำหนดแบบอักษรแบบย่อ	รูปแบบ น้ำหนัก ขนาด และชื่อแบบอักษรตามลำดับ

จากตารางคุณสมบัติที่เกี่ยวข้องกับการกำหนดแบบตัวอักษร สามารถนำไปประยุกต์ใช้ได้ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6

	CSS	HTML
1	.font1 {	
2	font-family: Arial, sans-serif;	ข้อความนี้ใช้ฟอนต์ Arial ตัวเอียง ขนาด
3	font-size: 18px;	18px และตัวหนา
4	font-style: italic;	
5	font-weight: bold;	
6	}	
7	.font2 {	ข้อความนี้ใช้ฟอนต์ Times New Roman
8	font: normal 400 20px "Times New	ขนาด 20px และน้ำหนักปกติ
8	Roman", serif;	
9	}	

จากตัวอย่างที่ 6 มีการสร้างอิลิเมนต์ p สองรายการ โดยมีการกำหนดแอตทริบิวต์ class="font1" และ class="font2" ตามลำดับ เพื่อแสดงการใช้งานคุณสมบัติเกี่ยวกับแบบอักษรในรูปแบบต่างกันด้วย CSS ดังนี้

- อิลิเมนต์ที่มีคลาส font1 ใช้ตัวเลือกแบบคลาสเพื่อกำหนดคุณสมบัติแบบแยก โดยใช้ font-family เป็น Arial และ fallback เป็น sans-serif (บรรทัดที่ 2), กำหนด font-size เป็น 18px (บรรทัดที่ 3), font-style เป็น italic (บรรทัดที่ 4) และ font-weight เป็น bold (บรรทัดที่ 5)
- อิลิเมนต์ที่มีคลาส font2 ใช้คำสั่งย่อ font (บรรทัดที่ 9) เพื่อระบุลักษณะของแบบอักษรในบรรทัดเดียว โดยเรียงลำดับค่าเป็น font-style, font-weight, font-size และ font-family ตามลำดับ ซึ่งในที่นี้กำหนดให้ใช้แบบอักษร Times New Roman และกำหนดแบบอักษรสำรอง (Fallback) เป็น serif ในกรณีที่ไม่สามารถแสดงแบบอักษรหลักที่กำหนดได้

ผลลัพธ์ที่ปรากฏบนเว็บเบราว์เซอร์จะเห็นว่าอิลิเมนต์ p ที่มีคลาส font1 แสดงข้อความด้วยแบบอักษร Arial ลักษณะเอียง ขนาด 18px และตัวหนา ส่วนอิลิเมนต์ p ที่มีคลาส font2 แสดงข้อความด้วยแบบอักษร Times New Roman ขนาด 20px ในลักษณะปกติและน้ำหนักมาตรฐานตามที่กำหนดไว้ในคำสั่งย่อ

1.5 คุณสมบัติในการจัดการพื้นหลังของอิลิเมนต์

การจัดการพื้นหลังของอิลิเมนต์ใน CSS เป็นเครื่องมือสำคัญที่ช่วยเพิ่มความน่าสนใจและความสวยงามให้กับเว็บแอปพลิเคชัน ไม่ว่าจะเป็นการกำหนดสีพื้นหลัง การแสดงภาพพื้นหลังซ้ำ การปรับตำแหน่งของภาพ ไปจนถึงการกำหนดขนาดและลักษณะการแสดงผลของภาพพื้นหลัง คุณสมบัติเหล่านี้ช่วยให้สามารถปรับแต่งอิลิเมนต์ได้อย่างยืดหยุ่นและเหมาะสมกับการใช้งานในบริบทต่าง ๆ คุณสมบัติในการจัดการพื้นหลังของอิลิเมนต์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชันแสดงดังตารางที่ 8-7

ตารางที่ 8-7 รายการคุณสมบัติในการจัดการพื้นหลังของอิลิเมนต์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
background-color	กำหนดสีพื้นหลังของอิลิเมนต์	ชื่อ, สีค่าสี
background-image	กำหนดภาพพื้นหลังของอิลิเมนต์	url("ชื่อไฟล์")
background-repeat	กำหนดว่าภาพพื้นหลังจะแสดงซ้ำหรือไม่	repeat, no-repeat, repeat-x, repeat-y
background-position	กำหนดตำแหน่งของภาพพื้นหลังภายในอิลิเมนต์	ชื่อตำแหน่ง, %, ค่าพิกัดตำแหน่ง
background-size	กำหนดขนาดของภาพพื้นหลัง	ค่าตามหน่วยวัด, auto, cover, contain
background-attachment	กำหนดลักษณะการเลื่อนของภาพพื้นหลังเมื่อเลื่อนหน้าจอ	scroll, fixed, local
background	กำหนดลักษณะของพื้นหลังแบบรวมหลายคุณสมบัติในคำสั่งเดียว	สี ภาพ ตำแหน่ง การแสดงซ้ำ การเลื่อน

คุณสมบัติที่แสดงในตารางที่ 8-7 สามารถนำมาใช้เพื่อปรับแต่งลักษณะของพื้นหลังของอิลิเมนต์ได้ โดยสามารถเลือกใช้คุณสมบัติเฉพาะที่ต้องการ หรือกำหนดแบบย่อที่รวมหลายคุณสมบัติไว้ในคำสั่งเดียวก็ได้ ทั้งนี้การจัดการพื้นหลังไม่เพียงแต่ช่วยสร้างความสวยงามให้กับหน้าเว็บ แต่ยังสามารถใช้เพื่อเน้นความสำคัญของเนื้อหาหรือแสดงอัตลักษณ์ของเว็บได้อีกด้วย ตัวอย่างต่อไปนี้แสดงวิธีการใช้งานคุณสมบัติพื้นหลัง

ตัวอย่างที่ 7

CSS

```
1  .bg1 {
2      background-color: #f9f9f9;
3  }
4  .bg2 {
5      background-image: url("pattern.png");
6      background-repeat: repeat;
7      background-position: top left;
8  }
9  .bg3 {
10     background: #e0f7fa url("banner.jpg") no-repeat center top;
11 }
```

HTML

```
1  <div class="bg1">    พื้นหลังของอิลิเมนต์นี้ใช้สีพื้นฐาน</div>
2  <div class="bg2">    พื้นหลังของอิลิเมนต์นี้ใช้ภาพซ้ำแบบลายพื้นหลัง</div>
3  <div class="bg3">    พื้นหลังของอิลิเมนต์นี้ใช้คำสั่งแบบย่อ </div>
```

จากตัวอย่างที่ 7 มีการกำหนดอิลิเมนต์ div จำนวน 3 รายการ โดยใช้แอตทริบิวต์ class ที่แตกต่างกัน ได้แก่ bg1, bg2 และ bg3 เพื่อแสดงการใช้คุณสมบัติพื้นหลังที่แตกต่างกัน

- อิลิเมนต์ที่มีคลาส bg1 ใช้คุณสมบัติ background-color เพื่อกำหนดสีพื้นหลังให้เป็นสีเทาอ่อน #f9f9f9 โดยไม่มีการใช้ภาพประกอบ
- อิลิเมนต์ที่มีคลาส bg2 ใช้คุณสมบัติ background-image, background-repeat และ background-position เพื่อกำหนดภาพพื้นหลังให้แสดงซ้ำจากมุมบนซ้ายของอิลิเมนต์ซึ่งเหมาะกับการทำลวดลายซ้ำต่อกัน

- อิลิเมนต์ที่มีคลาส bg3 ใช้คำสั่งแบบย่อ background ซึ่งรวมคุณสมบัติ background-color, background-image, background-repeat และ background-position เข้าไว้ด้วยกัน โดยภาพจะไม่ซ้ำและถูกจัดตำแหน่งที่กึ่งกลางด้านบน พร้อมพื้นหลังสี #e0f7fa

ผลลัพธ์ที่แสดงบนเว็บเบราว์เซอร์จะแสดงอิลิเมนต์ทั้งสามรายการโดยมีพื้นหลังที่แตกต่างกันตามคุณสมบัติที่กำหนดไว้

1.6 คุณสมบัติในการกำหนดลักษณะของลิงก์

การแสดงผลของลิงก์ตามค่าเริ่มต้นการแสดงผลของเว็บเบราว์เซอร์ จะแสดงข้อความลิงก์เป็นสีน้ำเงินที่มีเส้นใต้ ซึ่งเป็นรูปแบบของลิงก์ที่ยังไม่เคยคลิก และจะเปลี่ยนเป็นสีม่วงเมื่อลิงก์มีประวัติเคยถูกคลิกหรือเข้าเยี่ยมชมมาก่อน อย่างไรก็ตาม CSS สามารถใช้ในการควบคุมลักษณะของลิงก์ได้หลากหลายมากขึ้น เพื่อให้เหมาะสมกับรูปแบบของเว็บแอปพลิเคชัน ทั้งนี้ การกำหนดลักษณะของลิงก์จะอาศัยตัวเลือกพิเศษที่เรียกว่า pseudo-classes ได้แก่ :link, :visited, :hover, และ :active เพื่อบ่งบอกสถานะที่ต้องการกำหนดรูปแบบ ร่วมกับการกำหนดคุณสมบัติการแสดงผล เช่น สี ขนาด หรือการตกแต่งข้อความ เป็นต้น คุณสมบัติในการกำหนดลักษณะของลิงก์แสดงดังตารางที่ 8-8

ตารางที่ 8-8 รายการคุณสมบัติในการกำหนดลักษณะของลิงก์

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
color	กำหนดสีของข้อความในลิงก์	ชื่อสี, ค่าสี
text-decoration	กำหนดลักษณะของการตกแต่งข้อความ เช่น เส้นใต้ เส้นขีดทับ เป็นต้น	none, underline, overline, line-through
font-weight	กำหนดน้ำหนักหรือลักษณะความหนาของข้อความในลิงก์	normal, bold, 100–900

คุณสมบัติที่แสดงในตารางที่ 8-8 เป็นคุณสมบัติในกลุ่มการจัดการรูปแบบตัวอักษร สามารถนำมาใช้เพื่อควบคุมลักษณะการแสดงผลของลิงก์ให้เหมาะสมกับการออกแบบเว็บแอปพลิเคชัน เช่น การปรับเปลี่ยนสีข้อความให้ต่างจากข้อความทั่วไป การปรับปรับเปลี่ยนสีเมื่อผู้ใช้เลื่อนเมาส์เหนือลิงก์ เป็นต้น การเน้นน้ำหนักตัวอักษรของลิงก์ เพื่อให้ผู้ใช้งานสามารถแยกแยะลิงก์ได้อย่างชัดเจนจากเนื้อหาอื่น ๆ บนหน้าเว็บ ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 8

CSS

```
1  a.link1:link {
2      color: #1a73e8;
```

```
3       text-decoration: underline;
4       font-weight: normal;
5   }
7   a.link1:hover {
8       color: #0b57d0;
9       text-decoration: none;
10  }
11  a.link1:visited {
12      color: #6c757d;
13  }
```

HTML

```
1   <h2>ลิงก์ที่ไม่มีการกำหนดลักษณะด้วย CSS: </h2>
2   <a href="https://www.ajnok.com" target="_blank">ลิงก์เริ่มต้น</a>
3   <h2>ลิงก์ที่มีการกำหนดลักษณะด้วย CSS:</h2>
4   <a href="https://www.ajnok.com" class="link1" target="_blank">ลิงก์กำหนดลักษณะ</a>
```

จากตัวอย่างที่ 8 มีการสร้างอิลิเมนต์ a จำนวน 2 รายการ โดยรายการแรกไม่กำหนดแอตทริบิวต์ class ใด ๆ และใช้การแสดงผลตามค่าเริ่มต้นของเบราว์เซอร์ ขณะที่รายการที่สองกำหนดแอตทริบิวต์ class="link1" เพื่อใช้ควบคุมลักษณะการแสดงผลผ่านชุดกฎ CSS ที่กำหนดไว้ ในส่วนของ CSS มีการใช้ตัวเลือกแบบคลาสร่วมกับ pseudo-classes เพื่อควบคุมลักษณะของลิงก์ดังนี้

- บรรทัดที่ 2–4 เป็นการกำหนดคุณสมบัติของลิงก์ที่ยังไม่ถูกคลิก โดยใช้ตัวเลือก a.link1:link ซึ่งหมายถึงลิงก์ที่มีคลาสเป็น link1 และยังไม่ถูกเยี่ยมชม โดยกำหนดให้มีสี #1a73e8 มีเส้นใต้ และไม่เป็นตัวหนา
- บรรทัดที่ 8-9 ใช้ตัวเลือก a.link1:hover เพื่อควบคุมลิงก์เมื่อมีการวางเมาส์เหนือลิงก์ที่อยู่ในคลาส link1 โดยเปลี่ยนสีข้อความเป็น #0b57d0 และไม่แสดงเส้นใต้
- บรรทัดที่ 12 ใช้ตัวเลือก a.link1:visited เพื่อควบคุมลักษณะของลิงก์ที่อยู่ในคลาสส link1 และมีสถานะที่ผู้ใช้เคยคลิกลิงก์นั้น โดยเปลี่ยนสีข้อความเป็น #6c757d

ผลลัพธ์ที่แสดงบนเว็บเบราว์เซอร์ จะพบว่าอิลิเมนต์ a ที่ไม่มีการกำหนดคลาสจะปรากฏเป็นลิงก์สีน้ำเงินและมีเส้นใต้ตามค่าเริ่มต้นการแสดงผลของเว็บเบราว์เซอร์ ในขณะที่อิลิเมนต์ a ที่มีคลาส link1 จะแสดงลักษณะของลิงก์ตามสถานะต่าง ๆ ได้แก่ ลิงก์ที่ยังไม่ถูกคลิกจะเป็นสี #1a73e8 มีเส้นใต้ เมื่อนำเมาส์ไปวางเหนือลิงก์จะเปลี่ยนสีเป็น #0b57d0 และไม่เส้นใต้ ส่วนลิงก์ที่เคยถูกคลิกแล้วจะแสดงเป็นสีเทา #6c757d ซึ่งแสดงให้เห็นถึงการใช้ pseudo-class ในการจัดการรูปแบบการแสดงผลตามสถานะต่าง ๆ ของอิลิเมนต์

1.7 คุณสมบัติในการจัดรูปแบบตาราง

จากการสร้างตารางด้วยแท็กในภาษา HTML ที่ได้กล่าวถึงในบทที่ 6 พบว่าตารางที่สร้างขึ้นจะไม่มีการแสดงเส้นตารางแต่อย่างใด เนื่องจากเป็นการจัดโครงสร้างของข้อมูลในรูปแบบตาราง ดังนั้นจึงต้องอาศัย CSS ในการจัดรูปแบบตาราง โดยมีคุณสมบัติที่สำคัญในการจัดรูปแบบตารางดังตารางที่ 8-9

ตารางที่ 8-9 รายการคุณสมบัติในการจัดรูปแบบตาราง

ชื่อคุณสมบัติ	หน้าที่หลัก	ค่าที่ยอมรับ
border	กำหนดลักษณะของเส้นขอบตารางและเซลล์	ความหนา รูปแบบเส้น สี
border-collapse	กำหนดว่าจะรวมเส้นขอบของเซลล์ให้เป็นเส้นเดียวหรือแยกออกจากกัน	collapse, separate
border-spacing	กำหนดระยะห่างระหว่างเซลล์ของตาราง (ใช้ได้เมื่อ border-collapse: separate)	ค่าตามหน่วยวัด
caption-side	กำหนดตำแหน่งของคำอธิบายตาราง	top, bottom

จากตารางรายการคุณสมบัติในการจัดรูปแบบตารางข้างต้น จะมีการใช้คุณสมบัติ border เช่นเดียวกันกับอิลิเมนต์ที่ต้องการสร้างขอบเพื่อนำมาใช้ในการแสดงขอบของเซลล์ข้อมูลและขอบของตาราง โดยมีตัวอย่างการใช้งานดังนี้

ตัวอย่างที่ 9

CSS

```
1  .table-a {
2      border-collapse: separate;
3      border: 2px solid black;
4      caption-side: top;
5      display: inline-table;
6  }
7  .table-b {
8      border-collapse: collapse;
9      border: 2px solid red;
10     caption-side: bottom;
11     display: inline-table;
12 }
13 th, td {
14     border: 1px solid black;
15     padding: 8px;
16     text-align: center;
17 }
18 .caption {
19     font-weight: bold;
20     margin-bottom: 10px;
21 }
```

HTML

```
<table class="table-separate">
    <caption>รายการสินค้า (border-collapse: separate)</caption>
    <tr>
        <th>รหัสสินค้า</th>
        <th>ชื่อสินค้า</th>
        <th>ราคา</th>
    </tr>
    <tr>
        <td>P001</td>
        <td>สมุดโน้ต</td>
        <td>45</td>
    </tr>
    <tr>
        <td>P002</td>
        <td>ปากกาหมึกเจล</td>
        <td>25</td>
    </tr>
</table>
<table class="table-collapse">
    <caption>รายการสินค้า (border-collapse: collapse)</caption>
    <tr>
        <th>รหัสสินค้า</th>
        <th>ชื่อสินค้า</th>
        <th>ราคา</th>
    </tr>
    <tr>
        <td>P001</td>
        <td>สมุดโน้ต</td>
        <td>45</td>
    </tr>
    <tr>
```

```
32              <td>P002</td>
33              <td>ปากกาหมึกเจล</td>
34              <td>25</td>
35          </tr>
36      </table>
```

จากตัวอย่างที่ 9 มีการสร้างตารางจำนวน 2 ตารางเพื่อแสดงลักษณะการแสดงผลที่แตกต่างกันระหว่างค่าของคุณสมบัติ border-collapse และ border-separate โดยตารางแรกมีการกำหนดแอตทริบิวต์ class="table-a" มีชุดกฎของ CSS ที่เกี่ยวข้องกับการจัดรูปแบบตารางในบรรทัดที่ 1-6 และ 13-17 ดังนี้

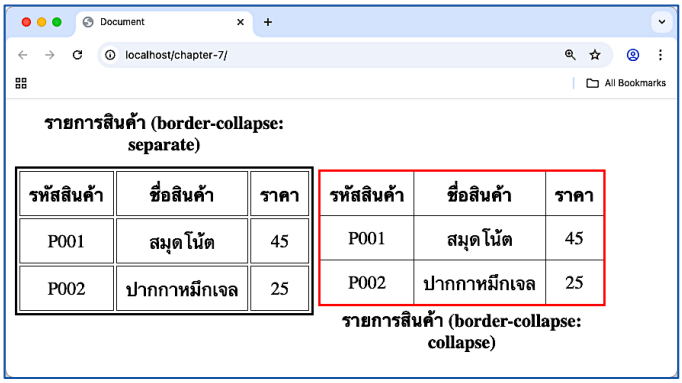
- บรรทัดที่ 2 ใช้คุณสมบัติ border-collapse: separate; เพื่อทำให้เส้นขอบของแต่ละเซลล์แยกออกจากกัน
- บรรทัดที่ 3 ใช้คุณสมบัติ border: 2px solid black; เพื่อกำหนดเส้นขอบโดยรอบตารางเป็นเส้นตรงที่มีความหนา 2px สีดำ
- บรรทัดที่ 4 ใช้คุณสมบัติ caption-side: top; เพื่อกำหนดให้คำอธิบายตารางอยู่เหนือตาราง
- บรรทัดที่ 5 ใช้คุณสมบัติ display: inline-table เพื่อทำให้ตารางมีคุณลักษณะการแสดงผลเป็นแบบ Inline เพื่อทำให้สองตารางแสดงอยู่ในแถวบรรทัดเดียวกัน
- บรรทัดที่ 13-17 เป็นการใช้คุณสมบัติร่วมกับทุกๆ อิลิเมนต์ th และ td โดยเป็นการกำหนดให้เส้นขอบของเซลล์ที่เป็นหัวตาราง และเซลล์ข้อมูลแสดงเป็นเส้นตรงสีดำความหนา 1px (บรรทัดที่ 14) มีระยะห่างภายในเซลล์ 8px (บรรทัดที่ 15) และจัดข้อความไว้กึ่งกลางเซลล์ (บรรทัดที่ 16)

สำหรับตารางที่กำหนดแอตทริบิวต์ class="table-b" มีชุดกฎของ CSS ที่เกี่ยวข้องกับการจัดรูปแบบตารางในบรรทัดที่ 7-12 และ 13-17 ซึ่งมีลักษณะคล้ายกับตารางก่อนหน้า แต่มีจุดที่แตกต่างกันดังนี้

- บรรทัดที่ 8 ใช้คุณสมบัติ border-collapse: collapse; เพื่อทำให้เส้นขอบของแต่ละเซลล์รวมเป็นเส้นเดียวกัน
- บรรทัดที่ 9 ใช้คุณสมบัติ border: 2px solid red; เพื่อกำหนดเส้นขอบโดยรอบตารางเป็นเส้นตรงที่มีความหนา 2px สีแดง
- บรรทัดที่ 10 ใช้คุณสมบัติ caption-side: bottom; เพื่อกำหนดให้คำอธิบายตารางอยู่ด้านล่างของตาราง

ผลลัพธ์ที่ได้ในเว็บเบราว์เซอร์จะพบว่าตารางแรก (คลาส table-a) มีการแสดงคำอธิบายตารางไว้เหนือตาราง และลักษณะของเส้นขอบเซลล์จะแยกจากกัน ส่วนตารางที่สอง (คลาส table-b) มีการแสดงคำอธิบายไว้ใต้ตาราง และลักษณะเส้นขอบทุกเส้นรวมทั้งเส้นขอบโดยรอบตารางมีการรวมกันเป็นเส้นเดียวดังภาพตัวอย่างด้านล่างนี้

ผลลัพธ์ของตัวอย่างที่ 9



2. หลักการพื้นฐานของการจัดวางโครงร่างด้วย CSS Flexbox และ CSS Grid

การจัดวางโครงร่าง (Layout) ขององค์ประกอบในหน้าเว็บเป็นหนึ่งในพื้นฐานสำคัญของการพัฒนาเว็บแอปพลิเคชัน โดยเฉพาะเมื่อมีองค์ประกอบจำนวนมากที่ต้องการการจัดตำแหน่งอย่างเหมาะสม และการตอบสนองต่ออุปกรณ์ที่มีขนาดหน้าจอที่แตกต่างกัน CSS ได้พัฒนาเครื่องมือที่เรียกว่า CSS Flexbox และ CSS Grid เพื่อใช้ในการควบคุมการวางตำแหน่งซึ่งทั้งสองแนวทางต่างมีหลักการที่เหมาะสมกับรูปแบบการจัดวางที่แตกต่างกัน

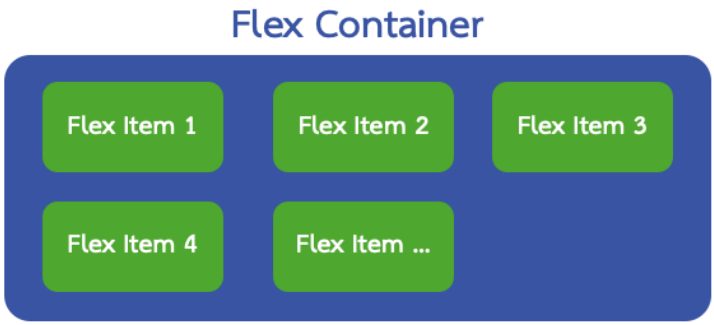
2.1 หลักการของ CSS Flexbox

Flexbox เป็นคำเรียกโดยย่อที่มาจากคำว่า Flexible Box Layout เป็นวิธีการจัดวางอิลิเมนต์ของ CSS ที่ออกแบบมาเพื่อจัดเรียงอิลิเมนต์ในลักษณะหนึ่งมิติ (One-Dimensional Layout) กล่าวคือ สามารถจัดเรียงได้ในแนวนอน (horizontal) หรือแนวตั้ง (vertical) อย่างใดอย่างหนึ่งในแต่ละครั้งโดยไม่รองรับการจัดวางสองมิติพร้อมกัน

การจัดเรียงในแนวนอนจะมีมุมมองในลักษณะของแถว เรียกว่า Flex Row ขณะที่การจัดเรียงในแนวตั้งจะมีมุมมองในลักษณะของคอลัมน์ เรียกว่า Flex Column ซึ่งทั้งสองรูปแบบจะช่วยให้การวางโครงร่างของหน้าเว็บมีความยืดหยุ่นมากขึ้น

2.1.1 องค์ประกอบหลักของ CSS Flexbox

องค์ประกอบหลักของ CSS Flexbox ประกอบด้วย 2 องค์ประกอบ ได้แก่ Flex Container และ Flex Items ดังภาพที่ 8.1



ภาพที่ 8.1 องค์ประกอบของ CSS Flexbox

1) Flex Container

Flex Container หรือองค์ประกอบบรรจุ คืออิลิเมนต์แม่ที่ทำหน้าที่ในการห่อหุ้มอิลิเมนต์ลูกที่อยู่ภายใน เช่น div, form, ul และ ol เป็นต้น โดยอิลิเมนต์ที่จะทำหน้าที่เป็น Flex Container จะถูกกำหนดคุณสมบัติ display: flex หรือ display: inline-flex เพื่อกำหนดให้อิลิเมนต์นั้นๆ ทำหน้าที่เป็น Flex Container เมื่อกำหนดแล้วอิลิเมนต์ลูกจะได้รับผลจากการจัดวางตามกฎของ Flexbox โดยอัตโนมัติ

Flex Container มีบทบาทสำคัญในการกำหนดทิศทางหรือแนวแกนที่จะใช้ในการจัดเรียง Flex Container ทำหน้าที่เป็นกรอบควบคุม ลักษณะการจัดเรียง ตลอดจน พฤติกรรมของอิลิเมนต์ลูก เช่น การจัดตำแหน่งในแนวแกนหลัก (justify-content) การจัดตำแหน่งในแนวแกนข้าม (align-items) การควบคุมการลำดับ (flex-direction) และการควบคุมการขึ้นบรรทัดใหม่ (flex-wrap) ตัวอย่างของการกำหนด Flex Container แสดงดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 10

	CSS	HTML
1	<style>	
2	.container {	Flex Items 1
3	display: flex;	Flex Items 2
4	flex-direction: row;	Flex Items 3
5	justify-content: space-between;	
6	}	
7	</style>	

จากตัวอย่างที่ 10 ในส่วนของ CSS มีการกำหนดให้อิลิเมนต์ที่มีแอตทริบิวต์ class="container" จะถูกกำหนดให้เป็น Flex Container โดยใช้ display: flex ในการกำหนด จากนั้นมีการ

กำหนดทิศทางของการจัดเรียงอิลิเมนต์ลูกหรือ Flex Item ในแนวนอนด้วยคุณสมบัติ flex-direction: row กำหนดให้จัดกระจายช่องว่างระหว่างองค์ประกอบลูกด้วยคุณสมบัติ justify-content: space-between เป็นต้น

2) Flex Items

Flex Items เป็นอิลิเมนต์ลูกที่อยู่ภายใน Flex Container ที่ต้องการจัดเรียงตามแนวแกนที่กำหนด อิลิเมนต์ใด ๆ ก็ตามที่อยู่ Flex Container และไม่ถูกตั้งค่าคุณสมบัติ display: none; จะถือว่าเป็น Flex Item ทั้งสิ้น

Flex Items จะถูกจัดวางเรียงตามแนวแกนที่กำหนดไว้ใน Flex Container และสามารถควบคุม ขนาด ตำแหน่ง และลักษณะการจัดเรียง ของแต่ละรายการได้อย่างยืดหยุ่น ทั้งนี้สามารถระบุพฤติกรรมเฉพาะของ Flex Item แต่ละรายการได้ด้วยคุณสมบัติเฉพาะ เช่น การกำหนดความสามารถในการขยายตัวของอิลิเมนต์เมื่อมีพื้นที่ว่างด้วยคุณสมบัติ flex-grow การกำหนดความสามารถในการหดตัวเมื่อมีพื้นที่จำกัดด้วยคุณสมบัติ flex-shrink กำหนดขนาดเริ่มต้นของอิลิเมนต์ก่อนนำไปคำนวณการจัดวางด้วยคุณสมบัติ flex-basis เป็นต้น ตัวอย่างการกำหนดพฤติกรรมของ Flex Items มีดังนี้

ตัวอย่างที่ 11

	CSS	HTML
1	.container {	
2	display: flex;	
3	height: 100px;	Flex Item 1
4	}	
5	.item1 {	
6	flex: 1;	Flex Item 2
7	background-color: lightblue;	
8	}	
9	.item2 {	
10	flex: 2;	
11	background-color: lightgreen;	
12	}	

จากตัวอย่างที่ 11 ในส่วนของ CSS มีการกำหนดให้อิลิเมนต์ที่มีแอตทริบิวต์ class="container" ทำหน้าที่เป็น Flex Container โดยใช้ display: flex ทำให้อิลิเมนต์ลูกเป็น Flex Items โดยอัตโนมัติ ในตัวอย่างนี้อิลิเมนต์ที่มีคลาสเป็น item1 ถูกกำหนดคุณสมบัติ flex: 1 หมายถึงการขยายพื้นที่ในอัตราส่วน 1 ส่วนของอิลิเมนต์แม่ ขณะที่อิลิเมนต์ที่มีคลาสเป็น item2 ถูกกำหนดคุณสมบัติ flex: 2 หมายถึงการขยายพื้นที่ในอัตราส่วน 2 ส่วนของอิลิเมนต์แม่ เมื่อเบราว์เซอร์แสดงผล Flex Container จะทำ

การจัดสรรพื้นที่ว่างที่เหลือให้กับอิลิเมนต์ลูกตามอัตราส่วน 1:2 ส่งผลให้อิลิเมนต์ที่มีคลาสเป็น item2 มีความกว้างมากกว่าอิลิเมนต์ที่มีคลาสเป็น item1 ถึงสองเท่า

2.1.2 ระบบแกนใน CSS Flexbox

หลักการของ Flexbox มุ่งเน้นการควบคุม ขนาด ตำแหน่ง และการกระจายตัวของ Flex Items ภายใน Flex Container โดยใช้ระบบแกน (Axis) เป็นพื้นฐานสำคัญในการจัดวาง โดยระบบแกนของ Flexbox แบ่งออกเป็น 2 ประเภท ได้แก่ แกนหลักและแกนข้าม

1) แกนหลัก

แกนหลัก (Main Axis) คือทิศทางที่ใช้จัดเรียง Flex Items โดยทิศทางของแกนหลักจะถูกกำหนดผ่านคุณสมบัติ flex-direction ให้กับ Flex Container ซึ่งสามารถตั้งค่าได้ 4 รูปแบบ ดังนี้

- **flex-direction: row**

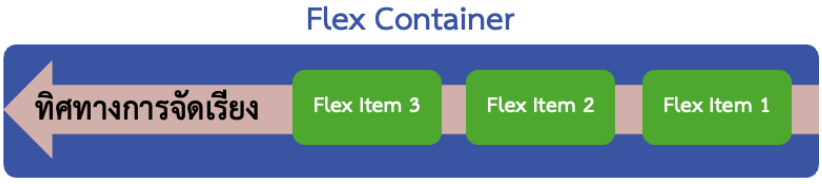
flex-direction: row เป็นการกำหนดให้ Flex Container ควบคุมการจัดเรียง Flex Items ในแนวนอนในลักษณะของแถวโดยเริ่มการวางอิลิเมนต์แรกจากซ้ายไปขวา แสดงดังภาพที่ 8.2



ภาพที่ 8.2 ทิศทางการจัดเรียง Flex Items แบบ row

- **flex-direction: row-reverse**

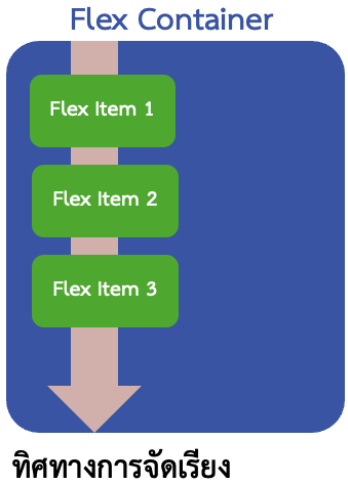
flex-direction: row-reverse เป็นการกำหนดให้ Flex Container ควบคุมการจัดเรียง Flex Items ในแนวนอนในลักษณะของแถวแต่จะกลับลำดับการวางอิลิเมนต์ลูกอันดับแรกจากขวาไปซ้ายแทน แสดงดังภาพที่ 8.3



ภาพที่ 8.3 ทิศทางการจัดเรียง Flex Items แบบ row-reverse

- **flex-direction: column**

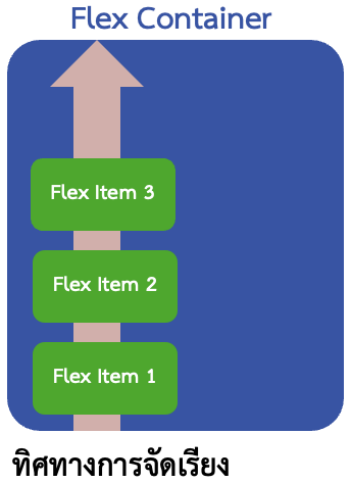
flex-direction: column เป็นการกำหนดให้ Flex Container ควบคุมการจัดเรียง Flex Items ในแนวตั้งในลักษณะของคอลัมน์โดยเริ่มการวางอิลิเมนต์แรกจากบนลงล่าง แสดงดังภาพที่ 8.4



ภาพที่ 8.4 ทิศทางการจัดเรียง Flex Items แบบ column

- **flex-direction: column-reverse**

flex-direction: column-reverse เป็นการกำหนดให้ Flex Container ควบคุมการจัดเรียง Flex Items ในแนวตั้งในลักษณะของคอลัมน์เช่นกัน แต่จะเริ่มการวางอิลิเมนต์แรกจากด้านล่างขึ้นไปด้านบนในลักษณะของคอลัมน์แสดงดังภาพที่ 8.5

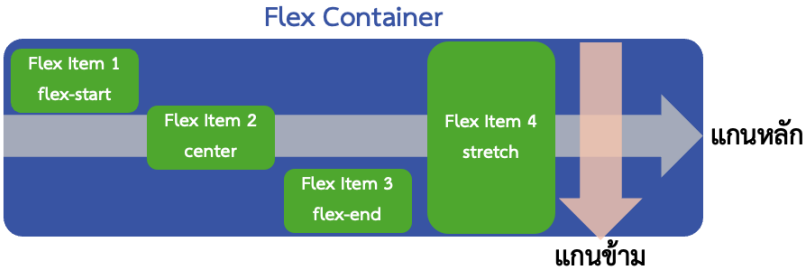


ภาพที่ 8.5 ทิศทางการจัดเรียง Flex Items แบบ column-reverse

2) แกนข้าม

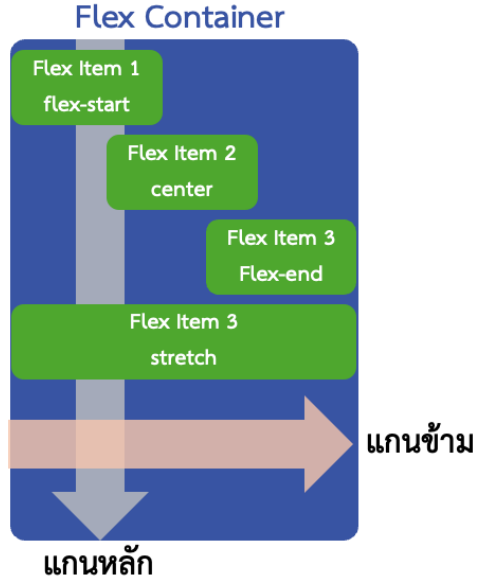
แกนข้าม (Cross Axis) หมายถึง แกนที่ตั้งฉากกับแกนหลัก โดยหลักการของ Flexbox จะใช้แกนข้ามในการจัดวางตำแหน่งของ Flex Items ในทิศทางที่ไม่ใช่แนวหลักโดยกำหนดคุณสมบัติ align-items ให้กับ Flex Items กล่าวคือ หาก Flex Container กำหนดแกนหลักเป็นแบบแนวนอนทั้ง row และ row-reverse

การจัดตำแหน่ง Flex Items ก็จะเป็นการจัดชิดด้านบน (align-items: flex-start) กึ่งกลางแนวตั้ง (align-items: center) และด้านล่าง (align-items: flex-end) ของ Flex Container รวมทั้งการยืดความสูงของ Flex Items ให้เต็มความสูงของ Flex Container (align-items: stretch) แสดงดังภาพที่ 8.6



ภาพที่ 8.6 การจัดตำแหน่งในแนวแกนข้ามเพื่อแกนหลักมีทิศทางในแนวนอน

หาก Flex Container กำหนดแกนหลักเป็นแบบแนวตั้งทั้ง column และ column-reverse การจัดตำแหน่ง Flex Items ก็จะเป็นการจัดชิดซ้าย (align-items: flex-start) กึ่งกลางแนวนอน (align-items: center) และด้านขวา (align-items: flex-end) ของ Flex Container รวมทั้งการยืดความยาวของ Flex Items ให้เต็มความยาวของ Flex Container (align-items: stretch) แสดงดังภาพที่ 8.7



ภาพที่ 8.7 การจัดตำแหน่งในแนวแกนข้ามเพื่อแกนหลักมีทิศทางในแนวตั้ง

2.2 หลักการของ CSS Grid

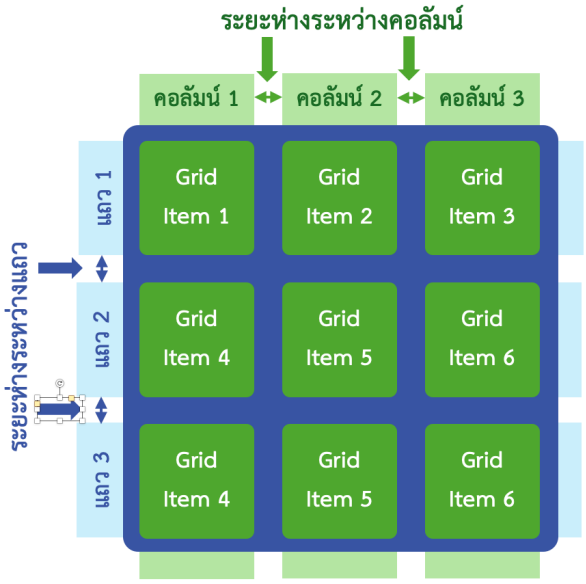
CSS Grid เป็นเทคนิคการจัดวางโครงร่างแบบสองมิติ (Two-Dimensional Layout) ในการจัดตำแหน่งของอิลิเมนต์บนหน้าเว็บทั้งในแนวนอน (Rows) และแนวตั้ง (Columns) พร้อมกัน แตกต่างจาก Flexbox

ซึ่งจัดวางได้เพียงหนึ่งมิติในแต่ละครั้ง ดังนั้น CSS Grid จึงเหมาะสำหรับการออกแบบการจัดวางที่ซับซ้อน เช่น การสร้างโครงร่างของทั้งหน้าเว็บ และแบบฟอร์ม เป็นต้น

หลักการพื้นฐานของ CSS Grid คือการแบ่งพื้นที่ของอิลิเมนต์แม่ที่เรียกว่า Grid Container ออกเป็นตารางของแถวและคอลัมน์ แล้วจัดวางอิลิเมนต์ลูกที่เรียกว่า Grid Items ลงในช่องของตารางนั้นตามตำแหน่งที่ระบุ โดยสามารถควบคุมขนาดของแถวและคอลัมน์ ความกว้างคงที่หรือแบบยืดหยุ่น ไปจนถึงการจัดตำแหน่งแบบซับซ้อนได้

2.2.1 องค์ประกอบหลักของ CSS Grid

องค์ประกอบหลักของ CSS Grid ประกอบด้วย 2 องค์ประกอบได้แก่ Grid Container และ Grid Items โดยมีโครงสร้างและการจัดวาง Grid Items ดังภาพที่ 8.8



ภาพที่ 8.8 โครงสร้างของ CSS Grid ที่มีการจัดวาง Grid Items แบบ 3 แถว 3 คอลัมน์

1) Grid Container

Grid Container คืออิลิเมนต์ที่ทำหน้าที่ห่อหุ้มและควบคุมโครงสร้างของอิลิเมนต์ลูกที่อยู่ภายใน เช่น div, section, form, หรือ article โดยอิลิเมนต์ที่จะทำหน้าที่เป็น Grid Container ต้องมีการกำหนดคุณสมบัติ display: grid หรือ display: inline-grid เพื่อกำหนดให้อิลิเมนต์นั้นๆ ทำหน้าที่เป็น Grid Container จากนั้นจะใช้คุณสมบัติ grid-template-rows และ grid-template-columns เพื่อกำหนดจำนวนแถวและจำนวนคอลัมน์ตามลำดับ เมื่อกำหนดอิลิเมนต์ให้เป็น Grid Container พร้อมรูปแบบของแถวและคอลัมน์แล้ว อิลิเมนต์ลูกภายในจะถูกจัดวางตามโครงสร้างที่ระบุไว้ใน Grid Container โดยอัตโนมัติตามลำดับก่อนหลังของอิลิเมนต์ลูก

Grid Container มีบทบาทสำคัญในการกำหนดจำนวนแถวและคอลัมน์ ขนาด และช่องว่างของโครงร่าง เช่น การใช้คุณสมบัติ grid-template-columns, grid-template-rows, และ gap เพื่อ

ควบคุมขนาดและระยะห่างของ Grid รวมถึงคุณสมบัติอื่น ๆ เช่น justify-items, align-items, grid-auto-flow และ grid-template-areas ที่ใช้ในการจัดวางตำแหน่งและทิศทางของอิลิเมนต์ลูก ตัวอย่างของการกำหนด Grid Container แสดงดังนี้

ตัวอย่างที่ 12

CSS

```
1  .grid-container {
2      display: grid;
3      grid-template-columns: repeat(3, 100px);
4      grid-template-rows: repeat(2, 100px);
5      gap: 10px;
6  }
```

HTML

```
1  <div class="grid-container">
2      <div>Item 1</div>
3      <div>Item 2</div>
4      <div>Item 3</div>
5      <div>Item 4</div>
6      <div>Item 5</div>
7      <div>Item 6</div>
8  </div>
```

จากตัวอย่างที่ 12 ในส่วนของ CSS มีการกำหนดให้อิลิเมนต์ที่มีแอตทริบิวต์ class="grid-container" ทำหน้าที่เป็น Grid Container โดยใช้ display: grid ร่วมกับการกำหนดจำนวนคอลัมน์ 3 คอลัมน์และแถว 2 แถว โดยใช้คุณสมบัติ grid-template-columns: repeat(3, 100px); และ grid-template-rows: repeat(2, 100px); พร้อมกับกำหนดระยะห่างระหว่างคอลัมน์และแถวด้วยคุณสมบัติ gap: 10px; ซึ่งจะทำให้ Grid Items ภายในถูกจัดวางให้อยู่ในลักษณะกริด 3 คอลัมน์ 2 แถว และมีระยะห่างระหว่างคอลัมน์และแถวเท่ากัน

2) Grid Items

Grid Item คืออิลิเมนต์ลูกที่อยู่ภายใน Grid Container ซึ่งจะได้รับผลจากการจัดวางตามโครงสร้างกริดโดยอัตโนมัติ โดยทุกอิลิเมนต์ที่อยู่ภายใน Grid Container จะถือว่าเป็น Grid Item ทั้งสิ้น เว้นแต่จะถูกตั้งค่า display: none; หรือไม่แสดงผล

Grid Item สามารถจัดวางตำแหน่งในตาราง Grid ได้อย่างอิสระและยืดหยุ่น เช่น การใช้คุณสมบัติ grid-column, grid-row และ grid-area เพื่อกำหนดตำแหน่งแบบระบุช่องในตาราง Grid ได้โดยตรง หรือจะใช้การจัดแนวโดยกำหนดคุณสมบัติ justify-self และ align-self ก็ได้เช่นกัน นอกจากนี้ยังสามารถปรับขนาดหรือผสานหลายคอลัมน์หรือหลายแถวเข้าด้วยกันได้โดยการใช้ grid-column-span และ grid-row-span ดังแสดงในตัวอย่างต่อไปนี้

ตัวอย่างที่ 13

CSS

```
1  .grid-container {
2      display: grid;
3      grid-template-columns: 100px 100px 100px;
4      grid-template-rows: 100px 100px;
5      gap: 10px;
6  }
7  .item1 {
8      background-color: lightblue;
9      grid-column: span 2;
10 }
11 .item2 {
12     background-color: lightgreen;
13 }
```

HTML

```
1  <div class="grid-container">
2      <div class="item1">Item 1</div>
3      <div class="item1">Item 2</div>
4      <div>Item 3</div>
5      <div>Item 4</div>
6      <div>Item 5</div>
7  </div>
```

จากตัวอย่างที่ 13 ได้กำหนดให้อิลิเมนต์ที่มีแอตทริบิวต์ class="grid-container" เป็น Grid Container (CSS บรรทัดที่ 2) และมีการกำหนดคอลัมน์จำนวน 3 คอลัมน์กว้างคอลัมน์ละ 100px (CSS

บรรทัดที่ 3) มีแถวจำนวน 2 แถวสูงแถวละ 100px เช่นกัน (CSS บรรทัดที่ 4) ทั้งนี้ อิลิเมนต์ที่มีแอตทริบิวต์ class="item1" ถูกกำหนดให้ใช้พื้นที่ 2 คอลัมน์ติดต่อกัน ด้วย grid-column: span 2; (CSS บรรทัดที่ 9) ทำให้ Grid Item นี้มีขนาดความกว้างเป็นสองเท่าของช่องกริดปกติ ในขณะที่ class="item2" จะอยู่ในตำแหน่งช่องถัดไปในกริดปกติ โดยลักษณะการจัดวางจะเป็นไปตามลำดับอัตโนมัติจากซ้ายไปขวาและบนลงล่าง

2.2.2 การกำหนดคอลัมน์และแถว

การสร้างโครงสร้างตารางใน CSS Grid จำเป็นต้องกำหนดจำนวนและขนาดของคอลัมน์และแถวให้ชัดเจนผ่านคุณสมบัติ grid-template-columns และ grid-template-rows ซึ่งเป็นหัวใจหลักของการควบคุมตำแหน่งและขนาดของ Grid Items ภายใน Grid Container โดยสามารถใช้หน่วยวัดต่าง ๆ เช่น พิกเซล (px), สัดส่วน (fr), หรือคำสั่งพิเศษ เช่น auto หรือ repeat() ได้ตามต้องการ

1) การกำหนดแถว (Grid Rows)

การกำหนดแถวใน CSS Grid ใช้คุณสมบัติ grid-template-rows เพื่อระบุจำนวนและความสูงของแถวใน Grid Container ซึ่งสามารถกำหนดแถวความสูงคงที่ เช่น 100px, กำหนดให้ปรับตามเนื้อหา (auto) หรือกำหนดแบบสัดส่วน (Fraction Unit: fr) ซึ่งเป็นหน่วยวัดเฉพาะของ Grid ก็ได้

ยกตัวอย่างเช่น grid-template-rows: 100px 150px auto; หมายถึงกำหนดให้มี 3 แถว โดยแถวแรกสูง 100px แถวที่สองสูง 150px และแถวสุดท้ายปรับความสูงตามขนาดของเนื้อหาภายใน เป็นต้น

ตัวอย่างที่ 14

CSS

```
1  .grid-container {
2      display: grid;
3      grid-template-columns: 80px 1fr 2fr;
4      background-color: #eee;
5  }
```

HTML

```
1  <div class="grid-container">
2      <div>Item 1</div>
3      <div>Item 2</div>
4      <div>Item 3</div>
5  </div>
```

จากตัวอย่างที่ 14 ในส่วนของ CSS มีการกำหนดแถวไว้ทั้งหมด 3 แถว (CSS บรรทัดที่ 3) โดยแถวแรกสูง 80px แถวที่สองและสามไม่ได้กำหนดค่าความสูงแบบแน่นอน แต่ใช้หน่วยวัดที่เรียกว่า fr ที่

หมายถึง ส่วนของพื้นที่ที่เหลืออยู่ภายหลังจากหักค่าคงที่ทั้งหมดแล้ว เช่น ถ้าหากว่า Grid Container ที่สร้างขึ้นนี้มีความสูงรวมทั้งหมด 380px ระบบจะหักความสูงของแถวแรกออกไปก่อน คือ 80px โดยจะเหลือพื้นที่อีก 300px ที่ต้องแบ่งให้แถวที่สองและสาม เมื่อกำหนด 1fr และ 2fr ให้กับแถวที่สองและสามตามลำดับ จะทำให้ระบบแบ่งพื้นที่ 300px นี้เป็น 3 ส่วนเท่าๆ กัน (1fr+2fr = 3fr) โดยแถวที่สองจะได้พื้นที่ 1 ส่วนจาก 3 ส่วน (1/3 ของพื้นที่ที่เหลือ) ซึ่งมีค่าเป็น 100px ส่วนแถวที่สามจะได้พื้นที่ 2 ส่วนจาก 3 ส่วน (2/3 ของพื้นที่ที่เหลือ) ซึ่งมีค่าเป็น 200px เป็นต้น

2) การกำหนดคอลัมน์

การกำหนดคอลัมน์ (Grid Columns) เป็นการวางโครงสร้างของอิลิเมนต์ในแนวนอนภายใน Grid Container โดยใช้คุณสมบัติ grid-template-columns ในการกำหนดจำนวนและขนาดของแต่ละคอลัมน์ที่ต้องการใช้งาน ค่าที่กำหนดสามารถเป็นค่าคงที่ เช่น px, em, หรือหน่วยแบบยืดหยุ่น เช่น % หรือ fr ได้ตามความเหมาะสมของการออกแบบเว็บแอปพลิเคชัน คุณสมบัติ grid-template-columns จะกำหนดว่าในแต่ละแถวของ Grid Container จะมีคอลัมน์จำนวนเท่าใด และแต่ละคอลัมน์จะมีความกว้างเท่าใด

ตัวอย่างที่ 15

CSS

```
1  .grid-container {
2      display: grid;
3      grid-template-columns: 100px 200px 100px;
4      gap: 10px;
5  }
6  .grid-item {
7      text-align: center;
8      padding: 10px;
9  }
```

HTML

```
1  <div class="grid-container">
2      <div class="grid-item">1</div>
3      <div class="grid-item">2</div>
4      <div class="grid-item">3</div>
5  </div>
```

จากตัวอย่างที่ 15 ในส่วนของ CSS ได้มีการกำหนดคุณสมบัติ display: grid ให้กับอิลิเมนต์ที่มีคลาสเป็น grid-container เพื่อแสดงผลแบบ Grid Layout และกำหนดคอลัมน์จำนวน 3 คอลัมน์ ด้วยความกว้าง 100px, 200px และ 100px ตามลำดับผ่านคุณสมบัติ grid-template-columns: 100px 200px 100px; (บรรทัดที่ 3) ส่งผลให้เมื่อ Grid Items ถูกแสดงในแต่ละแถว จะถูกจัดเรียงลงในคอลัมน์ตามขนาดที่กำหนด โดยมีระยะห่างระหว่างคอลัมน์เท่ากับ 10px ตามที่กำหนดไว้ในคุณสมบัติ gap

หากต้องการให้คอลัมน์มีขนาดยืดหยุ่นตามพื้นที่ของ Grid Container สามารถใช้หน่วย fr เช่น 1fr 2fr 1fr แทนค่าคงที่ ซึ่งหมายถึงการแบ่งพื้นที่เป็น 4 ส่วน โดยคอลัมน์ที่สองจะมีขนาดกว้างเป็นสองเท่าของคอลัมน์ที่หนึ่งและสามตามลำดับ

3. CSS Framework

จากเนื้อหาก่อนหน้าได้กล่าวถึงคุณสมบัติสำคัญของ CSS สำหรับควบคุมขนาด การจัดวาง และจัดรูปแบบองค์ประกอบในหน้าเว็บ ซึ่งเป็นพื้นฐานสำคัญของการพัฒนาส่วนหน้าของเว็บแอปพลิเคชัน แต่การกำหนดกฎ CSS ด้วยตนเองในหน้าเว็บทั้งหมดที่อาจมีองค์ประกอบจำนวนมากที่ต้องรักษารูปแบบให้สอดคล้องกันในทุกๆ หน้าเว็บ อาจนำไปสู่ความซ้ำซ้อน ใช้เวลานาน และยากต่อการจัดการ โดยเฉพาะอย่างยิ่งเมื่อออกแบบให้รองรับอุปกรณ์ที่หลากหลาย การใช้ CSS Framework จึงเป็นทางเลือกที่ช่วยลดข้อจำกัดเหล่านี้

3.1 ความหมาย CSS Framework

CSS Framework คือชุดกฎของ CSS ที่มีการเขียนไว้ล่วงหน้า เพื่อช่วยอำนวยความสะดวกในการจัดรูปแบบการแสดงผลของหน้าเว็บ โดยจะรวบรวมกฎการจัดรูปแบบในรูปของคลาส (class) ที่สามารถเรียกใช้งานได้ทันที ทั้งนี้เพื่อช่วยอำนวยความสะดวกแก่ผู้พัฒนาในการเขียนคำสั่ง CSS ซ้ำ ๆ และช่วยให้การจัดวางองค์ประกอบบนหน้าเว็บเป็นไปอย่างเป็นระบบและมีความสอดคล้องในทุกหน้าเว็บของระบบเดียวกัน (Ross, 2025; Ikechukwu, 2024)

3.2 ประเภทของ CSS Framework

CSS Framework ที่ใช้ร่วมกับภาษา HTML สามารถแบ่งออกได้เป็น 2 ประเภทตามแนวทางการใช้งาน ดังนี้ (Ikechukwu, 2024; Qudah, 2024)

3.2.1 Component-Based CSS Framework

Component-Based CSS Framework เป็น CSS Framework ที่มีการจัดเตรียมชุดกฎของ CSS ไว้สำหรับองค์ประกอบของส่วนติดต่อกับผู้ใช้ (User Interface Component) ที่สามารถนำไปใช้งานได้ทันที เช่น ปุ่ม แบบฟอร์ม เมนูนำทาง เป็นต้น โดยผู้พัฒนาสามารถเรียกใช้งานผ่านชื่อคลาสของ CSS ที่กำหนดไว้ เหมาะสำหรับผู้ที่ต้องการนำองค์ประกอบต่าง ๆ ไปใช้งานโดยไม่ต้องเขียนชุดกฎของ CSS เพิ่มเติม มีจุดเด่นในด้านการสร้างเว็บได้อย่างรวดเร็ว และองค์ประกอบมีรูปแบบที่สอดคล้องกัน อย่างไรก็ตาม ผู้พัฒนาจำเป็นต้องสร้างโครงสร้าง HTML ให้สอดคล้องกับข้อกำหนดของ CSS Framework นั้น ๆ รวมทั้งองค์ประกอบจะมีรูปแบบสำเร็จ

ทำให้ปรับแต่งในระดับย่อยได้ไม่มากนัก ตัวอย่างของ CSS Framework ในประเภทนี้ เช่น Bootstrap, Bulma และ Materialize เป็นต้น

3.2.2 Utility-First CSS Framework

Utility-First CSS Framework เป็น CSS Framework ที่เน้นการใช้คลาสขนาดเล็กได้มีการเตรียมไว้ ซึ่งแต่ละคลาสจะที่ทำหน้าที่เฉพาะอย่าง เช่น กำหนดสี กำหนดขนาด กำหนดระยะขอบ เป็นต้น โดยผู้พัฒนาจะต้องนำคลาสขนาดเล็กเหล่านี้มาใช้สร้างเป็นองค์ประกอบต่าง ๆ ที่จะแสดงในหน้าเว็บด้วยตนเอง เช่น ปุ่ม ตาราง หรือแบบฟอร์ม เป็นต้น ไม่เหมือนกับ Component-Based CSS Framework ที่ผู้พัฒนาสามารถเรียกใช้องค์ประกอบนั้นๆ ได้ทันที Utility-First CSS Framework นี้ช่วยให้ผู้พัฒนาสามารถควบคุมการจัดรูปแบบในระดับย่อยขององค์ประกอบ (Atomic Level) ได้เอง และไม่จำเป็นต้องสร้างโครงสร้างอิลิเมนต์ของ HTML ตามที่ CSS Framework นั้นกำหนด ทำให้องค์ประกอบที่สร้างขึ้นเป็นไปตามที่ผู้พัฒนากำหนดได้มากที่สุดซึ่งเป็นจุดเด่นที่สำคัญของ CSS Framework นี้ ตัวอย่างของ Utility-First CSS Framework เช่น Tailwind CSS และ Tachyons เป็นต้น

4. การใช้งาน Bootstrap สำหรับสร้างองค์ประกอบในเว็บแอปพลิเคชัน

จากประเภทของ CSS Framework ข้างต้น ในหนังสือเล่มนี้จะแนะนำการใช้ Bootstrap ในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากเป็น CSS Framework ที่ได้รับความนิยม ผู้พัฒนาสามารถเลือกใช้องค์ประกอบเว็บได้ทันที ง่ายต่อการเรียนรู้ ช่วยลดเวลาในการเขียนกฎของ CSS ได้ อีกทั้งมีเอกสารสนับสนุนการเรียนรู้และชุมชนนักพัฒนาที่สามารถใช้เป็นแหล่งข้อมูลสำหรับศึกษาเพิ่มเติมเป็นจำนวนมาก

4.1 ความรู้เบื้องต้นเกี่ยวกับ Bootstrap

Bootstrap เป็น CSS Framework ที่จัดอยู่ในประเภท Component-Based CSS Framework ซึ่งได้มีการจัดเตรียมชุดกฎของ CSS สำหรับองค์ประกอบของส่วนติดต่อผู้ใช้ที่พร้อมใช้งานได้ทันที เช่น ปุ่ม ตาราง แบบฟอร์ม ส่วนนำทาง และองค์ประกอบอื่น ๆ โดยผู้พัฒนาสามารถเรียกใช้งานได้ด้วยการกำหนดสร้างโครงสร้าง HTML และกำหนดชื่อคลาสที่ Bootstrap กำหนดไว้ให้กับอิลิเมนต์ ทำให้สามารถสร้างหน้าเว็บที่มีความสวยงามและมีความสอดคล้องกันได้อย่างรวดเร็ว

Bootstrap ถูกสร้างขึ้นในช่วงกลางปีพ.ศ. 2553 โดยนักออกแบบและนักพัฒนาจากบริษัท Twitter โดยในระยะแรก Bootstrap มีชื่อว่า Twitter Blueprint และถูกใช้เป็นคู่มือมาตรฐานในการพัฒนาระบบภายในของบริษัท ต่อมาในวันที่ 19 สิงหาคม พ.ศ. 2554 ได้มีการเผยแพร่เป็นโอเพนซอร์สอย่างเป็นทางการ และกลายเป็นหนึ่งในโครงการ CSS Framework ที่มีผู้ใช้งานและมีส่วนร่วมในการพัฒนาเป็นจำนวนมาก (Bootstrap, n.d.a)

Boostrap มีพัฒนาการมาอย่างต่อเนื่องโดยนับตั้งแต่การเผยแพร่ให้ใช้งานโดยทั่วไป ซึ่งมีการปรับปรุงและเผยแพร่เป็น Bootsrap เวอร์ชันต่าง ๆ ซึ่งการเรียกชื่อเวอร์ชันของ Bootstrap นั้น ใช้หลักการเรียกแบบ Semantic Versioning (SemVer) ซึ่งมีโครงสร้างเป็น Major.Minor.Patch โดย Major version หมายถึงการ

เปลี่ยนแปลงที่อาจมีผลกระทบต่อการใช้งานเดิม (Breaking Change) Minor version หมายถึงการเพิ่มฟีเจอร์ใหม่ที่เข้ากันได้กับเวอร์ชันก่อนหน้า และ Patch version หมายถึงการแก้ไขข้อผิดพลาดโดยไม่เปลี่ยนพฤติกรรมการทำงานหลัก พัฒนาการที่สำคัญของ Bootstrap ในแต่ละรุ่นมีดังนี้

- Bootstrap 2 มีการเพิ่มความสามารถในการออกแบบเว็บแบบตอบสนองเพื่อรองรับการแสดงผลในอุปกรณ์ที่มีขนาดหน้าจอที่แตกต่างกันและระบบโครงร่างแบบ 12-Columns Grid
- Bootstrap 3 มีการปรับแนวทางการใช้งานตามแนวคิด Mobile-First ซึ่งหมายถึง แนวคิดในการออกแบบและพัฒนาเว็บไซต์โดยคำนึงถึงการแสดงผลในอุปกรณ์ที่มีหน้าจอขนาดเล็กที่สุดก่อนเป็นอันดับแรก โดยเฉพาะอุปกรณ์ในกลุ่มสมาร์ตโฟน จากนั้นจึงค่อยขยายการออกแบบให้รองรับอุปกรณ์ที่มีขนาดหน้าจอใหญ่ขึ้น เช่น แท็บเล็ตและคอมพิวเตอร์ เป็นต้น
- Bootstrap 4 มีการเปลี่ยน CSS Preprocessor ซึ่งเป็นเทคโนโลยีที่ใช้ประมวลผลการเขียน CSS ที่มีการนำโครงสร้างของภาษาประมวลผลให้เป็น CSS มาตรฐาน โดยเปลี่ยนจาก LESS (Learner Style Sheets) มาเป็น SASS (Syntactically Awesome Stylesheets) แทน และมีการเพิ่มสนับสนุนการใช้ Flexbox ในการจัดวางเค้าโครง
- Bootstrap 5 ได้ทำลดการพึ่งพาการใช้ jQuery ซึ่งเป็น JavaScript Framework และใช้ JavaScript แบบมาตรฐานทั่วไปร่วมกับความสามารถของ CSS ที่มีการปรับปรุงความสามารถเพิ่มเติม นอกจากนี้ยังมีการปรับปรุงองค์ประกอบต่าง ๆ ที่มีอยู่เดิมให้มีประสิทธิภาพมากขึ้น รวมทั้งลดการรองรับการใช้งานในเบราว์เซอร์รุ่นเก่า

จากพัฒนาการของ Bootstrap ข้างต้นจะพบว่า Bootstrap มีการพัฒนาสอดรับกับการเปลี่ยนแปลงทางด้านเทคโนโลยีมาอย่างต่อเนื่องดังนั้นผู้พัฒนาควรติดตามการเปลี่ยนแปลงจากเว็บไซต์ของ Bootstrap

4.2 การใช้งาน Bootstrap

จากการที่ Bootstrap มีลักษณะเป็นไฟล์ CSS ที่มีการสร้างชุดกฎของ CSS ที่พร้อมเรียกใช้งานผ่านชื่อคลาส ดังนั้นหน้าเว็บทุกๆ หน้าที่จะเรียกใช้งานคลาสต่าง ๆ ของ Bootstrap จะต้องมีการเรียกใช้ไฟล์ CSS ของ Bootstrap และไฟล์ JavaScript ที่ต้องใช้ร่วมกับ CSS โดยมีวิธีการใช้งาน 2 วิธีดังนี้ (Bootstrap, n.d.b)

4.2.1 การใช้งาน Bootstrap ผ่านเครือข่าย CDN

การใช้งาน Bootstrap ผ่านเครือข่าย CDN เป็นวิธีที่ง่ายและรวดเร็วในการใช้งานโดยไม่ต้องดาวน์โหลดหรือติดตั้งไฟล์ใด ๆ ลงในเครื่องผู้พัฒนา วิธีนี้เหมาะสำหรับการทดลอง การเรียนรู้ และการพัฒนาเว็บแอปพลิเคชันขนาดเล็กหรือสร้างต้นแบบ เพราะช่วยลดเวลาในการติดตั้งและสามารถใช้งานร่วมกับเอกสารเว็บได้ทันที

ในการใช้งาน Bootstrap ผ่าน CDN จำเป็นต้องเชื่อมโยงกับไฟล์ CSS และ JavaScript ที่อยู่ภายนอกเครื่องผู้พัฒนา ซึ่งไฟล์ต่าง ๆ เหล่านี้จะอยู่ที่เครื่องผู้ให้บริการ Bootstrap CDN โดยการเชื่อมโยงกับไฟล์ CSS และ JavaScript จะใช้ความรู้ที่ได้กล่าวถึงในบทที่ 5 และบทที่ 7 ในหัวข้อการเชื่อมโยงไฟล์ JavaScript และ

การใช้ CSS แบบภายนอก ตามลำดับ ซึ่งการเชื่อมโยงกับไฟล์ CSS จะใช้แท็ก <link> ส่วนการเชื่อมโยงไฟล์ JavaScript จะใช้แท็ก <script> โดยมีตำแหน่งการวางแท็กทั้ง 2 ตามรูปแบบต่อไปนี้

รูปแบบ

```
1  <!DOCTYPE html>
2  <html lang="en">
3      <head>
4          <meta charset="UTF-8">
5          <meta name="viewport" content="width=device-width, initial-scale=1.0">
6          <title>Document</title>
7          <!-- ตำแหน่งการเชื่อมโยงไฟล์ CSS ของ Bootstrap ผ่าน CDN -->
8          <link href="URL" rel="stylesheet" >
9      </head>
10     <body>
11         <!-- ตำแหน่งการเชื่อมโยงไฟล์ JavaScript ของ Bootstrap ผ่าน CDN -->
12         <script  src="URL"></script>
13     </body>
14 </html>
```

จากรูปแบบการใช้งานข้างต้น ตามข้อกำหนดการใช้งาน Bootstrap ได้มีข้อกำหนดเบื้องต้นว่าเอกสารเว็บจำเป็นต้องมีการประกาศประเภทของเอกสารเว็บตามมาตรฐานปัจจุบันด้วย <!DOCTYPE html> ดังที่แสดงในบรรทัดที่ 1 เพื่อให้ Bootstrap แสดงผลการทำงานได้ถูกต้อง รวมทั้งการตอบสนองต่อแนวคิด Mobile-First ที่ถูกนำมาใช้เป็นแกนหลักในการทำงานของ Bootstrap จึงจำเป็นต้องมีการกำหนดขนาดของหน้าต่างมุมมองด้วยแท็ก <meta> ดังที่แสดงในบรรทัดที่ 5 ซึ่งจะช่วยให้เว็บเบราว์เซอร์สามารถปรับขนาดการแสดงผลให้เหมาะสมกับขนาดหน้าจอของอุปกรณ์

บรรทัดที่ 8 เป็นตำแหน่งของการวางแท็ก <link> เพื่อเชื่อมโยงไปยัง URL ของไฟล์ Bootstrap CSS โดย Bootstrap เวอร์ชันที่เขียนหนังสือเล่มนี้คือ 5.3.7 มี URL เป็น https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/css/bootstrap.min.css ทั้งนี้ ในกรณีที่ผู้พัฒนามีไฟล์ CSS ที่เขียนขึ้นเองและต้องการนำมาใช้ในหน้าเว็บเช่นเดียวกับ Bootstrap ผู้พัฒนาควรเชื่อมโยงไฟล์ CSS ของผู้พัฒนาหลังจากที่ได้เชื่อมโยงไฟล์ Bootstrap CSS เพื่อให้เว็บเบราว์เซอร์ใช้ชุดกฎของ CSS จากการกำหนดของผู้พัฒนาในกรณีที่มีการกำหนดชุดกฎซ้ำกัน

บรรทัดที่ 12 เป็นตำแหน่งของการวางแท็ก <script> เพื่อเชื่อมโยงไปยัง URL ของไฟล์ Bootstrap JavaScript โดย URL ของไฟล์ดังกล่าวที่ใช้ร่วมกับ Bootstrap 5.3.7 คือ https://cdn.jsdelivr.net/npm/bootstrap@5.3.7/dist/js/bootstrap.bundle.min.js ตำแหน่งของการเชื่อมโยงนี้จะอยู่ก่อนแท็กปิด </body> เสมอ ทั้งนี้หากผู้พัฒนามีไฟล์ JavaScript ที่เขียนขึ้นมาเอง ลำดับการเชื่อมโยงไฟล์ JavaScript จะเริ่มจาก Bootstrap JavaScript จากนั้นจึงเป็น JavaScript ที่ผู้พัฒนาเขียนขึ้นมาเอง เพื่อให้การประมวลผลมีลำดับที่ถูกต้อง

ข้อดีของการใช้ Bootstrap ผ่าน CDN คือการที่ไม่ต้องดาวน์โหลดไฟล์มาจัดเก็บไว้ในเครื่อง สามารถปรับเปลี่ยนเวอร์ชันของ Bootstrap ที่ต้องการใช้งานได้ง่ายด้วยการเปลี่ยน URL และในบางกรณีอาจช่วยเพิ่มประสิทธิภาพในการโหลดหน้าเว็บ เนื่องจากไฟล์จาก CDN มักถูกจัดเก็บไว้ในระบบของผู้ให้บริการหรือเครื่องของผู้ใช้โดยเพื่อนำไปใช้ซ้ำโดยไม่ต้องโหลดจากเครื่องผู้ให้บริการ (Caching) จึงช่วยลดระยะเวลาในการโหลดหน้าเว็บ และลดการใช้ทรัพยากรเครือข่ายลงได้

การใช้วิธีนี้อาจมีข้อจำกัดบางประการ เช่น ไม่สามารถทำงานได้ในสภาวะที่ไม่มีการเชื่อมต่ออินเทอร์เน็ต (Offline) และมีความเสี่ยงด้านความปลอดภัยเนื่องจากต้องพึ่งพาไฟล์จากแหล่งภายนอก นอกจากนี้ยังอาจพบปัญหาที่เกี่ยวข้องกับกลไกการด้านความปลอดภัยของเว็บเบราว์เซอร์ ซึ่งควบคุมการร้องขอข้อมูลระหว่างเว็บไซต์ที่มีต้นทาง (Origin) แตกต่างกัน เรียกว่า CORS (Cross-Origin Resource Sharing) โดยหากเซิร์ฟเวอร์ CDN ไม่อนุญาตให้ร้องขอจากต้นทางที่กำลังพัฒนา (เช่น localhost) เบราว์เซอร์อาจปฏิเสธการโหลดไฟล์ CSS หรือ JavaScript จากภายนอก จึงเป็นข้อควรพิจารณาเมื่อต้องพัฒนาในสภาพแวดล้อมภายในเครื่อง

4.2.2 การใช้งาน Boostrap ด้วยการดาวน์โหลดไฟล์

การใช้งาน Bootstrap ด้วยการดาวน์โหลดไฟล์เป็นวิธีที่ผู้พัฒนาดำเนินการจัดเก็บไฟล์ของ Bootstrap ไว้ภายในโฟลเดอร์ของโครงการเว็บแอปพลิเคชันในเครื่องของผู้พัฒนาเองโดยไม่ต้องเชื่อมต่อกับผู้ให้บริการภายนอก (CDN) ขั้นตอนการใช้ Bootstrap ด้วยการดาวน์โหลดไฟล์มีดังนี้

ขั้นตอนที่ 1) ดาวน์โหลดไฟล์

เปิดเว็บเบราว์และเข้าสู่เว็บไซต์ https://getbootstrap.com เพื่อดาวน์โหลดไฟล์ ทั้งนี้ให้เลือกดาวน์โหลดไฟล์ CSS และ JavaScript ภายใต้หัวข้อ Compiled CSS and JS เนื่องจากเป็นไฟล์ที่พร้อมใช้งานโดยไม่ต้องทำการแปลงไฟล์แต่อย่างใด เมื่อได้ไฟล์มาแล้วให้ทำการแตกไฟล์ที่มีการบีบอัดก่อน (Extract File)

ขั้นตอนที่ 2) คัดลอกไฟล์

ไฟล์ต่าง ๆ ที่ได้จากการดำเนินการในขั้นตอนที่ 1 จะถูกแยกจัดเก็บตามประเภทของไฟล์ โดยไฟล์ CSS จะถูกจัดเก็บไว้ในโฟลเดอร์ css และไฟล์ JavaScript จะถูกจัดเก็บไว้ในโฟลเดอร์ js จากกรณีศึกษาการสร้าง LAMP Stack ด้วย Docker Desktop ที่ได้นำเสนอไว้ในบทที่ 1 ซึ่งมีการออกแบบโฟลเดอร์เพื่อจัดเก็บทรัพยากรในการพัฒนาเว็บไซต์ไว้แล้ว ให้ทำการคัดลอกไฟล์ชื่อ bootstrap.min.css ที่อยู่ในโฟลเดอร์ css ไปไว้ในโฟลเดอร์ c:\webapp\source\css และไฟล์ชื่อ bootstrap.bundle.min.js ที่อยู่ในโฟลเดอร์ js ไปไว้ในโฟลเดอร์ c:\webapp\source\js ตามลำดับ

ขั้นตอนที่ 3) สร้างไฟล์เอกสารเว็บ

สร้างไฟล์เอกสารเว็บตามมาตรฐานของภาษา HTML 5 และมีรูปแบบการใช้งานเช่นเดียวกับการใช้งานผ่านเครือข่าย CDN ในหัวข้อก่อนหน้า แต่จะเปลี่ยนจาก URL เป็นชื่อไฟล์แทน โดยรูปแบบต่อไปนี้จะสมมติให้ไฟล์เอกสารเว็บที่ต้องการใช้ Bootstrap Framework ชื่อ page1.html ซึ่งจัดเก็บอยู่ในตำแหน่ง c:\webapp\source\ รูปแบบของการใช้เชื่อมโยงไฟล์ Bootstrap CSS และ Bootstrap JavaScript มีดังนี้

รูปแบบ **ไฟล์ c:\webapp\source\page1.html**

```
<!DOCTYPE  html>
<html>
    <head>
        ...
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link href="css/bootstrap.min.css" rel="stylesheet" >
    </head>
    <body>
        ...
        <script  src="js/bootstrap.bundle.min.js"></script>
    </body>
</html>
```

ข้อดีของการใช้ Bootstrap ด้วยการดาวน์โหลดไฟล์ คือผู้พัฒนาสามารถควบคุมไฟล์ที่ใช้งานได้โดยตรง สามารถใช้งานได้ในสภาพแวดล้อมแบบออฟไลน์โดยไม่ต้องพึ่งพาการเชื่อมต่ออินเทอร์เน็ตหรือเครื่องผู้ให้บริการภายนอกในระหว่างที่ทำการพัฒนา และเหมาะกับการพัฒนาเว็บแอปพลิเคชันที่ต้องการความมั่นคงปลอดภัยสูงหรืออยู่ในระบบเครือข่ายปิดภายในองค์กร นอกจากนี้ยังลดปัญหาด้าน CORS

วิธีนี้อาจมีข้อจำกัดบางประการ เช่น ผู้พัฒนาต้องดำเนินการดาวน์โหลดไฟล์ Bootstrap ด้วยตนเองทั้งในการเริ่มต้นใช้งานและเมื่อมีการอัปเดตเวอร์ชันของ Bootstrap ซึ่งจะต้องดาวน์โหลดและจัดการไฟล์ให้อยู่ในตำแหน่งจัดเก็บที่ถูกต้อง นอกจากนี้การเก็บไฟล์ไว้ในโครงการพัฒนาเว็บแอปพลิเคชันจะเพิ่มขนาดของระบบและอาจเกิดความซ้ำซ้อนหากมีหลายโครงการที่ใช้งาน Bootstrap แยกกัน อีกทั้งยังไม่สามารถใช้ประโยชน์จากการแคชไฟล์ร่วมกันเหมือนการใช้ CDN ซึ่งอาจทำให้การโหลดหน้าเว็บครั้งแรกในฝั่งผู้ใช้งานใช้เวลานานขึ้นเล็กน้อยในบางกรณี โดยเฉพาะเมื่อระบบมีขนาดใหญ่หรือองค์ประกอบบนหน้าเว็บมีความซับซ้อน

4.3 การใช้งานองค์ประกอบพื้นฐานของ Bootstrap

Bootstrap มีการจัดเตรียมองค์ประกอบของส่วนติดต่อผู้ใช้ที่ใช้งานได้ทันทีไว้หลากหลายรูปแบบ โดยออกแบบให้มีลักษณะที่เหมาะสมกับการใช้งานทั่วไป ทั้งในเชิงความสวยงามและการตอบสนองต่อขนาดของหน้าจอ (Responsive UI) องค์ประกอบเหล่านี้ถูกจัดอยู่ในรูปของคลาส CSS ที่สามารถเรียกใช้งานผ่าน HTML โดยตรง เพื่อช่วยลดเวลาในการเขียน CSS และสามารถสร้างต้นแบบเว็บแอปพลิเคชันได้อย่างรวดเร็ว โดยในหัวข้อนี้จะนำเสนอการใช้งานองค์ประกอบพื้นฐานที่พบบ่อยในเว็บแอปพลิเคชัน ดังนี้

4.3.1 การกำหนดสีตัวอักษรและพื้นหลังให้กับอิลิเมนต์

Bootstrap มีการกำหนดชุดคลาสสำเร็จรูปสำหรับการจัดสีตัวอักษรและพื้นหลังของอิลิเมนต์ โดยแบ่งออกเป็น 2 กลุ่มหลัก ได้แก่ คลาสสำหรับกำหนดสีข้อความ (text-*) และคลาสสำหรับกำหนดสีพื้นหลัง (bg-*) ซึ่งแต่ละคลาสจะสื่อความหมายเชิงบริบท เช่น สีหลัก (Primary) สีเตือน (Warning) หรือสีอันตราย (Danger) เป็นต้น โดยผู้พัฒนาสามารถนำคลาสเหล่านี้ไปใช้ร่วมกับอิลิเมนต์ใด ๆ ที่ต้องการปรับแต่งสีเพื่อสื่อความหมายหรือจัดองค์ประกอบของหน้าเว็บให้มีความโดดเด่นและอ่านง่ายมากยิ่งขึ้น

ตารางที่ 8-10 คลาสที่ใช้ในการกำหนดสีเชิงบริบทใน Bootstrap

ความหมาย/บริบท	คลาสสำหรับข้อความ (text-*)	คลาสสำหรับพื้นหลัง (bg-*)
หลัก (Primary)	text-primary	bg-primary
รอง (Secondary)	text-secondary	bg-secondary
สำเร็จ (Success)	text-success	bg-success
อันตราย (Danger)	text-danger	bg-danger
คำเตือน (Warning)	text-warning	bg-warning
ข้อมูล (Info)	text-info	bg-info
พื้นหลังอ่อน (Light)	text-light	bg-light
พื้นหลังเข้ม (Dark)	text-dark	bg-dark
ข้อความจาง (Muted)	text-muted	ไม่รองรับ
สีขาว (White)	text-white	bg-white

จากตารางที่ 8-10 แสดงรายชื่อคลาสสำหรับการกำหนดสีของตัวอักษรและพื้นหลังจะเห็นได้ว่า Bootstrap ได้จัดเตรียมคลาสสำเร็จรูปจำนวนมากให้สามารถใช้งานได้อย่างสะดวก โดยมีการตั้งชื่อคลาสตามแนวทางเชิงความหมาย (Semantic Naming) เช่น text-success, bg-warning, text-danger เป็นต้น ซึ่งช่วยให้ผู้พัฒนาสามารถเลือกใช้คลาสให้เหมาะสมกับเจตนาของข้อความหรือบริบทของข้อมูลที่ต้องการนำเสนอ เช่น การแจ้งสถานะผลลัพธ์ การเตือนผู้ใช้ หรือการแสดงข้อความสำคัญอย่างชัดเจน ตัวอย่างต่อไปนี้จะแสดงให้เห็นวิธีการนำคลาสเหล่านี้มาใช้งานร่วมกับอิลิเมนต์

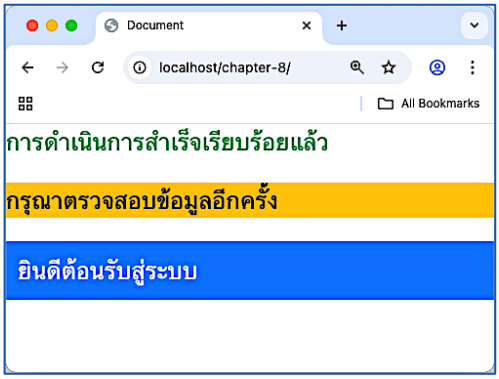
รูปแบบ

<ชื่อแท็ก class="text-* bg-*">ข้อความ</ชื่อแท็ก>

ตัวอย่างที่ 16

```
1  <p class="text-success">การดำเนินการสำเร็จเรียบร้อยแล้ว</p>
2  <p class="bg-warning text-dark">กรุณาตรวจสอบข้อมูลอีกครั้ง</p>
3  <div class="bg-primary text-white">ยินดีต้อนรับสู่ระบบ</div>
```

ผลลัพธ์ของตัวอย่างที่ 16



จากตัวอย่างที่ 16 มีการใช้อิลิเมนต์ p และ div ร่วมกับคลาสของ Bootstrap เพื่อกำหนดสีตัวอักษรและพื้นหลัง ดังนี้

- บรรทัดที่ 1 ใช้คลาส text-success กับอิลิเมนต์ p เพื่อแสดงข้อความเป็นสีเขียว ซึ่งใช้ในบริบทที่ต้องการสื่อถึงสถานการณ์ดำเนินการที่สำเร็จเรียบร้อยแล้ว
- บรรทัดที่ 2 ใช้คลาส bg-warning เพื่อกำหนดพื้นหลังสีเหลืองให้กับอิลิเมนต์ p และใช้คลาส text-dark เพื่อกำหนดให้ข้อความแสดงผลเป็นสีดำเพื่อให้อ่านง่าย เหมาะกับกรณีที่ต้องการเตือนหรือเน้นข้อควรระวัง
- บรรทัดที่ 3 ใช้อิลิเมนต์ div ร่วมกับคลาส bg-primary เพื่อกำหนดพื้นหลังสีน้ำเงินเข้ม และคลาส text-white เพื่อแสดงข้อความเป็นสีขาว

4.3.2 ปุ่ม

ปุ่ม (Button) ใน Bootstrap เป็นองค์ประกอบของส่วนติดต่อผู้ใช้ที่ถูกออกแบบให้สามารถใช้งานได้อย่างสะดวกและรองรับหลายรูปแบบการแสดงผล โดย Bootstrap กำหนดให้ใช้คลาสหลัก btn เพื่อกำหนดให้อิลิเมนต์แสดงผลในลักษณะปุ่ม และสามารถระบุรูปแบบของปุ่มเพิ่มเติมด้วยคลาส btn-* ซึ่งเรียกว่าคลาสกำหนดลักษณะปุ่ม (Button Variant Class) เช่น btn-primary, btn-danger, btn-info เป็นต้น เพื่อใช้สีใน

การสื่อความหมายหรือบริบทของการกระทำที่ปุ่มนั้นเกี่ยวข้อง ผู้พัฒนาสามารถใช้คลาสปุ่มร่วมกับอิลิเมนต์ button, a หรือ input ได้ โดยรูปแบบทั่วไปมีดังนี้:

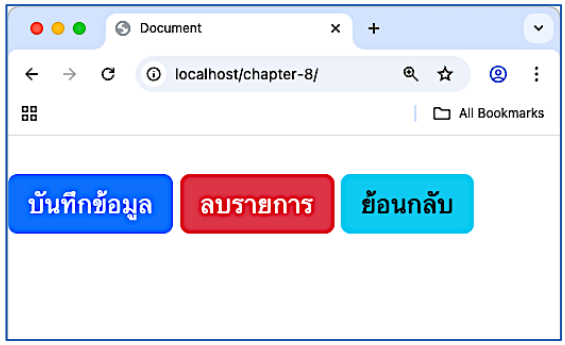
รูปแบบ

<ชื่อแท็ก type="button" class="btn ชื่อคลาสรูปแบบของปุ่ม">ข้อความบนปุ่ม</ชื่อแท็ก>

ตัวอย่างที่ 17

```
1 <button type="button" class="btn btn-primary">บันทึกข้อมูล</button>
2 <button type="button" class="btn btn-danger">ลบรายการ</button>
3 <a href="index.html" class="btn btn-info">ย้อนกลับ</a>
```

ผลลัพธ์ของตัวอย่างที่ 17



จากตัวอย่างที่ 17 ปุ่มแรกสร้างจากแท็ก <button> ใช้คลาส btn เพื่อกำหนดให้อิลิเมนต์แสดงผลในลักษณะปุ่ม และใช้ร่วมกับคลาส btn-primary เพื่อใช้ปุ่มที่มีรูปแบบพื้นหลังสีน้ำเงินเข้ม รวมทั้งมีข้อความว่า *บันทึกข้อมูล* อยู่บนปุ่ม ส่วนปุ่มที่สองสร้างจากแท็ก <button> ใช้คลาส btn ร่วมกับ btn-danger เพื่อสร้างปุ่มที่มีพื้นหลังสีแดง มีข้อความว่า *ลบรายการ* เพื่อแสดงเจตนาเตือนหรือกระทำที่มีผลกระทบต่อข้อมูล

อิลิเมนต์ a ในตัวอย่างบรรทัดที่ 3 แม้จะไม่ใช่ปุ่มโดยธรรมชาติ แต่เมื่อใช้คลาส btn จะทำให้การแสดงผลของอิลิเมนต์นี้เป็นปุ่มแทนที่จะเป็นลิงก์ตามค่าแสดงผลเริ่มต้น นอกจากนี้มีการกำหนดคลาสเป็น btn-info ร่วมด้วย ทำให้ปุ่มที่ได้มีสีพื้นหลังสีฟ้าอ่อน มีข้อความว่า *ย้อนกลับ* ซึ่งสามารถนำผู้ใช้งานกลับไปยังหน้า index.html ได้

4.3.3 การจัดตำแหน่งข้อความ

การจัดตำแหน่งข้อความ (Text Alignment) เป็นองค์ประกอบที่ใช้ควบคุมทิศทางการจัดวางข้อความภายในอิลิเมนต์ HTML ให้เหมาะสมกับบริบทของการนำเสนอเนื้อหา เช่น การจัดข้อความให้อยู่ชิดด้านซ้าย (Left-aligned) อยู่ตรงกลาง (Centered) หรืออยู่ชิดด้านขวา (Right-aligned) ซึ่ง Bootstrap ได้กำหนดคลาสสำเร็จรูปสำหรับจัดตำแหน่งข้อความโดยใช้รูปแบบ text-* ตามด้วยตำแหน่งของข้อความที่ต้องการจัดวาง

หากต้องการจัดข้อความให้ชิดด้านซ้ายของอิลิเมนต์ ให้ใช้คลาส text-start หากต้องการจัดข้อความให้อยู่ตรงกลางของพื้นที่ ให้ใช้คลาส text-center ขณะที่การจัดข้อความให้ชิดขอบด้านขวาของอิลิเมนต์ให้ใช้คลาส text-end

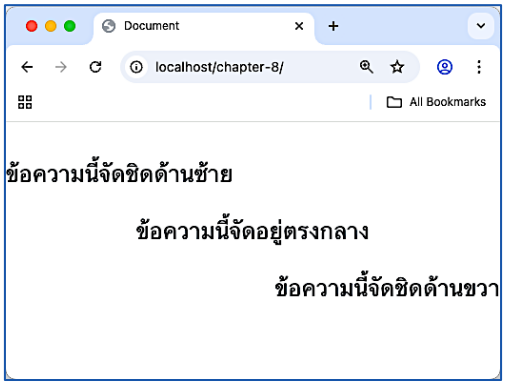
รูปแบบ

<ชื่อแท็ก class="text-*">ข้อความ</ชื่อแท็ก>

ตัวอย่างที่ 18

```
1  <p class="text-start">ข้อความนี้จัดชิดด้านซ้าย</p>
2  <p class="text-center">ข้อความนี้จัดอยู่ตรงกลาง</p>
3  <p class="text-end">ข้อความนี้จัดชิดด้านขวา</p>
```

ผลลัพธ์ของตัวอย่างที่ 18



จากตัวอย่างที่ 18 มีการใช้อิลิเมนต์ p จำนวน 3 รายการ โดยแต่ละรายการกำหนดค่าของแอตทริบิวต์ class แตกต่างกันตามตำแหน่งที่ต้องการจัดข้อความในแนวนอน ได้แก่ text-start, text-center และ text-end ตามลำดับ

ผลลัพธ์ที่ปรากฏบนเว็บเบราว์เซอร์จะพบว่าแต่ละข้อความในอิลิเมนต์ p มีตำแหน่งที่แตกต่างกันตามคลาสที่กำหนดไว้ในแอตทริบิวต์ class โดยไม่ต้องเขียนคำสั่ง CSS เพิ่มเติม ซึ่งช่วยให้การจัดข้อความในหน้าเว็บทำได้สะดวกรวดเร็วขึ้น

4.3.4 กลุ่มรายการ

กลุ่มรายการ (List Group) เป็นองค์ประกอบของ Bootstrap ที่ใช้แสดงรายการข้อมูลแบบเรียงลำดับและไม่เรียงลำดับ ให้อยู่ในรูปแบบที่มีการจัดรูปแบบเดียวกันโดยใช้คลาส list-group ร่วมกับ list-group-item เพื่อกำหนดให้อิลิเมนต์ ul หรือ div แสดงรายการที่มีลักษณะเป็นกล่องแสดงผลที่จัดเรียงตามลำดับบรรทัด และสามารถนำไปใช้แสดงรายการข้อมูลหรือเมนูย่อยได้ง่ายขึ้น

รูปแบบ

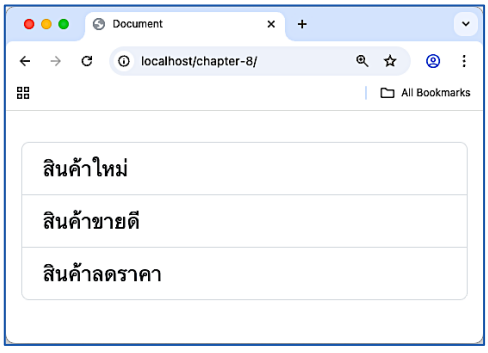
```
<ul class="list-group">
    <li class="list-group-item">รายการที่ 1</li>
    <li class="list-group-item">รายการที่ 2</li>
    <li class="list-group-item">รายการที่ 3</li>
</ul>
```

ตัวอย่างที่ 19

```
1  <ul class="list-group">
2      <li class="list-group-item">สินค้าใหม่</li>
3      <li class="list-group-item">สินค้าขายดี</li>
4      <li class="list-group-item">สินค้าลดราคา</li>
5  </ul>
```

จากตัวอย่างที่ 19 เป็นการสร้างรายการแบบไม่เรียงลำดับโดยใช้แท็ก <ul> และ <li> โดยมีการกำหนดคลาส list-group ให้กับอิลิเมนต์ เพื่อกำหนดให้รายการภายในมีลักษณะของกลุ่มรายการ และใช้อิลิเมนต์ พร้อมกับคลาส list-group-item สำหรับแต่ละรายการ เพื่อให้แสดงผลในลักษณะกล่องรายการแต่ละบรรทัดที่มีระยะห่างและกรอบเหมาะสม เว็บเบราว์เซอร์จะแสดงผลรายการทั้งสามเป็นกล่องข้อความที่เรียงลำดับในแนวตั้ง โดยแต่ละรายการมีเส้นขอบแยกชัดเจนไม่แสดงสัญลักษณ์หน้ารายการ

ผลลัพธ์ของตัวอย่างที่ 19



4.3.5 รูปภาพ

รูปภาพ (Image) เป็นองค์ประกอบพื้นฐานที่ใช้ในการแสดงภาพกราฟิกภายในเว็บแอปพลิเคชัน โดย Bootstrap มีคลาสที่ช่วยให้สามารถปรับขนาด ปรับลักษณะขอบ และจัดตำแหน่งของรูปภาพได้ง่าย โดยคลาสที่นิยมใช้ในระดับเบื้องต้น ได้แก่ img-fluid, rounded, และ mx-auto d-block

คลาส img-fluid ใช้เพื่อปรับขนาดรูปภาพให้เหมาะสมกับขนาดของอิลิเมนต์แม่ คลาส rounded ใช้เพื่อให้รูปภาพมีมุมโค้งมน และคลาสmx-auto d-block ใช้สำหรับจัดตำแหน่งรูปภาพให้อยู่กึ่งกลางแนวนอน ส่วนคลาส mx-auto d-block ใช้สำหรับจัดตำแหน่งรูปภาพให้อยู่กึ่งกลางแนวนอน

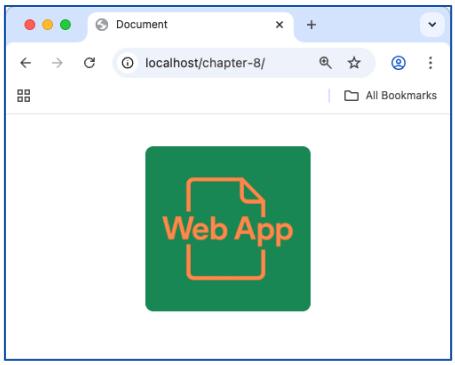
รูปแบบ

```
<img src="ไฟล์รูปภาพ" class="img-fluid rounded mx-auto d-block" alt="คำอธิบายภาพ">
```

ตัวอย่างที่ 20

```
1  <img src="webapp.png" class="img-fluid rounded mx-auto d-block" alt="โลโก้เว็บไซต์">
```

ผลลัพธ์ของตัวอย่างที่ 20



จากตัวอย่างที่ 20 เป็นการแสดงรูปภาพโดยใช้แท็ก <img> ร่วมกับคลาส img-fluid เพื่อให้รูปภาพปรับขนาดอัตโนมัติตามอุปกรณ์แสดงผล คลาส rounded ใช้เพื่อให้มีมุมโค้ง และคลาส mx-auto d-block เพื่อจัดตำแหน่งให้อยู่กึ่งกลางของพื้นที่ที่แสดงผล

4.3.6 เมนูนำทาง

เมนูนำทางหรือ Navbar ใน Bootstrap คือองค์ประกอบสำเร็จรูปที่ใช้ในการจัดทำแถบเมนูนำทางสำหรับหน้าเว็บ องค์ประกอบของเมนูนำทางพื้นฐานประกอบด้วยรูปแบบการใช้และองค์ประกอบดังนี้

รูปแบบ

```
1  <nav  class="navbar  nav-expand-{breakpoint}  คลาสกำหนดสีพื้นหลัง">
2        <div  class="container-fluid">
3              <div class="navbar-nav">
4                    <a class="nav-link active" aria-current="page"  href="#">หน้าปัจจุบัน</a
5                    <a class="nav-link" href="#">ข้อความ</a>
6                    <a class="nav-link" href="#">ข้อความ</a>
```

```
7                   <a class="nav-link disabled" aria-disabled="true">ลิงก์ที่ถูกปิดใช้งาน</a>
8               <div>
9           </div>
10  </nav>
```

จากรูปแบบการสร้างเมนูนำทางข้างต้นมีองค์ประกอบและการกำหนดคลาสที่สำคัญดังนี้

1) อิลิเมนต์หลักของเมนูนำทาง

แท็ก <nav> ในบรรทัดที่ 1 เป็นการสร้างอิลิเมนต์หลักขอเมนูนำทางที่มีคลาสสำคัญที่ต้องระบุอย่างน้อย 2 คลาส ดังนี้

- navbar เป็นคลาสหลักที่ใช้กำหนดส่วนเมนูนำทาง
- navbar-expand-{breakpoint} เป็นคลาสที่ใช้กำหนดจุดเปลี่ยนการแสดงผล ถ้าไม่มีการกำหนดคลาสนี้ การแสดงผลของลิงก์จะเป็นลักษณะคอลัมน์ในแนวตั้ง ค่า {breakpoint} นี้สามารถกำหนดโดยใช้ตัวย่อดังแสดงในตารางที่ 8-11 ซึ่งใช้แทนความกว้างของหน้าจอขนาดต่าง ๆ เช่น navbar-expand-sm หมายถึง เมนูนำทางจะเปลี่ยนการแสดงลิงก์จากแนวตั้งเป็นแนวนอน (เรียงลำดับต่อการเป็นแถว) เมื่อพื้นที่การแสดงผลมีขนาดตั้งแต่ 576px ขึ้นไป เป็นต้น

ในกรณีที่ต้องการกำหนดสีพื้นหลังของเมนูนำทางด้วยคลาสสีสำหรับพื้นหลัง (bg-ชื่อสีตามบริบท) สามารถกำหนดเป็นคลาสเพิ่มเติมได้และควรใช้คู่กับแอตทริบิวต์ data-bs-theme="dark" สำหรับพื้นหลังที่เป็นสีเข้ม และ data-bs-theme="light" สำหรับพื้นหลังที่มีสีโทนอ่อน เพื่อให้มีการเลือกใช้สีของลิงก์ที่แตกต่างจากสีของพื้นหลัง ยกตัวอย่างเช่น <nav class="navbar navbar-expand-sm bg-primary" data-bs-theme="light" > เป็นต้น

ตารางที่ 8-11 ชื่อตัวย่อของ Breakpoint

ชื่อ Breakpoint	ตัวย่อที่ใช้กำกับร่วมกับคลาส	ขนาดหน้าจอ
Extra small	ไม่ต้องระบุ	น้อยกว่า 576px
Small	sm	576px - 767px
Medium	md	768px - 991px
Large	lg	992px - 1199px
Extra large	xl	1200px - 1399px
Extra extra large	xxl	1400px ขึ้นไป

1) อิลิเมนต์ห่อหุ้มเนื้อหาของเมนูนำทาง

แท็ก ในบรรทัดที่ 2 เป็นการสร้างอิลิเมนต์ที่ใช้ห่อหุ้มเนื้อหาของเมนูนำทาง โดยจะต้องมีการระบุคลาส container-fluid เพื่อให้จัดวางลิงก์ภายในขอบเขตเมนูนำทางได้เหมาะสม

2) อิลิเมนต์ห่อหุ้มรายการลิงก์เชื่อมโยง

แท็ก ในบรรทัดที่ 3 เป็นการสร้างอิลิเมนต์ที่ใช้ห่อหุ้มรายการลิงก์เชื่อมโยง ซึ่งจะต้องกำหนดคลาสเป็น navbar-nav

3) รายการลิงก์เชื่อมโยง

รายการลิงก์เชื่อมโยงที่แสดงในบรรทัดที่ 4-7 เป็นการสร้างลิงก์ด้วยแท็ก a และแต่ละแท็กจะต้องมีการกำหนดคลาสเป็น nav-link ทั้งนี้หากลิงก์นั้นเป็นลิงก์ที่แสดงหน้าปัจจุบัน สามารถเพิ่มคลาส active เพิ่มเข้าไป และต้องใช้คู่กับแอตทริบิวต์ aria-current="page" เสมอดังแสดงในบรรทัดที่ 4 และถ้าต้องการปิดการคลิกลิงก์ใดจะต้องเพิ่มคลาส disabled และต้องใช้คู่กับแอตทริบิวต์ aira-disabled="true" ร่วมเสมอเช่นกันดังแสดงในบรรทัดที่ 7 นอกจากนี้สามารถใช้คลาส link-{ชื่อสีตามบริบท} เช่น link-primary, link-warning และ link-light เพื่อกำหนดสีของลิงก์ได้อีกด้วย

ตัวอย่างที่ 21

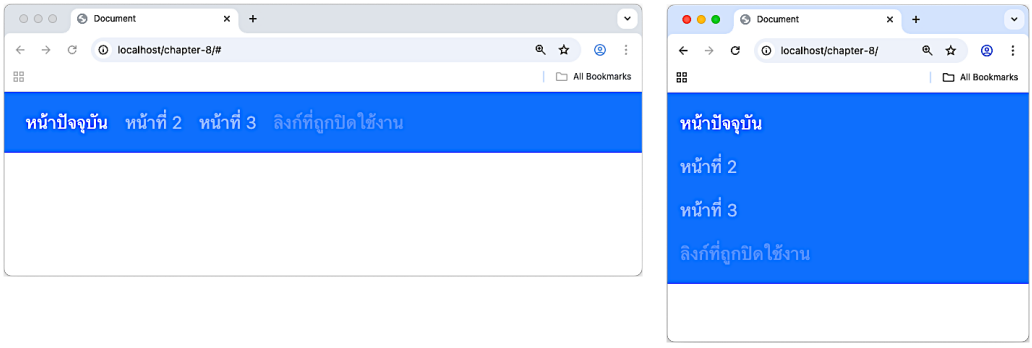
```
1  <nav class="navbar  navbar-expand-sm  bg-primary"  data-bs-theme="dark">
2        <div class="container-fluid">
3             <div class="navbar-nav">
4              <a class="nav-link active" aria-current="page"  href="#">หน้าปัจจุบัน</a
5              <a class="nav-link" href="#">หน้าที่ 2</a>
6              <a class="nav-link" href="#">หน้าที่ 3</a>
7              <a class="nav-link disabled" aria-disabled="true">ลิงก์ที่ถูกปิดใช้งาน</a>
8             </div>
9        </div>
10 </nav>
```

จากตัวอย่างที่ 21 ในบรรทัดที่ 1 เป็นการสร้างเมนูนำทางที่กำหนดให้แสดงรายการลิงก์เป็นแนวนอนเมื่อพื้นที่การแสดงผลมีความกว้างตั้งแต่ 576px ขึ้นไป ซึ่งกำหนดโดยใช้คลาส navbar-expand-sm มีการกำหนดสีพื้นหลังของเมนูนำทางเป็นโทนสีน้ำเงินด้วยคลาส bg-primary เนื่องจากเป็นพื้นหลังที่มีโทนสีเข้ม จึงมีการกำหนดคู่กับแอตทริบิวต์ data-bs-theme="dark"

ในส่วนของลิงก์บรรทัดที่ 4-7 แต่ละแท็ก <a> มีการกำหนดคลาส nav-link เพื่อแสดงถึงการเป็นลิงก์ในเมนูนำทาง อีกทั้งมีการกำหนดให้ลิงก์ลำดับแรกเป็นลิงก์ที่แสดงหน้าเว็บปัจจุบันด้วยการกำหนดคลาส active คู่กับการใช้แอตทริบิวต์ aria-current="page" นอกจากนี้ ลิงก์รายการสุดท้ายมีการปิดการใช้งานเพื่อไม่ให้

ผู้ใช้คลิกแต่ยังมีการแสดงรายการลิงก์นี้อยู่ โดยกำหนดคลาส disabled คู่กับการใช้แอตทริบิวต์ aria-disabled="true" ตามข้อกำหนดของการใช้งาน

ผลลัพธ์ของตัวอย่างที่ 21



ผลลัพธ์ของตัวอย่างที่ 21 เมนูนำทางจะแสดงเป็นแถบเต็มความกว้างของเว็บเบราว์เซอร์ซึ่งมีพื้นหลังเป็นโทนสีน้ำเงินและตัวอักษรของลิงก์เป็นสีขาว อันเป็นผลมาจากคลาส bg-primary และแอตทริบิวต์ data-bs-them="dark" โดยรูปทางซ้ายมือเป็นการขยายเว็บเบราว์เซอร์ให้มีความกว้างมากกว่า 576px จะทำให้รายการเมนูแสดงเป็นแถวเดียวกัน ส่วนรูปทางขวามือเป็นการย่อเว็บเบราว์เซอร์ให้มีความกว้างน้อยกว่า 576px ทำให้เมนูนำทางแสดงรายการลิงก์เป็นแบบแนวตั้ง นอกจากนี้ ลิงก์ลำดับที่ 1 จะมีการเน้นมากกว่าลิงก์อื่นเนื่องจากมีการควบคุมให้เป็นลิงก์ของหน้าเว็บปัจจุบัน ส่วนลิงก์สุดท้ายจะมีลักษณะจางกว่าลิงก์อื่นๆ อันเนื่องมาจากการกำหนดปิดการใช้งานด้วยคลาส disabled ร่วมกับแอตทริบิวต์ aria-disabled="true"

4.3.7 ตาราง

ตารางใน Bootstrap เป็นองค์ประกอบที่ใช้จัดแสดงข้อมูลในลักษณะแถวและคอลัมน์ โดย Bootstrap ได้จัดเตรียมคลาสสำหรับตกแต่งและควบคุมพฤติกรรมของตารางให้สามารถใช้งานได้ง่ายและแสดงผลได้สวยงาม โดยมีรูปแบบการใช้งานเบื้องต้นดังนี้

รูปแบบ

```
1  <table  class="table  คลาสปรับแต่งรูปแบบตาราง">
2      <tr class="คลาสปรับแต่งรูปแบบแถว">
3          <td class="คลาสปรับแต่งรูปแบบคอลัมน์">ข้อความ</th>
4      </tr>
5  </table>
```

จากรูปแบบการสร้างตารางข้างต้นมีองค์ประกอบและการกำหนดคลาสที่สำคัญดังนี้

1) อิลิเมนต์ตาราง

แท็ก <table> ในบรรทัดที่ 1 เป็นการสร้างอิลิเมนต์ตารางที่มีคลาสสำคัญที่ต้องระบุอย่างน้อย 1 คลาส คือ table โดย Bootstrap จะทำการแสดงตารางมีเส้นระหว่างแถวของตารางแบบรวมเส้นขอบภายในให้โดยอัตโนมัติ ทั้งนี้เพื่อตกแต่งตารางเพิ่มเติมสามารถเพิ่มคลาสสำหรับปรับแต่งรูปแบบตารางดังต่อไปนี้

- table-border เป็นคลาสที่ทำให้ตารางมีเส้นขอบทั้งภายในและภายนอกตารางแบบรวมเส้นขอบให้โดยอัตโนมัติ
- table-borderless เป็นคลาสที่ทำให้ไม่แสดงเส้นตาราง
- table-{ชื่อสีตามบริบท} เป็นคลาสที่ใช้กำหนดโทนสีพื้นหลังทั้งตาราง เช่น table-primary เป็นการทำให้พื้นหลังของตารางเป็นโทนสีฟ้า
- table-striped เป็นคลาสที่ทำให้แถวที่อยู่ในอิลิเมนต์ tbody ซึ่งเป็นส่วนข้อมูลของตารางมีการสลับสีเข้ม-อ่อนตามโทนสีของตารางที่กำหนดด้วย table-{ชื่อสีตามบริบท} ที่กล่าวถึงก่อนหน้า
- table-hover เป็นคลาสที่ทำให้สีพื้นหลังของแถวเปลี่ยนมานำเมาส์อยู่เหนือแถวนั้นๆ

2) อิลิเมนต์แถวและเซลล์ของตาราง

แท็ก <tr> ในบรรทัดที่ 2 และแท็ก <td> เป็นการสร้างแถวและเซลล์ของตาราง (ทั้งนี้จะรวมถึง th ที่ใช้สร้างเซลล์ที่เป็นหัวคอลัมน์ด้วย) โดยคลาสที่สามารถใช้กับอิลิเมนต์ทั้งสอง มีดังนี้

- table-{ชื่อสีตามบริบท} เป็นคลาสที่ใช้กำหนดโทนสีพื้นหลังเช่นเดียวกับการกำหนดสีพื้นหลังของตาราง ซึ่งหานำคลาสนี้มาใช้กับแถวหรือเซลล์ใด (tr, td และ th) จะส่งผลให้แถวหรือเซลล์นั้นแสดงพื้นหลังตามโทนสีที่กำหนด
- table-active เป็นคลาสที่ใช้สร้างความโดดเด่นหรือไฮไลท์ให้กับแถวหรือเซลล์ (ทั้ง td และ th)

ตัวอย่างที่ 22

```
1  <table class="table table-bordered table-success table-striped table-hover">
2      <thead>
3          <tr>
4           <th>คอลัมน์ 1</th>
5           <th>คอลัมน์ 2</th>
6           <th>คอลัมน์ 3</th>
7          </tr>
8      </thead>
```

```
9         <tbody >
10              <tr >
11               <td>ข้อมูลรายการที่ 1-1</td>
12  <td>ข้อมูลรายการที่ 1-2</td>
13               <td>ข้อมูลรายการที่ 1-3</td>
14              </tr>
15              <tr>
16               <td>ข้อมูลรายการที่ 2-1</td>
17               <td>ข้อมูลรายการที่ 2-2</td>
18               <td>ข้อมูลรายการที่ 2-3</td>
19              </tr>
20              <tr>
21               <td>ข้อมูลรายการที่ 3-1</td>
22               <td>ข้อมูลรายการที่ 3-2</td>
23               <td>ข้อมูลรายการที่ 3-3</td>
24              </tr>
25        </tbody>
26  </table>
```

จากตัวอย่างที่ 22 ในบรรทัดที่ 1 เป็นการสร้างตารางโดยมีการกำหนดคลาส table เพื่อให้ใช้รูปแบบของ Bootstrap และมีการกำหนดการแสดงเส้นตารางทั้งหมดด้วยคลาส table-bordered และกำหนดสีพื้นหลังของตารางเป็นโทนสีเขียวด้วยคลาส .table-success รวมทั้งมีการกำหนดการสลับสีในแต่ละแถวข้อมูลที่อยู่ในอิลิเมนต์ tbody (บรรทัดที่ 9-25) และมีการใช้คลาส table-hover เพื่อให้เกิดการเปลี่ยนสีในแถวที่มีการเลื่อนเมาส์มาไว้เหนือแถวนั้นๆ

ผลลัพธ์ของตัวอย่างที่ 22



5. สรุปท้ายบท

การกำหนดลักษณะการแสดงผลของเว็บแอปพลิเคชันด้วย CSS เป็นกระบวนการสำคัญที่ช่วยให้เว็บแอปพลิเคชันสามารถนำเสนอข้อมูลได้อย่างเป็นระเบียบ สวยงาม ส่งเสริมการยกระดับประสบการณ์การใช้งานของผู้ใช้ให้ดียิ่งขึ้น โดยคุณสมบัติ CSS พื้นฐาน เช่น การควบคุมขนาดอิลิเมนต์ การจัดการเส้นขอบ ระยะห่าง การจัดตำแหน่งตัวอักษร ฟอนต์ พื้นหลัง และลิงก์ ล้วนเป็นเครื่องมือสำคัญที่ช่วยให้ผู้พัฒนาสามารถปรับแต่งลักษณะของอิลิเมนต์ให้เหมาะสมกับบริบทของการใช้งานในแต่ละส่วนของหน้าเว็บ

การทำความเข้าใจหลักการพื้นฐานของ Flexbox และ CSS Grid มีความสำคัญต่อการจัดวางโครงร่างหน้าเว็บอย่างยืดหยุ่นและเป็นระบบ โดย Flexbox เหมาะสำหรับการจัดเรียงอิลิเมนต์ในแนวแกนเดียว ไม่ว่าจะเป็นแนวนอนหรือแนวตั้ง และสามารถควบคุมการจัดตำแหน่งและการเว้นระยะระหว่างอิลิเมนต์ได้อย่างแม่นยำ ส่วน CSS Grid เหมาะสำหรับการจัดวางองค์ประกอบในรูปแบบสองมิติ คือทั้งแนวนอนและแนวตั้งพร้อมกัน ซึ่งช่วยให้สามารถกำหนดโครงร่างของหน้าเว็บได้อย่างหลากหลาย เป็นพื้นฐานสำคัญที่นำไปสู่การออกแบบเว็บแอปพลิเคชันที่ตอบสนองต่ออุปกรณ์และขนาดหน้าจอที่แตกต่างกันได้อย่างมีประสิทธิภาพ

นอกจากนี้ CSS Framework ยังมีบทบาทสำคัญในการส่งเสริมประสิทธิภาพการพัฒนาเว็บแอปพลิเคชัน โดยช่วยลดความซ้ำซ้อนในการเขียนคำสั่ง CSS และสนับสนุนการออกแบบที่เป็นมาตรฐานด้วยองค์ประกอบสำเร็จรูปที่สามารถนำไปประยุกต์ใช้ได้ทันที เช่น ปุ่ม กลุ่มรายการ เมนูนำทาง ตาราง การจัดตำแหน่งข้อความ และสีพื้นหลัง ส่งผลให้ผู้พัฒนาสามารถสร้างส่วนติดต่อผู้ใช้ที่มีความสม่ำเสมอและเหมาะสมต่อการใช้งานในบริบทต่าง ๆ ได้อย่างรวดเร็ว

ความเข้าใจในแนวทางการใช้ CSS ควบคู่กับองค์ประกอบพื้นฐานของ Bootstrap จะช่วยให้ผู้พัฒนามีทักษะในการสร้างเว็บแอปพลิเคชันที่ทั้งมีประสิทธิภาพและมีความสวยงาม ซึ่งเป็นพื้นฐานสำคัญต่อการพัฒนาในระดับที่สูงขึ้น

6. กิจกรรมท้ายบท

6.1 จงอธิบายหน้าที่ของคุณสมบัติ CSS ที่ใช้ในการจัดรูปแบบอิลิเมนต์ ประกอบด้วย การควบคุมขนาด เส้นขอบ ระยะห่างภายใน ระยะห่างภายนอก การจัดตำแหน่งข้อความ แบบตัวอักษร พื้นหลัง ลิงก์ และตาราง พร้อมยกตัวอย่างประกอบ

6.2 จงเขียนรหัส HTML และ CSS เพื่อสร้างอิลิเมนต์ div จำนวน 2 อิลิเมนต์ โดยกำหนดขนาดความสูง/ความกว้าง (width/height) เส้นขอบ (border) ระยะห่างภายใน (padding) และระยะห่างภายนอก (margin) ให้แตกต่างกัน พร้อมอธิบายผลลัพธ์ที่เกิดขึ้นกับอิลิเมนต์ทั้งสองตามหลักการ Box Model

6.3 จงอธิบายหลักการจัดวางอิลิเมนต์ด้วย CSS Flexbox โดยกล่าวถึงแนวคิดของแกนหลัก (Main Axis) และแกนข้าม (Cross Axis) ทิศทางการจัดวาง (flex-direction) และการจัดตำแหน่ง (justify-content และ align-items) พร้อมยกตัวอย่างกรณีที่ควรใช้ Flexbox ในการพัฒนาเว็บแอปพลิเคชัน

6.4 จงอธิบายหลักการจัดวางอิลิเมนต์ด้วย CSS Grid โดยกล่าวถึงแนวคิดของแถว (Row) คอลัมน์ (Column) และพื้นที่ (Grid Area) พร้อมยกตัวอย่างกรณีที่ควรเลือกใช้ Grid แทน Flexbox

6.5 จงอธิบายความหมายและประโยชน์ของ CSS Framework

6.6 Bootstrap CSS Framework มีหลักการทำงานร่วมกับ HTML อย่างไร? จงอธิบายแนวทางการใช้ Bootstrap โดยอ้างอิงจากกลไกการทำงานของคลาสใน HTML และระบบ Grid ของ Bootstrap

6.7 จงสร้างเอกสาร HTML ชื่อ booking.html และใช้ Bootstrap CSS Framework เพื่อสร้างหน้าเว็บแสดงข้อมูลระบบจองห้องพัก โดยให้มีโครงสร้างและองค์ประกอบใกล้เคียงกับภาพที่ 8.9 ทั้งนี้ไม่จำเป็นต้องจัดตำแหน่งหรือสีให้เหมือนภาพทุกประการ แต่ต้องใช้องค์ประกอบของ Bootstrap ให้ถูกต้องตามที่ศึกษาในบท โดยมีข้อกำหนดดังนี้



ภาพที่ 8.9 ผลลัพธ์ที่คาดหวังในการใช้ Bootstrap CSS Framework

6.7.1 สร้างส่วนหัวของหน้าเว็บ โดยใช้คลาส .container หรือ .container-fluid และแสดงข้อความชื่อระบบว่า “ระบบจองห้องพัก” พร้อมข้อความภาษาอังกฤษขนาดเล็กว่า “Hotel Room Booking System”

6.7.2 สร้างปุ่มจำนวน 2 ปุ่ม ได้แก่ “จองห้อง” และ “ดูรายการจอง” โดยใช้คลาส .btn และคลาสสีของ Bootstrap เช่น .btn-primary, .btn-secondary หรือ .btn-outline-primary

6.7.3 สร้างตารางแสดงรายการห้องพักจำนวนอย่างน้อย 3 แถว โดยใช้คลาส .table และ .table-striped และกำหนดให้ตารางมีคอลัมน์ดังนี้: ลำดับ, หมายเลขห้อง, ประเภทห้อง, ราคา/คืน (บาท) และ สถานะ

6.7.4 กำหนดข้อมูลในตารางให้มีอย่างน้อย 3 รายการ เช่น ห้องเดี่ยว ห้องคู่ และห้องสวีท พร้อมกำหนดสถานะของห้อง เช่น “ว่าง” หรือ “ไม่ว่าง”

6.7.5 ใช้คลาส Bootstrap เพิ่มเติมเพื่อปรับลักษณะของตารางหรือสถานะของห้องอย่างน้อย 1 รายการ เช่น .table-bordered, .table-hover, .badge, .text-bg-success หรือ .text-bg-danger ตามความเหมาะสม

6.7.6 ใช้การจัดวางโครงร่างด้วย Flex หรือ Grid เพื่อจัดวางส่วนหัว ส่วนปุ่ม และส่วนตาราง



ภาษา JavaScript

บทที่ 9 มุ่งเน้นการนำเสนอพื้นฐานของภาษา JavaScript ซึ่งเป็นเครื่องมือสำคัญในการเพิ่มความสามารถแบบโต้ตอบให้กับเว็บแอปพลิเคชัน โดยผู้อ่านจะได้เรียนรู้โครงสร้างภาษา กลไกการทำงาน และรูปแบบการเขียนคำสั่งที่จำเป็นสำหรับการควบคุมพฤติกรรมของเอกสาร HTML แบบไดนามิก เพื่อเป็นพื้นฐานสำหรับการเรียนรู้การจัดการข้อมูลและโต้ตอบกับหน้าเว็บด้วย JavaScript ในบทถัดไป หัวข้อหลักในบทนี้ประกอบด้วย

1. ความรู้เบื้องต้นเกี่ยวกับภาษา JavaScript
2. องค์ประกอบของภาษา JavaScript
3. ชนิดของข้อมูลในภาษา JavaScript
4. โครงสร้างควบคุม
5. สรุปท้ายบท
6. กิจกรรมท้ายบท

1. ความรู้เบื้องต้นเกี่ยวกับภาษา JavaScript

ภาษา JavaScript เป็นภาษาคอมพิวเตอร์ที่มีบทบาทสำคัญในการพัฒนาเว็บ โดยทำหน้าที่ควบคุมพฤติกรรมและการโต้ตอบของเอกสาร HTML เพื่อเพิ่มความยืดหยุ่นและความสามารถในการตอบสนองต่อผู้ใช้ เนื้อหาต่อไปจะกล่าวถึงความหมายและบทบาท พัฒนาการและมาตรฐาน ตลอดจนกลไกการแปลและการทำงานของภาษา JavaScript

1.1 ความหมายและบทบาทของภาษา JavaScript

ภาษา JavaScript เป็นภาษาคอมพิวเตอร์ที่ถูกออกแบบมาเพื่อสร้างปฏิสัมพันธ์บนเว็บเพจ เช่น การแสดงแอนิเมชัน การตอบสนองต่อเหตุการณ์ของผู้ใช้ และการปรับเปลี่ยนเนื้อหาของอิลิเมนต์ HTML แบบพลวัต (Dynamic) เป็นต้น ภาษา JavaScript เป็นหนึ่งในสามเทคโนโลยีหลักของการพัฒนาเว็บร่วมกับภาษา HTML และภาษา CSS ซึ่งภาษา HTML ทำหน้าที่จัดโครงสร้างข้อมูล ภาษา CSS ทำหน้าที่ในการจัดรูปแบบ และภาษา JavaScript ทำหน้าที่ควบคุมพฤติกรรมและการโต้ตอบของเว็บเพจ (จีราวุธ วารินทร์, 2566; W3Schools, n.d.; ศุภชัย สมพานิช, 2565)

ภาษา JavaScript สนับสนุนรูปแบบการเขียนโปรแกรมหลายแบบ (Multi-paradigm) เช่น การเขียนแบบเชิงคำสั่ง (Imperative) เชิงฟังก์ชัน (Functional) และเชิงวัตถุ (Object-oriented) เป็นต้น ในบริบทของการทำงานในเว็บเบราว์เซอร์ ภาษา JavaScript ทำงานร่วมกับเว็บเบราว์เซอร์และโครงสร้าง DOM ซึ่งเป็นตัวแทนเชิงโครงสร้างของเอกสารเว็บ เพื่อควบคุมองค์ประกอบต่าง ๆ เช่น การตอบสนองต่อเหตุการณ์ของผู้ใช้ (Event Handling) การจัดการแบบฟอร์ม และการนำทางภายในหน้าเว็บ เป็นต้น (Mozilla Developer Network, 2024)

นอกจากนี้ ภาษา JavaScript ยังสามารถนำไปใช้ในเครื่องผู้ให้บริการได้ในสภาพแวดล้อมอื่นๆ เช่น Node.js, Deno และ Apache CouchDB เป็นต้น (จีราวุธ วารินทร์, 2566; ศุภชัย สมพานิช, 2565) ซึ่งช่วยให้ผู้พัฒนาสามารถเขียนแอปพลิเคชันโดยใช้ภาษาเดียวกันทั้งฝั่งผู้ใช้และฝั่งเครื่องผู้ให้บริการ รวมทั้งทั้งสามารถเชื่อมต่อฐานข้อมูล ประมวลผลไฟล์ หรือสร้าง API ได้อีกด้วย

แม้ว่าชื่อภาษา JavaScript จะคล้ายกับภาษา Java แต่ทั้งสองภาษามีมีความแตกต่างกันอย่างมีนัยสำคัญ ทั้งในด้านแนวคิด โครงสร้าง และวัตถุประสงค์ในการใช้งาน เช่น ในประเด็นของการใช้ตัวแปรเพื่อจัดเก็บข้อมูล ภาษา JavaScript เป็นภาษาสคริปต์ที่มีระบบการกำหนดชนิดข้อมูลแบบไดนามิก (Dynamically Typed) ซึ่งผู้พัฒนาไม่จำเป็นต้องประกาศชนิดข้อมูลล่วงหน้า แต่ภาษา Java มีข้อกำหนดในการระบุชนิดข้อมูลอย่างเคร่งครัด เป็นต้น

จากบริบทของภาษา JavaScript ที่สามารถทำงานได้ทั้งในเว็บเบราว์เซอร์และเครื่องผู้ให้บริการ ในหนังสือเล่มนี้มุ่งเน้นนำเสนอการใช้งานภาษา JavaScript ในบริบทของการใช้ในเว็บเบราว์เซอร์ เพื่อจัดการองค์ประกอบของเอกสารเว็บ อย่างไรก็ตาม การเรียนรู้รูปแบบและคำสั่งต่าง ๆ สามารถนำไปใช้กับบริบทของการทำงานฝั่งเครื่องผู้ให้บริการได้ด้วยเช่นกัน

1.2 พัฒนาการและมาตรฐานของภาษา JavaScript

ภาษา JavaScript เริ่มต้นจากการพัฒนาโดย Brendan Eich นักพัฒนาของบริษัท Netscape ในปี พ.ศ. 2538 ภายใต้ชื่อ LiveScript ซึ่งถูกออกแบบให้ทำงานบนเว็บเบราว์เซอร์ Netscape Navigator ต่อมาได้เปลี่ยนชื่อเป็น JavaScript เพื่อให้สอดคล้องกับความนิยมของภาษา Java ในขณะนั้น (Deno Land Inc., 2025) แม้ทั้งสองภาษาจะมีความคล้ายคลึงกันในด้านไวยากรณ์เบื้องต้น แต่มีความแตกต่างกันอย่างสิ้นเชิงในเชิงแนวคิดและการใช้งาน

ไม่นานหลังจากนั้น Microsoft ได้พัฒนาภาษา JavaScript ในแบบฉบับของตนเองในชื่อ JScript สำหรับการใช้ในเว็บเบราว์เซอร์ Internet Explorer 3.0 ซึ่งนำไปสู่การกำหนดมาตรฐานกลางสำหรับภาษาลักษณะนี้เพื่อให้สามารถทำงานกับเว็บเบราว์เซอร์ที่แตกต่างกันได้ โดย Netscape ได้นำเสนอภาษา JavaScript ต่อองค์กร Ecma International ซึ่งเป็นองค์กรที่ทำหน้าที่กำหนดมาตรฐานเทคโนโลยีสารสนเทศและการสื่อสารในระดับสากล (ECMA International, n.d.) และในปี พ.ศ. 2540 ได้ประกาศใช้มาตรฐาน ECMA-262 ซึ่งเป็นฉบับแรกของภาษาสคริปต์ที่เรียกว่า ECMAScript เพื่อเป็นกรอบอ้างอิงกลางสำหรับผู้พัฒนาที่ต้องการใช้งานภาษานี้

การเปลี่ยนผ่านจากช่วงพัฒนาสู่การกำหนดมาตรฐานอย่างเป็นทางการนั้น เริ่มมีความชัดเจนมากขึ้นในมาตรฐาน ECMAScript เวอร์ชัน 5 (ES5) ซึ่งประกาศใช้ในปีพ.ศ. 2552 และนับเป็นรุ่นแรกที่มีการระบุหมายเลขรุ่นอย่างชัดเจน หลังจากนั้นมาตรฐาน ECMAScript 6 หรือ ES6 (ES2015) ได้รับการประกาศใช้ในปีพ.ศ. 2558 ถือเป็นจุดเปลี่ยนสำคัญที่สร้างพื้นฐานใหม่ให้กับภาษา JavaScript ทั้งในด้านโครงสร้างภาษา ความสามารถใหม่และแนวทางการเขียนโปรแกรมสมัยใหม่ อีกทั้งเป็นจุดอ้างอิงสำคัญของการพัฒนามาตรฐาน ECMAScript ในรุ่นต่อมา

ภายหลังจากมาตรฐาน ES6 เป็นต้นมา มาตรฐาน ECMAScript ได้รับการพัฒนาอย่างต่อเนื่องในลักษณะของการออกรุ่นประจำปี โดยมีการอ้างอิงตามปีที่ประกาศ เช่น ES2016, ES2017 เป็นต้น ซึ่งแต่ละรุ่นจะประกอบด้วยคุณลักษณะใหม่ ๆ หรือการปรับปรุงความสามารถของภาษาให้ทันสมัยและสอดคล้องกับแนวโน้มการพัฒนาเว็บแอปพลิเคชันในยุคปัจจุบัน ผู้พัฒนาจึงควรติดตามความเปลี่ยนแปลงของมาตรฐาน ECMAScript อย่างต่อเนื่อง เนื่องจากแต่ละรุ่นอาจมีการเพิ่มคุณลักษณะใหม่หรืออาจมีการยกเลิกหรือเลิกใช้ (Deprecate) โครงสร้างคำสั่งบางประการ การเข้าใจและประยุกต์ใช้มาตรฐานใหม่อย่างถูกต้องจะช่วยให้สามารถบำรุงรักษาเว็บแอปพลิเคชันที่พัฒนาขึ้นตามมาตรฐานเดิมให้สามารถทำงานร่วมกับสภาพแวดล้อมใหม่ได้อย่างราบรื่น และทำให้ผู้พัฒนาสามารถปรับตัวให้ทันต่อการเปลี่ยนแปลงทางเทคโนโลยีในระยะยาวได้อย่างมีประสิทธิภาพ

1.3 กลไกการแปลและการทำงานของ JavaScript

ภาษา JavaScript จัดเป็นภาษาสคริปต์ (Scripting Language) ซึ่งหมายถึงภาษาคอมพิวเตอร์ประเภทหนึ่งที่ใช้เขียนคำสั่งเพื่อควบคุมหรือดำเนินการภายในระบบหรือโปรแกรมอื่น มีลักษณะโดยรวมคล้ายกับภาษาประมวลผลทั่วไปที่ผู้พัฒนาจะใช้ภาษาที่มนุษย์เข้าใจในการเรียบเรียงเป็นชุดคำสั่ง เช่น ภาษา C หรือภาษา Java เป็นต้น แต่มีความแตกต่างอย่างมีนัยสำคัญ กล่าวคือ ภาษาที่มีกลไกการแปลลักษณะเดียวกับภาษา C หรือภาษา

Java นั้น รหัสคำสั่งที่เขียนโดยผู้พัฒนาจะถูกแปลโดยตัวแปลภาษาที่เรียกว่าคอมไพเลอร์ (Compiler) ให้เป็นชุดรหัสที่หน่วยประมวลผลกลางสามารถปฏิบัติตามได้ซึ่งเรียกว่าภาษาเครื่อง โดยชุดรหัสที่ได้นี้ไม่ได้อยู่ในรูปแบบภาษาที่มนุษย์สามารถอ่านได้ กระบวนการแปลลักษณะนี้เรียกว่า การคอมไพล์ และหากมีการเปลี่ยนแปลงใดๆ ในรหัสต้นฉบับ ก็จะต้องมีการคอมไพล์ใหม่ทุกเครื่องเพื่อให้ได้ภาษาเครื่องสำหรับการปฏิบัติการชุดใหม่

ในส่วนการแปลของภาษาสคริปต์ รหัสคำสั่งที่เขียนด้วยภาษาสคริปต์ เช่น ภาษา JavaScript หรือภาษา PHP จะถูกอ่านและแปลให้เป็นภาษาเครื่องแบบทันทีในขณะทำงาน (Interpret) โดยไม่ต้องผ่านกระบวนการแปลโดยตัวแปลภาษาให้เป็นภาษาเครื่องดังที่ได้กล่าวถึงข้างต้น ทำให้สามารถพัฒนาและปรับแก้ไขรหัสคำสั่งได้สะดวกและรวดเร็ว

ในบริบทของภาษา JavaScript ที่ทำงานภายในเว็บเบราว์เซอร์ กลไกที่ทำหน้าที่แปลและประมวลผลรหัสคำสั่งของภาษา JavaScript จะเรียกว่า JavaScript Engine ซึ่งเว็บเบราว์เซอร์แต่ละรายก็จะใช้กลไกการแปลภาษา JavaScript ที่ใช้ต่างกันไป เช่น V8, SpiderMonkey และ JavaScriptCore ดังที่ได้กล่าวถึงมาแล้วในบทที่ 1

กลไกการทำงานของภาษา JavaScript เริ่มต้นจากการที่เว็บเบราว์เซอร์โหลดหน้าเว็บและพบคำสั่งภาษา JavaScript ที่ฝังอยู่ภายในเอกสารเว็บหรือเชื่อมโยงกับไฟล์ภายนอก จากนั้นเว็บเบราว์เซอร์จะส่งคำสั่งเหล่านั้นเข้าสู่การแปลและประมวลผลโดย JavaScript Engine ซึ่งเริ่มจากการวิเคราะห์ทางไวยากรณ์เบื้องต้น (Lexical Analysis) ซึ่งทำหน้าที่แยกสคริปต์ออกเป็นองค์ประกอบพื้นฐาน ได้แก่ โทเคน (Tokens) ตัวอักขระควบคุมรูปแบบ (Format Control Characters) ตัวแบ่งบรรทัด (Line Terminators) ข้อความหมายเหตุ (Comments) และช่องว่างหรือช่องว่างแท็บ (Whitespace) ซึ่งใช้เพื่อการจัดรูปแบบรหัสคำสั่งให้ผู้พัฒนาอ่านได้ง่ายขึ้น หลังจากผ่านกระบวนการวิเคราะห์เบื้องต้นแล้ว ตัวแปลภาษาก็จะดำเนินการประมวลผลคำสั่งตามลำดับที่ผู้พัฒนากำหนดขึ้น ผลลัพธ์จากการประมวลผลก็จะถูกส่งกลับไปยังกลไกการทำงานของเว็บเบราว์เซอร์เพื่อนำไปดำเนินการส่วนอื่นๆ ต่อไป อย่างไรก็ตาม ผลลัพธ์ที่ได้จากการทำงานของสคริปต์จะหายไปเมื่อผู้ใช้ทำการโหลดหน้าเว็บใหม่ หรือเปลี่ยนเส้นทางไปยังหน้าเว็บอื่น ยกเว้นในกรณีที่มีการเขียนคำสั่งให้จัดเก็บข้อมูลไว้โดยเฉพาะ เช่น การใช้ localStorage หรือ sessionStorage เป็นต้น (Web.Dev, 2024)

2. องค์ประกอบของภาษา JavaScript

ภาษา JavaScript มีโครงสร้างที่เรียบง่ายและยืดหยุ่น สามารถเริ่มต้นเขียนคำสั่งได้ทันทีโดยไม่ต้องมีส่วนหัวหรือกำหนดค่าพิเศษเหมือนภาษาอื่น เช่น ภาษา HTML ที่ต้องมีการประกาศประเภทเอกสารไว้ที่บรรทัดแรกและมีอิลิเมนต์ html เป็นอิลิเมนต์หลัก เป็นต้น โครงสร้างคำสั่งส่วนใหญ่ของภาษา JavaScript สามารถเขียนเป็นคำสั่งเดี่ยวหรือกลุ่มของคำสั่งที่ประกอบเป็นฟังก์ชันหรือโครงสร้างควบคุมได้ โดยภาษานี้ยึดหลักการอ่านจากซ้ายไปขวาและจากบนลงล่าง เช่นเดียวกับภาษาโปรแกรมทั่วไป

2.1 ประโยคคำสั่ง

ประโยคคำสั่ง (Statement) หมายถึงหน่วยคำสั่งที่ใช้ควบคุมหรือสั่งให้หน่วยประมวลผลดำเนินการอย่างใดอย่างหนึ่ง เช่น การสั่งให้ดำเนินการสร้างตัวแปร การคำนวณ การเรียกใช้ฟังก์ชัน หรือการควบคุมทิศทางการทำงานของโปรแกรม เป็นต้น ประโยคคำสั่งเป็นองค์ประกอบสำคัญของภาษาโปรแกรมทุกประเภทรวมถึงภาษา JavaScript โดยทำหน้าที่กำหนดลำดับการทำงานของโปรแกรมตามที่ผู้พัฒนาออกแบบไว้เพื่อให้เกิดผลลัพธ์ตามวัตถุประสงค์ ภาษา JavaScript มีรูปแบบการเขียนประโยคคำสั่งที่ได้รับอิทธิพลมาจากภาษา C ภาษา Java และภาษา C++ ทำให้ภาษา JavaScript มีรูปแบบการเขียนประโยคคำสั่งตลอดจนการจัดกลุ่มคำสั่งที่คล้ายคลึงกัน (Mozilla Developer Network, 2025) โดยมีรูปแบบการเขียนประโยคคำสั่งทั่วไปและประโยคคำสั่งแบบกลุ่มดังนี้

2.1.1 การเขียนประโยคคำสั่งทั่วไป

การเขียนประโยคคำสั่งทั่วไป จะมีการแสดงจุดสิ้นสุดของประโยคคำสั่งโดยใช้เครื่องหมายอัฒภาค (;) ท้ายประโยคคำสั่ง และสามารถเขียนเรียงประโยคคำสั่งต่อกันเป็นบรรทัดเดียวได้เมื่อใช้เครื่องหมายนี้ในท้ายแต่ละประโยคคำสั่ง โดยมีรูปแบบดังนี้

รูปแบบที่ 1

ประโยคคำสั่ง 1;

... ;

ประโยคคำสั่ง n;

รูปแบบที่ 2

ประโยคคำสั่ง 1; ... ; ประโยคคำสั่ง n;

จากรูปแบบที่ 1 ของการเขียนประโยคคำสั่งเป็นการเขียนประโยคละ 1 บรรทัด ซึ่งการเขียนรูปแบบนี้อาจละเครื่องหมายอัฒภาคท้ายประโยคได้ซึ่งเป็นผลจากมีคุณลักษณะของ JavaScript ที่เรียกว่าการแทรกอัฒภาคโดยอัตโนมัติ (Automatic Semicolon Insertion: ASI) อย่างไรก็ตาม การละเครื่องหมายดังกล่าวอาจทำให้เกิดข้อผิดพลาดหรือความคลุมเครือในบางกรณีได้ ด้วยเหตุนี้ แนวปฏิบัติที่ดีในการเขียนภาษา JavaScript จึงควรระบุเครื่องหมายอัฒภาคไว้ทุกครั้งเมื่อสิ้นสุดประโยคคำสั่ง

สำหรับรูปแบบที่ 2 ของการเขียนประโยคคำสั่ง เป็นการเขียนหลายประโยคคำสั่งในบรรทัดเดียวกันซึ่งแต่ละประโยคมีจุดสิ้นสุดโดยใช้เครื่องหมายอัฒภาค แม้ว่าสามารถเขียนประโยคในรูปแบบนี้ได้ แต่จะทำให้เกิดความยุ่งยากต่อการอ่านของผู้พัฒนาและผู้ร่วมงานอื่นด้วย

อนึ่ง เพื่อให้รหัสคำสั่งในภาษา JavaScript อ่านและบำรุงรักษาได้ง่าย โดยทั่วไปนักพัฒนานิยมเขียนคำสั่งในแต่ละบรรทัดไม่ให้ยาวเกิน 80 ตัวอักษร โดยเฉพาะอย่างยิ่งในกรณีที่ประโยคคำสั่งมีความซับซ้อน หากประโยคคำสั่งมีความยาวมาก ควรขึ้นบรรทัดใหม่หลังเครื่องหมายตัวดำเนินการ หรือเครื่องหมายกำหนดค่า เช่น +, -, &&, || = เป็นต้น เพื่อให้รหัสคำสั่งอ่านง่ายและไม่ส่งผลกระทบต่อความหมายของคำสั่ง ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 1

```
1  let  fullName = firstName + " " +
           lastName + " (" +
           nickName + ")";
2  document.getElementById("answer").innerHTML =
           "ข้อความที่ใช้แสดงผลในอิลิเมนต์ที่มี id เป็น answer";
```

จากตัวอย่างที่ 1 ในบรรทัดคำสั่งแรกเป็นการขึ้นบรรทัดใหม่หลังตำแหน่งเครื่องหมาย + เนื่องจากประโยคคำสั่งมีความยาวมากเกินไป เช่นเดียวกับบรรทัดที่ 2 ซึ่งมีการขึ้นบรรทัดใหม่หลังเครื่องหมาย = อย่างไรก็ตามทั้งสองคำสั่งมีเครื่องหมายอัฒภาคปิดท้ายประโยคของตนเสมอ

2.1.2 การเขียนประโยคคำสั่งแบบกลุ่ม

การเขียนประโยคคำสั่งแบบกลุ่ม (Block Statement) เป็นการรวมประโยคคำสั่งตั้งแต่ 1 ประโยคให้อยู่ระหว่างเครื่องหมายวงเล็บปีกกา { และ } เพื่อใช้ในกรณีควบคุมทิศทางการทำงานของโปรแกรม เช่น ภายหลังคำสั่ง if, for, while เป็นต้น รวมทั้งใช้ในการกำหนดของเขตของฟังก์ชันที่ผู้พัฒนากำหนดขึ้นเอง โดยมีตัวอย่างดังนี้

ตัวอย่างที่ 2

```
1  if (price > 5000) {
2        discount = 0.05;
3        message = "คุณได้รับส่วนลด 5%";
4  }
5  function  showResult() {
6        console.log(price);
7        document.getElementById("result").innerHTML = price;
8  }
```

จากตัวอย่างที่ 2 แสดงการเขียนประโยคแบบกลุ่มโดยใช้เครื่องหมายวงเล็บปีกกา ในบรรทัดที่ 1-4 เป็นการจัดกลุ่มประโยคคำสั่งของการตรวจสอบเงื่อนไข หากค่าในตัวแปร price มีค่ามากกว่า 50 จะมีการดำเนินการตั้งแต่หลังเครื่องหมาย { ในบรรทัดที่ 1 และสิ้นสุดที่ } ในบรรทัดที่ 4

ในบรรทัดที่ 5 - 8 เสนอตัวอย่างของการกำหนดกลุ่มของประโยคคำสั่งสำหรับการดำเนินการของฟังก์ชัน showResult() เมื่อมีการเรียกใช้ฟังก์ชันนี้จะเริ่มดำเนินการตั้งแต่หลังเครื่องหมาย { ในบรรทัดที่ 5 และสิ้นสุดที่ } ในบรรทัดที่ 8

2.2 นิพจน์และค่าคงที่

นิพจน์และค่าคงเป็นส่วนประกอบที่สำคัญของประโยคคำสั่งในภาษา JavaScript โดยมีความหมายดังนี้

2.2.1 นิพจน์

นิพจน์ (Expression) หมายถึง หน่วยของรหัสคำสั่งที่ประกอบด้วยค่าคงที่ ตัวแปร และเครื่องหมายดำเนินการ ซึ่งเมื่อ JavaScript Engine นำหน่วยของรหัสคำสั่งนี้มาประมวลผลตามลำดับและตรรกะของเครื่องหมายดำเนินการแล้ว จะให้ผลลัพธ์ค่าหนึ่งซึ่งเรียกว่าค่าผลลัพธ์ของนิพจน์ และเรียกกระบวนการประมวลผลตามเครื่องหมายดำเนินการเพื่อให้ได้ผลลัพธ์นี้ว่าการประเมินค่า (Evaluation)

ค่าผลลัพธ์ของนิพจน์สามารถนำไปใช้ในประโยคคำสั่งเดียวกัน หรือส่งต่อไปยังประโยคคำสั่งอื่นผ่านการกำหนดค่าในตัวแปร หรืออาจใช้เป็นเงื่อนไขควบคุมทิศทางการทำงานของโปรแกรม ตัวอย่างของนิพจน์ เช่น 2+3 เป็นนิพจน์ที่คำนวณผลลัพธ์ออกมาเป็นเลข 5 หรือ price > 5000 เป็นนิพจน์ที่เปรียบเทียบค่าในตัวแปร price กับตัวเลข 5000 โดยจะให้ผลลัพธ์เป็นค่าตรรกะจริงหรือเท็จซึ่งขึ้นอยู่กับค่าในตัวแปร price และ a=20 เป็นนิพจน์กำหนดค่าคงที่ให้ตัวแปร a เป็นต้น

1) นิพจน์เชิงคำนวณ

นิพจน์เชิงคำนวณ (Arithmetic Expression) คือ นิพจน์ที่ประกอบด้วยค่าคงที่หรือตัวแปรร่วมกับเครื่องหมายดำเนินการทางคณิตศาสตร์เพื่อประเมินค่าผลลัพธ์เป็นตัวเลข เช่น ผลรวม ผลต่าง ผลคูณ หรือผลหาร เป็นต้น เครื่องหมายดำเนินการทางคณิตศาสตร์ของภาษา JavaScript แสดงดังตารางที่ 9-1

ตารางที่ 9-1 เครื่องหมายดำเนินการทางคณิตศาสตร์

เครื่องหมาย	ความหมาย	ตัวอย่าง
+	บวก	10 + 5 หมายถึง นำ a บวกด้วย b
-	ลบ	10 - 5 หมายถึง นำ 10 ลบด้วย 5
*	คูณ	10 * 5 หมายถึง นำ 10 คูณด้วย 5
/	หาร	10 / 5 หมายถึง นำ 10 หารด้วย 5
%	หารเอาเศษ	10 % 5 หมายถึง เศษที่เหลือจากการนำ 10 หารด้วย 5
**	ยกกำลัง	10 ** 5 หมายถึง 10 ยกกำลัง 5

ตัวอย่างที่ 3

```
1  5000+1000;
2  a * 200;
3  (5000*0.1)/100;
```

จากตัวอย่างที่ 3 เป็นตัวอย่างของประโยคคำสั่งในภาษา JavaScript ซึ่งประกอบด้วย 3 ประโยคคำสั่ง ประโยคคำสั่งแรก (บรรทัดที่ 1) ส่วนที่เป็นนิพจน์เชิงคำนวณคือ 5000+1000 ซึ่งเป็นนิพจน์ที่ประกอบด้วยค่าคงที่และเครื่องหมายดำเนินการทางคณิตศาสตร์ เมื่อ JavaScript Engine ประเมินนิพจน์โดยดำเนินการบวกค่าคงที่ทั้งสอง จะได้ค่าผลลัพธ์ของนิพจน์เป็น 6000

ประโยคคำสั่งที่ 2 (บรรทัดที่ 2) ส่วนที่เป็นนิพจน์คือ a * 200 ซึ่งเป็นนิพจน์ที่ประกอบด้วยตัวแปร ค่าคงที่และตัวดำเนินการทางคณิตศาสตร์ ค่าผลลัพธ์ของนิพจน์คือผลลัพธ์ที่ได้จากการนำค่าในตัวแปร a คูณกับ 200

ประโยคคำสั่งที่ 3 (บรรทัดที่ 3) ส่วนที่เป็นนิพจน์เชิงคำนวณคือ (5000*0.01)/100 ซึ่งเป็นนิพจน์ที่ประกอบด้วยค่าคงที่ เครื่องหมายดำเนินการ รวมทั้งเครื่องหมายควบคุมลำดับการประเมินค่า (วงเล็บเปิดและปิด) โดย JavaScript Engine จะทำการประเมินค่านิพจน์ย่อยในวงเล็บก่อน คือ 5000*0.01 ค่าที่ได้จากนิพจน์คือ 50 จะถูกนำไปใช้ในการประเมินค่าในนิพจน์หลักคือการนำไปหารด้วย 100 ผลลัพธ์ของนิพจน์นี้ทั้งหมดคือ 0.5

2) นิพจน์เปรียบเทียบ

นิพจน์เปรียบเทียบ (Comparison Expression) คือ นิพจน์ที่ใช้ในการเปรียบเทียบค่าระหว่างค่าคงที่ ตัวแปร หรือค่าผลลัพธ์จากนิพจน์อื่น โดยผลลัพธ์ที่ได้จากการเปรียบเทียบจะเป็นค่าตรรกะ (Boolean) คือ true (จริง) หรือ false (เท็จ) เท่านั้น นิพจน์เปรียบเทียบมักใช้เป็นเงื่อนไขในคำสั่งควบคุม เช่น if, while เป็นต้น หรือใช้ร่วมกับนิพจน์ตรรกะเพื่อสร้างเงื่อนไขที่ซับซ้อนยิ่งขึ้น เครื่องหมายที่ใช้ในการเปรียบเทียบในภาษา JavaScript แสดงดังตารางที่ 9-2

ตารางที่ 9-2 เครื่องหมายเปรียบเทียบ

เครื่องหมาย	ความหมาย	ตัวอย่าง
==	เท่ากับ (ไม่พิจารณาชนิดข้อมูล)	a == b หมายถึง a เท่ากับ b
===	เท่ากับ (พิจารณาชนิดข้อมูล)	a === b หมายถึง a เท่ากับ b ทั้งชนิดและค่า
!=	ไม่เท่ากัน (ไม่พิจารณาชนิดข้อมูล)	a != b หมายถึง a ไม่เท่ากับ b
!==	ไม่เท่ากัน (พิจารณาชนิดข้อมูล)	a !== b หมายถึง a ไม่เท่ากับ b ทั้งชนิดและค่า
>	มากกว่า	a > b หมายถึง a มากกว่า b
<	น้อยกว่า	a < b หมายถึง a น้อยกว่า b
>=	มากกว่าหรือเท่ากับ	a >= b หมายถึง a มากกว่าหรือเท่ากับ b
<=	น้อยกว่าหรือเท่ากับ	a <= b หมายถึง a น้อยกว่าหรือเท่ากับ b
==	เท่ากับ (ไม่พิจารณาชนิดข้อมูล)	a == b หมายถึง a เท่ากับ b

ตารางที่ 9-2 (ต่อ)

เครื่องหมาย	ความหมาย	ตัวอย่าง
===	เท่ากับ (พิจารณาชนิดข้อมูล)	a === b หมายถึง a เท่ากับ b ทั้งชนิดและค่า
!=	ไม่เท่ากัน (ไม่พิจารณาชนิดข้อมูล)	a != b หมายถึง a ไม่เท่ากับ b
!==	ไม่เท่ากัน (พิจารณาชนิดข้อมูล)	a !== b หมายถึง a ไม่เท่ากับ b ทั้งชนิดและค่า
>	มากกว่า	a > b หมายถึง a มากกว่า b
<	น้อยกว่า	a < b หมายถึง a น้อยกว่า b
>=	มากกว่าหรือเท่ากับ	a >= b หมายถึง a มากกว่าหรือเท่ากับ b
<=	น้อยกว่าหรือเท่ากับ	a <= b หมายถึง a น้อยกว่าหรือเท่ากับ b

จากตารางที่ 9-2 การเปรียบเทียบที่มีการพิจารณาชนิดข้อมูลหมายถึง การเปรียบเทียบที่ไม่เพียงแค่ตรวจสอบว่าค่าที่เปรียบเทียบกันนั้นเท่ากันหรือไม่ แต่ยังตรวจสอบด้วยว่าค่าทั้งสองมีชนิดข้อมูลตรงกันด้วยหรือไม่ เช่น '5' === 5 จะให้ผลลัพธ์เป็น false เนื่องจาก '5' เป็นข้อมูลชนิดข้อความแต่ค่าที่นำมาเปรียบเทียบคือ 5 เป็นข้อมูลชนิดตัวเลข จึงไม่ถือว่าเท่ากันทั้งในแง่ของค่าและชนิดข้อมูล ในทางกลับกัน '5' == 5 จะให้ผลลัพธ์เป็น true เนื่องจากไม่สนใจว่าข้อมูลเป็นชนิดใด โดย JavaScript Engine จะทำการแปลงค่าตัวอักษรให้เป็นตัวเลขซึ่งจะได้ผลลัพธ์เป็น 5 แล้วจึงนำมาเปรียบเทียบกับตัวเลข เครื่องหมายเปรียบเทียบที่พิจารณาทั้งค่าและชนิดของข้อมูลเรียกว่าเครื่องหมายเปรียบเทียบแบบเข้มงวด (Strict Comparation Operator) ในขณะที่เครื่องหมายเปรียบเทียบที่พิจารณาเฉพาะค่าเพียงอย่างเดียวเรียกว่าเครื่องหมายเปรียบเทียบแบบไม่เข้มงวด (Loose Comparation Operator)

ตัวอย่างที่ 4

```
1  555 == '500';
2  if(price > 500) { ... }
```

จากตัวอย่างที่ 4 เป็นตัวอย่างของประโยคคำสั่งในภาษา JavaScript ที่ประกอบด้วยนิพจน์เชิงเปรียบเทียบทั้งสองประโยค โดยในประโยคคำสั่งแรก (บรรทัดที่ 1) คือ 555 == '500'; เป็นนิพจน์ที่ประกอบด้วยค่าคงที่สองค่า ได้แก่ จำนวนเต็ม 555 และข้อความ ‘500’ มีการใช้เครื่องหมายเปรียบเทียบ == ซึ่งเป็นเครื่องหมายเปรียบเทียบแบบไม่เข้มงวด ทำให้ JavaScript ทำการแปลงสตริง ‘500’ ให้เป็นตัวเลขก่อนการเปรียบเทียบ ผลลัพธ์จากนิพจน์นี้คือ false เนื่องจาก 555 ไม่เท่ากับ 500

ประโยคคำสั่งที่สอง (บรรทัดที่ 2) คือ if(price > 500) { ... } เป็นประโยคคำสั่งควบคุมทิศทาง (Control Flow Statement) โดยในวงเล็บ () คือ นิพจน์เปรียบเทียบ price > 500 ซึ่งประกอบด้วยตัวแปร price ค่าคงที่ 500 และเครื่องหมายเปรียบเทียบ > ค่าผลลัพธ์จากนิพจน์จะมีค่าเป็น true หรือ false ขึ้นอยู่

กับค่าของตัวแปร price โดยจะเป็นตัวกำหนดการทำงานของคำสั่งในกลุ่ม { ... } ที่ตามหลัง if กล่าวคือ หากนิพจน์มีค่าประเมินเป็น true จึงจะเข้าสู่การดำเนินการของคำสั่งที่อยู่ระหว่างวงเล็บปีกกาหลังคำสั่ง if( )

3) นิพจน์กำหนดค่า

นิพจน์กำหนดค่า (Assignment Expression) คือ นิพจน์ที่ใช้สำหรับการกำหนดค่าคงที่หรือค่าผลลัพธ์ของนิพจน์อื่นให้กับตัวแปรโดยใช้เครื่องหมายกำหนดค่า (Assignment Operators) ซึ่งนิยมใช้กันมากที่สุดคือเครื่องหมายเท่ากับ (=) และยังมีเครื่องหมายกำหนดค่าร่วมกับการดำเนินการอื่น เพื่อย่อรูปแบบการกำหนดค่าและการคำนวณในคำสั่งเดียวกัน เครื่องหมายที่ใช้ในนิพจน์กำหนดค่าแสดงในตารางที่ 9-3

ตารางที่ 9-3 เครื่องหมายกำหนดค่า

เครื่องหมาย	ความหมาย	ตัวอย่าง	ค่าของตัวแปร
=	กำหนดค่า	x = 10	x มีค่าเป็น 10
+=	บวกและกำหนดค่า	x += 5	x มีค่าเพิ่มขึ้น 5 (x = x+5)
-=	ลบและกำหนดค่า	x -= 3	x มีค่าลดลง 3 (x = x-3)
*=	คูณและกำหนดค่า	x *= 2	x คูณด้วย 2 (x = x*2)
/=	หารและกำหนดค่า	x /= 4	x หารด้วย 4 (x = x/4)
%=	หารเอาเศษและกำหนดค่า	x %= 3	x เป็นเศษจากหาร 3 (x = x%3)
**=	ยกกำลังและกำหนดค่า	x **= 2	x ยกกำลัง 2 (x = x**2)

ตัวอย่างที่ 5

```
1  count = 10;
2  count += 5;
3  total = count * 2;
```

จากตัวอย่างที่ 5 ประโยคคำสั่งแรกในบรรทัดที่ 1 เป็นนิพจน์ที่กำหนดค่าตัวเลข 10 ให้กับตัวแปร count

ประโยคคำสั่งที่สอง (บรรทัดที่ 2) เป็นนิพจน์ที่ใช้เครื่องหมาย += เพื่อเพิ่มค่าของ count อีก 5 ซึ่งเมื่อประเมินค่าแล้ว count จะมีค่าใหม่เป็น 15 โดยนิพจน์ count += 5 มีความหมายเหมือนกับ count = count + 5 ซึ่งในเชิงกลไกการทำงานของ JavaScript ประกอบด้วย 2 นิพจน์ ได้แก่ นิพจน์เชิงคำนวณ count + 5 ซึ่งให้ค่าผลลัพธ์เป็น 15 และนิพจน์กำหนดค่า count = 15 ซึ่งนำค่าผลลัพธ์จากนิพจน์แรกมากำหนดค่าใหม่ให้กับ count

ส่วนประโยคคำสั่งที่ 3 ประกอบด้วย 2 นิพจน์คือนิพจน์เชิงคำนวณและนิพจน์กำหนดค่า โดยจะดำเนินการประเมินค่านิพจน์เชิงคำนวณคือ count * 2 เป็นอันดับแรก ทำให้ได้ค่าผลลัพธ์คือ 30 จากนั้นจึงนำค่าผลลัพธ์นี้ไปกำหนดค่าให้กับตัวแปร total

4) นิพจน์เชิงตรรกะ

นิพจน์เชิงตรรกะ (Logical Expression) คือ นิพจน์ที่ทำการประเมินค่าโดยใช้เครื่องหมายตรรกะ (Logical Operators) เพื่อให้ได้ผลลัพธ์เป็นค่าตรรกะ (Boolean) ได้แก่ true หรือ false นิพจน์เชิงตรรกะมักถูกใช้ร่วมกับนิพจน์เปรียบเทียบเพื่อสร้างเงื่อนไขที่ซับซ้อนยิ่งขึ้น เช่น การตรวจสอบเงื่อนไขมากกว่าหนึ่งรายการในประโยคคำสั่งควบคุม เช่น if, while เป็นต้น เครื่องหมายตรรกะที่ใช้ในภาษา JavaScript แสดงดังตารางที่ 9-4

ตารางที่ 9-4 เครื่องหมายตรรกะ

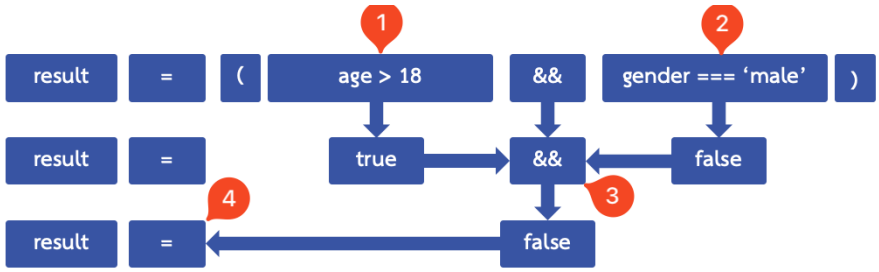
เครื่องหมาย	ความหมาย	ตัวอย่าง	ผลลัพธ์ของนิพจน์
&&	และ (AND)	true && false	true
||	หรือ (OR)	true || false	true
!	ไม่ (NOT)	!true	false

ตัวอย่างที่ 6

```
1  age = 25;
2  gender = 'female';
3  result = (age > 18 && gender === 'male');
```

จากตัวอย่างที่ 6 ประกอบด้วย 5 ประโยคคำสั่ง โดยประโยคคำสั่งแรก (บรรทัดที่ 1) เป็นการกำหนดค่าตัวแปร age ให้มีค่าเป็น 25 ประโยคคำสั่งที่ 2 (บรรทัดที่ 2) เป็นการกำหนดค่าตัวแปร gender เป็นข้อความ 'male'

ประโยคคำสั่งที่ 3 เป็นประโยคคำสั่งที่ประกอบด้วยนิพจน์เชิงเปรียบเทียบ นิพจน์เชิงตรรกะและนิพจน์กำหนดค่า โดยมีการใช้เครื่องหมาย ( ) เพื่อให้ดำเนินการนิพจน์ age > 18 && gender === 'male' ก่อน และนิพจน์ดังกล่าวก็แยกออกเป็นนิพจน์ age > 18 ซึ่งค่าของนิพจน์นี้คือ true ส่วนนิพจน์ gender === 'male' มีค่าของนิพจน์เป็น false เนื่องจากค่า gender เป็น 'female' ซึ่งไม่เท่ากับ 'male' มาพิจารณาร่วมกันจะได้เป็น true && false ทั้งนี้จากหลักตรรกศาสตร์ทำให้ค่านิพจน์นี้มีค่าเป็น false หลังจากนั้นจึงเป็นการกำหนดค่าผลลัพธ์นี้ให้กับตัวแปร result ลำดับของการประเมินนิพจน์แสดงดังภาพที่ 9.1



ภาพที่ 9.1 ตัวอย่างลำดับการประเมินค่าของนิพจน์เชิงตรรกะ

5) นิพจน์เรียกใช้ฟังก์ชัน

นิพจน์เรียกใช้ฟังก์ชัน (Function Call Expression) คือ นิพจน์ที่เกิดจากการเรียกใช้ฟังก์ชันเพื่อให้ฟังก์ชันนั้นทำงาน และผลลัพธ์ที่ได้จากการทำงานของฟังก์ชันจะถือเป็นค่าผลลัพธ์ของนิพจน์ ตัวอย่างเช่น การเรียกใช้ฟังก์ชันคำนวณทางคณิตศาสตร์หรือการเรียกใช้ฟังก์ชันภายในที่กำหนดเอง โดยรูปแบบพื้นฐานของการเรียกใช้ฟังก์ชันมีดังนี้

รูปแบบ

ชื่อฟังก์ชัน(อาร์กิวเมนต์_1, อาร์กิวเมนต์_2, ..., อาร์กิวเมนต์_n)

จากรูปแบบของการเรียกใช้ฟังก์ชัน จะประกอบด้วยชื่อฟังก์ชันที่ต้องการเรียกใช้ และภายในวงเล็บเปิด–ปิด ( ) จะเป็นตำแหน่งของอาร์กิวเมนต์ (Argument) ซึ่งหมายถึงค่าต่าง ๆ ที่ถูกส่งเข้าไปยังฟังก์ชันเพื่อใช้ในการประมวลผล โดยอาร์กิวเมนต์แต่ละค่าจะถูกคั่นด้วยเครื่องหมายจุลภาค (,) ทั้งนี้ จำนวนและลำดับของอาร์กิวเมนต์ที่ระบุจะต้องสอดคล้องกับพารามิเตอร์ที่กำหนดไว้ในประโยคประกาศฟังก์ชัน มิฉะนั้นอาจก่อให้เกิดข้อผิดพลาด หรือให้ผลลัพธ์ที่ไม่เป็นไปตามที่คาดหมาย

ตัวอย่างที่ 7

```
1  result = parseInt("123");
2  highest = Math.max(10, 25, 18);
3  console.log("ผลลัพธ์คือ", result);
```

จากตัวอย่างที่ 7 เป็นตัวอย่างของประโยคคำสั่งในภาษา JavaScript ที่ประกอบด้วยนิพจน์เรียกใช้ฟังก์ชันทั้งสามประโยค โดยในประโยคคำสั่งแรก (บรรทัดที่ 1) ซึ่ง parseInt("123") เป็นนิพจน์เรียกใช้ฟังก์ชันพื้นฐาน (Built-in function) สำหรับแปลงข้อความเป็นจำนวนเต็ม โดยข้อกำหนดของฟังก์ชัน parseInt() จะต้องรับอาร์กิวเมนต์อย่างน้อย 1 ค่า และเป็นค่าชนิดข้อความที่ต้องการแปลงให้เป็นตัวเลข ซึ่งในที่นี้คือ "123" ผลลัพธ์ที่ได้จากการประเมินนิพจน์เรียกใช้ฟังก์ชันนี้คือจำนวนเต็ม 123 และค่าดังกล่าวจะถูกนำไปกำหนดให้กับตัวแปร result ด้วยนิพจน์กำหนดค่า

ประโยคคำสั่งที่สอง (บรรทัดที่ 2) Math.max(10, 25, 18) เป็นนิพจน์เรียกใช้ฟังก์ชันชื่อ max( ) ซึ่งเป็นฟังก์ชันของอ็อบเจกต์ Math ที่ใช้สำหรับเปรียบเทียบหาค่าตัวเลขที่มากที่สุด โดยข้อกำหนดของฟังก์ชัน max( ) ได้ระบุให้รับอาร์กิวเมนต์เป็นตัวเลขจำนวนกี่ค่าก็ได้อย่างน้อยหนึ่งค่าขึ้นไป ซึ่งในที่นี้มีการส่งค่าคงที่ให้กับฟังก์ชัน max( ) จำนวน 3 ค่า ได้แก่ 10, 25 และ 18 ตามลำดับ ฟังก์ชัน max( ) จะประเมินค่าและคืนค่าตัวเลขที่มากที่สุดจากชุดตัวเลขที่รับเข้ามาคือ 25 เป็นค่าผลลัพธ์ของนิพจน์เรียกใช้ฟังก์ชัน จากนั้นค่าผลลัพธ์จะถูกกำหนดให้กับตัวแปร highest ด้วยนิพจน์กำหนดค่าที่ใช้เครื่องหมาย = ในการกำหนดค่า

ส่วนประโยคคำสั่งที่สาม (บรรทัดที่ 3) console.log("ผลลัพธ์คือ", result) เป็นนิพจน์เรียกใช้ฟังก์ชัน log() ซึ่งเป็นฟังก์ชันของอ็อบเจกต์ console ที่ใช้สำหรับแสดงผลข้อมูลไปยังคอนโซลของเว็บเบราว์เซอร์ โดยในที่นี้รับอาร์กิวเมนต์ 2 ค่า ได้แก่ ข้อความ "ผลลัพธ์คือ" และค่าของตัวแปร result ซึ่งมีค่าเป็น 123

2.2.2 ค่าคงที่

ค่าคงที่ (Fixed Value) หรือเรียกว่าค่า Literal เป็นค่าคงตัวที่ผู้พัฒนาเขียนไว้ในรหัสคำสั่งโดยตรงเพื่อแสดงค่าแบบแน่นอนที่ไม่ต้องอ้างผ่านตัวแปร และไม่เปลี่ยนแปลงระหว่างการประมวลผลของโปรแกรม โดยค่าคงที่ถือเป็นองค์ประกอบหนึ่งของนิพจน์ ซึ่งจะถูก JavaScript ประเมินค่าได้ทันทีโดยไม่ต้องอ้างอิงค่าจากตัวแปรหรือฟังก์ชันอื่น เช่น จากประโยคคำสั่ง a =5; ตัวเลข 5 คือค่า Literal ที่ผู้พัฒนาเป็นผู้ระบุโดยตรงในรหัสคำสั่ง เป็นต้น ค่าเหล่านี้ผู้พัฒนาเป็นผู้พิมพ์ระบุในรหัสคำสั่งโดยตรง ไม่ได้เกิดจากการประเมินค่าหรือการประมวลผลใดๆ ค่าคงที่ในภาษา JavaScript สามารถแบ่งออกได้ตามประเภทของข้อมูลพื้นฐาน โดยมีลักษณะและรูปแบบแตกต่างกันตามประเภทของข้อมูล ดังนี้

1) ค่าคงที่ชนิดตัวเลข

ค่าคงที่ชนิดตัวเลข (Numeric Literals) ในภาษา JavaScript หมายถึงค่าที่เป็นตัวเลขซึ่งระบุไว้ในรหัสคำสั่งโดยตรง สามารถจำแนกได้เป็น 2 กลุ่ม ได้แก่ ค่าคงที่จำนวนเต็ม และค่าคงที่จำนวนจริง

- **ค่าคงที่จำนวนเต็ม**

ค่าคงที่จำนวนเต็มสามารถระบุได้ในหลายระบบเลข ได้แก่ เลขฐานสิบ (Decimal) เขียนโดยใช้ตัวเลข 0–9 โดยตรง เช่น 0 , 117 เป็นต้น เลขฐานแปด (Octal) เขียนเริ่มต้นด้วย **0**o แล้วตามด้วยระบบเลขในฐานแปด เช่น **0**o527 เป็นต้น เลขฐานสิบหก (Hexadecimal) เขียนเริ่มต้นด้วย **0**x แล้วตามด้วยระบบเลขในฐานสิบหก เช่น **0**x5AF เป็นต้น เลขฐานสอง (Binary) เขียนเริ่มต้นด้วย **0**b แล้วตามด้วยระบบเลขในฐานสอง เช่น **0**b11011 เป็นต้น นอกจากยังสามารถระบุตัวเลขที่มีค่ามากเกินขอบเขตของตัวเลขจำนวนเต็มปกติที่เรียกว่า BigInt โดยจะเขียนตัวอักษร n ต่อท้ายค่าจำนวนเต็ม เช่น 555999555123755n เป็นต้น

- **ค่าคงที่จำนวนจริง**

ค่าคงที่จำนวนจริง (Floating-point Literals) ใช้สำหรับแทนค่าตัวเลขที่มีทศนิยม โดยมีโครงสร้างที่สามารถประกอบด้วย จำนวนเต็มไม่มีเครื่องหมาย จุดทศนิยมและเศษส่วนทศนิยมเช่น เช่น 3.15 หรือ .12345 เป็นต้น นอกจากนี้ยังสามารถระบุเป็นเลขชี้กำลัง (Exponent) โดยใช้ตัวอักษร e หรือ E ตามด้วย

จำนวนเต็ม ซึ่งอาจมีเครื่องหมาย + หรือ - นำหน้าได้ เช่น 1.23e+5 มีค่าเป็น 1.23×10^5 หรือ .25E-2 มีค่าเป็น 0.25×10^{-2} เป็นต้น

ตัวเลขที่มีค่าเป็นลบ เช่น -123.4 ไม่ถือว่าเป็นค่าคงที่เชิงตัวเลขโดยตรงเนื่องจากมีการใช้ตัวดำเนินการยูนารีคือเครื่องหมาย - กับค่าคงที่ 123.4 ทั้งนี้ ตัวดำเนินการยูนารี (Unary Operator) คือ ตัวดำเนินการที่ทำงานกับข้อมูลหรือตัวแปรเพียงหนึ่งตัว เช่น -, +, !, typeof, void เป็นต้น

2) ค่าคงที่ชนิดข้อความ

ค่าคงที่ชนิดข้อความ (String Literals) หมายถึง ข้อความที่ถูกระบุไว้โดยตรงในรหัสคำสั่งของภาษา JavaScript ซึ่งใช้สำหรับแสดงข้อมูลที่เป็นลำดับของอักขระ (Characters) เช่น ตัวอักษร ตัวเลข หรือสัญลักษณ์พิเศษต่าง ๆ ค่าคงที่เชิงข้อความจะต้องถูกล้อมรอบด้วยเครื่องหมายอัญประกาศ ซึ่งมีอยู่ 2 แบบที่ใช้งานได้ในภาษา JavaScript ได้แก่เครื่องหมายอัญประกาศเดี่ยว (Single quotes) เช่น 'การพัฒนาเว็บ' หรือ '081-1111234' เป็นต้น และเครื่องหมายอัญประกาศคู่ (Double quotes) เช่น "แอปพลิเคชัน" หรือ "abc@mail.app" เป็นต้น

JavaScript ยอมรับการใช้เครื่องหมายอัญประกาศทั้งสองรูปแบบโดยไม่มีความแตกต่างด้านการทำงาน อย่างไรก็ตาม การเลือกใช้อาจขึ้นอยู่กับบริบทหรือความสอดคล้องในการจัดรูปแบบข้อความ อย่างในกรณีที่ต้องการแทรกเครื่องหมายอัญประกาศคู่ในข้อความ ก็จะใช้เครื่องหมายอัญประกาศเดี่ยวล้อมรอบข้อความแทน เช่น ' Hi "JavaScript" ' เป็นต้น หรือหากต้องการแทรกเครื่องหมายอัญประกาศเดี่ยวไว้ในข้อความก็จะใช้เครื่องหมายอัญประกาศคู่ล้อมรอบข้อความแทน เช่น " It's a great day " เป็นต้น

นอกจากนี้ การกำหนดค่าคงที่เชิงข้อความสามารถใช้ร่วมกับอักขระควบคุม (Escape Characters) เพื่อแสดงอักขระพิเศษ เช่น \n (ขึ้นบรรทัดใหม่), \t (แท็บ) เช่น "บทที่ 9 \n JavaScript" เป็นต้น

3) ค่าคงที่แม่แบบ

ค่าคงที่แม่แบบ (Template Literals) หมายถึง รูปแบบการระบุข้อความในภาษา JavaScript ที่รองรับการแทรกค่าหรือนิพจน์ไว้ภายในข้อความโดยตรง พร้อมทั้งสามารถเขียนข้อความได้หลายบรรทัดโดยไม่ต้องใช้อักขระควบคุมเพื่อขึ้นบรรทัดใหม่ (\n) ซึ่งมีประโยชน์อย่างยิ่งในการสร้างข้อความที่ต้องมีการผสานค่าตัวแปรหรือลำดับตรรกะภายใน

ค่าคงที่แม่แบบจะต้องเขียนอยู่ระหว่างเครื่องหมาย Backtick หรือเครื่องหมาย Grave Accent (`) แทนที่จะใช้เครื่องหมายอัญประกาศเดี่ยวหรือคู่แบบค่าคงที่เชิงข้อความ และหากต้องการแทรกค่าหรือนิพจน์จะใช้รูปแบบ ${นิพจน์} ไว้ระหว่างเครื่องหมาย Backtick ดังกล่าว

ตัวอย่างที่ 8

```
1  name = "Nok";
2  age = 30;
3  greeting = `ชื่อของฉันคือ ${name} และฉันอายุ ${age} ปี`;
```

จากตัวอย่างที่ 8 ประกอบด้วยประโยคคำสั่ง 3 ประโยค ประโยคคำสั่งแรก (บรรทัดที่ 1) มีการกำหนดค่าคงที่เชิงข้อความให้กับตัวแปร name ซึ่งเป็นข้อความคำว่า *Nok* ประโยคคำสั่งที่สอง (บรรทัดที่ 2) มีการกำหนดค่าคงที่เชิงตัวเลขให้กับตัวแปร age ซึ่งมีค่าเป็น *30* ประโยคที่ 3 (บรรทัดที่ 3) เป็นการกำหนดค่าคงที่แบบแม่แบบให้กับตัวแปร greeting โดยใช้เครื่องหมาย Backtick ครอบข้อความและนิพนจน์ตัวแปรจำนวน 2 ตัว คือ name และ age เพื่อนำค่าจากตัวแปรทั้งสองมาแทรกลงในข้อความ ค่าผลลัพธ์ที่ได้จากการประเมินนิพจน์ของประโยคที่ 3 เป็นข้อความ *ชื่อของฉันคือ Nok และฉันอายุ 30 ปี*

4) ค่าคงที่ชนิดบูลีน

ค่าคงที่บูลีน (Boolean Literals) คือค่าคงที่ที่มีเพียงสองค่าคือ true และ false ซึ่งใช้เพื่อแสดงสถานะตรรกะของนิพจน์ เช่น ความจริงและความเท็จ ค่าคงที่ชนิดนี้มักปรากฏในนิพจน์เปรียบเทียบ นิพจน์เชิงตรรกะ หรือใช้กำหนดสถานะให้กับตัวแปรเพื่อใช้ในการควบคุมทิศทางการทำงานของโปรแกรม เช่น ในคำสั่ง if หรือ while เป็นต้น

ตัวอย่างที่ 9

```
1  isMember = true;
2  hasAccess = false;
```

จากตัวอย่างที่ 9 ประกอบด้วยประโยคคำสั่งจำนวน 2 ประโยค โดยในประโยคคำสั่งแรก (บรรทัดที่ 1) เป็นการกำหนดค่าคงที่เชิงบูลีน true ให้กับตัวแปร isMember เพื่อแสดงว่าผู้ใช้เป็นสมาชิก ส่วนประโยคคำสั่งที่สอง (บรรทัดที่ 2) เป็นการกำหนดค่าคงที่เชิงบูลีน false ให้กับตัวแปร hasAccess เพื่อแสดงว่ายังไม่มีสิทธิ์ในการเข้าถึงข้อมูล การใช้ค่าคงที่เชิงบูลีนลักษณะนี้เป็นแนวทางพื้นฐานที่สำคัญในการควบคุมเงื่อนไขในโปรแกรม

5) ค่าคงที่ชนิดแถวลำดับ

ค่าคงที่ชนิดแถวลำดับ (Array Literals) คือค่าคงที่ที่ใช้แสดงข้อมูลแบบลำดับโดยเรียงค่าหลายค่ารวมกันไว้ระหว่างวงเล็บเหลี่ยม [และ] โดยแต่ละค่าจะคั่นด้วยเครื่องหมายจุลภาค (,) ค่าต่าง ๆ ที่อยู่ในแถวลำดับสามารถเป็นค่าคงที่ประเภทเดียวกันหรือต่างชนิดกันก็ได้ เช่น ตัวเลข ข้อความ หรือแม้กระทั่งแถวลำดับซ้อนกันเอง (Nested Array)

ตัวอย่างที่ 10

```
1  numbers = [1, 2, 3, 4, 5];
2  mixed = [100, "Hello", true];
3  matrix = [[1, 2], [3, 4]];
```

จากตัวอย่างที่ 10 ประกอบด้วยประโยคคำสั่งจำนวน 3 ประโยค ซึ่งแต่ละประโยคเป็นการกำหนดค่าคงที่ชนิดแถวลำดับให้กับตัวแปรที่แตกต่างกัน โดยประโยคคำสั่งแรก (บรรทัดที่ 1) เป็นการกำหนดค่าคงที่แถวลำดับให้กับตัวแปร numbers ซึ่งประกอบด้วยค่าคงที่เชิงตัวเลขจำนวน 5 ค่า ได้แก่ 1, 2, 3, 4 และ 5 โดยจัดเก็บในรูปแบบของแถวลำดับที่ใช้เครื่องหมาย [ ] ครอบ และใช้เครื่องหมายจุลภาคคั่นค่าระหว่างแต่ละค่า

ประโยคคำสั่งที่สอง (บรรทัดที่ 2) เป็นการกำหนดค่าคงที่ชนิดแถวลำดับให้กับตัวแปร mixed โดยภายในประกอบด้วยค่าคงที่ต่างชนิดกัน ได้แก่ จำนวนเต็ม 100, ข้อความ “Hello” และค่าบูลีน true แสดงให้เห็นว่าในภาษา JavaScript สามารถสร้างแถวลำดับที่มีค่าหลากหลายชนิดได้โดยไม่มีข้อจำกัดเกี่ยวกับชนิดข้อมูล

ประโยคคำสั่งที่สาม (บรรทัดที่ 3) เป็นการกำหนดค่าคงที่ชนิดแถวลำดับให้กับตัวแปร matrix โดยเป็นแถวลำดับสองมิติ ซึ่งประกอบด้วยแถวลำดับย่อย 2 แถว ได้แก่ [1, 2] และ [3, 4] แสดงให้เห็นถึงความสามารถของ JavaScript ในการซ้อนแถวลำดับภายในแถวลำดับอื่นเพื่อสร้างโครงสร้างข้อมูลแบบเมทริกซ์

6) ค่าคงที่ชนิดอ็อบเจกต์

ค่าคงที่ชนิดอ็อบเจกต์ (Object Literals) ใช้สำหรับแทนกลุ่มข้อมูลที่อยู่ในรูปแบบคู่ของชื่อและค่า (key-value pairs) โดยอยู่ภายในวงเล็บปีกกา { } และแต่ละคู่จะคั่นด้วยเครื่องหมายจุลภาค (,) ชื่อของข้อมูล (Key) มักเป็นข้อความ และค่าที่กำหนด (Value) อาจเป็นค่าคงที่หรือนิพจน์ชนิดใดก็ได้ รวมถึงอ็อบเจกต์หรือแถวลำดับก็ได้เช่นกัน

ตัวอย่างที่ 11

```
1  person = { name: "Nok", age: 30, isStudent: false };
```

จากตัวอย่างที่ 11 ประกอบด้วยประโยคคำสั่งเพียงประโยคเดียว โดยเป็นการกำหนดค่าคงที่ชนิดอ็อบเจกต์ (Object Literal) ให้กับตัวแปร person ซึ่งมีลักษณะของการจับคู่ระหว่างชื่อคุณสมบัติ (Property Name) กับค่าของคุณสมบัติ (Property Value) หลายรายการ ได้แก่ name มีค่าคงที่เชิงข้อความคือ "Nok", age มีค่าคงที่เชิงตัวเลขคือ 30, และ isStudent มีค่าคงที่เชิงบูลีนคือ false โดยค่าคงที่ชนิดอ็อบเจกต์จะใช้เครื่องหมายวงเล็บปีกกา ครอบกลุ่มของคู่ข้อมูลทั้งหมด และใช้เครื่องหมายจุลภาค (,) ในการคั่นแต่ละคู่ข้อมูล

7) ค่าคงที่ที่ใช้แทนการไม่มีค่า

ค่าคงที่ที่ใช้แทนการไม่มีค่า (Null Literal) คือค่าคงที่ใน JavaScript ซึ่งเขียนโดยใช้คำว่า null เพื่อแสดงเจตนาของผู้พัฒนาว่าตัวแปรที่กำหนดค่านั้น ไม่มีข้อมูลใด ๆ อยู่ภายใน โดยถือว่าเป็นค่าว่างโดยตั้งใจ ซึ่งแตกต่างจากค่าศูนย์ (0) ข้อความว่าง ("") หรือค่าที่ยังไม่ได้กำหนด (Undefined) ที่อาจเกิดขึ้นจากระบบหรือกระบวนการประมวลผลของโปรแกรม เช่น phone = null; เป็นการกำหนดค่าของตัวแปร phone ให้เป็น null ซึ่งหมายความว่าผู้พัฒนาตั้งใจให้ตัวแปรนี้ไม่มีข้อมูล ทั้งนี้ การใช้ null มักพบในกรณีที่ต้องการเตรียมตัวแปรไว้ล่วงหน้าและจะกำหนดค่าให้ในภายหลัง

2.3 ตัวแปร

การเขียนโปรแกรมนอกจากจะเป็นการสั่งการให้หน่วยต่าง ๆ ของคอมพิวเตอร์ปฏิบัติตามคำสั่งแล้ว หนึ่งในกระบวนการที่สำคัญของการเขียนโปรแกรมคือการประมวลผลข้อมูล ไม่ว่าจะเป็นการคำนวณ การเปรียบเทียบ การจัดกลุ่ม หรือการแสดงผล ซึ่งการจะทำเช่นนั้นได้ จำเป็นต้องมีวิธีในการเก็บข้อมูลไว้ในหน่วยความจำและสามารถเรียกใช้งานข้อมูลเหล่านั้นในภายหลังได้อย่างรวดเร็ว ตัวแปรจึงเป็นกลไกสำคัญที่ใช้แทนตำแหน่งของข้อมูลในหน่วยความจำ โดยให้ผู้พัฒนาเรียกใช้งานผ่านชื่อที่กำหนดแทนการเข้าถึงหน่วยความจำโดยตรง

2.3.1 ความหมายของตัวแปร

ตัวแปร (Variable) คือชื่อที่ผู้พัฒนากำหนดขึ้นเพื่อใช้ในการอ้างถึงข้อมูลภายในโปรแกรม โดยทำหน้าที่เป็นตัวแทนเชิงสัญลักษณ์ (Symbolic Representation) ที่เชื่อมโยงกับข้อมูลในหน่วยความจำของคอมพิวเตอร์แทนการใช้งานหมายเลขตำแหน่งหน่วยความจำโดยตรง ช่วยให้การเรียกใช้และจัดการข้อมูลทำได้สะดวกและเข้าใจง่ายยิ่งขึ้น ในภาษา JavaScript การเชื่อมโยงระหว่างตัวแปรกับข้อมูลมีลักษณะที่แตกต่างกันตามชนิดของข้อมูล โดยแบ่งออกเป็นสองรูปแบบคือการเชื่อมโยงแบบค่า (Value Binding) สำหรับข้อมูลชนิดพื้นฐาน เช่น ตัวเลข สายอักขระ และค่าความจริง เป็นต้น และการเชื่อมโยงแบบอ้างอิง (Reference Binding) สำหรับข้อมูลประเภทอ้างอิง (Reference Types) เช่น ออบเจ็กต์ อาร์เรย์ และฟังก์ชัน เป็นต้น

2.3.2 การประกาศตัวแปร

การประกาศตัวแปร (Variable Declaration) คือกระบวนการแจ้งให้ JavaScript Engine ทราบว่าผู้พัฒนาต้องการสร้างตัวแปรใหม่เพื่อใช้ในการเก็บและจัดการข้อมูล เปรียบเสมือนการสร้างกล่องเก็บข้อมูลพร้อมติดป้ายชื่อเพื่อให้เรียกใช้ได้ง่าย โดยผู้พัฒนาจะเป็นผู้ระบุชื่อ (Identifier) ที่ใช้อ้างอิงถึงข้อมูลดังกล่าว

เมื่อมีการประกาศตัวแปร JavaScript Engine จะดำเนินการสร้างการเชื่อมโยง (Binding) ระหว่างชื่อตัวแปรและพิจารณาขอบเขตการเข้าถึง การจัดสรรพื้นที่หน่วยความจำที่เหมาะสมตามชนิดข้อมูล และการกำหนดค่าเริ่มต้นให้กับตัวแปรตามกฎของคำสงวนที่ใช้ในการประกาศ

ภาษา JavaScript มีคำสงวนสามแบบ ได้แก่ var, let และ const ซึ่งแต่ละคำมีพฤติกรรมที่แตกต่างกันในด้านขอบเขต (Scope) ความสามารถในการเปลี่ยนค่า (Mutability) และช่วงเวลาที่ตัวแปรสามารถใช้งานได้ (Temporal Dead Zone และ Hoisting) การเลือกใช้คำสงวนที่เหมาะสมไม่เพียงส่งผลต่อความถูกต้องของโปรแกรมเท่านั้น แต่ยังส่งผลต่อประสิทธิภาพและความปลอดภัยในการจัดการข้อมูลอีกด้วย

1) การประกาศตัวแปรด้วย var

var เป็นคำสงวนที่ใช้ในการประกาศตัวแปรโดยเป็นรูปแบบที่ใช้มาตั้งแต่ JavaScript เวอร์ชันแรก คุณสมบัติสำคัญของการประกาศตัวแปรด้วยวิธีนี้คือ ตัวแปรที่ประกาศด้วย var จะมีขอบเขตการเข้าถึงอยู่ในระดับฟังก์ชัน (Function Scope) กล่าวคือ หากมีการประกาศตัวแปรไว้ภายในฟังก์ชันที่ผู้พัฒนาสร้างขึ้น ตัวแปรนั้นจะสามารถถูกเรียกใช้ได้จากทุกตำแหน่งภายในฟังก์ชัน แม้ว่าจะอยู่ในบล็อกคำสั่งที่แตกต่างกันซึ่งครอบด้วยวงเล็บปีกกา { และ } ก็ตาม นอกจากนี้การประกาศตัวแปรด้วย var ยังมีพฤติกรรมที่เรียกว่า การยกตัว

แปรขึ้นบนสุดที่เรียกว่า Hoisting ซึ่งหมายถึง การที่ JavaScript Engine รับรู้การประกาศตัวแปรนั้นตั้งแต่ช่วงต้นของฟังก์ชัน แม้ผู้พัฒนาจะเขียนคำสั่งประกาศไว้ในลำดับท้ายของฟังก์ชันก็ตาม

ตัวแปรที่ประกาศด้วย var สามารถกำหนดค่าเริ่มต้นได้ทันทีในขณะประกาศ หรืออาจละการกำหนดค่าไว้ก่อนแล้วจึงกำหนดค่าในภายหลังก็ได้ นอกจากนี้ยังสามารถประกาศตัวแปรหลายตัวในประโยคเดียวกัน โดยใช้เครื่องหมายจุลภาคคั่นระหว่างชื่อตัวแปรแต่ละตัว โดยมีรูปแบบการประกาศตัวแปรดังนี้

รูปแบบที่ 1

```
var ชื่อตัวแปร_1, ชื่อตัวแปร_2, ... , ชื่อตัวแปร_n;
```

รูปแบบที่ 2

```
var ชื่อตัวแปร_1 = ค่าเริ่มต้น, ชื่อตัวแปร_2 = ค่าเริ่มต้น, ... ;
```

จากรูปแบบการประกาศตัวแปรด้วย var ในรูปแบบที่ 1 เป็นการประกาศตัวแปรที่ไม่ได้มีการกำหนดค่าเริ่มต้นให้กับตัวแปร ทำให้ตัวแปรที่ประกาศขึ้นมีค่าเป็น undefined โดยอัตโนมัติ ส่วนรูปแบบที่ 2 เป็นการประกาศตัวแปรพร้อมกำหนดค่าเริ่มต้นให้กับตัวแปร ซึ่งค่าเริ่มต้นนี้อาจเป็นค่าคงที่หรือนิพจน์ก็ได้ โดยผู้พัฒนาสามารถเปลี่ยนแปลงค่าของตัวแปรเหล่านี้ได้ในภายหลังโดยใช้นิพจน์กำหนดค่า

ตัวอย่างที่ 12

```
1  var  name, lastname;
2  var  point = 0, rate = 3/4;
```

จากตัวอย่างที่ 12 ประกอบด้วยประโยคคำสั่งจำนวน 2 ประโยค ซึ่งแต่ละประโยคเป็นการประกาศตัวแปรโดยใช้คำสงวน var พร้อมระบุชื่อของตัวแปรและค่าที่ต้องการกำหนด โดยประโยคคำสั่งแรก (บรรทัดที่ 1) เป็นการประกาศตัวแปรจำนวน 2 ตัว ได้แก่ name และ lastname โดยไม่ได้กำหนดค่าเริ่มต้นให้กับตัวแปรทั้งสอง ซึ่งจะทำให้ JavaScript กำหนดค่าเริ่มต้นเป็น undefined ให้โดยอัตโนมัติ

ส่วนประโยคคำสั่งที่สอง (บรรทัดที่ 2) เป็นการประกาศตัวแปรชื่อ point และ rate โดยมีการกำหนดค่าเริ่มต้นให้กับตัวแปรทันทีภายในประโยคเดียวกัน โดยตัวแปร point ถูกกำหนดค่าเริ่มต้นเป็นค่าคงที่จำนวนเต็ม 0 และตัวแปร rate ถูกกำหนดค่าเริ่มต้นเป็นผลลัพธ์ของนิพจน์คณิตศาสตร์ 3/4 ซึ่งมีค่าเท่ากับ 0.75

2) การประกาศตัวแปรด้วย let

let เป็นคำสงวนที่ใช้ในการประกาศตัวแปรในภาษา JavaScript ซึ่งถูกนำเสนอในมาตรฐาน ECMAScript 2015 (ES6) เพื่อใช้ในกรณีที่ต้องการควบคุมขอบเขตของตัวแปรให้อยู่ภายในระดับบล็อก (Block Scope) กล่าวคือ ตัวแปรจะสามารถเข้าถึงได้เฉพาะภายในกลุ่มประโยคคำสั่งที่อยู่ระหว่างเครื่องหมาย { } เดียวกันเท่านั้น ซึ่งช่วยลดโอกาสเกิดข้อผิดพลาดจากการเข้าถึงตัวแปรโดยไม่ตั้งใจ และป้องกันความสับสนที่อาจ

เกิดจากการใช้ชื่อตัวแปรซ้ำกันในกลุ่มคำสั่งต่างกัน ขอบเขตการเข้าถึงในระดับบล็อกนี้เป็นคุณสมบัติที่แตกต่างจากการประกาศตัวแปรด้วย var ซึ่งมีขอบเขตอยู่ในระดับฟังก์ชัน

นอกจากนี้ การประกาศตัวแปรด้วย let ยังอยู่ภายใต้ข้อกำหนดที่เรียกว่า ช่วงเวลาที่ตัวแปรสามารถใช้งานได้ (Temporal Dead Zone: TDZ) ซึ่งหมายความว่า หากมีการเรียกใช้ตัวแปรก่อนที่จะมีการประกาศ จะทำให้เกิดข้อผิดพลาด ReferenceError แม้ว่า JavaScript จะมีกลไก Hoisting ก็ตาม

ตัวแปรที่ประกาศด้วย let มีรูปแบบการเขียนเช่นเดียวกับการประกาศตัวแปรด้วย var แต่ต่างกันที่คำสงวนที่ใช้ โดยสามารถเขียนได้ในรูปแบบต่อไปนี้

รูปแบบที่ 1

let ชื่อตัวแปร_1, ชื่อตัวแปร_2, ... , ชื่อตัวแปร_n;

รูปแบบที่ 2

let ชื่อตัวแปร_1 = ค่าเริ่มต้น, ชื่อตัวแปร_2 = ค่าเริ่มต้น, ... ;

ตัวอย่างที่ 13

```
1  let title, year;
2  let price = 99.99, available = true;
```

จากตัวอย่างที่ 13 ประกอบด้วยประโยคคำสั่งจำนวน 2 ประโยค ซึ่งเป็นการประกาศตัวแปรโดยใช้คำสงวน let โดยประโยคคำสั่งแรก (บรรทัดที่ 1) เป็นการประกาศตัวแปรจำนวน 2 ตัว ได้แก่ title และ year โดยไม่ได้กำหนดค่าเริ่มต้นให้กับตัวแปรทั้งสอง ซึ่งจะทำให้ JavaScript กำหนดค่าเริ่มต้นของตัวแปรเหล่านี้เป็น undefined โดยอัตโนมัติ

ส่วนประโยคคำสั่งที่สอง (บรรทัดที่ 2) เป็นการประกาศตัวแปรชื่อ price และ available พร้อมกำหนดค่าเริ่มต้นให้กับตัวแปรทั้งสองภายในประโยคเดียวกัน โดย price ได้รับการกำหนดค่าเริ่มต้นเป็นเลขทศนิยม 99.99 และ available ได้รับการกำหนดค่าเริ่มต้นเป็นค่าคงที่แบบตรรกะ true

3) การประกาศค่าคงที่ด้วย const

const เป็นคำสงวนที่ใช้ในการประกาศตัวแปรให้มีค่าคงที่โดยไม่สามารถเปลี่ยนค่าของตัวแปรได้ในภายหลัง (Constant) ซึ่งถูกนำเสนอในมาตรฐาน ECMAScript 2015 (ES6) การใช้งานการประกาศลักษณะนี้เพื่อให้ตัวแปรมีค่าเดิมตลอดการทำงานของโปรแกรม เช่น ค่าคงที่ทางคณิตศาสตร์ อัตราภาษี หรือค่าคงที่ที่ใช้ซ้ำในโปรแกรม เป็นต้น

คุณสมบัติสำคัญของการประกาศค่าคงที่ด้วย const คือ ไม่สามารถกำหนดค่าใหม่ให้กับตัวแปรนั้นได้อีกหลังจากที่มีการกำหนดค่าเริ่มต้นครั้งแรกแล้ว และมีขอบเขตการใช้งานในระดับบล็อกเช่นเดียวกับ let กล่าวคือ ตัวแปรจะสามารถเข้าถึงได้เฉพาะภายในกลุ่มประโยคคำสั่งที่อยู่ภายในเครื่องหมาย { } เดียวกันเท่านั้น นอกจากนี้ การประกาศค่าคงที่ด้วย const ก็อยู่ภายใต้ข้อจำกัดของช่วงเวลาห้ามเข้าถึงตัวแปรเช่นเดียวกับ let ซึ่งหมายความว่า จะไม่สามารถเรียกใช้ตัวแปรก่อนบรรทัดที่มีการประกาศได้

สิ่งที่แตกต่างจาก let คือ การประกาศตัวแปรด้วย const จะต้อง กำหนดค่าเริ่มต้นในทันทีที่ประกาศ และไม่สามารถละการกำหนดค่าไว้ก่อนได้ หากไม่มีการกำหนดค่าเริ่มต้น จะเกิดข้อผิดพลาด SyntaxError ในขณะการทำงานของโปรแกรม

การประกาศค่าคงที่ด้วย const สามารถประกาศหลายตัวแปรในประโยคเดียวกันได้ เช่นเดียวกับ var และ let โดยใช้เครื่องหมายจุลภาค (,) คั่นระหว่างชื่อตัวแปรแต่ละตัว โดยมีรูปแบบการเขียนดังนี้

รูปแบบ

const ชื่อตัวแปร_1 = ค่าเริ่มต้น, ชื่อตัวแปร_2 = ค่าเริ่มต้น, ... ;

ตัวอย่างที่ 14

```
1   const  PI = 3.14159, VAT_RATE = 0.07;
```

จากตัวอย่างที่ 14 เป็นการประกาศค่าคงที่จำนวน 2 ตัว โดยใช้คำสงวน const และกำหนดค่าเริ่มต้นให้กับตัวแปรในทันที โดยตัวแปร PI ได้รับการกำหนดค่าเริ่มต้นเป็นค่าคงที่ทางคณิตศาสตร์ 3.14159 ซึ่งใช้แทนค่าของ π และตัวแปร VAT_RATE ได้รับการกำหนดค่าเริ่มต้นเป็นอัตราภาษีมูลค่าเพิ่ม 0.07 หรือ 7% ซึ่งเป็นค่าคงที่ที่มักใช้ในโปรแกรมที่เกี่ยวข้องกับระบบการคำนวณราคา

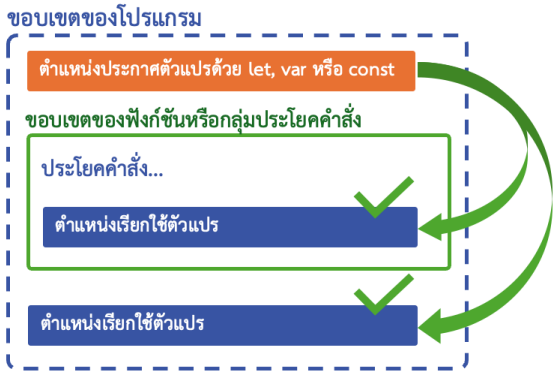
2.3.3 ขอบเขตของตัวแปร

ขอบเขตของตัวแปร (Variable Scope) คือ ขอบเขตการเข้าถึงและใช้งานตัวแปรจากแต่ละส่วนของโปรแกรม โดยเป็นการแสดงถึงตัวแปรที่ถูกประกาศสามารถถูกเรียกใช้งานได้จากตำแหน่งใดบ้างในรหัสคำสั่ง ซึ่งถือเป็นแนวคิดสำคัญในการช่วยลดความสับสนในการใช้งานตัวแปร และช่วยให้โปรแกรมทำงานได้อย่างถูกต้องตามที่ออกแบบไว้ โดยลักษณะของขอบเขตจะขึ้นอยู่กับคำสงวนที่ใช้ในการประกาศตัวแปรดังที่ได้กล่าวถึงในหัวข้อก่อนหน้า โดยใน JavaScript มีขอบเขตของตัวแปรจำแนกตามระดับของการเข้าถึง 3 ลักษณะดังนี้

1) ขอบเขตโกลบอล

ขอบเขตโกลบอล (Global Scope) คือ ขอบเขตการเข้าถึงและใช้งานตัวแปรที่สามารถเรียกใช้ได้จากทุกส่วนของโปรแกรมภายในไฟล์เดียวกัน โดยไม่จำกัดว่าจะอยู่ในฟังก์ชัน ลูป หรือโครงสร้างควบคุมใด ๆ ตัวแปรที่ถูกประกาศด้วยคำสำคัญ var, let หรือ const ภายนอกฟังก์ชันหรือโครงสร้างจัดกลุ่มประโยคคำสั่ง เช่น เงื่อนไข if หรือวงรอบ for จะถือว่าเป็นตัวแปรระดับโกลบอลโดยอัตโนมัติ ซึ่งหมายความว่าตัวแปรเหล่านี้จะถูกเก็บไว้ในบริบทการทำงานของโปรแกรมหลัก (Global Execution Context) และสามารถเรียกใช้งานหรือแก้ไขค่าได้จากทุกจุดในโปรแกรม

การประกาศตัวแปรในขอบเขตโกลบอลมีทั้งข้อดีและข้อควรระวัง ข้อดีคือช่วยให้ตัวแปรสามารถแชร์ข้อมูลระหว่างฟังก์ชันต่าง ๆ ได้โดยสะดวก แต่ในทางกลับกันอาจเกิดปัญหาเมื่อมีการแก้ไขค่าตัวแปรจากหลายตำแหน่งในโปรแกรม ทำให้เกิดผลกระทบโดยไม่ตั้งใจ (Side Effect) ตำแหน่งการประกาศตัวแปรแบบโกลบอลและขอบเขตการเรียกใช้งานแสดงดังภาพที่ 9.2



ภาพที่ 9.2 ตำแหน่งการประกาศตัวแปรแบบโกลบอลและขอบเขตการเรียกใช้งาน

จากภาพที่ 9.2 แสดงให้เห็นว่าตำแหน่งของการประกาศตัวแปรที่อยู่นอกกลุ่มคำสั่งที่มีเครื่องหมายวงเล็บปีกกา { และ } ไม่ว่าจะประกาศด้วย let, const หรือ var ก็ตาม จะถือว่าเป็นการประกาศในขอบเขตระดับโกลบอล ซึ่งตัวแปรที่ประกาศในตำแหน่งนี้จะสามารถเข้าถึงได้จากทุกตำแหน่งของโปรแกรมภายในไฟล์หรือสคริปต์เดียวกัน ไม่ว่าจะเป็นภายในฟังก์ชัน ภายในกลุ่มประโยคคำสั่ง หรือแม้แต่จากตำแหน่งอื่น ๆ ที่อยู่นอกโครงสร้างใด ๆ ของโปรแกรม

ตัวอย่างที่ 15

```
1  let x = 10;
2  const VAT = 0.07;
3  var y = 5;
4  function test() {
5          console.log(x);
6          console.log(VAT);
7          console.log(y);
8  }
9  if (x > 0) {
10         console.log(x);
11         console.log(VAT);
12         console.log(y);
13 }
14 console.log(x);
15 console.log(VAT);
16 console.log(y);
```

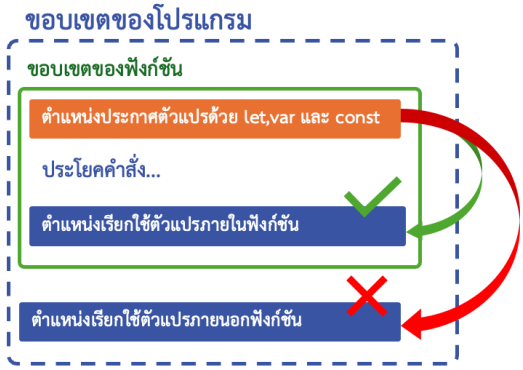
จากตัวอย่างที่ 15 แสดงให้เห็นว่ามีการประกาศตัวแปรจำนวน 3 ตัวในระดับโกลบอล (บรรทัดที่ 1–3) ได้แก่ ตัวแปร x ซึ่งประกาศและกำหนดค่าด้วย let เป็นจำนวนเต็ม 10, ตัวแปร VAT ซึ่งประกาศด้วย const เป็นค่าคงที่ 0.07 และตัวแปร y ซึ่งประกาศด้วย var เป็นจำนวนเต็ม 5 โดยตัวแปรทั้งสามสามารถถูกเรียกใช้งานได้จากทุกตำแหน่งภายในโปรแกรมที่อยู่ในไฟล์หรือสคริปต์เดียวกัน ดังนี้

- บรรทัดที่ 4-8 เป็นการกำหนดฟังก์ชัน test() ซึ่งภายในฟังก์ชันมีการเรียกใช้ตัวแปร x, VAT และ y ที่ประกาศไว้ในระดับโกลบอล ทำให้สามารถเรียกใช้งานได้โดยไม่มีข้อผิดพลาด
- บรรทัดที่ 9-13 เป็นโครงสร้างควบคุมแบบเงื่อนไข if(x > 0) ซึ่งแม้จะมีการจำกัดขอบเขตด้วย {} แต่ตัวแปร x, VAT และ y ที่อยู่ในระดับโกลบอลยังสามารถถูกเรียกใช้ได้จากภายในกลุ่มคำสั่งนี้เช่นกัน
- บรรทัดที่ 14-16 อยู่ในระดับเดียวกับจุดที่ประกาศตัวแปรทั้งสาม จึงสามารถเรียกใช้ x, VAT และ y ได้โดยตรงโดยไม่เกิดข้อผิดพลาด

จากตัวอย่างข้างต้นแสดงให้เห็นว่า ตัวแปรที่ถูกประกาศไว้ในระดับโกลบอลจะสามารถเข้าถึงและเรียกใช้งานได้จากทุกตำแหน่งของโปรแกรมภายใต้ไฟล์หรือสคริปต์เดียวกัน ไม่ว่าจะเป็นภายในฟังก์ชันโครงสร้างควบคุม หรือในระดับโกลบอลเอง

2) ขอบเขตของฟังก์ชัน

ขอบเขตของฟังก์ชัน (Function Scope) คือ ขอบเขตการเข้าถึงและใช้งานตัวแปรที่ถูกประกาศภายในฟังก์ชัน ซึ่งตัวแปรที่ประกาศด้วย var, let หรือ const ภายในฟังก์ชันใดจะสามารถถูกเรียกใช้ได้เฉพาะภายในฟังก์ชันนั้นเท่านั้น และไม่สามารถเข้าถึงได้จากภายนอกฟังก์ชัน แม้ว่าฟังก์ชันนั้นจะอยู่ในไฟล์หรือสคริปต์เดียวกันก็ตาม การประกาศตัวแปรภายในฟังก์ชันจึงถือเป็นการจำกัดขอบเขตให้แคบลงเพื่อป้องกันการรบกวนค่าระหว่างส่วนต่าง ๆ ของโปรแกรม และช่วยให้สามารถควบคุมการใช้งานตัวแปรได้อย่างปลอดภัยมากขึ้น นอกจากนี้ เมื่อฟังก์ชันสิ้นสุดการทำงาน ตัวแปรที่ประกาศภายในฟังก์ชันจะถูกทำลายออกจากหน่วยความจำโดยอัตโนมัติ ซึ่งแตกต่างจากตัวแปรโกลบอลที่ยังคงอยู่จนกว่าจะสิ้นสุดการทำงานของโปรแกรม ตำแหน่งการประกาศตัวแปรภายในฟังก์ชันและขอบเขตการเรียกใช้งานแสดงดังภาพที่ 9.3



ภาพที่ 9.3 ตำแหน่งการประกาศตัวแปรในฟังก์ชันและขอบเขตการเรียกใช้งาน

จากภาพที่ 9.3 แสดงให้เห็นถึงแนวคิดของขอบเขตฟังก์ชัน (Function Scope) ซึ่งตัวแปรที่ถูกประกาศด้วย let, const หรือ var ภายในฟังก์ชันจะสามารถถูกเรียกใช้งานได้เฉพาะภายในฟังก์ชันนั้นเท่านั้น กล่าวคือ ตำแหน่งการเรียกใช้งานตัวแปรที่อยู่ภายในฟังก์ชันสามารถเข้าถึงตัวแปรที่ประกาศภายในเดียวกันได้ (แสดงด้วยเครื่องหมายถูกสีเขียว) ขณะที่ตำแหน่งนอกฟังก์ชันจะไม่สามารถเข้าถึงตัวแปรภายในได้ (แสดงด้วยเครื่องหมายกากบาทสีแดง) ซึ่งเป็นหลักการสำคัญในการควบคุมขอบเขตของข้อมูลและการทำงานภายในโปรแกรม ไม่ให้เกิดการรบกวนหรือเปลี่ยนแปลงค่าจากภายนอกฟังก์ชันโดยไม่ได้ตั้งใจ

ตัวอย่างที่ 16

```
1   function test() {
2       let x = 10;
3       const VAT = 0.07;
4       var y = 5;
5       console.log(x);
6       console.log(VAT);
7       console.log(y);
8   }
9   test();
10  console.log(x);
11  console.log(VAT);
12  console.log(y);
```

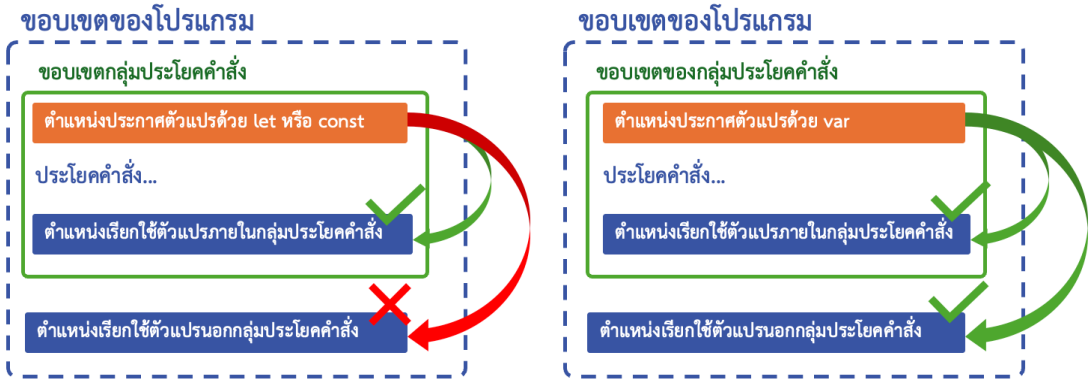
จากตัวอย่างที่ 16 บรรทัดที่ 1- 8 เป็นการสร้างฟังก์ชันชื่อ test( ) ซึ่งมีการประกาศตัวแปรภายในขอบเขตของฟังก์ชันในบรรทัดที่ 2-4 ประกอบด้วย ตัวแปร x ซึ่งถูกประกาศด้วย let มีการกำหนดค่าเป็น 10 ตัวแปร VAT ซึ่งถูกประกาศด้วย const มีการกำหนดค่าเป็น 0.07 และตัวแปร y ซึ่งถูกประกาศด้วย var มีการกำหนดค่าเป็น 5 โดยตัวแปรทั้งสามนี้สามารถถูกเรียกใช้ได้เฉพาะภายในฟังก์ชัน test( ) เท่านั้น

- บรรทัดที่ 5-7 เป็นคำสั่งแสดงค่าของตัวแปรที่ประกาศใช้ในฟังก์ชันไปยังคอนโซลของเว็บเบราว์เซอร์ ซึ่งหากมีการเรียกใช้ฟังก์ชัน test( ) จะมีการแสดงค่า10, 0.07 และ 5 ซึ่งเป็นค่าที่กำหนดในตัวแปรภายในฟังก์ชัน test( ) ตามลำดับ
- บรรทัดที่ 9 เป็นการเรียกใช้ฟังก์ชัน test( )
- บรรทัดที่ 10–12 เป็นการพยายามเรียกใช้ตัวแปร x, VAT และ y จากภายนอกฟังก์ชัน ซึ่งจะทำให้เกิด ข้อผิดพลาด ReferenceError เนื่องจากตัวแปรเหล่านี้ถูกจำกัดอยู่ในขอบเขตของฟังก์ชัน test( ) เท่านั้น

กรณีตัวอย่างนี้แสดงให้เห็นว่า ตัวแปรที่ถูกประกาศภายในฟังก์ชันจะไม่สามารถเข้าถึงได้จากภายนอกฟังก์ชัน ซึ่งเป็นหลักการสำคัญของ ขอบเขตฟังก์ชัน ที่ช่วยควบคุมการมองเห็น (Visibility) และการใช้งานตัวแปรให้จำกัดอยู่เฉพาะในบริบทที่ต้องการเท่านั้น

3) ขอบเขตของกลุ่มประโยคคำสั่ง

ขอบเขตของกลุ่มประโยคคำสั่ง (Block Scope) คือ ขอบเขตที่ถูกสร้างขึ้นจากการจัดกลุ่มประโยคคำสั่งให้อยู่ภายในเครื่องหมายวงเล็บปีกกา { และ } โดยไม่จำเป็นต้องเป็นฟังก์ชัน เช่น การใช้กับคำสั่งควบคุมทิศทางการทำงานของโปรแกรม เช่น if, for, หรือ while หรือแม้แต่การสร้างกลุ่มประโยคคำสั่งขึ้นมาเองโดยตรงก็ได้ หากมีการประกาศตัวแปรด้วย let หรือ const ภายในกลุ่มประโยคคำสั่งดังกล่าว ตัวแปรนั้นจะสามารถถูกเรียกใช้ได้เฉพาะภายในขอบเขตของกลุ่มประโยคคำสั่งเท่านั้น ไม่สามารถเรียกใช้ภายนอกกลุ่มประโยคคำสั่งได้ ขณะที่ตัวแปรที่ประกาศด้วย var จะไม่ยึดตามขอบเขตกลุ่มประโยคคำสั่ง สามารถเรียกใช้งานจากภายนอกกลุ่มประโยคคำสั่งได้ ขอบเขตของกลุ่มประโยคคำสั่งแสดงดังภาพที่ 9.4



ภาพที่ 9.4 ตำแหน่งการประกาศตัวแปรในกลุ่มประโยคคำสั่งและขอบเขตการเรียกใช้งาน

จากภาพด้านซ้ายของภาพที่ 9.4 แสดงให้เห็นว่าเมื่อมีการประกาศตัวแปรด้วย let หรือ const ภายในกลุ่มประโยคคำสั่ง จะสามารถเรียกใช้ได้เฉพาะภายในกลุ่มเท่านั้น หากพยายามเรียกใช้งานภายนอกกลุ่มจะเกิดข้อผิดพลาด ในทางตรงกันข้าม ภาพด้านขวาแสดงให้เห็นว่า หากประกาศตัวแปรด้วย var ภายในกลุ่มประโยคเดียวกัน ตัวแปรจะถูกยกระดับขึ้น (Hoisting) ซึ่งทำให้สามารถเรียกใช้งานได้ทั้งภายในและภายนอกกลุ่มประโยคคำสั่งนั้นได้

ตัวอย่างที่ 17

```
1   let key = 1;
2   if (key === 1) {
3       let x = 10;
4       const VAT = 0.07;
```

```
5           var y = 5;
6           console.log(x);
7           console.log(VAT);
8           console.log(y);
9       }
10      console.log(x);
11      console.log(VAT);
12      console.log(y);
```

จากตัวอย่างที่ 17 แสดงให้เห็นถึงการประกาศตัวแปรทั้งสามภายในขอบเขตของคำสั่งควบคุม if (key === 1) ซึ่งมีการใช้เครื่องหมาย { และ } เพื่อสร้างกลุ่มประโยคคำสั่ง โดยสามารถอธิบายเป็นลำดับได้ดังนี้

- บรรทัดที่ 1 ประกาศตัวแปร key ด้วย let และกำหนดค่าเป็น 1
- บรรทัดที่ 2–9 เป็นกลุ่มประโยคคำสั่งภายใต้คำสั่งควบคุม if(key===1) ซึ่งจะทำงานภายในกลุ่มก็ต่อเมื่อนิพจน์ key === 1 ให้ค่า true ซึ่งผลจากการประเมินค่าทำให้ได้ค่า true จึงมีการทำงานในกลุ่มประโยคคำสั่งต่อไปในบรรทัดที่ 4-9
- บรรทัดที่ 3 ประกาศตัวแปร x ภายในกลุ่มประโยคคำสั่งด้วย let และกำหนดค่าเป็น 10
- บรรทัดที่ 4 ประกาศตัวแปร VAT ภายในกลุ่มประโยคคำสั่งด้วย const และกำหนดค่าเป็น 0.07
- บรรทัดที่ 5 ประกาศตัวแปร y ภายในกลุ่มประโยคคำสั่งด้วย var และกำหนดค่าเป็น 5
- บรรทัดที่ 6–8 แสดงค่าของ x, VAT, และ y ไปยังคอนโซล ซึ่งจะมีการแสดงค่า 10, 0.07 และ 5 ตามลำดับ ไม่พบข้อผิดพลาดแต่อย่างใด เนื่องจากเป็นการใช้ตัวแปรภายในกลุ่มประโยคเดียวกัน
- บรรทัดที่ 10-11 เป็นการเรียกใช้ตัวแปร x และ VAT เพื่อแสดงค่าไปยังคอนโซล โดยตัวแปรทั้งสองถูกประกาศด้วย let และ const ภายในกลุ่มประโยคคำสั่ง จึงไม่สามารถเข้าถึงได้จากภายนอก ทำให้เกิดข้อผิดพลาด ReferenceError
- บรรทัดที่ 12 เป็นการเรียกใช้ตัวแปร y เพื่อแสดงค่าไปยังคอนโซล โดยตัวแปร y ถูกประกาศด้วย var ภายในกลุ่มประโยคคำสั่งซึ่งไม่ได้อยู่ในฟังก์ชัน ส่งผลให้สามารถเข้าถึงตัวแปร y ได้จากภายนอกกลุ่มคำสั่งโดยไม่เกิดข้อผิดพลาด และแสดงผลในคอนโซลเป็น 5 ทั้งนี้พฤติกรรมของ var จะจำกัดการเข้าถึงเมื่อมีการนำไปใช้ประกาศตัวแปรในขอบเขตของฟังก์ชันเท่านั้น ไม่รวมถึงกลุ่มประโยคคำสั่งทั่วไปหรือกลุ่มประโยคคำสั่งของคำสั่งควบคุม เช่น คำสั่ง if, for, หรือ while เป็นต้น

2.3.4 ข้อกำหนดทางไวยากรณ์ของการตั้งชื่อตัวแปรใน JavaScript

การตั้งชื่อตัวแปร (Variable Naming) เป็นกระบวนการสำคัญที่ส่งผลต่อการสร้างความเข้าใจในโปรแกรม การบำรุงรักษา และการตรวจสอบข้อผิดพลาด โดยภาษา JavaScript มีข้อกำหนดทางไวยากรณ์ (Syntax Rules) สำหรับการตั้งชื่อตัวแปร รวมถึงแนวปฏิบัติที่ดี (Best Practices) เพื่อให้การเขียนรหัสคำสั่งเป็นระบบและสื่อความหมายได้ดี การเลือกชื่อตัวแปรที่เหมาะสมจะช่วยให้รหัสคำสั่งอ่านง่าย ลดความสับสนเมื่อทำงานร่วมกับผู้อื่น และช่วยป้องกันข้อผิดพลาดที่อาจเกิดขึ้นจากการใช้ชื่อตัวแปรซ้ำหรือไม่สื่อความหมาย ข้อกำหนดทางไวยากรณ์ของการตั้งชื่อตัวแปรใน JavaScript มีดังนี้

- ชื่อตัวแปรต้องขึ้นต้นด้วยตัวอักษร (a–z หรือ A–Z) เครื่องหมายขีดล่าง (_) หรือเครื่องหมายดอลลาร์ ($) เท่านั้น โดยตัวอักษรถัดไปของชื่อตัวแปรสามารถเป็นตัวเลข 0–9 ได้
- ไม่สามารถใช้คำสงวนของภาษา JavaScript เป็นชื่อตัวแปรได้ เช่น let, const, var, if, function เป็นต้น
- ชื่อตัวแปรมีลักษณะ case-sensitive กล่าวคือ ตัวแปรที่ใช้ตัวพิมพ์เล็กและพิมพ์ใหญ่ในการพิมพ์ ถึงแม้จะใช้ชื่อเดียวก็ถือเป็นคนละตัวแปร เช่น score และ Score ถือเป็นคนละตัวแปร เป็นต้น
- ไม่สามารถมีช่องว่างหรืออักขระพิเศษอื่น ๆ อยู่ในชื่อได้ เช่น -, @, # ยกเว้นเครื่องหมาย $

ตัวอย่างของการตั้งชื่อตัวแปรทั้งที่เป็นไปตามข้อกำหนดและไม่เป็นไปตามข้อกำหนดแสดงดังตารางที่ 9-5

ตารางที่ 9-5 ตัวอย่างการตั้งชื่อตัวแปร

ตัวอย่างชื่อ	ความถูกต้องตามข้อกำหนด	เหตุผล
rate1	ถูกต้อง	ขึ้นต้นด้วยตัวอักษร ใช้ตัวเลขต่อท้ายได้
_value	ถูกต้อง	ขึ้นต้นด้วยขีดล่างได้
$temp	ถูกต้อง	เครื่องหมาย $ อนุญาตให้ใช้ขึ้นต้นได้
taxRate25	ถูกต้อง	ใช้ตัวอักษรผสมตัวเลขได้ ไม่ผิดกฎ
1stValue	ไม่ถูกต้อง	ขึ้นต้นด้วยตัวเลข
my-rate	ไม่ถูกต้อง	- เป็นสัญลักษณ์พิเศษ (เป็นตัวดำเนินการ)
p value	ไม่ถูกต้อง	มีช่องว่างในชื่อ
const	ไม่ถูกต้อง	เป็นคำสงวนในภาษา JavaScript
let	ไม่ถูกต้อง	เป็นคำสงวนในภาษา JavaScript
while	ไม่ถูกต้อง	เป็นคำสงวนในภาษา JavaScript

2.3.5 แนวปฏิบัติที่ดีในการตั้งชื่อตัวแปร

แม้ว่า JavaScript จะยอมรับชื่อตัวแปรได้หลากหลายรูปแบบที่เป็นไปตามข้อกำหนดทางไวยากรณ์แล้ว การเลือกชื่อตัวแปรที่ดีจะช่วยอำนวยความสะดวกแก่ผู้พัฒนาด้วยเช่นกัน โดยมีแนวปฏิบัติที่ดีในการตั้งตัวแปรดังนี้

- ใช้ชื่อภาษาอังกฤษที่สื่อความหมายและสอดคล้องกับลักษณะของข้อมูลที่จัดเก็บ เช่น price, totalAmount, isAvailable แทนการใช้ x, a1, temp
- ในการตั้งชื่อตัวแปรทั่วไปที่ประกาศด้วย let และ var ควรใช้รูปแบบ camelCase เช่น productName, netPrice เป็นต้น
- ในการตั้งชื่อตัวแปรที่เป็นค่าคงที่ที่ประกาศด้วย const ควรใช้ตัวอักษรพิมพ์ใหญ่ทั้งหมด หากเป็นชื่อที่มีคำมากกว่าหนึ่งคำควรใช้ขีดล่างเพื่อเชื่อมระหว่างคำ
- หลีกเลี่ยงการใช้ตัวย่อที่ไม่ชัดเจน ยกเว้นในกรณีที่เป็นมาตรฐานที่ใช้กันทั่วไป เช่น id, url เป็นต้น

2.4 คำอธิบาย

คำอธิบาย (Comment) คือข้อความที่ถูกแทรกไว้ในรหัสคำสั่งเพื่อให้ผู้อ่านหรือผู้พัฒนาเข้าใจจุดประสงค์ของรหัสคำสั่งแต่ละส่วน แต่จะไม่ถูกประมวลผลโดยตัวแปลภาษา JavaScript แต่อย่างใด จึงไม่ส่งผลต่อการทำงานของโปรแกรม ความสามารถในการแทรกคำอธิบายภายในรหัสคำสั่งเป็นสิ่งสำคัญที่ช่วยให้การพัฒนาและการดูแลรักษาโปรแกรมมีประสิทธิภาพมากยิ่งขึ้น โดยเฉพาะเมื่อรหัสคำสั่งมีความซับซ้อนหรือถูกใช้งานโดยนักพัฒนาหลายคนร่วมกัน JavaScript รองรับการเขียนคำอธิบายอยู่ 2 รูปแบบ ได้แก่ คำอธิบายแบบบรรทัดเดียว และคำอธิบายแบบหลายบรรทัด

2.4.1 คำอธิบายแบบบรรทัดเดียว

คำอธิบายแบบบรรทัดเดียว (Single-line Comment) จะใช้เครื่องหมาย // นำหน้าข้อความคำอธิบาย โดยจะมีผลเฉพาะข้อความที่อยู่ด้านหลังเครื่องหมาย // ในบรรทัดเดียวกันเท่านั้น ตัวแปลภาษา JavaScript จะละเว้นข้อความดังกล่าวในขณะประมวลผลโปรแกรม ซึ่งเหมาะสำหรับการเขียนคำอธิบายสั้น ๆ ประกอบรหัสคำสั่ง เช่น การอธิบายวัตถุประสงค์ของตัวแปรหรือคำสั่งเฉพาะจุดที่ต้องการความเข้าใจเพิ่มเติม เป็นต้น รูปแบบและตัวอย่างของการเขียนคำอธิบายแบบบรรทัดเดียวมีดังนี้

รูปแบบ

```
// ข้อความคำอธิบาย
```

ตัวอย่างที่ 18

```
1  // ประกาศตัวแปรชื่อ name สำหรับจัดเก็บชื่อ
2  let name = null;
3  console.log(name); // ตรวจสอบค่าในตัวแปร name
```

จากตัวอย่างที่ 18 ประกอบด้วยประโยคคำสั่งจำนวน 2 ประโยค โดยมีการแทรกคำอธิบายแบบบรรทัดเดียวไว้ 2 แห่งด้วยเครื่องหมาย // โดยในบรรทัดที่ 1 เป็นข้อความคำอธิบายว่า *ประกาศตัวแปรชื่อ name สำหรับจัดเก็บชื่อ* ซึ่งตัวแปลภาษาจะไม่ทำการประมวลผลข้อความในบรรทัดนี้เนื่องจากเป็นคำอธิบายเท่านั้น ส่วนบรรทัดที่ 3 เป็นการเขียนคำอธิบายแบบบรรทัดเดียวเช่นเดียวกัน แต่เขียนไว้ต่อท้ายประโยคคำสั่ง โดยตัวแปลภาษาจะประมวลผลเฉพาะส่วนของคำสั่งก่อนหน้าเครื่องหมาย // และจะไม่ทำการประมวลผลข้อความที่อยู่ถัดไปในบรรทัดเดียวกัน

2.4.2 คำอธิบายแบบหลายบรรทัด

คำอธิบายแบบหลายบรรทัด (Multi-line Comment) จะใช้เครื่องหมาย /* เพื่อเริ่มต้นข้อความคำอธิบาย และใช้เครื่องหมาย */ เพื่อสิ้นสุดข้อความคำอธิบาย โดยข้อความที่อยู่ระหว่างเครื่องหมายทั้งสองนี้จะไม่ถูกประมวลผลโดยตัวแปลภาษา JavaScript ซึ่งเหมาะสำหรับการเขียนคำอธิบายที่มีความยาวหลายบรรทัด เช่น อธิบายกระบวนการทำงานของฟังก์ชัน การวางโครงสร้างของโปรแกรม หรือการอธิบายแนวคิดเบื้องหลังการเขียนรหัสคำสั่งที่ซับซ้อน รูปแบบและตัวอย่างของการเขียนคำอธิบายแบบหลายบรรทัดมีดังนี้

รูปแบบ

```
/* ข้อความคำอธิบาย
   ที่เขียนได้หลายบรรทัด
   โดยตัวแปลภาษาจะไม่ประมวลผลข้อความนี้ */
```

ตัวอย่างที่ 19

```
1  /* ประกาศตัวแปรชื่อ name
2  สำหรับจัดเก็บชื่อของผู้ใช้ */
3  let name = "Nok";
4  /*
5  แสดงค่าของตัวแปร name
6  ไปยังคอนโซล
7  */
8  console.log(name);
```

จากตัวอย่างที่ 19 มีการเขียนคำอธิบายแบบหลายบรรทัดไว้ 2 ส่วนด้วยเครื่องหมาย /* และ */ โดยในบรรทัดที่ 1–2 เป็นคำอธิบายก่อนการประกาศตัวแปร name ซึ่งตัวแปลภาษาจะไม่ประมวลผลข้อความในสองบรรทัดนี้ ส่วนบรรทัดที่ 4-7 เป็นคำอธิบายก่อนการแสดงค่าตัวแปร name ไปยังคอนโซล ข้อความที่อยู่ภายในคำอธิบายแบบหลายบรรทัดทั้งหมดจะถูกเพิกเฉยโดยตัวแปลภาษา JavaScript และไม่มีผลต่อการทำงานของโปรแกรม โดยใช้เพื่อช่วยให้รหัสคำสั่งอ่านเข้าใจง่ายและสามารถบำรุงรักษาได้สะดวกมากยิ่งขึ้น

3. ชนิดของข้อมูล

ชนิดของข้อมูล (Data Type) เป็นแนวคิดพื้นฐานที่ใช้บ่งบอกถึงคุณสมบัติและลักษณะของข้อมูล มีความสำคัญต่อ JavaScript Engine ในการพิจารณาว่าสามารถดำเนินการใดกับข้อมูลนั้นได้บ้าง เช่น การคำนวณทางคณิตศาสตร์ การเปรียบเทียบค่า การแปลงประเภท หรือการแสดงผล เป็นต้น (จีราวุธ วารินทร์, 2566) รวมทั้งมีความสัมพันธ์กับการประกาศตัวแปรดังที่ได้กล่าวถึงในหัวข้อก่อนหน้านี้

3.1 การจัดการชนิดข้อมูลในภาษา JavaScript

JavaScript เป็นภาษาที่ใช้ระบบกำหนดชนิดข้อมูลแบบไม่ตายตัว (Dynamic Type System) ซึ่งแตกต่างจากภาษาที่ใช้ระบบชนิดข้อมูลแบบตายตัว (Static Type System) อย่างเช่น Java หรือ C++ เป็นต้น JavaScript มีการกำหนดชนิดข้อมูลแบบพลวัต (Dynamic Typing) กล่าวคือ ชนิดของข้อมูลจะถูกกำหนดในขณะทำงาน (Runtime) ไม่ใช่ขณะเขียนโปรแกรม ทำให้สามารถใช้ตัวแปรเดียวกันจัดเก็บข้อมูลต่างชนิดกันได้ในช่วงเวลาที่ต่างกัน

JavaScript ใช้การแปลงชนิดข้อมูลแบบอ่อน (Weak Typing หรือ Type Coercion)กล่าวคือ ภาษา JavaScript สามารถแปลงชนิดของข้อมูลโดยอัตโนมัติเมื่อนำไปใช้ในบริบทที่ต่างกัน ตัวอย่างเช่น นิพจน์ "123" + 2 ประกอบด้วยค่าคงที่ชนิดข้อความและตัวเลข แต่ใช้ตัวดำเนินการบวก ในกรณีนี้ JavaScript Engine จะแปลงตัวเลขให้เป็นข้อความและนำข้อความทั้งสองมาเชื่อมต่อกัน ทำให้ได้ผลลัพธ์เป็น "1232"

3.2 การจำแนกชนิดข้อมูลในภาษา JavaScript

ชนิดของข้อมูลในภาษา JavaScript สามารถจำแนกตามลักษณะการจัดเก็บและการจัดการได้เป็น 2 กลุ่ม คือ ข้อมูลประเภทพื้นฐาน และข้อมูลประเภทอ้างอิง

3.2.1 ข้อมูลประเภทพื้นฐาน

ข้อมูลประเภทพื้นฐาน (Primitive Data Types) เป็นข้อมูลที่เก็บค่าจริง (Actual Value) โดยตรงในหน่วยความจำ และมีลักษณะการจัดการแบบผ่านค่า (Pass-By-Value) กล่าวคือ เมื่อนำค่าไปกำหนดให้ตัวแปรอื่น หรือส่งเป็นอาร์กิวเมนต์ให้ฟังก์ชัน จะมีการสำเนาค่าจริงไปยังตำแหน่งใหม่ ไม่ใช่การอ้างอิงถึงตำแหน่งเดียวกัน ลักษณะการจัดการข้อมูลแบบผ่านค่าแสดงด้วยตัวอย่างต่อไปนี้

ตัวอย่างที่ 20

```
1  let a = 10;
2  let b = a;
3  a = 20;
4  console.log(b);
```

จากตัวอย่างที่ 20 ในบรรทัดที่ 1 เป็นการประกาศตัวแปรชื่อ a และกำหนดค่าเริ่มต้นเป็น 10 ส่วนบรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ b และกำหนดค่าเป็น a ซึ่งในขณะนี้ JavaScript Engine จะทำการสำเนาค่า 10 จากตัวแปร a ไปเก็บไว้ในตัวแปร b โดยไม่ได้อ้างอิงไปยังตำแหน่งเดียวกันในหน่วยความจำ

บรรทัดที่ 3 เป็นการเปลี่ยนค่าตัวแปร a เป็น 20 ซึ่งจะส่งผลให้เฉพาะตัวแปร a เท่านั้นที่มีค่าเปลี่ยนแปลง ส่วนตัวแปร b ยังคงเก็บค่า 10 ไว้เหมือนเดิม เนื่องจากเป็นการเก็บค่าแยกกันในหน่วยความจำ บรรทัดที่ 4 เป็นการแสดงผลค่าของตัวแปร b ซึ่งผลลัพธ์ที่ได้คือ 10 แสดงให้เห็นว่าการเปลี่ยนแปลงค่าของตัวแปร a ไม่ส่งผลกระทบต่อค่าของตัวแปร b ซึ่งเป็นลักษณะเฉพาะของการจัดการข้อมูลแบบผ่านค่าที่ใช้ในข้อมูลประเภทพื้นฐาน

นอกเหนือจากคุณลักษณะการจัดการข้อมูลดังกล่าวที่กล่าวถึงข้างต้นแล้ว ข้อมูลประเภทพื้นฐานประกอบด้วยค่าคงที่ (Literal) ดังที่ได้กล่าวถึงในหัวข้อ 2.2.2 ซึ่งมีลักษณะคงที่ไม่เปลี่ยนแปลง (Immutable) กล่าวคือ ค่าของข้อมูลไม่สามารถถูกเปลี่ยนแปลงโดยตรง ตัวแปรสามารถเปลี่ยนค่าที่ชี้ไปได้ แต่ค่าดั้งเดิมจะไม่ถูกปรับเปลี่ยนโดยการดำเนินการใดๆ อีกทั้งข้อมูลประเภทนี้ไม่มีคุณสมบัติ (Properties) หรือเมธอด (Methods) ของตัวเอง ทำให้เป็นข้อมูลรูปแบบที่เรียบง่ายที่สุดใน JavaScript ตามมาตรฐาน ECMAScript ข้อมูลพื้นฐานใน JavaScript มีทั้งหมด 7 ชนิด ดังนี้

1) ตัวเลข

ตัวเลข คือชนิดข้อมูลที่ใช้แทนค่าจำนวนจริงในภาษา JavaScript โดยสามารถเป็นได้ทั้งจำนวนเต็มและจำนวนทศนิยม โดยไม่จำแนกชนิดระหว่างจำนวนเต็ม (Integer) และจำนวนทศนิยม (Floating-point) ดังเช่นในภาษาอื่น เช่น Java หรือ C++ ทั้งนี้ JavaScript ใช้มาตรฐาน IEEE 754 แบบเลขทศนิยมความแม่นยำสองเท่า (Double-Precision Floating Point) ขนาด 64 บิต ในการจัดเก็บข้อมูล ทำให้สามารถแทนค่าตัวเลขในช่วงประมาณ $-(2^{53}-1)$ ถึง $2^{53}-1$ ข้อมูลชนิดนี้ สามารถนำไปใช้ในกระบวนการคำนวณ เช่น บวก ลบ คูณ หาร การเปรียบเทียบค่า และการดำเนินการทางคณิตศาสตร์ทั่วไปได้

2) ตัวเลขขนาดใหญ่

ตัวเลขขนาดใหญ่ (BigInt) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บจำนวนเต็มขนาดใหญ่ที่เกินขีดจำกัดของชนิดข้อมูล Number โดยได้รับการเพิ่มเข้ามาในภาษา JavaScript ตั้งแต่มาตรฐาน ECMAScript 2020 (ES11) เพื่อแก้ไขข้อจำกัดในการจัดเก็บจำนวนเต็มที่ข้อมูลชนิดตัวเลขรองรับได้อย่างแม่นยำในช่วง $-(2^{53}-1)$ ถึง $2^{53}-1$ และด้วยข้อมูลชนิดตัวเลขขนาดใหญ่ ผู้พัฒนาสามารถจัดการกับจำนวนเต็มที่มีขนาดใหญ่กว่าตัวเลขได้โดยไม่สูญเสียความแม่นยำ จึงเหมาะสำหรับการคำนวณที่ต้องการความเที่ยงตรงสูง เช่น การเข้ารหัสข้อมูล การประมวลผลตัวเลขขนาดใหญ่ และการจัดการเลขประจำตัวที่มีหลายหลัก เป็นต้นทั้งนี้ ตัวเลขขนาดใหญ่และตัวเลขเป็นชนิดข้อมูลที่แยกจากกัน ไม่สามารถดำเนินการทางคณิตศาสตร์ร่วมกันได้โดยตรง จึงต้องแปลงชนิดข้อมูลให้ตรงกันก่อนจึงจะสามารถคำนวณร่วมกันได้

3) สตริง

สตริง (String) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บข้อความหรือลำดับของอักขระ เช่น ตัวอักษร ตัวเลข สัญลักษณ์ และช่องว่าง โดย JavaScript ใช้การเข้ารหัสแบบ UTF-16 ในการจัดเก็บข้อมูลข้อความ ทำให้สามารถรองรับอักขระจากหลากหลายภาษา รวมถึงอีโมจิและสัญลักษณ์พิเศษต่าง ๆ ข้อมูลชนิดนี้มีลักษณะคงที่ไม่เปลี่ยนแปลง (Immutable) เช่นเดียวกับข้อมูลประเภทพื้นฐานอื่น ๆ กล่าวคือ เมื่อมีการดำเนินการใด ๆ กับข้อความ เช่น การตัดข้อความหรือการต่อข้อความ JavaScript จะสร้างข้อความใหม่ขึ้นมาแทนข้อความเดิม แทนที่จะปรับเปลี่ยนค่าที่มีอยู่โดยตรง

4) บูลีน

บูลีน (Boolean) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บค่าความจริง โดยมีค่าได้เพียง 2 ค่าเท่านั้น คือ true และ false ข้อมูลชนิดนี้มีความสำคัญอย่างมากในการเขียนโปรแกรม โดยเฉพาะในการควบคุมการทำงานของโปรแกรม การตัดสินใจ และการประเมินเงื่อนไขต่าง ๆ เช่น การใช้งานในคำสั่งควบคุมทางเลือก if-else การทำซ้ำด้วยการวนรอบ หรือการตรวจสอบสถานะของระบบ เป็นต้น ข้อมูลชนิดบูลีนมักเกิดขึ้นจากการเปรียบเทียบค่า การตรวจสอบเงื่อนไข หรือการดำเนินการทางตรรกศาสตร์ ภาษา JavaScript มีกลไกการแปลงค่าอื่นๆ เป็นบูลีนโดยอัตโนมัติ ซึ่งเรียกว่า "Truthy" และ "Falsy" values โดยค่าที่ถือว่าเป็น false ได้แก่ 0, "" (ข้อความว่าง), null, undefined, NaN, และ false ส่วนค่าอื่นๆ จะถือว่าเป็น true

5) ไม่ได้กำหนดค่า

ไม่ได้กำหนค่า (Undefined) คือชนิดข้อมูลที่แสดงถึงสถานะของตัวแปรที่ยังไม่ได้รับการกำหนดค่า หรือค่าที่ไม่มีอยู่จริงในระบบ ค่า undefined จะเกิดขึ้นโดยอัตโนมัติเมื่อมีการประกาศตัวแปรแต่ยังไม่ได้กำหนดค่าเริ่มต้นให้ หรือเมื่อพยายามเข้าถึงคุณสมบัติของออบเจกต์ที่ไม่มีอยู่ หรือเมื่อฟังก์ชันไม่ได้ส่งค่ากลับ (return) ชนิดข้อมูลนี้มีค่าเพียงค่าเดียวคือ undefined และมักใช้ในการตรวจสอบว่าตัวแปรหรือคุณสมบัติได้รับการกำหนดค่าแล้วหรือไม่ ความแตกต่างระหว่าง undefined และ null คือ undefined หมายถึงยังไม่ได้กำหนดค่า ส่วน null หมายถึงการกำหนดค่าว่างอย่างตั้งใจ

6) ค่าว่างเปล่า

ค่าว่างเปล่า (Null) คือชนิดข้อมูลที่ใช้สำหรับแสดงถึงค่าว่างที่ถูกกำหนดขึ้นอย่างตั้งใจ แตกต่างจาก undefined ที่เกิดขึ้นโดยอัตโนมัติเมื่อไม่ได้กำหนดค่า ค่า null มักใช้เพื่อระบุว่าตัวแปรนั้นมีค่าเป็นว่างเปล่าหรือไม่มีข้อมูลในขณะนั้น โดยเป็นการกำหนดอย่างชัดเจนจากผู้พัฒนา ชนิดข้อมูลนี้มีค่าเพียงค่าเดียวคือ null และมักใช้ในการเริ่มต้นตัวแปรที่จะเก็บออบเจกต์ การรีเซ็ตค่าตัวแปร หรือการแสดงให้เห็นว่าไม่มีข้อมูลในขณะนั้น

7) สัญลักษณ์

สัญลักษณ์ (Symbol) คือชนิดข้อมูลที่ได้รับการเพิ่มเข้ามาใน JavaScript ตั้งแต่ ECMAScript 2015 (ES6) โดยมีคุณลักษณะพิเศษคือการสร้างค่าที่ไม่ซ้ำกัน (Unique) ทุกครั้ง แม้ว่าจะใช้ข้อมูล

เดียวกันในการสร้างก็ตาม ข้อมูลชนิดสัญลักษณ์มักใช้เป็นคีย์ (Key) สำหรับคุณสมบัติของออบเจกต์เพื่อหลีกเลี่ยงการชนกันของชื่อคุณสมบัติ โดยเฉพาะในการพัฒนาไลบรารีหรือเฟรมเวิร์กที่ต้องการความปลอดภัยในการตั้งชื่อ

3.2.2 ข้อมูลประเภทอ้างอิง

ข้อมูลประเภทอ้างอิง (Reference Data Types) เป็นข้อมูลที่มีโครงสร้างซับซ้อนกว่าข้อมูลประเภทพื้นฐาน โดยตัวแปรจะเก็บการอ้างอิง (Reference) หรือที่อยู่ในหน่วยความจำที่ชี้ไปยังตำแหน่งจริงของข้อมูลแทนการเก็บค่าจริงโดยตรง ลักษณะการจัดการข้อมูลเป็นแบบผ่านการอ้างอิง (Pass-By-Reference) หมายความว่า เมื่อมีการกำหนดค่าให้ตัวแปรอื่นหรือส่งค่าไปยังฟังก์ชัน จะเป็นการสำเนาการอ้างอิงไปยังตำแหน่งเดียวกันในหน่วยความจำ ไม่ใช่การสำเนาข้อมูลจริง ทำให้การเปลี่ยนแปลงข้อมูลผ่านตัวแปรใดตัวแปรหนึ่งจะส่งผลต่อตัวแปรอื่นๆ ที่อ้างอิงไปยังข้อมูลเดียวกัน ข้อมูลประเภทอ้างอิงมีลักษณะเปลี่ยนแปลงได้ (Mutable) กล่าวคือ สามารถเปลี่ยนแปลงเนื้อหาภายในได้โดยไม่ต้องสร้างข้อมูลใหม่ นอกจากนี้ยังมีคุณสมบัติและเมธอดที่สามารถใช้งานได้ ลักษณะการจัดการข้อมูลแบบผ่านการอ้างอิงแสดงด้วยตัวอย่างต่อไปนี้

ตัวอย่างที่ 21

```
1  let person1 = { name: "Nok", age: 25 };
2  let person2 = person1;
3  person2.age = 30;
4  console.log(person1.age);
5  console.log(person2.age);
```

จากตัวอย่างที่ 21 บรรทัดที่ 1 เป็นการสร้างออบเจกต์ชื่อ person1 ที่มีคุณสมบัติ name เป็น "Nok" และ age เป็น 25 ส่วนบรรทัดที่ 2 เป็นการประกาศตัวแปรชื่อ person2 และกำหนดค่าเป็น person1 ซึ่งในขณะนี้ JavaScript Engine จะทำการสำเนาการอ้างอิง (Reference) จากตัวแปร person1 ไปยังตัวแปร person2 ไม่ใช่การคัดลอกค่าภายในออบเจกต์ ทำให้ทั้งสองตัวแปรชี้ไปยังออบเจกต์เดียวกันในหน่วยความจำ

บรรทัดที่ 3 เป็นการเปลี่ยนค่าคุณสมบัติ age ของออบเจกต์ผ่านตัวแปร person2 จากเดิม 25 เป็น 30 เนื่องจากทั้ง person1 และ person2 อ้างอิงไปยังออบเจกต์เดียวกัน การเปลี่ยนแปลงนี้จึงส่งผลต่อทั้งสองตัวแปร บรรทัดที่ 4 และ 5 เป็นการแสดงผลค่า age ของทั้งสองตัวแปร ซึ่งผลลัพธ์ที่ได้คือ 30 ทั้งคู่ แสดงให้เห็นว่าการเปลี่ยนแปลงผ่านตัวแปรใดตัวแปรหนึ่งจะส่งผลกระทบต่อตัวแปรอื่นที่อ้างอิงไปยังออบเจกต์เดียวกัน ซึ่งเป็นลักษณะเฉพาะของการจัดการแบบผ่านการอ้างอิงของข้อมูลประเภทอ้างอิง

ข้อมูลประเภทอ้างอิงใน JavaScript ประกอบด้วยชนิดหลักๆ 3 ชนิด ดังนี้

1) ออบเจกต์

ออบเจกต์ (Object) คือชนิดข้อมูลพื้นฐานของข้อมูลประเภทอ้างอิงใน JavaScript โดยเป็นโครงสร้างข้อมูลที่ใช้จัดเก็บข้อมูลในรูปแบบคู่ของคีย์และค่า (Key-Value Pairs) ซึ่งคีย์เป็นชื่อของคุณสมบัติ

(Property) และค่าเป็นข้อมูลที่เก็บไว้ในคุณสมบัตินั้น เช่น person1 = { name: "สมชาย", age: 25 }; คุณสมบัติชื่อ name มีค่าเป็นข้อความว่า Nok และคุณสมบัติ age มีค่าเป็นตัวเลข 25 เป็นต้น

ออบเจกต์สามารถเก็บข้อมูลได้หลากหลายชนิด ทั้งข้อมูลประเภทพื้นฐาน ข้อมูลประเภทอ้างอิงอื่น หรือแม้แต่ฟังก์ชัน ทำให้เป็นโครงสร้างข้อมูลที่มีความยืดหยุ่นสูงและเหมาะสำหรับการจำลองสิ่งต่าง ๆ ในโลกจริง เช่น ข้อมูลบุคคล สินค้า เป็นต้น

การสร้างออบเจกต์สามารถทำได้หลายวิธี เช่น การใช้รูปแบบค่าคงที่ออบเจ็กต์ การใช้ new Object() หรือการสร้างด้วยคลาส (Class) การเข้าถึงคุณสมบัติของออบเจกต์สามารถทำได้ผ่านสัญลักษณ์จุด (Dot Notation) เช่น obj.name หรือผ่านสัญลักษณ์วงเล็บเหลี่ยม (Bracket Notation) เช่น obj["name"] ทั้งนี้การเลือกวิธีเข้าถึงจะขึ้นอยู่กับลักษณะของชื่อคุณสมบัติและความเหมาะสมของการใช้งานในบริบทนั้น ๆ

2) อาร์เรย์

อาร์เรย์ (Array) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บข้อมูลหลายค่าในตัวแปรเดียว โดยจัดเรียงข้อมูลเป็นแถวลำดับและใช้หมายเลขดัชนี (Index) ในการอ้างอิงแต่ละองค์ประกอบ เริ่มต้นจากหมายเลข 0 เช่น value=[50,35,38,90] เป็นนิพจน์การสร้างข้อมูลชนิดอาร์เรย์ 1 มิติ โดย value[0] มีค่าเป็น 50 และ value[3] มีค่าเป็น 90 เป็นต้น อาร์เรย์ใน JavaScript มีความยืดหยุ่นสูง สามารถเก็บข้อมูลได้หลากหลายชนิดในอาร์เรย์เดียวกัน และมีขนาดที่สามารถเปลี่ยนแปลงได้ระหว่างการทำงานของโปรแกรม อาร์เรย์มีเมธอดมากมายที่ช่วยในการจัดการข้อมูล เช่น การเพิ่ม ลบ ค้นหา และจัดเรียงข้อมูล เป็นต้น

3) ฟังก์ชัน

ฟังก์ชัน (Function) คือชนิดข้อมูลพิเศษใน JavaScript ที่เป็นทั้งข้อมูลประเภทอ้างอิงและกลุ่มของรหัสคำสั่งที่สามารถเรียกใช้งานได้ ฟังก์ชันใน JavaScript เป็น First-Class Objects หมายความว่าสามารถนำไปใช้เหมือนข้อมูลชนิดอื่น เช่น กำหนดให้ตัวแปร ส่งเป็นพารามิเตอร์ให้ฟังก์ชันอื่น หรือคืนค่าจากฟังก์ชัน ฟังก์ชันมีความสำคัญอย่างมากในการเขียนโปรแกรมเนื่องจากช่วยให้สามารถแบ่งโปรแกรมเป็นส่วนย่อยๆ ที่สามารถใช้งานซ้ำได้และจัดการได้ง่าย

4. โครงสร้างการควบคุม

โครงสร้างการควบคุม (Control Structures) เป็นกลไกสำคัญที่ช่วยให้นักพัฒนาสามารถควบคุมการไหลของการทำงานในโปรแกรม แทนการทำงานแบบเรียงลำดับจากบนลงล่างเพียงอย่างเดียว โครงสร้างเหล่านี้ทำให้โปรแกรมสามารถตัดสินใจ วนซ้ำ และดำเนินการตามเงื่อนไขได้ ซึ่งเป็นพื้นฐานสำคัญในการสร้างโปรแกรมที่มีปฏิสัมพันธ์และความซับซ้อน โดยเฉพาะในการพัฒนาเว็บแอปพลิเคชันที่ต้องตอบสนองต่อการกระทำของผู้ใช้และจัดการกับข้อมูลที่เปลี่ยนแปลงได้

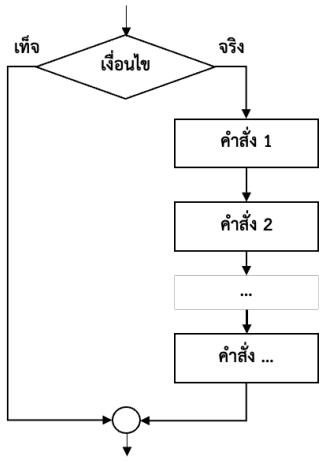
JavaScript มีโครงสร้างการควบคุม 2 ประเภท ได้แก่ โครงสร้างการตัดสินใจ (Decision Structures) ที่ใช้ในการเลือกเส้นทางการทำงานตามเงื่อนไข และโครงสร้างการวนซ้ำ (Loop Structures) ที่ใช้ในการทำงานซ้ำๆ จนกว่าจะครบตามเงื่อนไขที่กำหนด

4.1 โครงสร้างการตัดสินใจ

โครงสร้างการตัดสินใจเป็นกลไกที่ช่วยให้โปรแกรมสามารถเลือกทำงานในเส้นทางที่แตกต่างกันตามผลการประเมินเงื่อนไข โดยใช้ค่า Boolean (true หรือ false) เป็นเกณฑ์ในการตัดสินใจ ในกรณีที่เงื่อนไขเป็นจริง โปรแกรมจะดำเนินการตามชุดคำสั่งหนึ่ง และหากเงื่อนไขเป็นเท็จ โปรแกรมจะดำเนินการตามชุดคำสั่งอื่นหรืออาจข้ามไปยังส่วนถัดไป

4.1.1 โครงสร้างการตัดสินใจแบบทางเลือกเดียว

โครงสร้างการตัดสินใจแบบทางเลือกเดียว (Single-Selection Structure) เป็นรูปแบบการควบคุมการทำงานของโปรแกรมที่มีการตรวจสอบเงื่อนไขที่กำหนด และจะดำเนินการตามคำสั่งที่อยู่ในกลุ่มประโยคคำสั่งที่กำหนดเมื่อเงื่อนไขเป็นจริงเท่านั้น หากเงื่อนไขเป็นเท็จ โปรแกรมจะข้ามกลุ่มประโยคคำสั่งดังกล่าว และจะดำเนินการต่อในส่วนถัดไปโดยไม่มีการประมวลผลคำสั่งในส่วนนั้น โครงสร้างการตัดสินใจแบบทางเลือกเดียวมีลักษณะโครงสร้างดังภาพที่ 9.5



ภาพที่ 9.5 โครงสร้างการตัดสินใจแบบทางเลือกเดียว

จากภาพที่ 9.5 โครงสร้างการตัดสินใจแบบทางเลือกเดียวเหมาะกับกรณีที่ต้องการให้โปรแกรมทำงานบางอย่างเมื่อมีเงื่อนไขเฉพาะเจาะจง โดยไม่ต้องกำหนดทางเลือกอื่นเมื่อเงื่อนไขไม่เป็นจริง เช่น การบันทึกข้อมูลเมื่อผู้ใช้กดปุ่มยืนยัน หากผู้ใช้ไม่ยืนยันก็ไม่มีการดำเนินการใดๆ เป็นต้น การสร้างโครงสร้างการตัดสินใจแบบทางเลือกเดียวใน JavaScript ใช้คำสั่งควบคุม if โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

```
if (เงื่อนไข) {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง;
        ...;
}
```

ตัวอย่างที่ 22

```
1  let age = 18;
2  if (age >= 18) {
3          console.log("คุณมีสิทธิ์ลงคะแนนเสียง");
4  }
5  let score = 75;
6  if (score >= 80) {
7          console.log("คะแนนของคุณอยู่ในระดับดีเยี่ยม");
8  }
```

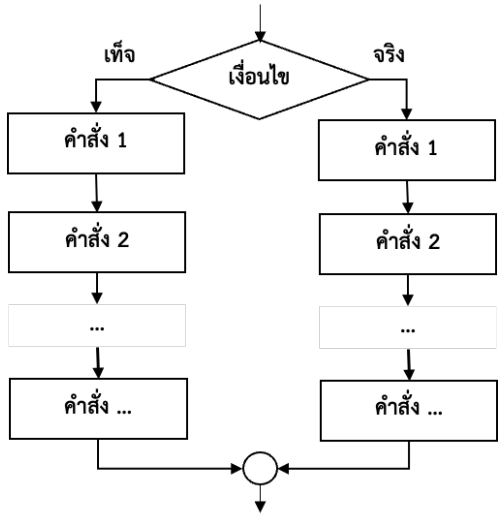
จากตัวอย่างที่ 22 บรรทัดที่ 1 เป็นการประกาศตัวแปร age กำหนดค่าเป็นตัวเลข 18 ในบรรทัดที่ 2 เป็นคำสั่งควบคุม if ที่ใช้นิพจน์เปรียบเทียบ age >= 18 ซึ่งเมื่อทำการประเมินค่านิพจน์แล้วทำให้ได้ค่าผลลัพธ์ของนิพจน์เป็น true เนื่องจาก age มีค่าเป็น 18 ทำให้มีการดำเนินการในกลุ่มประโยคคำสั่งบรรทัดที่ 3 เป็นคำสั่งให้แสดงข้อความ *คุณมีสิทธิ์ลงคะแนนเสียง* ออกทางคอนโซลของเว็บเบราว์เซอร์

ในบรรทัดที่ 5 เป็นการประกาศตัวแปร score และกำหนดค่าเป็นตัวเลข 75 และในบรรทัดที่ 6 เป็นคำสั่งควบคุม if ที่ใช้นิพจน์เปรียบเทียบ score >= 80 เมื่อมีการประเมินค่านิพจน์นี้แล้วจะได้ผลการประเมินค่าเป็น false เนื่องจากค่า score คือ 75 ซึ่งไม่เป็นไปตามค่าตรวจสอบเงื่อนไขที่จะต้องมีค่ามากกว่าหรือเท่ากับ 80 ทำให้ไม่ดำเนินการในกลุ่มประโยคคำสั่งของคำสั่งควบคุม if ในบรรทัดที่ 7

4.1.2 โครงสร้างการตัดสินใจแบบสองทางเลือก

โครงสร้างการตัดสินใจแบบสองทางเลือก (Double-Selection Structure) เป็นรูปแบบการควบคุมการทำงานของโปรแกรมที่มีการตรวจสอบเงื่อนไขที่กำหนด และจะดำเนินการตามคำสั่งที่อยู่ในกลุ่มประโยคคำสั่งชุดหนึ่งเมื่อเงื่อนไขเป็นจริง หรือดำเนินการตามคำสั่งที่อยู่ในกลุ่มประโยคคำสั่งอีกชุดหนึ่งเมื่อเงื่อนไขเป็นเท็จ โครงสร้างนี้แตกต่างจากการตัดสินใจแบบทางเลือกเดียวตรงที่มีการกำหนดทางเลือกสำหรับทั้งสองกรณี ทำให้โปรแกรมจะดำเนินการอย่างใดอย่างหนึ่งเสมอ

โครงสร้างการตัดสินใจลักษณะนี้เหมาะกับสถานการณ์ที่ต้องจัดการกับผลลัพธ์สองทาง เช่น การตรวจสอบสิทธิ์ผู้ใช้ หากผู้ใช้ป้อนรหัสผ่านถูกต้อง ระบบจะอนุญาตให้เข้าสู่ระบบ แต่หากป้อนผิดจะส่งข้อความแจ้งข้อผิดพลาดให้ผู้ใช้ทราบ เป็นต้น การออกแบบในลักษณะนี้ทำให้โปรแกรมมีความยืดหยุ่นและสามารถควบคุมการไหลของโปรแกรมโดยการกำหนดพฤติกรรมที่แตกต่างกันในแต่ละกรณี โครงสร้างการตัดสินใจแบบสองทางเลือกมีลักษณะดังภาพที่ 9.6



ภาพที่ 9.6 โครงสร้างการตัดสินใจแบบสองทางเลือก

จากภาพที่ 9.6 สามารถนำไปสร้างโครงสร้างการตัดสินใจแบบสองทางเลือกในภาษา JavaScript โดยใช้คำสั่งควบคุม if-else ที่มีรูปแบบและตัวอย่างดังนี้

รูปแบบ

```
if (เงื่อนไข) {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง;
        ...;
} else {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขไม่เป็นจริง;
        ...;
}
```

ตัวอย่างที่ 23

```
1  let age = 20;
2  if (age >= 18) {
3          console.log("คุณเป็นผู้ใหญ่แล้ว สามารถออกเสียงเลือกตั้งได้");
4  } else {
5          console.log("คุณยังเป็นเด็กและเยาวชน ยังไม่สามารถออกเสียงเลือกตั้งได้");
6  }
```

จากตัวอย่างที่ 23 เป็นรหัสคำสั่งที่ใช้ตรวจสอบสิทธิ์ในการเลือกตั้งโดยจะมีสิทธิ์เมื่ออายุ 18 ปีขึ้นไป บรรทัดที่ 1 เป็นการประกาศตัวแปร age และกำหนดค่าเป็นตัวเลข 20 ในบรรทัดที่ 2 เป็นคำสั่งควบคุม if ที่ใช้นิพจน์เปรียบเทียบ age >= 18 ซึ่งเมื่อทำการประเมินค่านิพจน์แล้วจะได้ค่าผลลัพธ์ของนิพจน์เป็น true เนื่องจาก age มีค่าเป็น 20 ซึ่งมากกว่า 18 ทำให้มีการดำเนินการในกลุ่มประโยคคำสั่งในบรรทัดที่ 3 เป็นคำสั่งให้แสดงข้อความ *คุณเป็นผู้ใหญ่แล้ว สามารถออกเสียงเลือกตั้งได้* ออกทางคอนโซลของเว็บเบราว์เซอร์ เมื่อเงื่อนไขของคำสั่ง if เป็นจริงแล้ว คำสั่งในส่วน else ซึ่งอยู่ในบรรทัดที่ 4-6 จะไม่ถูกดำเนินการ ดังนั้นข้อความ *คุณยังเป็นเด็กและเยาวชน ยังไม่สามารถออกเสียงเลือกตั้งได้* ในบรรทัดที่ 6 จะไม่ถูกแสดงผล

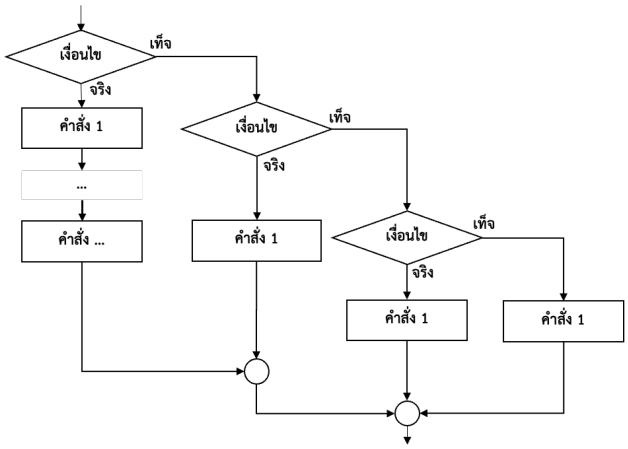
หากต้องการทดสอบการทำงานของส่วน else สามารถเปลี่ยนค่าของตัวแปร age เป็นค่าที่น้อยกว่า 18 เช่น กำหนดให้ age = 15 เมื่อดำเนินการประเมินค่านิพจน์ age >= 18 จะได้ผลลัพธ์เป็น false ทำให้ข้ามการดำเนินการในกลุ่มประโยคคำสั่งของ if และดำเนินการในกลุ่มประโยคคำสั่งของ else แทน ซึ่งจะแสดงข้อความ *คุณยังเป็นเด็กและเยาวชน ยังไม่สามารถออกเสียงเลือกตั้งได้* ออกทางคอนโซลของเว็บเบราว์เซอร์

4.1.3 โครงสร้างการตัดสินใจแบบหลายทางเลือก

โครงสร้างการตัดสินใจแบบหลายทางเลือก (Multiple-Selection Structure) เป็นรูปแบบของการควบคุมลำดับการทำงานของโปรแกรมที่ใช้ในกรณีที่ต้องการตรวจสอบเงื่อนไขหลายเงื่อนไขตามลำดับ เพื่อเลือกเส้นทางการทำงานที่เหมาะสมจากทางเลือกที่มีอยู่หลายทาง กลไกของโครงสร้างนี้เริ่มต้นจากการตรวจสอบเงื่อนไขแรกที่ถูกกำหนดไว้ หากเงื่อนไขดังกล่าวเป็นจริง โปรแกรมจะทำการประมวลผลตามกลุ่มคำสั่งที่กำหนดสำหรับกรณีนั้นทันที และเมื่อดำเนินการเสร็จสิ้นแล้ว โปรแกรมจะออกจากโครงสร้างการตัดสินใจโดยไม่ตรวจสอบเงื่อนไขอื่น ๆ ต่อไป

อย่างไรก็ตาม หากการตรวจสอบเงื่อนไขแรกพบว่าเป็นเท็จ โปรแกรมจะทำการตรวจสอบเงื่อนไขถัดไปตามลำดับที่กำหนดไว้ โดยกระบวนการนี้จะดำเนินไปเรื่อย ๆ จนกระทั่งพบเงื่อนไขที่เป็นจริง หรือหากตรวจสอบจนถึงเงื่อนไขสุดท้ายแล้วยังไม่พบเงื่อนไขใดที่เป็นจริง โปรแกรมจะทำการประมวลผลตามกลุ่มคำสั่งที่จัดเตรียมไว้สำหรับกรณีอื่น ๆ ซึ่งเป็นส่วนสำรองสำหรับสถานการณ์ที่ไม่ตรงกับเงื่อนไขใด ๆ ที่กำหนดไว้

โครงสร้างการตัดสินใจลักษณะนี้มักใช้ในกรณีที่ต้องเลือกการทำงานจากชุดทางเลือกที่มีจำนวนมาก เช่น การเลือกรูปแบบการคำนวณตามประเภทข้อมูล การกำหนดผลลัพธ์ที่แตกต่างกันตามค่าที่ผู้ใช้ป้อนเข้ามา หรือการจัดการเมนูคำสั่งในโปรแกรม เป็นต้น โดยข้อดีของโครงสร้างนี้คือช่วยให้รหัสคำสั่งอ่านได้ง่าย มีลำดับการตรวจสอบเงื่อนไข และสามารถจัดการกับกรณีที่ซับซ้อนได้ โครงสร้างดังกล่าวสามารถอธิบายได้ตามผังลำดับขั้นในภาพที่ 9.7 ซึ่งแสดงให้เห็นการตรวจสอบเงื่อนไขทีละขั้นก่อนเลือกการทำงานที่เหมาะสมต่อไป



ภาพที่ 9.7 โครงสร้างการตัดสินใจแบบหลายทางเลือก

จากภาพที่ 9.7 เป็นโครงสร้างการตัดสินใจแบบหลายทางเลือก เหมาะสำหรับสถานการณ์ที่มีทางเลือกมากกว่า 2 ทางเลือก และแต่ละทางเลือกมีเงื่อนไขที่แตกต่างกัน การสร้างโครงสร้างการตัดสินใจแบบหลายเลือกใน JavaScript ใช้คำสั่งควบคุม if-else if-else โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

```
if (เงื่อนไขที่ 1) {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง;
        ...;
} else if(เงื่อนไขที่ 2) {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขที่ 2 เป็นจริง;
        ...;
} else {
        คำสั่งที่จะทำงานเมื่อไม่มีเงื่อนไขใดๆ ที่เป็นจริง;
        ...;
}
```

ตัวอย่างที่ 24

```
1   let value = -20;
2   if (value > 0) {
3           console.log("จำนวนบวก");
4   } else if (value == 0) {
5           console.log("จำนวนศูนย์");
```

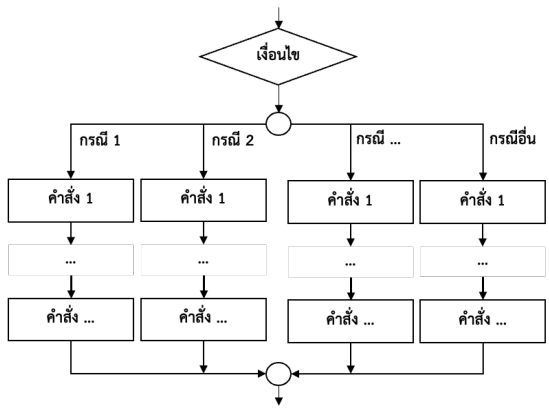
```
6   } else {
7           console.log("จำนวนลบ");
8   }
```

จากตัวอย่างที่ 24 เป็นรหัสคำสั่งที่ใช้ตรวจสอบว่าค่าที่ตรวจสอบเป็นจำนวนบวก ศูนย์หรือลบ โดยบรรทัดที่ 1 เป็นการประกาศตัวแปร value และกำหนดค่าเป็นตัวเลข -20 ในบรรทัดที่ 2 เป็นคำสั่งควบคุม if ที่ใช้นิพจน์เปรียบเทียบ value > 0 ซึ่งเมื่อทำการประเมินค่านิพจน์แล้วจะได้ค่าผลลัพธ์เป็น false เนื่องจาก -20 ไม่มากกว่า 0 ทำให้ข้ามไปตรวจสอบเงื่อนไขถัดไป ในบรรทัดที่ 4 เป็นคำสั่ง else if ที่ตรวจสอบเงื่อนไข value == 0 ซึ่งเมื่อทำการประเมินค่านิพจน์แล้วจะได้ค่าผลลัพธ์เป็น false เนื่องจาก -20 ไม่เท่ากับ 0 ทำให้ข้ามไปยังส่วน else ในบรรทัดที่ 7 และดำเนินการในกลุ่มประโยคคำสั่งในบรรทัดที่ 8 ซึ่งเป็นคำสั่งให้แสดงข้อความ *จำนวนลบ* ออกทางคอนโซลของเว็บเบราว์เซอร์

4.1.4 โครงสร้างการตัดสินใจแบบหลายกรณี

โครงสร้างการตัดสินใจแบบหลายกรณี (Switch Structure) เป็นรูปแบบการควบคุมการทำงานของโปรแกรมที่ทำหน้าที่เป็นทางเลือกแทนการใช้โครงสร้างการตัดสินใจแบบหลายเลือก (if-else if-else) โดยเฉพาะอย่างยิ่งในกรณีที่ต้องการตรวจสอบค่าของตัวแปรหรือนิพจน์เพียงค่าหนึ่งกับค่าคงที่หลายค่า ซึ่งโครงสร้างนี้ใช้การเปรียบเทียบแบบเท่ากัน (===) เพื่อกำหนดว่าคำสั่งใดจะถูกดำเนินการต่อไป การใช้ switch ช่วยให้รหัสคำสั่งกระชับ อ่านง่าย และจัดการกรณีจำนวนมากได้อย่างเป็นระเบียบ

อย่างไรก็ตาม โครงสร้างการตัดสินใจแบบหลายกรณีไม่เหมาะสมกับสถานการณ์ที่ต้องการตรวจสอบเงื่อนไขที่ซับซ้อน เช่น การเปรียบเทียบค่าตามช่วง (มากกว่า หรือน้อยกว่า) หรือการใช้เงื่อนไขแบบเชิงตรรกะ เนื่องจาก switch สามารถตรวจสอบได้เฉพาะค่าที่เท่ากันอย่างเคร่งครัดเท่านั้น นอกจากนี้ ภายในโครงสร้าง switch แต่ละกรณี (case) มักต้องมีคำสั่ง break เพื่อป้องกันการไหลไปยังกรณีถัดไปโดยไม่ได้ตั้งใจ ซึ่งเป็นลักษณะเฉพาะของการทำงานใน switch โครงสร้างการตัดสินใจแบบหลายกรณีนี้สามารถอธิบายได้จากผังลำดับขั้นในภาพที่ 9.8



ภาพที่ 9.8 โครงสร้างการตัดสินใจแบบหลายกรณี

จากภาพที่ 9.8 แสดงโครงสร้างการตัดสินใจแบบหลายกรณี ซึ่งโครงสร้างแบบนี้มีจุดเด่นที่ทำให้รหัสคำสั่งอ่านง่ายและเป็นระเบียบกว่าการใช้ if-else ซ้อนหลายชั้น โดยการใช้คำสั่ง switch จะทำการตรวจสอบค่าของตัวแปรหรือนิพจน์กับค่าคงที่แต่ละกรณีตามลำดับ และเมื่อพบกรณีที่ตรงกันจะดำเนินการคำสั่งในกลุ่มนั้นแล้วหยุดการดำเนินการเมื่อพบคำสั่ง break แล้วจึงทำคำสั่งในส่วนอื่นนอกคำสั่ง switch ส่วนประกอบของคำสั่ง switch ประกอบด้วย

- คำสั่งควบคุม switch เป็นคำสั่งเริ่มต้นของโครงสร้าง และนิพจน์หรือตัวแปรที่เป็นค่าที่จะถูกนำมาเปรียบเทียบกับค่าใน case ต่าง ๆ
- คำสงวน case ค่าคงที่: ใช้กำหนดค่าคงที่ที่จะเปรียบเทียบกับตัวแปรหรือนิพจน์ในคำสั่งควบคุม switch โดยจะมีเครื่องหมาย : หลังค่าคงที่เสมอ
- คำสงวน break ใช้สำหรับออกจากโครงสร้าง switch เพื่อป้องกันไม่ให้ดำเนินการในกรณีถัดไป
- คำสงวน default: กำหนดกลุ่มคำสั่งสำหรับกรณีที่ไม่มีกรณีใดตรงกัน ท้ายคำสงวนจะต้องมีเครื่องหมาย : เสมอ ทั้งนี้ส่วนนี้อาจไม่ระบุก็ได้

รูปแบบของคำสั่งควบคุมสำหรับโครงสร้างการตัดสินใจแบบหลายกรณีในภาษา JavaScript มีดังนี้

รูปแบบ

```
switch (ตัวแปรหรือนิพจน์) {
    case ค่าคงที่_1:
        คำสั่งที่ต้องดำเนินการเมื่อตัวแปรหรือนิพิจน์ตรวจสอบมีค่าเท่ากับ ค่าคงที่_1;
        ...;
        break;
    case ค่าคงที่_2:
        คำสั่งที่ต้องดำเนินการเมื่อตัวแปรหรือนิพิจน์ตรวจสอบมีค่าเท่ากับ ค่าคงที่_1;
        ...;
        break;
    default:
        คำสั่งที่ต้องดำเนินการเมื่อตัวแปรหรือนิพิจน์ตรวจสอบไม่ตรงกับค่าคงที่ใดๆ
        ...;
}
```

ตัวอย่างที่ 25

```
1   let  option=2;
2   switch (option) {
3         case  1:
4             console.log("หนึ่ง");
5             break;
6         case  2:
7             console.log("สอง");
8             break;
9         default:
10            console.log("ไม่ตรงกับกรณีใดๆ")
11  }
12  console.log("สิ้นสุดโปรแกรม");
```

จากตัวอย่างที่ 25 เป็นรหัสคำสั่งสำหรับการตรวจสอบว่าค่าที่ตรวจสอบมีค่าเป็น 1 หรือ 2 โดยใน บรรทัดที่ 1 เป็นการประกาศตัวแปร option และกำหนดค่าเป็นตัวเลข 2 ในบรรทัดที่ 2 เป็นคำสั่งควบคุม switch ที่เปรียบเทียบค่าของ option กับค่าใน case ต่าง ๆ โดยเริ่มตรวจสอบจาก case 1 ในบรรทัดที่ 3 ซึ่งไม่ตรงกับค่า option ที่เป็น 2 จึงข้ามไปตรวจสอบ case 2 ในบรรทัดที่ 6 เมื่อพบว่าค่าตรงกัน จึงดำเนินการแสดงข้อความ "สอง" ในบรรทัดที่ 7 และใช้คำสั่ง break ในบรรทัดที่ 8 เพื่อออกจากโครงสร้าง switch โดยข้าม default ในบรรทัดที่ 9-10 สุดท้ายในบรรทัดที่ 12 จะแสดงข้อความ "สิ้นสุดโปรแกรม" ออกทางคอนโซลของเว็บเบราว์เซอร์

ข้อควรพิจารณาในการใช้งานคำสั่งควบคุม switch ประกอบด้วย 1) ควรระบุประโยคคำสั่ง break; เมื่อสิ้นการทำงานในแต่ละกรณีเสมอ เพื่อหยุดการทำงานของโครงสร้าง switch และป้องกันไม่ให้คำสั่งในกรณีถัดไปถูกดำเนินการโดยไม่ตั้งใจ (Fall-Through) 2) ควรตรวจสอบประเภทข้อมูลเนื่องจากการเปรียบเทียบเงื่อนไขของ switch จะเป็นแบบพิจารณาชนิดข้อมูลด้วย 3) ค่าที่ใช้ระบุในแต่ละ case ต้องเป็นค่าคงที่ (Literal) ไม่สามารถใช้ตัวแปรหรือนิพจน์ที่ซับซ้อนได้ และ 4) แม้ว่าการใช้ default สามารถละไว้ได้ แต่ควรใช้เสมอเพื่อจัดการในกรณีที่ไม่คาดคิด

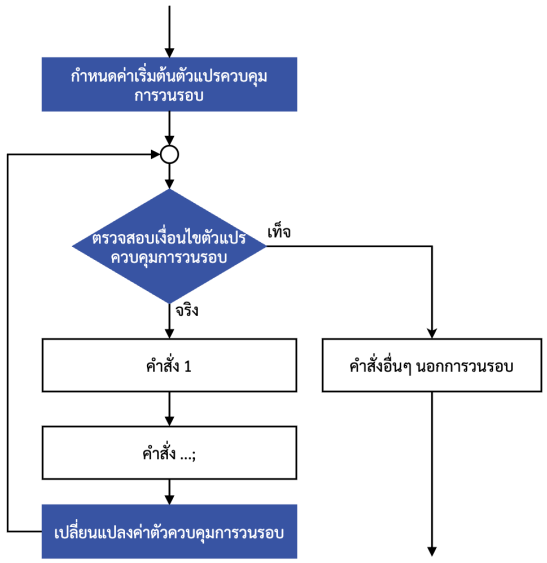
4.2 โครงสร้างการวนซ้ำ

โครงสร้างการวนซ้ำ (Loop Structure) เป็นรูปแบบการควบคุมการทำงานของโปรแกรมที่ช่วยให้สามารถดำเนินการชุดคำสั่งเดียวกันซ้ำหลายครั้งตามเงื่อนไขที่กำหนด โครงสร้างนี้มีความสำคัญอย่างยิ่งในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากช่วยลดการเขียนรหัสคำสั่งซ้ำซ้อน เพิ่มประสิทธิภาพ และทำให้การจัดการข้อมูลจำนวนมากเป็นไปได้อย่างมีประสิทธิภาพ โครงสร้างการวนซ้ำจะประกอบด้วยส่วนประกอบสำคัญ 3 ส่วน ได้แก่

เงื่อนไขการวนซ้ำ (Loop Condition) ซึ่งเป็นตัวกำหนดว่าการวนซ้ำจะดำเนินการต่อหรือหยุด กลุ่มประโยคคำสั่งที่จะทำซ้ำ (Loop Body) ซึ่งเป็นชุดคำสั่งที่จะถูกดำเนินการในแต่ละรอบของการวนซ้ำ และการปรับปรุงตัวแปรควบคุม (Loop Variable Update) ซึ่งเป็นการเปลี่ยนแปลงค่าตัวแปรที่ใช้ในเงื่อนไขการวนซ้ำเพื่อให้การวนซ้ำสามารถสิ้นสุดได้

4.2.1 โครงสร้างการวนซ้ำแบบ for

โครงสร้างการวนซ้ำแบบ for (for Loop) เป็นรูปแบบการวนซ้ำที่เหมาะสำหรับกรณีที่ทราบจำนวนรอบการวนซ้ำที่แน่นอน หรือต้องการควบคุมการวนซ้ำด้วยตัวแปรนับ โดยการทำงานจะเริ่มจากการกำหนดค่าเริ่มต้นให้กับตัวแปรนับ ตรวจสอบเงื่อนไขการวนซ้ำ และปรับค่าเพิ่มหรือลดของตัวแปรนับในแต่ละรอบ ลำดับการทำงานของโครงสร้างการวนซ้ำแบบ for มีความเป็นระบบและชัดเจน ทำให้มักถูกนำไปใช้ในกรณีที่ต้องทำงานซ้ำกับกลุ่มข้อมูลหรือทำงานที่มีจำนวนรอบแน่นอน เช่น การแสดงผลข้อมูลในรายการ หรือลูปผ่านค่าตัวเลขต่อเนื่อง เป็นต้น ลำดับการทำงานของโครงสร้างการวนซ้ำแบบ for แสดงดังภาพที่ 9.9



ภาพที่ 9.9 โครงสร้างการวนซ้ำแบบ for

จากภาพที่ 9.9 การทำงานของการวนซ้ำแบบ for จะเริ่มจากการกำหนดค่าเริ่มต้นของตัวแปรควบคุมการวนซ้ำ (Initialization) จากนั้นจะทำการตรวจสอบเงื่อนไขการวนซ้ำ หากเงื่อนไขเป็นจริงก็จะทำคำสั่งในกลุ่มประโยคคำสั่งที่กำหนด เมื่อทำคำสั่งเสร็จแล้วก็จะทำการเปลี่ยนแปลงค่าตัวแปรควบคุมการวนซ้ำ แล้ววนกลับไปตรวจสอบเงื่อนไขการวนซ้ำอีกครั้ง หากตรวจสอบแล้วได้ผลเป็นเท็จก็จะสิ้นสุดการวนซ้ำและทำคำสั่งที่อยู่ถัดไป

โครงสร้างของคำสั่ง for จะประกอบด้วย 3 ส่วนหลัก คือ การกำหนดค่าเริ่มต้น (Initialization) เงื่อนไขในการทำซ้ำ (Condition) และการเปลี่ยนแปลงค่า (Update) ซึ่งจะถูกรวมไว้ในบรรทัดเดียวกันโดยมีรูปแบบและตัวอย่างดังต่อไปนี้

รูปแบบ

```
for (นิพจน์กำหนดค่าเริ่มต้น; นิพจน์ตรวจสอบเงื่อนไข; นิพจน์เปลี่ยนแปลงค่าตัวแปรควบคุม) {
    คำสั่งที่ 1;
    ...;
}
```

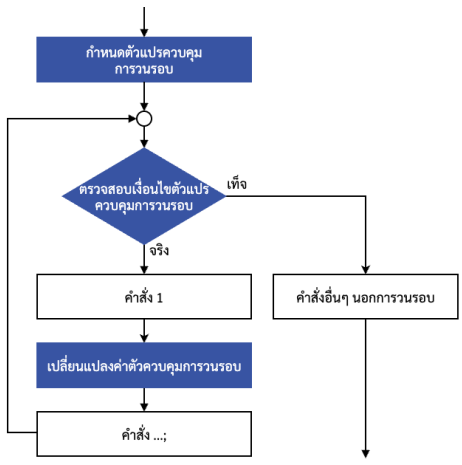
ตัวอย่างที่ 26

```
1  for (let loop = 1; loop <= 5; loop=loop+1) {
2          console.log("ลำดับที่ " + loop);
3  }
4  console.log("จบการวนซ้ำ");
```

จากตัวอย่างที่ 26 บรรทัดที่ 1 เป็นคำสั่งควบคุมการวนซ้ำ for ที่มีการกำหนดค่าเริ่มต้นของตัวแปร loop เป็น 1 เงื่อนไขการวนซ้ำคือ loop น้อยกว่าหรือเท่ากับ 5 และการปรับปรุงตัวแปรด้วยการเพิ่มค่าครั้งละ 1 ในแต่ละรอบ ในรอบแรกของการวนซ้ำ ตัวแปร loop มีค่าเป็น 1 ซึ่งตรงตามเงื่อนไขการตรวจสอบที่จะต้องน้อยกว่าหรือเท่ากับ 5 จึงมีการทำงานในกลุ่มประโยคคำสั่งในบรรทัดที่ 2 ซึ่งเป็นการแสดงข้อความ ลำดับที่ และค่าของตัวแปร loop คือ 1 (ลำดับที่ 1) จากนั้นจะมีการเพิ่มค่า loop เป็น 2 และตรวจสอบเงื่อนไขใหม่ กระบวนการนี้จะดำเนินต่อไปจนกระทั่ง loop มีค่าเป็น 6 ซึ่งไม่ตรงตามเงื่อนไข loop <= 5 การวนซ้ำจึงสิ้นสุดและดำเนินการคำสั่งในบรรทัดที่ 4 เพื่อแสดงข้อความ *จบการวนซ้ำ*

4.2.2 โครงสร้างการวนซ้ำแบบ while

โครงสร้างการวนซ้ำแบบ while (while Loop) เป็นรูปแบบการวนซ้ำที่เหมาะสำหรับกรณีที่ไม่ทราบจำนวนรอบการวนซ้ำที่แน่นอน แต่มีเงื่อนไขการหยุดที่ชัดเจนโดยใช้ค่าของตัวแปรเป็นเกณฑ์ ซึ่งมีโครงสร้างแสดงดังภาพที่ 9.10



ภาพที่ 9.10 โครงสร้างการวนซ้ำแบบ while

จากภาพที่ 9.10 แสดงโครงสร้างการวนซ้ำแบบ while ซึ่งใช้ตัวแปรควบคุมการวนซ้ำภายนอกโครงสร้างคำสั่ง กล่าวคือ โปรแกรมจะต้องมีการประกาศและกำหนดค่าของตัวแปรควบคุมการวนซ้ำไว้ล่วงหน้าโดยไม่เป็นส่วนหนึ่งของคำสั่ง while ต่างจากคำสั่ง for ที่สามารถกำหนดค่าเริ่มต้นไว้ภายในคำสั่งควบคุมการวนซ้ำได้โดยตรง

คำสั่ง while เป็นโครงสร้างควบคุมแบบตรวจสอบเงื่อนไขก่อนทำงาน (Entry-Controlled Loop) กล่าวคือ โปรแกรมจะทำการประเมินค่าของเงื่อนไขก่อน หากเงื่อนไขเป็นจริงจึงจะเข้าสู่การทำงานในกลุ่มคำสั่งภายในลูป และวนกลับมาตรวจสอบเงื่อนไขอีกครั้งในรอบถัดไป อย่างไรก็ตาม หากเงื่อนไขเป็นเท็จตั้งแต่เริ่มต้นกลุ่มคำสั่งภายในลูปจะไม่ถูกดำเนินการ

ภายในกลุ่มประโยคคำสั่งของ while ควรมีคำสั่งที่ทำให้ค่าของตัวแปรควบคุมการวนซ้ำเปลี่ยนแปลง เช่น การเพิ่มค่าหรือลดค่าตัวแปร เพื่อให้เงื่อนไขของการวนซ้ำมีโอกาสเป็นเท็จในอนาคต มิฉะนั้นจะเกิดการวนซ้ำอย่างไม่สิ้นสุด (Infinite Loop) และไม่สามารถออกจากลูปได้ จนกว่าจะหยุดการทำงานของโปรแกรม โครงสร้างการวนซ้ำแบบ while เหมาะกับสถานการณ์ที่ค่าควบคุมการวนซ้ำอาจเปลี่ยนแปลงตามกระบวนการคำนวณหรือการรับค่าจากผู้ใช้ โดยมีรูปแบบคำสั่งและตัวอย่างดังนี้

รูปแบบ

```
while (นิพจน์ตรวจสอบเงื่อนไข) {
    คำสั่งที่ 1;
    ...;
}
```

ตัวอย่างที่ 27

```
1  let count = 1;
2  while (count <= 3) {
3          console.log("รอบที่ " + count);
4          count = count+1;
5  }
6  console.log("สิ้นสุดการนับ");
```

จากรูปแบบการใช้คำสั่งควบคุมการวนซ้ำ while จะเห็นว่า ไม่ได้มีข้อบังคับในการกำหนดค่าเริ่มต้นและการเปลี่ยนแปลงค่าก่อนการวนซ้ำ จากตัวอย่างที่ 27 บรรทัดที่ 1 เป็นการประกาศตัวแปร count และกำหนดค่าเริ่มต้นเป็น 1 ซึ่งทำก่อนเข้าสู่คำสั่ง while ในบรรทัดที่ 2 เป็นคำสั่ง while ที่ตรวจสอบเงื่อนไข count <= 3

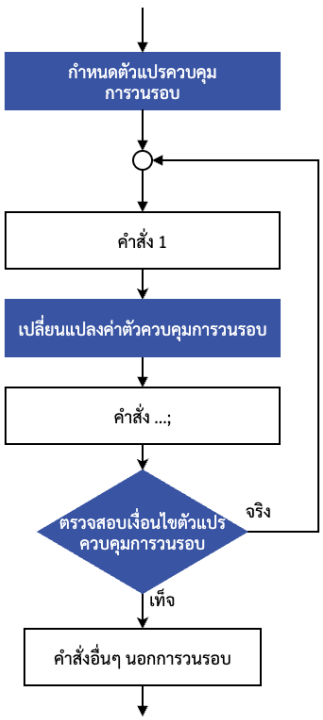
ในรอบแรก count มีค่าเป็น 1 ซึ่งตรงตามเงื่อนไข ทำให้มีการดำเนินการกลุ่มประโยคคำสั่งบรรทัดที่ 3-4 ผลของคำสั่งในบรรทัดที่ 3 เป็นการแสดงข้อความ *รอบที่ 1* ในคอนโซลของเว็บเบราว์เซอร์ จากนั้น

จึงเพิ่มค่า count ขึ้นอีก 1 ในบรรทัดที่ 4 (สามารถใช้ count++ แทน count=count+1 ได้) และจะวนกลับไปตรวจสอบเงื่อนไขใหม่ กระบวนการนี้จะดำเนินต่อไปจนกระทั่ง count มีค่าเป็น 4 ซึ่งไม่ตรงตามเงื่อนไข count <= 3 การวนซ้ำจึงสิ้นสุดและดำเนินการคำสั่งในบรรทัดที่ 6 ซึ่งเป็นการแสดงข้อความ *สิ้นสุดการนับ* ในคอนโซลของเว็บเบราว์เซอร์

4.2.3 โครงสร้างการวนซ้ำแบบ do-while

โครงสร้างการวนซ้ำแบบ do-while เป็นโครงสร้างควบคุมที่มีลักษณะเฉพาะ กล่าวคือ จะดำเนินการกลุ่มประโยคคำสั่งภายในลูปอย่างน้อยหนึ่งครั้งเสมอก่อนที่จะตรวจสอบเงื่อนไข แม้ว่าเงื่อนไขจะเป็นเท็จตั้งแต่เริ่มต้นก็ตาม ลักษณะนี้ทำให้โครงสร้าง do-while แตกต่างจากโครงสร้าง for และ while ที่ต้องประเมินเงื่อนไขก่อนการทำงาน (Entry-Controlled Loop) ในทางกลับกัน do-while เป็นโครงสร้างการวนซ้ำแบบตรวจสอบเงื่อนไขหลังการทำงาน (Exit-Controlled Loop) จึงเหมาะสำหรับสถานการณ์ที่ต้องการให้คำสั่งในลูปถูกประมวลผลอย่างน้อยหนึ่งครั้ง เช่น การรับข้อมูลจากผู้ใช้ก่อนตรวจสอบความถูกต้อง หรือการสร้างเมนูหลักที่แสดงผลอย่างน้อยหนึ่งครั้งก่อนรอการเลือกเมนู

โครงสร้างการวนซ้ำแบบ do-while มีรูปแบบไวยากรณ์ที่ง่ายต่อการเข้าใจ โดยใช้คำสำคัญ do ตามด้วยกลุ่มคำสั่งภายในบล็อก และตามด้วย while เพื่อตรวจสอบเงื่อนไขท้ายลูป ลักษณะของกระบวนการทำงานนี้แสดงดังภาพที่ 9.11



ภาพที่ 9.11 โครงสร้างการวนซ้ำแบบ do-while

จากภาพที่ 9.11 โครงสร้างการวนซ้ำแบบ do-while มีลำดับการทำงานโดยก่อนการเข้าสู่โครงสร้างการวนซ้ำอาจมีการกำหนดตัวแปรควบคุมการวนซ้ำก่อน ทั้งนี้การกำหนดดังกล่าวไม่ได้เป็นส่วนหนึ่งของคำสั่งควบคุมการวนซ้ำ do-while จากนั้นจะเข้าสู่กลุ่มประโยคคำสั่งของการวนซ้ำของ do-while อย่างน้อยหนึ่งรอบแล้วจึงทำการตรวจสอบเงื่อนไขการวนซ้ำ หากเงื่อนไขเป็นจริงก็จะย้อนกลับไปทำคำสั่งในกลุ่มประโยคคำสั่งอีกครั้ง แต่หากเงื่อนไขเป็นเท็จก็จะสิ้นสุดการวนซ้ำและทำคำสั่งอื่นที่อยู่ถัดจากคำสั่งควบคุมการวนซ้ำ

โครงสร้างนี้เหมาะสำหรับกรณีที่ต้องการให้คำสั่งอย่างน้อยหนึ่งรอบ เช่น การรับข้อมูลจากผู้ใช้แล้วตรวจสอบความถูกต้อง หรือการทำงานที่ต้องให้ผู้ใช้เลือกเมนูซ้ำจนกว่าจะเลือกออกจากโปรแกรม เป็นต้น รูปแบบและตัวอย่างของคำสั่งมีดังนี้

รูปแบบ

```
do {
    คำสั่งที่ 1;
    ...;
} while (นิพจน์ตรวจสอบเงื่อนไข);
```

ตัวอย่างที่ 28

```
1  let number = 10;
2  do {
3          console.log("ค่าปัจจุบันคือ " + number);
4          number++;
5  } while (number < 10);
6  console.log("จบการทำงาน");
```

จากตัวอย่างที่ 28 บรรทัดที่ 1 เป็นการประกาศตัวแปร number และกำหนดค่าเป็น 10 ในบรรทัดที่ 2 เป็นคำสั่ง do ที่จะดำเนินการกลุ่มคำสั่งในบรรทัดที่ 3-4 ก่อน ซึ่งจะแสดงข้อความ *ค่าปัจจุบันคือ 10* ในส่วนคอนโซลของเว็บเบราว์เซอร์และเพิ่มค่า number เป็น 11 จากนั้นในบรรทัดที่ 5 จึงตรวจสอบเงื่อนไข number < 10 ซึ่งได้ผลลัพธ์เป็น false เนื่องจาก 11 ไม่น้อยกว่า 10 ทำให้การวนซ้ำสิ้นสุดและดำเนินการคำสั่งในบรรทัดที่ 6 แม้ว่าเงื่อนไขจะเป็นเท็จ แต่คำสั่งภายในกลุ่มประโยคคำสั่ง do ได้ถูกดำเนินการไปหนึ่งครั้งแล้ว

5. สรุปท้ายบท

JavaScript ถือเป็นภาษาสคริปต์ที่มีบทบาทสำคัญอย่างยิ่งต่อการพัฒนาเว็บแอปพลิเคชันในยุคปัจจุบัน ด้วยคุณสมบัติในการควบคุมพฤติกรรมและการโต้ตอบของเว็บเพจที่มีความยืดหยุ่นสูง ทำให้สามารถสร้างประสบการณ์ผู้ใช้ (User Experience) ที่ตอบสนองได้อย่างรวดเร็วและมีความเป็นพลวัต ภาษานี้ทำงานควบคู่กับภาษา HTML

ซึ่งทำหน้าที่จัดโครงสร้างข้อมูลของเอกสาร และภาษา CSS ที่ทำหน้าที่จัดรูปแบบการแสดงผล เพื่อทำให้เว็บเพจมีความสมบูรณ์ทั้งด้านโครงสร้าง ความสวยงาม และการโต้ตอบในเชิงตรรกะ JavaScript ได้รับการพัฒนาภายใต้กรอบมาตรฐาน ECMAScript โดยองค์กร ECMA International ซึ่งเป็นมาตรฐานกลางที่กำหนดเพื่อให้ภาษานี้ทำงานได้อย่างสอดคล้องกันในเว็บเบราว์เซอร์หลากหลายชนิด ลดปัญหาความไม่เข้ากันระหว่างเบราว์เซอร์

ภาษา JavaScript สนับสนุนแนวคิดการเขียนโปรแกรมหลากหลายรูปแบบ ทั้งการเขียนโปรแกรมเชิงคำสั่ง (Imperative Programming) การเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming: OOP) และการเขียนโปรแกรมเชิงฟังก์ชัน (Functional Programming) อันเป็นการเพิ่มขีดความสามารถให้กับนักพัฒนาในการสร้างรหัสคำสั่งที่มีประสิทธิภาพและสามารถดูแลรักษาได้ง่าย ทั้งนี้ JavaScript ยังมีศักยภาพในการประยุกต์ใช้งานได้ทั้งฝั่งผู้ใช้ (Client-side) ผ่านเว็บเบราว์เซอร์ และฝั่งผู้ให้บริการ (Server-side) ผ่านสภาพแวดล้อมการทำงาน เช่น Node.js, Deno และ Apache CouchDB ซึ่งช่วยให้การพัฒนาแอปพลิเคชันแบบ Full Stack สามารถทำงานได้ด้วยภาษาเดียวกันตลอดกระบวนการพัฒนา

องค์ประกอบหลักของภาษา JavaScript ประกอบด้วยโครงสร้างพื้นฐานหลายรูปแบบที่ทำให้ภาษานี้มีความยืดหยุ่นและทรงพลัง ประโยคคำสั่ง (Statements) ทำหน้าที่ควบคุมลำดับการทำงานของโปรแกรม โดยสามารถกำหนดให้รหัสคำสั่งทำงานตามเงื่อนไขหรือทำงานแบบวนซ้ำได้ตามความจำเป็น นิพจน์ (Expressions) เป็นหน่วยย่อยของรหัสที่ให้ค่าผลลัพธ์และสามารถนำไปใช้ร่วมกับคำสั่งอื่นเพื่อสร้างกลไกการทำงานที่ซับซ้อนขึ้น ในขณะที่ค่าคงที่ (Literals) แสดงถึงข้อมูลที่กำหนดไว้โดยตรง เช่น ตัวเลข ข้อความ หรือค่าตรรกะ และเป็นส่วนสำคัญของการนิยามข้อมูล

การจัดการตัวแปร (Variables) มีความสมสำคัญในการจัดเก็บและจัดการข้อมูลในหน่วยความจำของโปรแกรม โดยตัวแปรที่ประกาศด้วย var, let และ const มีพฤติกรรมการทำงานที่แตกต่างกันทั้งในด้านขอบเขตการเข้าถึง (Scope) ความสามารถในการเปลี่ยนค่า (Mutability) และการจัดสรรหน่วยความจำ กลไกการยกตัวแปรขึ้นบนสุด (Hoisting) และช่วงเวลาที่ตัวแปรสามารถเข้าถึงได้ (Temporal Dead Zone: TDZ)

ชนิดของข้อมูล (Data Types) ในภาษา JavaScript ถูกจำแนกออกเป็นสองกลุ่มใหญ่ตามลักษณะการจัดการ ได้แก่ ข้อมูลประเภทพื้นฐาน (Primitive Data Types) และข้อมูลประเภทอ้างอิง (Reference Data Types) ข้อมูลประเภทพื้นฐาน ได้แก่ Number, String, Boolean, Undefined, Null, Bigint และ Symbol ซึ่งมีลักษณะเป็น Immutable กล่าวคือ ค่าของข้อมูลจะไม่สามารถถูกเปลี่ยนแปลงโดยตรงได้ และเมื่อมีการกำหนดค่าหรือส่งผ่านไปยังฟังก์ชันจะมีการจัดการแบบผ่านค่า (Pass-By-Value

ข้อมูลประเภทอ้างอิง เช่น object, array และ function มีลักษณะเป็น Mutable และจัดการแบบผ่านการอ้างอิง (Pass-By-Reference) ทำให้การเปลี่ยนแปลงข้อมูลผ่านตัวแปรหนึ่งจะมีผลกระทบต่อข้อมูลที่ตัวแปรอื่นอ้างอิงอยู่ด้วย การทำความเข้าใจความแตกต่างของการจัดการข้อมูลทั้งสองกลุ่มนี้มีความสำคัญต่อการพัฒนาโปรแกรมที่ปลอดภัย ปราศจากข้อผิดพลาด และสามารถบำรุงรักษาได้ง่ายในระยะยาว

โครงสร้างการควบคุม (Control Structures) ในภาษา JavaScript มีบทบาทสำคัญอย่างยิ่งในการกำหนดลำดับและทิศทางการทำงานของโปรแกรม โครงสร้างการตัดสินใจ เช่น if, if-else, if-else if และ switch ช่วยให้โปรแกรมสามารถประเมินเงื่อนไขและเลือกทำงานตามเส้นทางที่กำหนดไว้ได้อย่างมีระบบ

โครงสร้างการวนซ้ำ เช่น for, while และ do-while ช่วยให้สามารถทำงานซ้ำกับกลุ่มคำสั่งได้ตามจำนวนครั้งที่กำหนด หรือจนกว่าเงื่อนไขที่ระบุจะเป็นเท็จ โครงสร้างเหล่านี้ทำให้โปรแกรมสามารถจัดการกับข้อมูลจำนวนมาก ตอบสนองต่อการกระทำของผู้ใช้ และสนับสนุนการสร้างตรรกะที่ซับซ้อน เช่น การประมวลผลข้อมูลจากฟอร์ม การสร้างการเคลื่อนไหวขององค์ประกอบในหน้าเว็บ หรือการทำงานซ้ำสำหรับการสร้างตารางและกราฟิก

6. กิจกรรมท้ายบท

6.1 จงอธิบายความหมายของภาษา JavaScript และบทบาทของภาษา JavaScript ในการพัฒนาเว็บแอปพลิเคชัน พร้อมระบุว่ามาตรฐาน ECMAScript มีความสำคัญต่อการพัฒนาและใช้งาน JavaScript อย่างไร

6.2 ประโยคคำสั่งในภาษา JavaScript มีลักษณะอย่างไร? จงยกตัวอย่างประโยคคำสั่งที่ถูกต้องมา 2 ตัวอย่าง พร้อมอธิบายหน้าที่ของแต่ละคำสั่ง

6.3 จงอธิบายความหมายและยกตัวอย่างของนิพจน์ในภาษา JavaScript พร้อมทั้งระบุประเภทของนิพจน์อย่างน้อย 3 ประเภท

6.4 จงเปรียบเทียบความแตกต่างระหว่างการประกาศตัวแปรด้วย var, let, และ const ในด้านขอบเขตการใช้งาน (Scope) การกำหนดค่าใหม่ และการยกระดับ (Hoisting) พร้อมยกตัวอย่างการใช้งานที่เหมาะสมของแต่ละชนิด

6.5 ข้อมูลประเภทพื้นฐาน (Primitive Data Types) ของภาษา JavaScript มีอะไรบ้าง? จงอธิบายแต่ละประเภทโดยสังเขป พร้อมยกตัวอย่างค่าของแต่ละประเภท

6.6 จงเขียนรหัส JavaScript เพื่อประมวลผลต่อไปนี้

6.6.1 ประกาศตัวแปรเก็บค่าคะแนน (0–100) ด้วย let จากนั้นใช้โครงสร้างการตัดสินใจแบบ if-else เพื่อตรวจสอบว่าคะแนนอยู่ในระดับ "ผ่าน" (คะแนนมากกว่าหรือเท่ากับ 50) หรือ "ไม่ผ่าน (คะแนนน้อยกว่า 50)" แล้วแสดงผลด้วย console.log()

6.6.2 เขียนโครงสร้างการวนซ้ำแบบ for เพื่อแสดงตัวเลข 1 ถึง 5 ออกทาง console พร้อมอธิบายการทำงานของโครงสร้างการวนซ้ำที่ใช้



การจัดการข้อมูลและอิลิเมนต์ของหน้าเว็บด้วย JavaScript

บทที่ 10 มุ่งเน้นการนำเสนอหลักการใช้ภาษา JavaScript ในการจัดการข้อมูลและอิลิเมนต์ของหน้าเว็บผ่านกลไก DOM การตอบสนองต่อเหตุการณ์ของผู้ใช้ และการจัดการแบบฟอร์มเพื่อรับข้อมูลจากผู้ใช้ เพื่อเป็นพื้นฐานสำหรับการเรียนรู้เทคโนโลยีการพัฒนาเว็บส่วนหน้า หัวข้อหลักในบทนี้ประกอบด้วย

1. ฟังก์ชัน
2. ออบเจกต์
3. อาร์เรย์
4. การจัดการเอกสารเว็บด้วย JavaScript
5. สรุปท้ายบท
6. กิจกรรมท้ายบท

1. ฟังก์ชัน

ฟังก์ชัน (Function) เป็นหนึ่งในแนวคิดพื้นฐานที่สำคัญในการเขียนโปรแกรม ทั้งในภาษา JavaScript และภาษาโปรแกรมอื่นๆ โดยทำหน้าที่เป็นกลไกในการจัดกลุ่มชุดคำสั่งที่เกี่ยวข้องกัน เช่น การคำนวณ การแสดงผล หรือการประมวลผลข้อมูล ให้อยู่ภายใต้ชื่อเดียวกัน เพื่อให้สามารถเรียกใช้งานซ้ำได้โดยไม่ต้องเขียนรหัสคำสั่งซ้ำ

การใช้งานฟังก์ชันช่วยเพิ่มประสิทธิภาพในการพัฒนาเว็บแอปพลิเคชัน ทั้งในด้านการจัดระเบียบรหัสคำสั่ง การลดความซับซ้อนของโปรแกรม และการบำรุงรักษาในระยะยาว ฟังก์ชันในภาษา JavaScript มีหลายรูปแบบ ได้แก่ ฟังก์ชันแบบดั้งเดิม (Function Declaration) ฟังก์ชันแบบนิพจน์ (Function Expression) และฟังก์ชันแบบลูกศร (Arrow Function) ซึ่งแต่ละรูปแบบมีลักษณะการใช้งานที่เหมาะสมกับบริบทที่แตกต่างกัน

1.1 ความหมายและบทบาทของฟังก์ชันใน JavaScript

ฟังก์ชัน (Function) ในภาษา JavaScript คือกลุ่มคำสั่งที่ถูกกำหนดไว้ภายใต้ชื่อที่ระบุ เพื่อให้สามารถเรียกใช้งานซ้ำได้เมื่อต้องการ โดยไม่จำเป็นต้องเขียนรหัสคำสั่งเดิมซ้ำอีกครั้ง ฟังก์ชันสามารถรับค่าจากภายนอกผ่านทางพารามิเตอร์ และสามารถส่งค่าผลลัพธ์กลับออกไปยังจุดที่เรียกใช้ได้ จึงสามารถทำงานที่มีความซับซ้อนและต้องการนำกลับมาใช้ใหม่ได้อย่างมีประสิทธิภาพ

ในบริบทของการพัฒนาเว็บแอปพลิเคชัน ฟังก์ชันมีบทบาทสำคัญในการแยกส่วนการทำงานของโปรแกรมออกเป็นหน่วยย่อยที่มีความรับผิดชอบเฉพาะ เช่น ฟังก์ชันที่คำนวณผลรวมของข้อมูล ฟังก์ชันที่ตรวจสอบความถูกต้องของฟอร์ม หรือฟังก์ชันที่จัดการการตอบสนองของผู้ใช้เมื่อคลิกปุ่มใดปุ่มหนึ่ง ซึ่งช่วยให้รหัสคำสั่งมีความเป็นระบบ ง่ายต่อการตรวจสอบ แก้ไข และนำกลับมาใช้ใหม่ นอกจากนี้ ฟังก์ชันยังเป็นพื้นฐานของการจัดการเหตุการณ์ (Event Handling) ในหน้าเว็บซึ่งเป็นกลไกหลักในการโต้ตอบกับผู้ใช้ผ่าน DOM

1.2 รูปแบบของฟังก์ชันใน JavaScript

ฟังก์ชันในภาษา JavaScript สามารถประกาศได้หลายรูปแบบ โดยแต่ละรูปแบบมีลักษณะการใช้งานและพฤติกรรมเฉพาะ ซึ่งเหมาะสมกับสถานการณ์ที่แตกต่างกัน โดยทั่วไปสามารถจำแนกรูปแบบของฟังก์ชันได้เป็น 3 ลักษณะหลัก ได้แก่ ฟังก์ชันแบบประกาศ ฟังก์ชันแบบนิพจน์ และฟังก์ชันแบบลูกศร

1.2.1 ฟังก์ชันแบบประกาศ

ฟังก์ชันแบบประกาศ (Function Declaration) เป็นรูปแบบการสร้างฟังก์ชันที่ใช้คำสำคัญ function นำหน้าตามด้วยชื่อฟังก์ชัน วงเล็บกลมสำหรับรับพารามิเตอร์ และวงเล็บปีกกาเพื่อครอบกลุ่มประโยคคำสั่งที่ต้องการให้ทำงานเมื่อมีการเรียกใช้ฟังก์ชัน พารามิเตอร์ที่อยู่ในวงเล็บจะทำหน้าที่เป็นตัวแปรรองรับค่าจากภายนอกที่ส่งเข้ามาในขณะเรียกใช้งาน ฟังก์ชันรูปแบบนี้มีคุณสมบัติที่สามารถถูกเรียกใช้งานได้ก่อนตำแหน่งที่ประกาศ เนื่องจากถูกจัดการโดยกลไก Hoisting ของ JavaScript รูปแบบการสร้างฟังก์ชันมีดังนี้

รูปแบบ

```
function ชื่อฟังก์ชัน(พารามิเตอร์1, พารามิเตอร์2, ...) {
    คำสั่ง;
}
```

ตัวอย่างที่ 1

```
1  function greet(name) {
2      console.log("สวัสดี " + name);
3  }
4  greet("Nok");
```

จากตัวอย่างที่ 1 บรรทัดที่ 1–3 เป็นส่วนของการสร้างฟังก์ชันแบบประกาศ โดยกำหนดชื่อฟังก์ชันว่า greet และกำหนดพารามิเตอร์ชื่อ name เพื่อใช้รับข้อมูลจากภายนอกในขณะมีการเรียกใช้ฟังก์ชัน พารามิเตอร์นี้จะทำหน้าที่รองรับค่าที่ส่งเข้ามาเพื่อประมวลผลภายในฟังก์ชัน

บรรทัดที่ 2 เป็นประโยคคำสั่งภายในฟังก์ชัน ซึ่งทำหน้าที่แสดงข้อความออกทางคอนโซล โดยนำคำว่า “สวัสดี” มาต่อกับค่าของพารามิเตอร์ name ด้วยเครื่องหมาย + ซึ่งเป็นตัวดำเนินการสำหรับการต่อข้อความ

บรรทัดที่ 4 เป็นการเรียกใช้ฟังก์ชัน greet() โดยส่งอาร์กิวเมนต์ (argument) ซึ่งเป็นข้อความ "Nok" เข้าไป ฟังก์ชันจะนำค่าดังกล่าวไปกำหนดให้กับพารามิเตอร์ name และดำเนินการตามคำสั่งในบรรทัดที่ 2 ส่งผลให้ข้อความ “สวัสดี Nok” ปรากฏในคอนโซลของเว็บเบราว์เซอร์

1.2.2 ฟังก์ชันแบบนิพจน์

ฟังก์ชันแบบนิพจน์ (Function Expression) เป็นรูปแบบของการสร้างฟังก์ชันที่นำฟังก์ชันมาประกาศในลักษณะของนิพจน์แล้วกำหนดให้กับตัวแปร ซึ่งหมายความว่า ฟังก์ชันจะไม่มีชื่อแยกต่างหาก และจะเริ่มทำงานได้หลังจากที่ตัวแปรได้รับนิยามของฟังก์ชันแล้วเท่านั้น ฟังก์ชันแบบนิพจน์จะไม่มีคุณสมบัติ Hoisting จึงไม่สามารถเรียกใช้งานก่อนตำแหน่งที่มีการประกาศได้ ฟังก์ชันรูปแบบนี้เหมาะกับสถานการณ์ที่ต้องการกำหนดฟังก์ชันเป็นค่าให้กับตัวแปรหรือใช้งานฟังก์ชันตามเงื่อนไขที่เปลี่ยนไปได้ โดยรูปแบบการประกาศมีลักษณะดังตัวอย่างต่อไปนี้

รูปแบบ

```
const  ชื่อตัวแปร = function (พารามิเตอร์1, พารามิเตอร์2, ...) {
      คำสั่ง;
}
```

ตัวอย่างที่ 2

```
1  const calculateArea = function(width, height) {
2        let area = width * height;
3        return area;
4  };
5  console.log(calculateArea(10, 5));
```

จากตัวอย่างที่ 2 บรรทัดที่ 1–4 เป็นการสร้างฟังก์ชันแบบนิพจน์โดยกำหนดฟังก์ชันที่ไม่มีชื่อ (Anonymous Function) ให้กับตัวแปร calculateArea ฟังก์ชันนี้รับพารามิเตอร์สองตัว คือ width และ height เพื่อนำไปคำนวณพื้นที่

บรรทัดที่ 2 เป็นคำสั่งภายในฟังก์ชันที่ทำการประกาศตัวแปรชื่อ area พร้อมกับกำหนดค่าให้กับตัวแปร area ด้วยการประเมินนิพจน์ width x height

บรรทัดที่ 3 เป็นการส่งค่าของตัวแปร area กลับไปยังตำแหน่งที่เรียกใช้ฟังก์ชันด้วยคำสั่ง return

บรรทัดที่ 5 เป็นการเรียกใช้ฟังก์ชัน calculateArea โดยส่งอาร์กิวเมนต์ 10 และ 5 เข้าไป ซึ่งจะได้ผลลัพธ์เป็น 50 ซึ่งหลังจากที่เรียกใช้ฟังก์ชันและได้รับค่ากลับมาจะทำให้บรรทัดที่ 5 ได้รับการประเมินนิพจน์เป็น console.log(50) จากนั้นจึงดำเนินการประมวลผลซึ่งเป็นการแสดงค่า 50 ในส่วนคอนโซลของเว็บเบราว์เซอร์

1.2.3 ฟังก์ชันแบบลูกศร

ฟังก์ชันแบบลูกศร (Arrow Function) เป็นรูปแบบการเขียนฟังก์ชันที่ถูกเพิ่มเข้ามาใน ECMAScript 2015 (ES6) โดยมีไวยากรณ์ที่กระชับและสั้นกว่าฟังก์ชันแบบดั้งเดิม ลักษณะเด่นของฟังก์ชันชนิดนี้คือการใช้เครื่องหมาย => (เรียกว่า Fat Arrow) เพื่อระบุการกำหนดนิพจน์ของฟังก์ชัน แทนการใช้คำสำคัญ function เหมือนในรูปแบบเดิม

ฟังก์ชันแบบลูกศรเหมาะสำหรับการเขียนฟังก์ชันที่สั้น กระชับ และไม่มีความซับซ้อนมากนัก โดยเฉพาะอย่างยิ่งในบริบทของการส่งฟังก์ชันเป็นอาร์กิวเมนต์ (Callback Function)

รูปแบบที่ 1

```
const ชื่อตัวแปร = (พารามิเตอร์1, พารามิเตอร์2) => {
    คำสั่ง;
};
```

ตัวอย่างที่ 3

```
1  const calculateArea = (width, height) => {
2      let area = width * height;
3      return area;
4  };
5  console.log(calculateArea(10, 5));
```

จากตัวอย่างที่ 3 บรรทัดที่ 1–4 เป็นการสร้างฟังก์ชันแบบลูกศรโดยใช้พารามิเตอร์สองตัว ได้แก่ width และ height แล้วกำหนดฟังก์ชันนี้ให้กับตัวแปร calculateArea เพื่อคำนวณพื้นที่ของรูปสี่เหลี่ยม

บรรทัดที่ 2 เป็นคำสั่งภายในฟังก์ชันที่ประกาศตัวแปรชื่อ area พร้อมกับกำหนดค่าให้กับ area โดยการคูณค่าของ width กับ height

บรรทัดที่ 3 เป็นการส่งค่าของตัวแปร area กลับไปยังตำแหน่งที่เรียกใช้ฟังก์ชันด้วยคำสั่ง return

บรรทัดที่ 5 เป็นการเรียกใช้ฟังก์ชัน calculateArea โดยส่งอาร์กิวเมนต์ 10 และ 5 เข้าไป ซึ่งจะได้ผลลัพธ์เป็น 50 และแสดงผลทางคอนโซลของเว็บเบราว์เซอร์ด้วยคำสั่ง console.log เป็นค่า 50

ในตัวอย่างที่ 3 จะพบว่ากลุ่มประโยคคำสั่งของฟังก์ชันมีเพียงสองคำสั่ง ซึ่งคำสั่งแรกเป็นการประมวลผล และคำสั่งที่ส่งเป็นการส่งค่ากลับ ดังนั้นสามารถเขียนให้สั้นลงโดยโดยละคำสั่ง return เพื่อการส่งค่ากลับโดยใช้รูปแบบที่ 2 ดังนี้

รูปแบบ 2

const ชื่อตัวแปร = (พารามิเตอร์) => ค่าที่ต้องการส่งกลับ;

ตัวอย่างที่ 4

```
1  const calculateArea = (width, height) => width * height;
```

จากตัวอย่างที่ 4 ฟังก์ชันแบบลูกศรสามารถเขียนให้อยู่ในรูปแบบที่กระชับยิ่งขึ้น หากมีเพียงประโยคคำสั่งเดียว และผลของประโยคนั้นใช้ส่งค่ากลับ (return statement) โดยตรง ในกรณีนี้สามารถละทั้งคำว่า return และเครื่องหมายวงเล็บปีกกา {} ได้ ซึ่งเป็นคุณสมบัติที่ใช้กับฟังก์ชันแบบลูกศร เท่านั้น

1.3 พารามิเตอร์และค่าที่ส่งกลับจากฟังก์ชัน

พารามิเตอร์และค่าที่ส่งกลับเป็นกลไกสำคัญที่ทำให้ฟังก์ชันสามารถรับข้อมูลจากภายนอก ประมวลผล และส่งผลลัพธ์กลับออกมาได้ การเข้าใจการทำงานของพารามิเตอร์และการคืนค่าจะช่วยให้สามารถสร้างฟังก์ชันที่มีประสิทธิภาพและนำกลับมาใช้ได้ในหลากหลายสถานการณ์

1.3.1 การรับพารามิเตอร์

ในการใช้งานฟังก์ชันของภาษา JavaScript พารามิเตอร์ (Parameter) คือชื่อตัวแปรที่กำหนดไว้ในวงเล็บขณะประกาศฟังก์ชัน เพื่อใช้รองรับค่าที่จะถูกส่งเข้ามาเมื่อต้องการเรียกใช้งานฟังก์ชันนั้น ค่าที่ส่งเข้ามาเรียกว่าอาร์กิวเมนต์ (Argument) ซึ่งจะถูกจับคู่กับพารามิเตอร์ตามลำดับตำแหน่งโดยไม่ต้องระบุชนิดข้อมูลล่วงหน้า เนื่องจาก JavaScript เป็นภาษาที่ไม่มีการกำหนดชนิดข้อมูลอย่างตายตัว

ตัวอย่างที่ 5

```
1  function sayHello(name) {
2          console.log("สวัสดี " + name);
3  }
4  sayHello("Nok");
```

จากตัวอย่างที่ 5 บรรทัดที่ 1 เป็นการประกาศฟังก์ชันชื่อ sayHello ซึ่งมีพารามิเตอร์ชื่อ name สำหรับรับค่าจากภายนอก บรรทัดที่ 2 เป็นคำสั่งแสดงข้อความ โดยนำคำว่า “สวัสดี” มาต่อกับค่าที่รับผ่านพารามิเตอร์ name และบรรทัดที่ 4 เป็นการเรียกใช้ฟังก์ชัน sayHello("Nok") โดยส่งข้อความ "Nok" เป็น

อาร์กิวเมนต์ให้กับพารามิเตอร์ name ทำให้ฟังก์ชันแสดงข้อความว่า สวัสดี Nok บนคอนโซล ในกรณีที่ไม่มีการส่งอาร์กิวเมนต์ พารามิเตอร์จะมีค่าเป็น undefined โดยอัตโนมัติ ตัวอย่างเช่น sayHello(); ผลลัพธ์ที่แสดงในคอนโซลคือ สวัสดี undefined ซึ่งอาจไม่ตรงตามความต้องการ จึงควรมีการกำหนดค่าพารามิเตอร์เริ่มต้นเพื่อรองรับกรณีที่ไม่ได้ส่งค่าเข้ามาดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 6

```
1  function sayHello(name = "ผู้ใช้") {
2      console.log("สวัสดี " + name);
3  }
```

อนึ่ง พารามิเตอร์สามารถมีได้หลายตัวตามรูปแบบของการสร้างฟังก์ชันที่ได้กล่าวถึงในหัวข้อก่อนหน้า โดยคั่นด้วยเครื่องหมายจุลภาค เช่น function add(a, b, c) { ... } ซึ่งจะทำให้ฟังก์ชันสามารถรับค่าได้หลายตัวพร้อมกันตามลำดับที่กำหนด

1.3.2 การส่งค่ากลับจากฟังก์ชัน

การส่งค่ากลับจากฟังก์ชันในภาษา JavaScript เป็นกลไกที่ใช้คืนค่าผลลัพธ์จากการประมวลผลภายในฟังก์ชันกลับไปยังตำแหน่งที่เรียกใช้งาน โดยใช้คำสั่ง return ซึ่งทำหน้าที่สิ้นสุดการทำงานของฟังก์ชันและส่งค่าตัวแปรหรือนิพจน์ที่ระบุไว้ถัดจากคำสั่ง return กลับออกไปยังบริบทภายนอก รูปแบบทั่วไปของการส่งค่ากลับมีดังนี้

รูปแบบ

```
function ชื่อฟังก์ชัน(พารามิเตอร์) {
    คำสั่ง;
    return ค่าที่ต้องการส่งกลับ;
}
```

ตัวอย่างที่ 7

```
1  function multiply(x, y) {
2      return x * y;
3  }
4  let result = multiply(4, 5);
5  console.log(result);
```

จากตัวอย่างที่ 7 บรรทัดที่ 1–3 เป็นการประกาศฟังก์ชันชื่อ multiply ที่รับพารามิเตอร์ x และ y แล้วส่งค่าผลคูณของทั้งสองตัวกลับ ในบรรทัดที่ 5 มีการเรียกใช้ฟังก์ชันและรับค่าผลลัพธ์จาก multiply(4, 5) ซึ่งจะได้ค่า 20 และในบรรทัดที่ 6 เป็นการแสดงค่าที่ได้บนคอนโซล

ถ้าฟังก์ชันไม่มีคำสั่ง return หรือไม่มีค่าถัดจาก return ฟังก์ชันจะส่งค่ากลับเป็น undefined โดยอัตโนมัติ ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 8

```
1  function noReturnValue() {
2    console.log("ไม่มีการคืนค่า");
3  }
4  let value = noReturnValue();
5  console.log(value); // แสดงค่า undefined
```

จากตัวอย่างที่ 8 บรรทัดที่ 1–3: เป็นการประกาศฟังก์ชันชื่อ noReturnValue ซึ่งไม่มีพารามิเตอร์ และมีเพียงคำสั่ง console.log() เพื่อแสดงข้อความ “ไม่มีการคืนค่า” ไปยังคอนโซล สังเกตว่า ไม่มีคำสั่ง return อยู่ในฟังก์ชันนี้เลย ดังนั้น เมื่อฟังก์ชันทำงานเสร็จ ระบบจะคืนค่าเป็น undefined โดยอัตโนมัติ

บรรทัดที่ 4: เรียกใช้ฟังก์ชัน noReturnValue() ซึ่งจะทำให้ข้อความ *ไม่มีการคืนค่า* ปรากฏในคอนโซลทันที ค่าที่ได้จากการเรียกฟังก์ชันจะถูกนำไปเก็บในตัวแปร value ซึ่งในกรณีนี้คือ undefined เนื่องจากไม่มีการ return ค่าใดออกจากฟังก์ชัน และในบรรทัดที่ 5: เป็นการแสดงค่าของตัวแปร value ซึ่งผลลัพธ์จะเป็น undefined

1.4 ฟังก์ชันแบบไม่ระบุชื่อและฟังก์ชันแบบ Callback

ฟังก์ชันแบบไม่ระบุชื่อ (Anonymous Function) คือฟังก์ชันที่ไม่มีการกำหนดชื่อให้กับฟังก์ชันขณะประกาศ โดยมักใช้งานร่วมกับการกำหนดค่าให้กับตัวแปร หรือส่งเป็นอาร์กิวเมนต์ให้กับฟังก์ชันอื่น จุดเด่นของฟังก์ชันชนิดนี้คือสามารถสร้างและใช้งานแบบชั่วคราวได้โดยไม่ต้องประกาศแยกชื่อให้ชัดเจนเหมือนฟังก์ชันแบบดั้งเดิม

หนึ่งในกรณีการใช้งานฟังก์ชันแบบไม่ระบุชื่อที่พบได้บ่อยคือการส่งฟังก์ชันเข้าไปยังอีกฟังก์ชันหนึ่งในรูปแบบของพารามิเตอร์ ซึ่งเรียกว่า ฟังก์ชันเรียกกลับ หรือ Callback Function โดยมักใช้ในการดำเนินการตามลำดับ เช่น การทำงานเมื่อมีเหตุการณ์เกิดขึ้น การประมวลผลชุดข้อมูล หรือการรอผลลัพธ์จากฟังก์ชันแบบอะซิงโครนัส (Asynchronous)

ตัวอย่างที่ 9

```
1  function processUserInput(callback) {
2        const name = "Nok";
```

```
3       callback(name);
4   }
5   processUserInput(function(name) {
6       console.log("ยินดีต้อนรับ " + name);
7   });
```

จากตัวอย่างที่ 9 บรรทัดที่ 1–3 เป็นการประกาศฟังก์ชันชื่อ processUserInput ซึ่งรับพารามิเตอร์ callback ซึ่งคาดว่าจะเป็นฟังก์ชัน

ภายในฟังก์ชัน processUserInput มีการประกาศตัวแปร name และเรียกใช้ฟังก์ชัน callback โดยส่งค่า name เข้าไป

บรรทัดที่ 5–7 เป็นการเรียกใช้ processUserInput โดยส่งฟังก์ชันแบบไม่ระบุชื่อเป็นพารามิเตอร์ ซึ่งรับพารามิเตอร์ name และแสดงข้อความต้อนรับออกทางคอนโซลผลลัพธ์ที่ได้คือแสดงข้อความ *ยินดีต้อนรับ Nok*

การใช้ฟังก์ชันแบบ Callback ทำให้สามารถควบคุมลำดับการทำงานได้อย่างยืดหยุ่น และสามารถแยกความรับผิดชอบของรหัสคำสั่งโดยเฉพาะในกรณีที่มีหลายขั้นตอน เช่น การโหลดข้อมูล การตรวจสอบผลลัพธ์ หรือการทำงานแบบหน่วงเวลา เป็นต้น

2. ออบเจกต์

ออบเจกต์เป็นโครงสร้างข้อมูลพื้นฐานที่สำคัญที่สุดในภาษา JavaScript ซึ่งช่วยให้นักพัฒนาสามารถจัดเก็บและจัดระเบียบข้อมูลที่ซับซ้อนได้อย่างมีประสิทธิภาพ ในการพัฒนาเว็บแอปพลิเคชัน ออบเจกต์ทำหน้าที่เป็นกลไกหลักในการจัดการข้อมูลที่ได้รับจากผู้ใช้ การสื่อสารกับเซิร์ฟเวอร์ และการแสดงผลบนหน้าเว็บ การเข้าใจแนวคิดของออบเจกต์จึงเป็นพื้นฐานสำคัญที่จะทำให้สามารถพัฒนาเว็บแอปพลิเคชันที่มีโครงสร้างเป็นระเบียบและบำรุงรักษาได้ง่าย

2.1 ความหมายของและบทบาทของออบเจกต์

ออบเจกต์ (Object) ในภาษา JavaScript หมายถึง โครงสร้างข้อมูลที่ประกอบด้วยคู่ของคีย์-ค่า (Key-Value Pairs) ซึ่งเรียกว่าคุณสมบัติ (Property) โดยแต่ละคุณสมบัติสามารถเป็นข้อมูลประเภทใดก็ได้ในภาษา JavaScript ไม่ว่าจะเป็นข้อมูลชนิดพื้นฐานอย่างเช่น ตัวเลข ข้อความ ฟังก์ชัน หรือแม้กระทั่งอาร์เรย์และออบเจกต์อื่น ออบเจกต์จึงเป็นโครงสร้างที่ยืดหยุ่นและมีประสิทธิภาพในการจัดการข้อมูลที่มีความซับซ้อน

ออบเจกต์จัดอยู่ในกลุ่ม ข้อมูลแบบอ้างอิง (Reference Type) ซึ่งแตกต่างจากข้อมูลพื้นฐานที่จัดเป็นข้อมูลแบบค่าคงที่ (Primitive Type) โดยมีลักษณะสำคัญคือออบเจกต์สามารถเปลี่ยนแปลงค่าได้ (Mutable) และ

ค่าจะถูกจัดเก็บในหน่วยความจำโดยใช้การอ้างอิง ไม่ใช่ค่าจริงโดยตรง ดังนั้นเมื่อมีการส่งออบเจกต์ไปยังฟังก์ชันหรือกำหนดให้กับตัวแปรอื่น ค่าที่ถูกอ้างถึงจะยังคงชี้ไปยังข้อมูลต้นฉบับเดิม (Web.Dev, 2024)

ในการพัฒนาเว็บแอปพลิเคชัน ออบเจกต์มีบทบาทสำคัญในการจัดระเบียบข้อมูลที่มีความสัมพันธ์กันให้รวมอยู่ในโครงสร้างเดียว เช่น ข้อมูลของผู้ใช้ สินค้า หรือผลลัพธ์จากแบบฟอร์ม โดยสามารถเข้าถึงคุณสมบัติแต่ละรายการได้อย่างชัดเจน ทำให้รหัสคำสั่งอ่านง่ายและดูแลรักษาได้สะดวก ดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 10

```
1  const user = {
2      name: "Nok",
3      age: 30,
4      email: "nok@example.com"
5  };
```

จากตัวอย่างที่ 10 ออบเจกต์ชื่อ user ประกอบด้วย 3 คุณสมบัติ ได้แก่ name, age และ email ซึ่งค่าของแต่ละคุณสมบัติมีชนิดที่แตกต่างกัน และสามารถเรียกใช้งานได้โดยใช้ชื่อคีย์ร่วมกับเครื่องหมายจุด เช่น user.name เป็นต้น

2.2 การสร้างออบเจกต์

การสร้างออบเจกต์ (Object Creation) ในภาษา JavaScript สามารถทำได้หลายวิธี โดยแต่ละวิธีมีข้อดีที่เหมาะสมกับสถานการณ์การใช้งานที่แตกต่างกัน ในหัวข้อนี้จะกล่าวถึง 3 วิธีที่พบบ่อย ได้แก่ การสร้างออบเจกต์แบบลิตเทอรัล การสร้างออบเจกต์ด้วยฟังก์ชันคอนสตรัคเตอร์ และการสร้างออบเจกต์ด้วยคลาส

2.2.1 การสร้างออบเจกต์แบบลิตเทอรัล

การสร้างออบเจกต์แบบลิตเทอรัล (Object Literal Notation) เป็นวิธีที่ง่ายและพบได้บ่อยในการสร้างออบเจกต์ เป็นการกำหนดคุณสมบัติของออบเจกต์ภายในเครื่องหมายปีกกา { และ } เพื่อครอบคู่ของคีย์และค่าซึ่งเป็นคุณสมบัติของออบเจกต์ โดยแยกแต่ละคุณสมบัติด้วยเครื่องหมายจุลภาค โดยมีรูปแบบและตัวอย่างดังต่อไปนี้

รูปแบบ

```
const  ชื่อตัวแปร = {
    ชื่อคุณสมบัติ_1: ค่าหรือฟังก์ชัน,
    ...
    ชื่อคุณสมบัติ_n: ค่าหรือฟังก์ชัน,
};
```

ตัวอย่างที่ 11

```
1  const person = {
2      name: "Nok",
3      age: 30,
4      greet: function() {
5          console.log("สวัสดี ฉันชื่อ " + this.name);
6      }
7  };
```

จากตัวอย่างที่ 11 แสดงการสร้างออบเจกต์ชื่อ person เพื่อจัดเก็บข้อมูลของบุคคลที่ประกอบด้วยชื่อ อายุ และมีพฤติกรรมการทักทายด้วย greet โดยภายในมีคุณสมบัติ 3 รายการ ได้แก่

- name เป็นคุณสมบัติที่เก็บค่าข้อมูลชนิดข้อความมีค่าเป็น Nok
- age เป็นคุณสมบัติที่เก็บค่าข้อมูลชนิดตัวเลขมีค่าเป็น 30
- greet เป็นคุณสมบัติที่มีค่าเป็นประเภทฟังก์ชัน ซึ่งในเชิงแนวคิดการพัฒนาโปรแกรมเรียกว่าเมธอด (Method) เนื่องจากเป็นพฤติกรรมที่ทำให้ออบเจกต์สามารถดำเนินการบางอย่างได้เมื่อมีการเรียกใช้งาน สำหรับคุณสมบัติ greet นี้ เป็นการแสดงข้อความผ่าน console.log() โดยนำค่าจากคุณสมบัติ name มาต่อข้อความด้วยการใช้ this.name เพื่อบ่งชี้ว่าให้ใช้ค่าของคุณสมบัติ name ที่เป็นของออบเจกต์เดียวกันกับที่ฟังก์ชันนี้เป็นส่วนหนึ่งอยู่ ซึ่งเป็นรูปแบบของการเข้าถึงคุณสมบัติในออบเจกต์ของตัวเอง

แม้ตามมาตรฐานของภาษา JavaScript คุณสมบัติทุกตัวของออบเจกต์ล้วนเรียกว่า property ไม่ว่าค่าจะเป็นข้อมูลพื้นฐานหรือฟังก์ชัน แต่ในทางปฏิบัติจะใช้คำว่าเมธอดสำหรับคุณสมบัติที่เป็นฟังก์ชัน เพื่อสะท้อนความหมายว่าฟังก์ชันดังกล่าวเป็นพฤติกรรมของออบเจกต์นั้นโดยเฉพาะ จากตัวอย่างที่ 11 หากมีการเข้าถึงตัวแปร person ด้วยประโยคคำสั่ง person.greet() จะเป็นการเรียกใช้ฟังก์ชันที่เป็นค่าของคุณสมบัติ greet ซึ่งจะแสดงข้อความ *สวัสดี ฉันชื่อ Nok* ในคอนโซลของเว็บเบราว์เซอร์

นอกจากนี้ การประกาศออบเจกต์ด้วยคำสำคัญ const เป็นแนวปฏิบัติมาตรฐานที่แนะนำให้ใช้ เนื่องจาก const ทำให้ตัวแปรมีลักษณะการอ้างอิงแบบคงที่ (Immutable Binding) กล่าวคือ ไม่สามารถเปลี่ยนให้ person ชี้ไปยังออบเจกต์อื่นได้อีก เช่น หากมีการสร้างออบเจกต์ใหม่ชื่อ staff = { ... } จะไม่สามารถกำหนด person = staff; ได้ ซึ่งจะทำให้เกิดข้อผิดพลาด TypeError: Assignment to constant variable

อย่างไรก็ตาม คุณสมบัติภายในออบเจกต์ person ยังสามารถเปลี่ยนแปลงได้ เช่น person.name = "Ann" หรือเพิ่มคุณสมบัติใหม่เข้าไป เช่น person.email = "ann@example.com" ได้ตามปกติ เนื่องจากออบเจกต์ในภาษา JavaScript เป็นชนิดข้อมูลแบบอ้างอิงที่สามารถเปลี่ยนแปลงค่าภายในได้ (Mutable)

2.2.2 การสร้างออบเจกต์ด้วย new Object()

การสร้างออบเจกต์ด้วยคำสั่ง new Object() เป็นอีกหนึ่งแนวทางพื้นฐานในภาษา JavaScript ที่ใช้ในการสร้างออบเจกต์เปล่าที่ยังไม่มีคุณสมบัติใด ๆ จากนั้นจึงเพิ่มคุณสมบัติเข้าไปภายหลัง ซึ่งแม้ว่าในปัจจุบันจะไม่เป็นที่นิยมเท่าการสร้างแบบลิตเทอรัล แต่ก็ยังมีความสำคัญในการทำงานกับระบบที่ต้องการการสร้างออบเจกต์แบบพลวัตหรือใช้งานร่วมกับคำสั่งที่ต้องอ้างอิงการสร้างออบเจกต์ด้วยคำสำคัญ (Keyword) new โดยมีรูปแบบและตัวอย่างดังต่อไปนี้

รูปแบบ

```
const ชื่อตัวแปร = new Object();
ชื่อตัวแปร.ชื่อคุณสมบัติ_1 = ค่าหรือฟังก์ชัน;
...
ชื่อตัวแปร.ชื่อคุณสมบัติ_n = ค่า;
```

จากรูปแบบของการสร้างออบเจกต์ด้วยวิธีนี้ จะประกอบด้วยสองส่วนคือส่วนสร้างออบเจกต์เปล่าด้วยประโยคคำสั่ง const ชื่อตัวแปร = new Object(); จากนั้นจะเป็นการกำหนดคุณสมบัติและค่าให้กับออบเจกต์ด้วยประโยคคำสั่ง ชื่อตัวแปร.ชื่อคุณสมบัติ_1 = ค่า; ซึ่งค่าอาจเป็นชนิดข้อมูลพื้นฐานหรือเป็นฟังก์ชันเพื่อใช้เมธอดของออบเจกต์ดังที่ได้กล่าวถึงในหัวข้อก่อนหน้า

ตัวอย่างที่ 12

```
1  const person = new Object();
2  person.name = "Nok";
3  person.age = 30;
4  person.greet = function () {
5          console.log("สวัสดี ฉันชื่อ " + this.name);
6  };
```

จากตัวอย่างที่ 12 เป็นการสร้างและกำหนดคุณสมบัติของออบเจกต์ให้มีลักษณะเช่นเดียวกับตัวอย่างก่อนหน้า โดยในบรรทัดที่ 1 เป็นการสร้างออบเจกต์เปล่าด้วยนิพจน์ new Object() และกำหนดให้ตัวแปร person อ้างอิงไปยังออบเจกต์นั้น ในบรรทัดที่ 2–3 เป็นการเพิ่มคุณสมบัติ name และ age ให้กับออบเจกต์ โดยกำหนดค่าให้กับคุณสมบัติ name เป็นข้อมูลชนิดสตริง และคุณสมบัติ age เป็นข้อมูลชนิดตัวเลขตามลำดับ ในบรรทัดที่ 4–6 เป็นการเพิ่มคุณสมบัติ greet โดยกำหนดให้เป็นฟังก์ชันที่ทำหน้าที่แสดงข้อความต้อนรับผ่าน console.log() โดยมีการใช้ this.name เพื่อเข้าถึงค่าของคุณสมบัติ name ภายในออบเจกต์เดียวกัน

2.2.3 การสร้างออบเจกต์ด้วย Constructor Function

การสร้างออบเจกต์ด้วย Constructor Function เป็นแนวทางการสร้างออบเจกต์แบบแม่แบบที่ช่วยให้สามารถสร้างออบเจกต์ที่มีโครงสร้างคุณสมบัติที่เหมือนกันได้หลายออบเจกต์โดยไม่ต้องเขียนคำสั่งซ้ำหลายครั้ง เหมาะสำหรับกรณีที่ต้องสร้างออบเจกต์ที่มีโครงสร้างคล้ายกัน เช่น ข้อมูลของพนักงาน ลูกค้า หรือสินค้า เป็นต้น

Constructor Function ในภาษา JavaScript คือฟังก์ชันทั่วไปที่ใช้กำหนดโครงสร้างและค่าเริ่มต้นของออบเจกต์ มีรูปแบบการสร้างออบเจกต์ดังนี้

รูปแบบ

```
function ชื่อฟังก์ชัน(พารามิเตอร์_1, ..., พารามิเตอร์_n) {
    this.ชื่อคุณสมบัติ_1= พารามิเตอร์_1;
    ...
    this.ชื่อคุณสมบัติ_n = พารามิเตอร์_n;
}
const ชื่อตัวแปร = new ชื่อฟังก์ชัน(อาร์กิวเมนต์_1, ..., อาร์กิวเมนต์_n);
```

จากรูปแบบการสร้างออบเจกต์ข้างต้น มีลำดับการสร้างออบเจกต์ดังนี้

1) สร้างฟังก์ชันแม่แบบ (Constructor Function) โดยตามแนวปฏิบัติที่ดีของการตั้งชื่อฟังก์ชันแม่แบบ ชื่อฟังก์ชันควรขึ้นต้นด้วยตัวพิมพ์ใหญ่ เพื่อแยกแยะว่าเป็นฟังก์ชันสำหรับใช้สร้างออบเจกต์ พร้อมระบุพารามิเตอร์เพื่อนำไปใช้กำหนดค่าให้กับคุณสมบัติของออบเจกต์

2) ภายในฟังก์ชันแม่แบบจะใช้คำสำคัญ this เพื่อกำหนดคุณสมบัติให้กับออบเจกต์ใหม่ที่จะถูกสร้างขึ้น โดย this ในบริบทนี้เป็นการอ้างอิงถึงออบเจกต์ใหม่ที่กำลังถูกสร้างขึ้น (Instance Reference) ซึ่งแตกต่างจากการใช้ this ในรูปแบบการสร้างออบเจกต์ในสองวิธีก่อนหน้าซึ่งจะใช้ this ในบริบทของการอ้างถึงคุณสมบัติในออบเจกต์เดียวกันที่ถูกสร้างขึ้นแล้ว (Owner Object Reference) การกำหนดค่าให้กับคุณสมบัติของออบเจกต์ในระหว่างการสร้างออบเจกต์ จะใช้พารามิเตอร์แต่ละค่ามากำหนดให้กับคุณสมบัติ ซึ่งการตั้งชื่อคุณสมบัติไม่จำเป็นต้องเหมือนกับชื่อพารามิเตอร์ แต่เพื่อความเข้าใจง่าย ควรตั้งชื่อให้สอดคล้องกัน

3) หลังจากที่สร้างฟังก์ชันแม่แบบแล้ว การสร้างออบเจกต์จะใช้คำสำคัญ new นำหน้าชื่อฟังก์ชันที่เป็นแม่แบบของออบเจกต์พร้อมระบุอาร์กิวเมนต์ซึ่งเรียงลำดับตรงกันกับพารามิเตอร์ของฟังก์ชันแม่แบบ โดยคำสำคัญ new จะสร้างออบเจกต์ใหม่และกำหนดให้ this ในฟังก์ชันแม่แบบให้อ้างอิงถึงออบเจกต์ใหม่นั้น ทั้งนี้ Constructor Function จะส่งคืนออบเจกต์ที่สร้างขึ้นโดยอัตโนมัติ โดยไม่จำเป็นต้องใช้คำสั่ง return ในฟังก์ชันแม่แบบ

ตัวอย่างที่ 13

```
1  function  Person(name, age) {
2        this.firstname = name;
3        this.age = age;
4        this.greet = function() {
5              console.log("สวัสดี ฉันชื่อ " + this.firstname);
6        }
7  }
8  const  staff1 = new Person("Nok", 30);
9  staff1.greet();
```

จากตัวอย่างที่ 13 ในบรรทัดที่ 1 มีการสร้างฟังก์ชันแม่แบบชื่อ Person ที่มีพารามิเตอร์ 2 ตัว คือ name และ age ซึ่งจะใช้รับค่าชื่อและอายุจากคำสั่งสร้างออบเจกต์

บรรทัดที่ 2-3 เป็นการนำค่าพารามิเตอร์ name มากำหนดเป็นค่าให้กับคุณสมบัติ firstname และพารามิเตอร์ age มากำหนดค่าให้กับคุณสมบัติ age ของออบเจกต์ใหม่ที่กำลังสร้างขึ้นตามลำดับ

บรรทัดที่ 4-6 เป็นการกำหนดเมธอดให้กับออบเจกต์ใหม่โดยทำหน้าที่แสดงข้อความ สวัสดี ฉันชื่อ และค่าคุณสมบัติ firstname ของออบเจกต์นั้นๆ โดยใช้ this.firstname

บรรทัดที่ 8 เป็นการสร้างออบเจกต์ชื่อ staff1 โดยส่งค่าอาร์กิวเมนต์เป็นข้อความคำว่า *Nok* และตัวเลข *30* ให้กับฟังก์ชัน Person(name, age)

หลังจากที่ได้มีการสร้างออบเจกต์ staff1 แล้ว ในบรรทัดที่ 9 เป็นเรียกใช้เมธอด greet( ) ของออบเจกต์ staff1 ซึ่งจะมีการแสดงข้อความ *สวัสดี ฉันชื่อ Nok* ทางคอนโซลของเว็บเบราว์เซอร์ เหตุผลที่แสดงคุณสมบัติ firstname เป็น *Nok* ออกมาได้เนื่องจาก this.firstname ในบรรทัดที่ 5 เป็นการอ้างอิงถึงคุณสมบัติ firstname ของออบเจกต์ staff1 ซึ่งมีค่าเป็นข้อความคำว่า *Nok* ที่ได้รับจากการสร้างออบเจกต์ ซึ่งแสดงให้เห็นถึงการทำงานของ this ในบริบท Instance Reference ที่อ้างอิงถึงออบเจกต์ที่กำลังถูกเรียกใช้เมธอด

2.3 การเข้าถึงและจัดการคุณสมบัติของออบเจกต์

ออบเจกต์ในภาษา JavaScript สามารถอ่าน ปรับปรุง เพิ่มหรือลบคุณสมบัติได้ในระหว่างการทำงานของโปรแกรม ความสามารถนี้สะท้อนลักษณะของ JavaScript ที่เป็นภาษาแบบไดนามิก (Dynamic Language) ซึ่งออบเจกต์สามารถเปลี่ยนแปลงโครงสร้างได้ระหว่างการทำงาน

2.3.1 การเข้าถึงคุณสมบัติและเมธอดของออบเจกต์

การเข้าถึงค่าของคุณสมบัติและเรียกใช้เมธอดของออบเจกต์สามารถทำได้ 2 วิธี ได้แก่ Dot Notation และ Bracket Notation ซึ่งแต่ละวิธีมีลักษณะการใช้งานที่ต่างกัน ดังนี้

1) รูปแบบ Dot Notation

การเข้าถึงคุณสมบัติและเรียกใช้เมธอดของออบเจกต์ในรูปแบบ Dot Notation เป็นรูปแบบที่อ่านง่ายและเป็นที่นิยม โดยจะใช้เครื่องหมายจุด (.) ระหว่างชื่อออบเจกต์และคุณสมบัติหรือเมธอดของออบเจกต์ โดยมีรูปแบบและตัวอย่างดังต่อนี้

รูปแบบ	
1	**ชื่อออบเจกต์.ชื่อคุณสมบัติ**
2	**ชื่อออบเจกต์.ชื่อเมธอด()**
3	**ชื่อออบเจกต์.ชื่อเมธอด(อาร์กิวเมนต์_1,...,อาร์กิวเมนต์_n)**

จากรูปแบบของ Dot Notation ข้างต้น รูปแบบที่ 1 เป็นรูปแบบของการเข้าถึงคุณสมบัติของออบเจกต์ รูปแบบที่ 2 เป็นรูปแบบการเรียกใช้เมธอดของออบเจกต์โดยไม่ส่งอาร์กิวเมนต์ไปให้เมธอด และรูปแบบที่ 3 เป็นรูปแบบการการเรียกใช้เมธอดของออบเจกต์พร้อมอาร์กิวเมนต์ไปให้เมธอด

ตัวอย่างที่ 14

```
1  const product = {
2      name: "ตู้เย็น",
3      price: 25000,
4      showInfo: function() {
5          console.log("สินค้า: " + this.name);
6      },
7      calculateDiscount: function(percent) {
8          return this.price * (1 - percent/100);
9      }
10 };
11 console.log(product.name);
12 product.showInfo();
13 let discountPrice = product.calculateDiscount(10);
```

จากตัวอย่างที่ 14 บรรทัดที่ 1-10 เป็นการสร้างออบเจกต์ชื่อ product โดยมีคุณสมบัติ 2 รายการคือ name และ price รวมทั้งมีเมธอดจำนวน 2 รายการคือ showInfo( ) และ calculateDiscount โดยเมธอด calculateDiscount รับพารามิเตอร์ percent เพื่อใช้ในการคำนวณราคาหลังหักส่วนลด

บรรทัดที่ 11 เป็นการเข้าถึงคุณสมบัติ name ของออบเจกต์ product เพื่อนำไปแสดงผลในคอนโซล ซึ่งจะแสดงเป็นข้อความคำว่า *ตู้เย็น*

บรรทัดที่ 12 เป็นการเรียกใช้เมธอด showInfo( ) ของออบเจกต์ product โดยไม่มีการส่งค่าอาร์กิวเมนต์ใดๆ ไปให้กับเมธอด การทำงานของเมธอดนี้ซึ่งแสดงในบรรทัดที่ 4-6 จะใช้ this.name เพื่อเข้าถึงคุณสมบัติ name ของออบเจกต์เดียวกัน และแสดงข้อความ *สินค้า: ตู้เย็น* ในคอนโซลของเว็บเบราว์เซอร์

บรรทัดที่ 12 เป็นการกำหนดค่าให้กับตัวแปร discountPrice โดยใช้ค่าจากเมธอด calculateDiscount() ของออบเจกต์ product ซึ่งมีการส่งค่าเป็นตัวเลข 10 ไปให้กับเมธอด การทำงานของเมธอดที่แสดงในบรรทัดที่ 7-9 พร้อมส่งอาร์กิวเมนต์ 10 เมธอดนี้คำนวณราคาหลังหักส่วนลดโดยใช้สูตร this.price × (1 - percent/100) ซึ่งเมื่อส่ง 10 เป็นอาร์กิวเมนต์ จะได้ 25000 × (1 - 10/100) = 22500 และส่งค่านี้กลับมากำหนดให้กับตัวแปร discountPrice

ในบรรทัดที่ 12 เป็นการกำหนดค่าให้กับตัวแปร discountPrice โดยใช้ค่าจากเมธอด calculateDiscount( ) ของออบเจกต์ product และมีการส่งอาร์กิวเมนต์เป็นตัวเลข 10 ไปให้กับเมธอด การทำงานของเมธอดที่แสดงในบรรทัดที่ 7-9 จะนำค่าที่ได้รับคือ 10 ซึ่งเป็นค่าพารามิเตอร์ percent ไปทำการคำนวณในบรรทัดที่ 8 โดยนำค่า price ของออบเจกต์ product คูณกับส่วนลดที่ได้จากการคำนวณ 1 - percent/100 ทำให้ได้ผลลัพธ์เป็น 22500 และจะส่งค่านี้กลับมากำหนดให้กับตัวแปร discountPrice

จากการเข้าถึงคุณสมบัติและเมธอดของออบเจกต์โดยใช้รูปแบบ Dot Notation มีข้อจำกัดที่สำคัญคือ ชื่อคุณสมบัติต้องได้รับการตั้งชื่อตามกฎของ JavaScript กล่าวคือ ไม่มีช่องว่าง ไม่ขึ้นต้นด้วยตัวเลข ไม่ใช้คำสงวนและไม่มีอักขระพิเศษยกเว้นขีดล่างและเครื่องหมาย $

2) รูปแบบ Bracket Notation

การเข้าถึงคุณสมบัติและเรียกใช้เมธอดของออบเจกต์ด้วย Bracket Notation ใช้เครื่องหมายวงเล็บเหลี่ยม [ ] โดยระบุชื่อคุณสมบัติหรือเมธอดเป็นสตริงภายในวงเล็บ เป็นรูปแบบที่เหมาะกับการเข้าถึงแบบพลวัต กล่าวคือ สามารถกำหนดคุณสมบัติที่ต้องการเข้าถึงในระหว่างการทำงานของโปรแกรมโดยใช้ค่าของตัวแปรได้แทนที่จะระบุเป็นค่าคงที่ลงในรหัสคำสั่ง โดยมีรูปแบบและตัวอย่างดังต่อไปนี้

รูปแบบ	
1	**ชื่อออบเจกต์["ชื่อคุณสมบัติ"]**
2	**ชื่อออบเจกต์[ตัวแปร]**
3	**ชื่อออบเจกต์["ชื่อคุณสมบัติ"]()**
4	**ชื่อออบเจกต์[ตัวแปร]()**
5	**ชื่อออบเจกต์["ชื่อเมธอด"](อาร์กิวเมนต์_1, ..., อาร์กิวเมนต์_n)**

จากรูปแบบของ Bracket Notation ข้างต้น สามารถแบ่งได้เป็น 5 รูปแบบดังนี้

- รูปแบบที่ 1 เป็นรูปแบบการเข้าถึงคุณสมบัติของออบเจกต์โดยระบุชื่อคุณสมบัติเป็นข้อความภายในเครื่องหมายวงเล็บเหลี่ยม โดยชื่อของคุณสมบัติจะเขียนระหว่างเครื่องหมายอัญประกาศคู่
- รูปแบบที่ 2 เป็นรูปแบบการเข้าถึงคุณสมบัติของออบเจกต์โดยใช้ตัวแปรที่เก็บชื่อคุณสมบัติไว้ ทำให้สามารถเข้าถึงคุณสมบัติแบบพลวัตได้
- รูปแบบที่ 3 เป็นรูปแบบการเรียกใช้เมธอดของออบเจกต์โดยระบุชื่อเมธอดเป็นข้อความและไม่ส่งอาร์กิวเมนต์ไปให้เมธอด
- รูปแบบที่ 4 เป็นรูปแบบการเรียกใช้เมธอดของออบเจกต์โดยใช้ตัวแปรที่เก็บชื่อเมธอดไว้และไม่ส่งอาร์กิวเมนต์ไปให้เมธอด ทำให้สามารถเรียกใช้เมธอดแบบพลวัตได้เช่นกัน
- รูปแบบที่ 5 เป็นรูปแบบการเรียกใช้เมธอดของออบเจกต์โดยระบุชื่อเมธอดเป็นข้อความและส่งค่าไปให้เมธอดของออบเจกต์ด้วย

อ้างถึงการสร้างออบเจกต์ product ในตัวอย่างที่ 14 มีวิธีการเข้าถึงคุณสมบัติและออบเจกต์โดยใช้รูปแบบ Bracket Notation ทั้ง 5 รูปแบบดังตัวอย่างต่อไปนี้

ตัวอย่างที่ 15

```
    //รูปแบบที่ 1
1   console.log(product["name"]);
    //รูปแบบที่ 2
2   let  propertyName = "name";
3   console.log(product[propertyName
    //รูปแบบที่ 3
4   product["showInfo"]( );
    //รูปแบบที่ 4
5   let  methodName = "showInfo";
6   product[methodName]( );
    //รูปแบบที่ 5
7   let  discount = product["calculateDiscount"](10);
```

จากตัวอย่างที่ 15 แสดงการเข้าถึงคุณสมบัติและเมธอดของออบเจกต์ product โดยใช้ Bracket Notation ในรูปแบบต่าง ๆ ทั้งหมด 5 รูปแบบ โดยอ้างอิงจากออบเจกต์ product ที่สร้างขึ้นด้วยการสร้างออบเจกจากตัวอย่างที่ 14 สามารถอธิบายได้ดังนี้

- บรรทัดที่ 1 เป็นการเข้าถึงคุณสมบัติ name ของออบเจกต์ product ตามรูปแบบที่ 1 ซึ่งระบุชื่อคุณสมบัติ name ไว้ระหว่างเครื่องหมายคำพูดภายในวงเล็บเหลี่ยม โดยจะแสดงผลลัพธ์ในคอนโซลของเว็บเบราว์เซอร์เป็นข้อความคำว่า ตู้เย็น
- บรรทัดที่ 2-3 เป็นการใช้ตัวแปรในการเข้าถึงคุณสมบัติของออบเจกต์ product ตามรูปแบบที่ 2 โดยในบรรทัดที่ 2 มีการกำหนดค่าข้อความ name ให้กับตัวแปร propertyName และบรรทัดที่ 5 ระบุตัวแปร propertyName ไว้ภายในวงเล็บเหลี่ยมโดยไม่ต้องใส่เครื่องหมายคำพูดเนื่องจากเป็นชื่อตัวแปร โดยจะแสดงผลลัพธ์ในคอนโซลของเว็บเบราว์เซอร์เป็นข้อความคำว่า ตู้เย็น เช่นเดียวกับบรรทัดที่ 1
- บรรทัดที่ 4 เป็นการเรียกใช้เมธอด showInfo() ของออบเจกต์ product ตามรูปแบบที่ 3 ที่ระบุชื่อเมธอด showInfo ไว้ระหว่างเครื่องหมายคำพูดภายในวงเล็บเหลี่ยม ตามด้วยวงเล็บเปิด-ปิดด้านหลังวงเล็บเหลี่ยมเพื่อเรียกใช้เมธอด ผลลัพธ์ที่แสดงในคอนโซลของเว็บเบราว์เซอร์จะเป็นข้อความคำว่า สินค้า: ตู้เย็น
- บรรทัดที่ 5-6 เป็นการใช้ตัวแปรในการเรียกใช้เมธอด showInfo() ของออบเจกต์ product ตามรูปแบบที่ 3 โดยบรรทัดที่ 5 กำหนดค่าข้อความ showInfo ให้กับตัวแปร methodName ซึ่งข้อความนี้จะต้องตรงกับชื่อของเมธอดที่จะเรียกใช้ และบรรทัดที่ 10 ระบุตัวแปร methodName ไว้ภายในวงเล็บเหลี่ยมเพื่อเรียกใช้เมธอด ผลลัพธ์ที่ได้จะแสดงในคอนโซลของเว็บเบราว์เซอร์เหมือนกับรูปแบบที่ 3 คือข้อความคำว่า สินค้า: ตู้เย็น
- บรรทัดที่ 7 เป็นการเรียกใช้เมธอด calculateDiscount() ของออบเจกต์ product ตามรูปแบบที่ 5 โดยมีการส่งค่าตัวเลข 10 ให้กับเมธอด ผลการทำงานของเมธอด calculateDiscount() ในออบเจกต์ product จะได้เป็น 22500 ซึ่งนำมาใช้กำหนดค่าดังกล่าวให้กับตัวแปร discount

2.3.2 การเปลี่ยนแปลงค่าของคุณสมบัติที่มีอยู่ในออบเจกต์

การเปลี่ยนแปลงค่าของคุณสมบัติในออบเจกต์สามารถทำได้โดยตรงผ่านการกำหนดค่าใหม่ให้กับคุณสมบัตินั้น ๆ โดยใช้ได้ทั้ง Dot Notation และ Bracket Notation ซึ่งเป็นรูปแบบเดียวกับการเข้าถึงคุณสมบัติที่มีอยู่แล้ว ความสามารถในการปรับเปลี่ยนค่าของคุณสมบัตินี้เป็นผลมาจากลักษณะของออบเจกต์ที่เป็นชนิดข้อมูลแบบอ้างอิงและมีความสามารถในการเปลี่ยนแปลงค่าได้ รูปแบบและตัวอย่างของการเปลี่ยนค่าคุณสมบัติของออบเจกต์ทั้งแบบ Dot Notation และ Bracket Notation มีดังนี้

รูปแบบ

```
1  ชื่อออบเจกต์.ชื่อคุณสมบัติ = ค่าใหม่ที่ต้องการเปลี่ยน;
2  ชื่อออบเจกต์["ชื่อคุณสมบัติ"] = ค่าใหม่ที่ต้องการเปลี่ยน;
```

ตัวอย่างที่ 16

```
// อ้างถึงการสร้างออบเจกต์ product ในตัวอย่างที่ 14
// รูปแบบที่ 1 Dot Notation
```

```
1   product.name = "เครื่องปรับอากาศ";
    // รูปแบบที่ 2 Bracket Notation
2   product["price"] = "50000";
```

จากตัวอย่างที่ 16 แสดงการเปลี่ยนแปลงค่าของคุณสมบัติในออบเจกต์ product โดยอ้างอิงจากการสร้างออบเจกต์ product ที่แสดงในตัวอย่างที่ 14 ซึ่งเดิมคุณสมบัติ name ของออบเจกต์ product มีค่าเป็นข้อความคำว่า *ตู้เย็น* สามารถอธิบายการเปลี่ยนค่าโดยใช้รูปแบบต่าง ดังนี้

- บรรทัดที่ 1 เป็นการเปลี่ยนแปลงค่าของคุณสมบัติ name ในออบเจกต์ product โดยใช้รูปแบบ Dot Notation ซึ่งกำหนดค่าใหม่เป็นข้อความ เครื่องปรับอากาศ แทนค่าเดิมที่เป็น ตู้เย็น การเปลี่ยนแปลงนี้จะส่งผลให้เมื่อมีการเข้าถึงคุณสมบัติ name ของออบเจกต์ product ในครั้งต่อไป จะได้ค่าใหม่เป็นเครื่องปรับอากาศ
- บรรทัดที่ 5 เป็นการเปลี่ยนแปลงค่าของคุณสมบัติ price ในออบเจกต์ product โดยใช้รูปแบบ Bracket Notation ซึ่งกำหนดค่าใหม่เป็นข้อความ 50000 แทนค่าเดิมที่เป็นตัวเลข 25000 การเปลี่ยนแปลงนี้จะส่งผลให้เมื่อมีการเข้าถึงคุณสมบัติ price ของออบเจกต์ product ในครั้งต่อไป จะได้ค่าใหม่เป็น 50000

2.3.3 การเพิ่มคุณสมบัติใหม่ให้กับออบเจกต์

การเพิ่มคุณสมบัติใหม่ให้กับออบเจกต์ในภาษา JavaScript สามารถทำได้โดยใช้รูปแบบเดียวกับการกำหนดค่าคุณสมบัติเดิม ทั้งแบบ Dot Notation และ Bracket Notation ซึ่งช่วยให้สามารถขยายออบเจกต์ให้มีคุณสมบัติหรือพฤติกรรมเพิ่มเติมได้โดยไม่จำเป็นต้องสร้างออบเจกต์ใหม่ รูปแบบและตัวอย่างของการเปลี่ยนค่าคุณสมบัติของออบเจกต์ทั้งแบบ Dot Notation และ Bracket Notation มีดังนี้

รูปแบบ

```
1   ชื่อออบเจกต์.ชื่อคุณสมบัติใหม่ = ค่า;
2   ชื่อออบเจกต์["ชื่อคุณสมบัติใหม่"] = ค่า;
```

ตัวอย่างที่ 17

```
    // อ้างถึงการสร้างออบเจกต์ product ในตัวอย่างที่ 14
    // รูปแบบที่ 1 Dot Notation
1   product.brand = "PYT";
    // รูปแบบที่ 2 Bracket Notation
2   product["year"] = 2568;
```

จากตัวอย่างที่ 17 แสดงการเพิ่มคุณสมบัติใหม่ให้กับออบเจกต์ product โดยอ้างอิงจากออบเจกต์ product ที่สร้างขึ้นในตัวอย่างที่ 14 ซึ่งเดิมมีเพียงคุณสมบัติ name และ price พร้อมทั้งเมธอด

showInfo() และ calculateDiscount() โดยการเพิ่มคุณสมบัติใหม่ใช้ทั้ง Dot Notation และ Bracket Notation สามารถอธิบายได้ดังนี้

- บรรทัดที่ 1 เป็นการเพิ่มคุณสมบัติใหม่ชื่อ brand ให้กับออบเจกต์ product โดยใช้รูปแบบ Dot Notation และกำหนดค่าเป็นข้อความ PYT คุณสมบัติ brand นี้ไม่ได้อยู่ในออบเจกต์ตั้งแต่เริ่มต้น แต่หลังจากการกำหนดค่าในบรรทัดนี้ ออบเจกต์ product จะมีคุณสมบัติ brand เพิ่มขึ้นมาและสามารถเข้าถึงได้เหมือนกับคุณสมบัติอื่นๆ ที่มีอยู่เดิม
- บรรทัดที่ 3 เป็นการเพิ่มคุณสมบัติใหม่ชื่อ year ให้กับออบเจกต์ product โดยใช้รูปแบบ Bracket Notation และกำหนดค่าเป็นตัวเลข 2568 เช่นเดียวกับคุณสมบัติ brand หลังจากการกำหนดค่าในบรรทัดนี้ ออบเจกต์ product จะมีคุณสมบัติ year เพิ่มขึ้นมาและสามารถเข้าถึงได้โดยใช้ทั้ง Dot Notation และ Bracket Notation

หลังจากการดำเนินการในตัวอย่างที่ 17 ออบเจกต์ product จะมีคุณสมบัติทั้งหมด 4 รายการ คือ name, price, brand และ year พร้อมด้วยเมธอด showInfo() และ calculateDiscount() ที่มีอยู่เดิม ความสามารถในการเพิ่มคุณสมบัติใหม่นี้แสดงให้เห็นถึงความยืดหยุ่นของออบเจกต์ในภาษา JavaScript ที่สามารถปรับเปลี่ยนและขยายโครงสร้างได้แม้หลังจากการสร้างแล้ว

2.3.4 การเปลี่ยนแปลงเมธอดของออบเจกต์

แม้ว่าคุณสมบัติที่เป็นเมธอดจะถูกกำหนดไว้ตั้งแต่ขั้นตอนการสร้างออบเจกต์ แต่เนื่องจากออบเจกต์ในภาษา JavaScript เป็นชนิดข้อมูลแบบอ้างอิงที่มีลักษณะเปลี่ยนแปลงค่าได้ จึงสามารถเปลี่ยนแปลงเมธอดที่มีอยู่เดิมให้มีคำสั่งการทำงานชุดใหม่ได้ การเปลี่ยนแปลงเมธอดนี้จะเป็นการแทนที่ชุดคำสั่งเดิมของเมธอดที่ระบุทั้งหมดด้วยชุดคำสั่งใหม่ ไม่ใช่การเพิ่มหรือแก้ไขบางส่วนและไม่สามารถเปลี่ยนชื่อได้ หากต้องการเปลี่ยนชื่อเมธอด จะต้องใช้วิธีเพิ่มเมธอดใหม่แล้วจึงลบเมธอดเดิม โดยรูปแบบการเปลี่ยนแปลงชุดคำสั่งของเมธอด สามารถทำได้ทั้งรูปแบบ Dot Notation และ Bracket Notation ดังรูปแบบ

รูปแบบที่ 1

```
ชื่อออบเจกต์.ชื่อเมธอด = function(พารามิเตอร์1, ...,พารามิเตอร์_n) {
    คำสั่ง; ...;
};
```

รูปแบบที่ 2

```
ชื่อออบเจกต์["ชื่อเมธอด"] = function(พารามิเตอร์1, ...,พารามิเตอร์_n) {
    คำสั่ง; ...;
};
```

จากรูปแบบที่ 1 และรูปแบบที่ 2 ของการเปลี่ยนแปลงเมธอดจะใช้รูปแบบ Dot Notation และ Bracket Notation ตามลำดับ โดยทั้งสองรูปแบบจะกำหนดฟังก์ชันใหม่ตามรูปแบบฟังก์ชันแบบประกาศที่กล่าวถึงในหัวข้อที่ 1.2.1 ให้กับเมธอดที่ต้องการเปลี่ยนแปลง ซึ่งจะเป็นตัวกำหนดพฤติกรรมการทำงานใหม่ของเมธอดนั้น ทั้งนี้การกำหนดฟังก์ชันใหม่จะมีพารามิเตอร์หรือไม่ก็ได้

ตัวอย่างที่ 18

```
   /* อ้างถึงการสร้างออบเจกต์ product ในตัวอย่างที่ 14
   และการเปลี่ยนค่าคุณสมบัติในตัวอย่างที่ 16 ที่เปลี่ยนค่าของคุณสมบัติ name เป็น ข้อความคำว่า
   เครื่องปรับอากาศ */
   // รูปแบบที่ 1 Dot Notation
1  product.showInfo = function( ) {
2        alert("ผลิตภัณฑ์ใหม่: " + this.name);
3  };
   // รูปแบบที่ 2 Bracket Notation
4  product["calculateDiscount"] = function(voucher, percent) {
5        return (this.price * (1 - percent/100)) - voucher;
6  }
```

จากตัวอย่างที่ 18 แสดงการเปลี่ยนแปลงเมธอดของออบเจกต์ product โดยอ้างอิงจากการสร้างออบเจกต์ product ที่แสดงในตัวอย่างที่ 14 ซึ่งเดิมมีเมธอด showInfo() ที่ทำหน้าที่แสดงข้อความเป็นคำว่า *สินค้า: เครื่องปรับอากาศ* และเมธอด calculateDiscount() ที่ทำหน้าที่คำนวณราคาหลังหักส่วนลดโดยรับพารามิเตอร์ percent เท่านั้น สามารถอธิบายการเปลี่ยนแปลงเมธอดทั้งสองดังกล่าวโดยใช้รูปแบบที่แตกต่างกันดังนี้

- บรรทัดที่ 1-3 เป็นการเปลี่ยนแปลงเมธอด showInfo() ของออบเจกต์ product โดยใช้รูปแบบ Dot Notation ซึ่งกำหนดชุดการทำงานของฟังก์ชัน showInfo() เป็นชุดใหม่ในบรรทัดที่ 4 ซึ่งเป็นการแสดงข้อความ ผลิตภัณฑ์ใหม่: เครื่องปรับอากาศ ในหน้าต่างแจ้งเตือนแทนการแสดงผลในคอนโซลของเว็บเบราว์เซอร์ การเปลี่ยนแปลงนี้จะส่งผลให้เมื่อมีการเรียกใช้เมธอด showInfo() ของออบเจกต์ product ในครั้งต่อไป จะแสดงผลเป็นหน้าต่างแจ้งเตือนแทนการแสดงผลในคอนโซล
- บรรทัดที่ 4-6 เป็นการเปลี่ยนแปลงเมธอด calculateDiscount() ของออบเจกต์ product โดยใช้รูปแบบ Bracket Notation ซึ่งมีการเปลี่ยนแปลงพารามิเตอร์ในบรรทัดที่ 4 จากเดิมมีเพียง percent เพียงอย่างเดียว เปลี่ยนเป็นรับค่าพารามิเตอร์จำนวน 2 ตัว คือ voucher ที่เป็นค่าคูปองส่วนลดและ percent ที่เป็นอัตราส่วนลดปกติ นอกจากนี้ ยังมีการกำหนดชุดการทำงานของฟังก์ชัน calculateDiscount() ใหม่ในบรรทัดที่ 5 โดยเป็นการคำนวณราคาหลังหักส่วนลด (percent) และหักคูปองส่วนลด (voucher) เพิ่มเติม

ซึ่งชุดคำสั่งเดิมจะคำนวณเฉพาะส่วนลดเปอร์เซ็นต์เพียงอย่างเดียว การเปลี่ยนแปลงนี้จะส่งผลให้เมื่อมีการเรียกใช้เมธอด calculateDiscount() ของออบเจกต์ product ในครั้งต่อไป จะต้องส่งพารามิเตอร์ 2 ตัวและจะได้ผลลัพธ์ที่หักทั้งส่วนลดเปอร์เซ็นต์และคูปองส่วนลด

จากตัวอย่างข้างต้น ทั้งสองรูปแบบสามารถใช้ในการเปลี่ยนแปลงเมธอดได้เหมือนกัน โดยความแตกต่างหลักอยู่ที่วิธีการระบุชื่อเมธอดเท่านั้น

2.3.5 การเพิ่มเมธอดให้กับออบเจกต์

การเพิ่มเมธอดให้กับออบเจกต์ในภาษา JavaScript สามารถดำเนินการได้โดยใช้รูปแบบเดียวกันกับการเปลี่ยนแปลงเมธอดของออบเจกต์ที่ได้กล่าวถึงในหัวข้อที่ 2.3.4 แตกต่างเพียงแค่ชื่อของเมธอดจะต้องเป็นชื่อใหม่ที่ไม่ปรากฏอยู่ในออบเจกต์เดิม โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบที่ 1

```
ชื่อออบเจกต์.ชื่อเมธอดใหม่ = function(พารามิเตอร์1, ...,พารามิเตอร์_n) {
    คำสั่ง;
};
```

รูปแบบที่ 2

```
ชื่อออบเจกต์["ชื่อเมธอดใหม่"] = function(พารามิเตอร์1, ...,พารามิเตอร์_n) {
    คำสั่ง;
};
```

ตัวอย่างที่ 19

```
    /* อ้างถึงการสร้างออบเจกต์ product ในตัวอย่างที่ 14 และการเปลี่ยนแปลงอื่นๆ ในตัวอย่างที่ 16-18
    */
    // รูปแบบที่ 1 Dot Notation
1   product.getFullInfo = function() {
2       return "สินค้า: " + this.name + " ยี่ห้อ: " + this.brand + " ปีที่ผลิต: " + this.year;
3   };
    // รูปแบบที่ 2 Bracket Notation
4   product["checkStock"] = function(quantity) {
5       if (quantity > 0) {
6           return "มีสินค้าในสต็อก จำนวน " + quantity + " ชิ้น";
7       } else {
8           return "สินค้าหมด";
9       }
10  };
```

จากตัวอย่างที่ 19 แสดงการเพิ่มเมธอดใหม่ให้กับออบเจกต์ product โดยอ้างอิงจากการสร้างออบเจกต์ product ที่แสดงในตัวอย่างที่ 14 และการเปลี่ยนแปลงอื่นๆ ในตัวอย่างที่ 16-18 ออบเจกต์ product มีคุณสมบัติ name, price, brand และ year ที่ใช้เก็บชื่อสินค้า ชื่อยี่ห้อและปีที่ผลิตตามลำดับ รวมทั้งเมธอด showInfo() มีหน้าที่แสดงชื่อผลิตภัณฑ์ใหม่ โดยใช้ค่าคุณสมบัติ name และเมธอด calculateDiscount() ที่มีหน้าที่คำนวณราคาสินค้าที่หักจากส่วนลดปกติและคูปองส่วนลด

ในตัวอย่างนี้ เป็นเพิ่มเมธอดให้กับออบเจกต์ product จำนวน 2 เมธอด ประกอบด้วยเมธอด getFullInfo( ) ใช้สำหรับแสดงข้อมูลสินค้าพร้อมยี่ห้อและปีที่ผลิต และเมธอด checkStock() ที่จะรับค่าพารามิเตอร์ quantity เพื่อตรวจสอบสถานะสินค้าในสต็อก

- บรรทัดที่ 1-3 เป็นการเพิ่มเมธอดใหม่ชื่อ getFullInfo() ให้กับออบเจกต์ product โดยใช้รูปแบบ Dot Notation ซึ่งเป็นเมธอดที่ไม่มีการรับค่า ทำหน้าที่รวบรวมข้อมูลชื่อสินค้า ยี่ห้อและปีที่ผลิตเป็นข้อความเดียว โดยใช้คุณสมบัติ name, brand และ year ของออบเจกต์เดียวกันนี้โดยใช้คำสำคัญ this และส่งกลับเป็นข้อความที่มีรูปแบบเป็น สินค้า: เครื่องปรับอากาศ ยี่ห้อ: PYT ปีที่ผลิต: 2568 บาท
- บรรทัดที่ 4-10 เป็นการเพิ่มเมธอดใหม่ชื่อ checkStock() ให้กับออบเจกต์ product โดยใช้รูปแบบ Bracket Notation ซึ่งเป็นเมธอดที่รับพารามิเตอร์ quantity เพื่อตรวจสอบสถานะของสินค้าในสต็อก โดยใช้โครงสร้างควบคุม if-else ในการตรวจสอบค่าที่ได้รับ หากค่า quantity มากกว่า 0 จะส่งกลับข้อความแสดงจำนวนสินค้าในสต็อก แต่หากค่า quantity เท่ากับ 0 หรือน้อยกว่า จะส่งกลับข้อความ สินค้าหมด

หลังจากการดำเนินการในตัวอย่างที่ 19 ออบเจกต์ product จะมีเมธอดทั้งหมด 4 รายการ คือ showInfo(), calculateDiscount(), getFullInfo() และ checkStock() ซึ่งแสดงให้เห็นถึงความยืดหยุ่นของออบเจกต์ในการขยายความสามารถได้แม้หลังจากการสร้างแล้ว

2.3.6 การลบคุณสมบัติหรือเมธอดออกจากออบเจกต์

ภาษา JavaScript มีความยืดหยุ่นในการจัดการกับออบเจกต์ โดยอนุญาตให้ผู้พัฒนาสามารถลบคุณสมบัติหรือเมธอดที่ไม่ต้องการออกจากออบเจกต์ได้ในระหว่างการทำงานของโปรแกรม โดยใช้คำสั่ง delete ซึ่งเป็นตัวดำเนินการเฉพาะในภาษา JavaScript ที่ทำหน้าที่ลบคู่คีย์-ค่า (key-value pair) ออกจากออบเจกต์ โดยรูปแบบและตัวอย่างการลบคุณสมบัติหรือเมธอดออกจากออบเจกต์สามารถทำได้ทั้งรูปแบบ Dot Notation และ Bracket Notation ดังนี้

รูปแบบ

```
1  delete ชื่อออบเจกต์.ชื่อคุณสมบัติหรือเมธอด;
2  delete ชื่อออบเจกต์["ชื่อคุณสมบัติหรือเมธอด"];
```

ตัวอย่างที่ 20

```
/* อ้างถึงการสร้างออบเจกต์ product ในตัวอย่างที่ 14 และการเปลี่ยนแปลงอื่นๆ ในตัวอย่างที่ 16-19
*/
```

```
   // รูปแบบที่ 1 Dot Notation
1  delete  product.price;
   // รูปแบบที่ 2 Bracket Notation
2  delete  product["showInfo"];
```

จากตัวอย่างที่ 20 แสดงการลบคุณสมบัติและเมธอดของออบเจกต์ product โดยอ้างอิงจากการสร้างออบเจกต์ product ที่แสดงในตัวอย่างที่ 14 และการเปลี่ยนแปลงอื่นๆ ในตัวอย่างที่ 16-19 ซึ่งขณะนี้ออบเจกต์ product มีคุณสมบัติ name, price, brand และ year รวมทั้งเมธอด showInfo(), calculateDiscount(), getFullInfo() และ checkStock() สามารถอธิบายการลบคุณสมบัติและเมธอดโดยใช้รูปแบบที่แตกต่างกัน ดังนี้

- บรรทัดที่ 1 เป็นการลบคุณสมบัติ price ออกจากออบเจกต์ product โดยใช้รูปแบบ Dot Notation ร่วมกับตัวดำเนินการ delete ซึ่งจะทำให้คุณสมบัติ price ถูกเอาออกจากออบเจกต์อย่างถาวร การลบนี้จะส่งผลให้เมื่อมีการเข้าถึงคุณสมบัติ price ของออบเจกต์ product ในครั้งต่อไป จะได้ค่า undefined และเมธอดที่อ้างอิงถึงคุณสมบัติ price เช่น calculateDiscount() อาจทำงานผิดพลาดได้
- บรรทัดที่ 2 เป็นการลบเมธอด showInfo() ออกจากออบเจกต์ product โดยใช้รูปแบบ Bracket Notation ร่วมกับตัวดำเนินการ delete ซึ่งจะทำให้เมธอด showInfo() ถูกเอาออกจากออบเจกต์อย่างถาวร การลบนี้จะส่งผลให้เมื่อมีการเรียกใช้เมธอด showInfo() ของออบเจกต์ product ในครั้งต่อไป จะเกิดข้อผิดพลาดประเภท TypeError เนื่องจาก showInfo ไม่ใช่ฟังก์ชัน

หลังจากการดำเนินการในตัวอย่างที่ 20 ออบเจกต์ product จะมีคุณสมบัติเหลือ 3 รายการ คือ name, brand และ year และมีเมธอดเหลือ 3 รายการ คือ calculateDiscount(), getFullInfo() และ checkStock() การใช้ตัวดำเนินการ delete ต้องใช้ด้วยความระมัดระวัง เนื่องจากอาจส่งผลกระทบต่อการทำงานของเมธอดอื่นที่อาจอ้างอิงถึงคุณสมบัติหรือเมธอดที่ถูกลบ

3. อาร์เรย์

อาร์เรย์เป็นกลไกหนึ่งของการเขียนโปรแกรมที่ช่วยให้ผู้พัฒนาสามารถจัดการข้อมูลหลายรายการภายใต้ตัวแปรเดียว อาร์เรย์จึงเป็นเครื่องมือพื้นฐานสำคัญในการพัฒนาเว็บแอปพลิเคชัน

3.1 ความหมายและบทบาทของอาร์เรย์

อาร์เรย์ (Array) เป็นโครงสร้างข้อมูลที่ใช้สำหรับจัดเก็บข้อมูลหลายค่าไว้ในตัวแปรเดียวกัน สามารถเก็บข้อมูลหลายชนิดในอาร์เรย์เดียวกันได้ รวมทั้งมีขนาดที่สามารถเปลี่ยนแปลงได้ตามการ โดยข้อมูลเหล่านั้นจะถูกจัดเรียงตามลำดับและมีดัชนี (Index) ที่เริ่มนับจาก 0 สำหรับการอ้างอิงไปยังตำแหน่งของข้อมูลแต่ละตัว แสดงดังภาพที่ 10.1



ภาพที่ 10.1 โครงสร้างอาร์เรย์

จากภาพที่ 10.1 จะเห็นว่าลำดับการนับข้อมูลตามปกติจะนับเป็นลำดับที่ 1, 2, 3, ... ไปจนถึงตัวสุดท้ายซึ่งกำหนดให้มีค่าเป็น n ข้อมูลที่อยู่ในลำดับที่ 1 จะมีดัชนีชี้ตำแหน่งเป็น 0 และข้อมูลลำดับที่ 2 จะมีค่าดัชนีเป็น 1 และเพิ่มค่าดัชนีทีละ 1 ค่าเช่นนี้ไปเรื่อยๆ จนถึงข้อมูลลำดับสุดท้ายคือ n จะมีค่าดัชนีเป็น n-1

ในบริบทของการพัฒนาเว็บแอปพลิเคชัน อาร์เรย์มีบทบาทสำคัญในการจัดการอิลิเมนต์ต่าง ๆ ทั้งในด้าน เช่น การจัดเก็บข้อมูลที่มีลักษณะเป็นรายการ การสร้างอิลิเมนต์ของเอกสารเว็บแบบพลวัต การประมวลผลและการวิเคราะห์ข้อมูล รวมทั้งการจัดการสถานะและการโต้ตอบ เป็นต้น

3.2 การสร้างอาร์เรย์

การสร้างอาร์เรย์ใน JavaScript สามารถทำได้ 2 วิธี คือ การสร้างอาร์เรย์ด้วยเครื่องหมายวงเล็บสี่เหลี่ยม และการสร้างอาร์เรย์ด้วย Array Constructor

3.2.1 การสร้างอาร์เรย์ด้วยเครื่องหมายวงเล็บสี่เหลี่ยม

การสร้างอาร์เรย์ด้วยเครื่องหมายวงเล็บสี่เหลี่ยม เป็นการใช้เครื่องหมายวงเล็บเหลี่ยม [ ] ในการประกาศอาร์เรย์และระบุข้อมูลเริ่มต้น ตลอดจนสามารถสร้างอาร์เรย์ว่างเพื่อเพิ่มข้อมูลภายหลังได้

รูปแบบ

```
let ชื่อตัวแปร = [ข้อมูล_1, ข้อมูล_2, ..., ข้อมูล_n];
```

ตัวอย่างที่ 21

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  let numbers = [5, 1, 5, 5, 2, 10];
3  let mixed = ["hello", 42, true, null];
4  let empty = [ ];
```

จากตัวอย่างที่ 21 บรรทัดที่ 1 เป็นการสร้างอาร์เรย์ที่มีข้อมูลเป็นชนิดข้อความทั้งหมด 3 รายการ บรรทัดที่ 2 เป็นการสร้างอาร์เรย์ที่มีข้อมูลเป็นตัวเลขทั้งหมด 6 รายการ บรรทัดที่ 3 เป็นการสร้างอาร์เรย์ที่เก็บข้อมูลหลายประเภทรวมกัน และบรรทัดที่ 4 เป็นการสร้างอาร์เรย์ว่างหมายความว่าไม่มีสมาชิกในอาร์เรย์

3.2.2 การสร้างอาร์เรย์ด้วย Array Constructor

การสร้างอาร์เรย์ด้วย Array Constructor เป็นอีกวิธีหนึ่งในการสร้างอาร์เรย์ใน JavaScript โดยใช้คำสั่ง new Array() แม้ว่าวิธีนี้จะไม่ได้รับความนิยมมากเท่ากับการใช้วงเล็บเหลี่ยม แต่ก็เป็นวิธีที่สามารถใช้งานได้และมีประโยชน์ในบางสถานการณ์

รูปแบบ

let ชื่อตัวแปร = new Array(ข้อมูล_1, ข้อมูล_2, ..., ข้อมูล_n);

ตัวอย่างที่ 22

```
1  let language = new Array("HTML", "CSS", "JavaScript");
2  let numbers = new Array(5, 1, 5, 5, 2, 10);
3  let mixed = new Array("hello", 42, true, null);
4  let empty = new Array();
```

จากตัวอย่างที่ 22 เป็นการสร้างอาร์เรย์ด้วย Array Constructor โดยแต่ละบรรทัดจะให้ผลเหมือนกับการสร้างอาร์เรย์ด้วยวงเล็บสี่เหลี่ยมที่แสดงในตัวอย่างที่ 21

3.3 การเข้าถึงและเปลี่ยนแปลงข้อมูลในอาร์เรย์

หลังจากสร้างอาร์เรย์แล้ว การเข้าถึงและเปลี่ยนแปลงข้อมูลภายในอาร์เรย์เป็นการดำเนินการพื้นฐานที่สำคัญในการใช้งานอาร์เรย์ โดยการเข้าถึงข้อมูลจะใช้ดัชนี (Index) ซึ่งเป็นตัวชี้ลำดับของข้อมูลในอาร์เรย์ซึ่งจะเริ่มนับจากเลข 0 นอกจากนี้ ข้อมูลในอาร์เรย์สามารถเปลี่ยนแปลงข้อมูลได้หลังจากที่สร้างแล้ว

3.3.1 การเข้าถึงข้อมูลในอาร์เรย์

การเข้าถึงข้อมูลในอาร์เรย์ทำได้โดยการระบุดัชนีของตำแหน่งที่ต้องการภายในเครื่องหมายวงเล็บเหลี่ยม โดยมีรูปแบบและตัวอย่างการใช้งานดังต่อไปนี้

รูปแบบ

ชื่ออาร์เรย์[ดัชนี]

ตัวอย่างที่ 23

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  console.log(language[0]);
3  console.log(language[2]);
4  console.log(language[3]);
5  console.log(language.length);
```

จากตัวอย่างที่ 23 ในบรรทัดที่ 1 เป็นการสร้างตัวแปร language ให้จัดเก็บข้อมูลประเภทอาร์เรย์ที่ประกอบด้วยข้อมูลชนิดข้อความจำนวน 3 รายการ คือ HTML, CSS และ JavaScript ตามลำดับ บรรทัดที่ 2-5 เป็นการแสดงข้อมูลออกทางคอนโซลของเว็บเบราว์เซอร์โดยมีรายละเอียดดังนี้

- บรรทัดที่ 2 เป็นการแสดงค่าของอาร์เรย์ที่มีดัชนีเป็น 0 ซึ่งเป็นการอ้างถึงตำแหน่งแรกของอาร์เรย์ที่มีข้อมูลเป็นข้อความว่า HTML
- บรรทัดที่ 3 เป็นการแสดงค่าของอาร์เรย์ที่มีดัชนีเป็น 2 ซึ่งเป็นการอ้างถึงตำแหน่งที่ 3 ของอาร์เรย์ที่มีข้อมูลเป็นข้อความว่า JavaScript
- บรรทัดที่ 4 เป็นการแสดงค่าของอาร์เรย์ที่มีดัชนีเป็น 3 ซึ่งเป็นการเข้าถึงตำแหน่งที่ไม่มีอยู่ในอาร์เรย์ จึงได้ค่า undefined
- บรรทัดที่ 5 เป็นการแสดงจำนวนข้อมูลทั้งหมดของอาร์เรย์โดยใช้คุณสมบัติ length จะให้ผลลัพธ์เป็นเลข 3 เนื่องจากอาร์เรย์มีข้อมูลจำนวน 3 รายการ ถึงแม้ว่าตัวแปร language จะไม่ได้มีการประกาศเป็นออบเจกต์ แต่ใน JavaScript อาร์เรย์เป็นชนิดข้อมูลพิเศษที่สืบทอดมาจากออบเจกต์ จึงมีคุณสมบัติและเมธอดในตัวที่สามารถใช้งานได้ เช่น คุณสมบัติ length ที่จะอัปเดตค่าโดยอัตโนมัติเมื่อมีการเพิ่มหรือลบข้อมูลในอาร์เรย์ ทำให้สามารถทราบจำนวนข้อมูลในอาร์เรย์ได้ตลอดเวลา

3.3.2 การเปลี่ยนแปลงข้อมูลในอาร์เรย์

อาร์เรย์ใน JavaScript สามารถเปลี่ยนแปลงข้อมูลได้หลังจากที่สร้างแล้ว โดยสามารถกำหนดค่าใหม่ให้กับตำแหน่งใดตำแหน่งหนึ่งหรือเพิ่มข้อมูลใหม่ได้ การเปลี่ยนแปลงข้อมูลทำได้โดยการระบุดัชนีของตำแหน่งที่ต้องการเปลี่ยนแปลงและกำหนดค่าใหม่ให้โดยมีรูปแบบและตัวอย่างดังต่อไปนี้

รูปแบบ

ชื่ออาร์เรย์[ดัชนี] = ค่าใหม่;

ตัวอย่างที่ 24

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  console.log("ก่อนเปลี่ยนแปลง:", language[1]);
3  language[1] = "Bootstrap";
4  console.log("หลังเปลี่ยนแปลง:", language[1]);
5  language[3] = "JQuery";
6  console.log("หลังเพิ่มข้อมูล:", language);
7  language[language.length] = "PHP";
8  console.log("หลังเพิ่มด้วย length:", language);
9  language[6] = "Web App";
10 console.log("หลังข้ามดัชนี:", language);
```

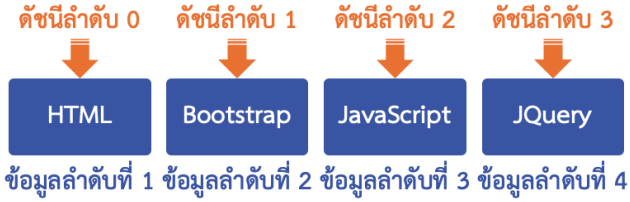
```
11 | console.log("ขนาดใหม่:", language.length);
```

จากตัวอย่างที่ 24 เป็นตัวอย่างของการเปลี่ยนแปลงข้อมูลในอาร์เรย์หลังจากที่มีการสร้างอาร์เรย์ขึ้นมาแล้ว ในบรรทัดที่ 1 เป็นการสร้างตัวแปร language ที่เก็บข้อมูลชนิดอาร์เรย์ที่มีรายการข้อมูล 3 รายการแสดงดังภาพที่ 10.2 โดยคำสั่งอื่นๆ มีการทำงานกับอาร์เรย์ที่สำคัญดังนี้



ภาพที่ 10.2 ตัวอย่างอาร์เรย์ที่มีข้อมูลจำนวน 3 รายการ

- บรรทัดที่ 2-4 เป็นการแสดงให้เห็นถึงข้อมูลก่อนและหลังเปลี่ยนแปลงข้อมูลในตำแหน่งดัชนีลำดับ 1 โดยค่าเดิมเป็นข้อความคำว่า CSS เมื่อมีการเปลี่ยนแปลงข้อมูลตามคำสั่งในบรรทัดที่ 3 จะทำให้ข้อความคำว่า Bootstrap ไปแทนที่ค่าเดิม ในบรรทัดที่ 4 เป็นคำสั่งที่จะแสดงผลเป็นข้อความคำว่า Bootstrap ในคอนโซล
- บรรทัดที่ 5-6 แสดงการเพิ่มข้อมูลใหม่โดยการกำหนดเป็นข้อความคำว่า JQuery ให้กับตำแหน่งดัชนีลำดับ 3 ซึ่งเป็นตำแหน่งถัดจากตำแหน่งสุดท้ายของอาร์เรย์ (ดัชนีสุดท้ายคือ 2) ทำให้ข้อมูลใหม่ถูกเพิ่มต่อท้ายอาร์เรย์ โดยในบรรทัดที่ 6 จะแสดงข้อมูลของอาร์เรย์ทั้งหมดซึ่งมีข้อมูลจำนวน 4 รายการดังภาพที่ 10.3



ภาพที่ 10.3 ตัวอย่างผลลัพธ์ของการเพิ่มข้อมูลในอาร์เรย์

- บรรทัดที่ 7–8 ใช้คุณสมบัติ length ของอาร์เรย์ language เพื่อใช้เป็นดัชนีของข้อมูลที่จะเพิ่มเข้ามาในอาร์เรย์ กล่าวคือ language.length จะได้ค่า 4 เนื่องจากมีรายการข้อมูล 4 รายการ จากนั้นนำค่าที่ได้มาใช้เป็นดัชนีสำหรับเพิ่มข้อมูลใหม่ซึ่งจะได้เป็น language[4] = "PHP" แสดงดังภาพที่ 10.4 การใช้คุณสมบัติ length ของอาร์เรย์นี้ จะช่วยให้เพิ่มข้อมูลในตำแหน่งสุดท้ายของอาร์เรย์ได้โดยไม่ต้องระบุดัชนีด้วยตนเอง และช่วยลดข้อผิดพลาดจากการนับดัชนีที่ไม่ถูกต้อง



ภาพที่ 10.4 ตัวอย่างของผลลัพธ์การเพิ่มข้อมูลใหม่ด้วยการใช้ค่าคุณสมบัติ length เป็นดัชนี

- บรรทัดที่ 9–10 แสดงการกำหนดค่าเป็นข้อความคำว่า PHP ไว้ในตำแหน่งดัชนีลำดับ 6 ของอาร์เรย์ ซึ่งดัชนีดังกล่าวไม่มีอยู่ในอาร์เรย์ (ค่าสูงสุดคือ 4) การกระทำนี้จะทำให้ขยายความยาวของอาร์เรย์โดยอัตโนมัติ และข้อมูลที่อยู่ตำแหน่งดัชนีลำดับ 5 จะมีค่าเป็น undefined แสดงดังภาพที่ 10.5



ภาพที่ 10.5 ตัวอย่างผลการเพิ่มข้อมูลด้วยการระบุดัชนีให้มีค่าเกินกว่าดัชนีสูงสุดเดิมของอาร์เรย์

- บรรทัดที่ 11 เป็นการแสดงค่าความยาวของอาร์เรย์ด้วย language.length ซึ่งหลังจากมีการกำหนดค่าถึงดัชนีลำดับ 6 แล้ว ความยาวของอาร์เรย์จึงกลายเป็น 7 แม้จะมีช่องว่างเกิดขึ้นในอาร์เรย์

จากตัวอย่างจะพบว่า การเปลี่ยนแปลงข้อมูลในอาร์เรย์ด้วยการระบุลำดับของดัชนีมีสิ่งที่ต้องระวัง คือ หากกำหนดดัชนีที่ข้ามตำแหน่งไปจะทำให้เกิดช่องว่าง (Empty Slots) ในอาร์เรย์ ซึ่งอาจส่งผลต่อการทำงานของโปรแกรม วิธีที่แนะนำสำหรับการเพิ่มข้อมูลต่อท้ายอาร์เรย์ด้วยการระบุดัชนีคือการใช้คุณสมบัติ length ของอาร์เรย์เข้ามาช่วยกำหนดค่าดัชนีตำแหน่งถัดไปสำหรับการเพิ่มข้อมูลต่อท้ายอาร์เรย์ ซึ่งเป็นการช่วยให้ลำดับดัชนีมีความต่อเนื่อง ไม่ทำให้เกิดช่องว่างในอาร์เรย์ ทั้งนี้ การข้ามดัชนีเป็นสิ่งที่ควรหลีกเลี่ยงเพราะจะทำให้เกิดช่องว่างที่ไม่พึงประสงค์

นอกจากนี้ การใช้ดัชนีไม่สามารถเพิ่มข้อมูลในช่วงต้นและกลางของอาร์เรย์ได้ เนื่องจากจะเป็นการเขียนทับข้อมูลเดิมแทนการแทรกข้อมูลใหม่ การดำเนินการเช่นนี้จำเป็นต้องใช้เมธอดของอาร์เรย์ที่จะกล่าวถึงในหัวข้อถัดไป

3.4 เมธอดสำคัญของอาร์เรย์

การจัดการข้อมูลในอาร์เรย์นอกเหนือจากการใช้ดัชนี JavaScript มีเมธอด หรือฟังก์ชันในตัวที่ช่วยให้การดำเนินการกับอาร์เรย์เป็นไปได้อย่างมีประสิทธิภาพและสะดวกมากขึ้น เมธอดเหล่านี้ครอบคลุมการเพิ่มและลบข้อมูล การค้นหาและกรองข้อมูล ตลอดจนการประมวลผลข้อมูลในรูปแบบต่าง ๆ ดังนี้

3.4.1 เมธอดเพิ่มและลบข้อมูล

JavaScript มีเมธอดสำหรับจัดการข้อมูลในอาร์เรย์ที่ใช้งานบ่อย ได้แก่ push() และ pop() สำหรับจัดการข้อมูลที่ท้ายอาร์เรย์ และ unshift() กับ shift() สำหรับจัดการข้อมูลที่ต้นอาร์เรย์

1) การเพิ่มข้อมูลท้ายอาร์เรย์ด้วย push()

เมธอด push() ใช้เพิ่มข้อมูลใหม่ลงท้ายอาร์เรย์ สามารถเพิ่มได้หลายรายการในครั้งเดียว และจะเพิ่มความยาวของอาร์เรย์โดยอัตโนมัติ เมธอดนี้จะคืนค่าเป็นความยาวใหม่ของอาร์เรย์ มีรูปแบบและตัวอย่างดังนี้

รูปแบบ

ชื่ออาร์เรย์.push(ข้อมูล_1, ข้อมูล_2, ..., ข้อมูล_n);

ตัวอย่างที่ 25

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  console.log("ก่อนเพิ่ม:", language);
   // เพิ่มรายการเดียว
3  let newLength1 = language.push("jQuery");
4  console.log("หลังเพิ่ม jQuery:", language);
5  console.log("ความยาวใหม่:", newLength1);
   // เพิ่มหลายรายการพร้อมกัน
6  let newLength2 = language.push("React", "Vue", "Angular");
7  console.log("หลังเพิ่มหลายรายการ:", language);
8  console.log("ความยาวสุดท้าย:", newLength2);
```

จากตัวอย่างที่ 25 แสดงการใช้เมธอด push() ในสองรูปแบบ ในบรรทัดที่ 1 เป็นการประกาศอาร์เรย์ language ที่มีข้อมูลเริ่มต้นจำนวน 3 รายการ ในบรรทัดที่ 3 ใช้เมธอด push() เพิ่มข้อมูลที่เป็นข้อความคำว่า *jQuery* รายการเดียว และในบรรทัดที่ 6 แสดงการเพิ่มข้อมูลที่เป็นข้อความหลายรายการพร้อมกัน ได้แก่ *React*, *Vue* และ *Angular* โดยเมธอดจะคืนค่าความยาวใหม่หรือจำนวนข้อมูลของอาร์เรย์หลังจากเพิ่มข้อมูลแล้ว ส่วนการเปลี่ยนแปลงข้อมูลจะเกิดขึ้นอาร์เรย์เดิมโดยไม่ต้องสร้างตัวแปรขึ้นมารับการเปลี่ยนแปลงนี้

2) การลบข้อมูลท้ายอาร์เรย์ด้วย pop()

เมธอด pop() ใช้ลบข้อมูลรายการสุดท้ายออกจากอาร์เรย์ เมธอดนี้จะคืนค่าเป็นข้อมูลที่ถูกลบ หากอาร์เรย์ว่างเปล่าจะคืนค่า undefined และจะลดความยาวของอาร์เรย์โดยอัตโนมัติ มีรูปแบบและตัวอย่างดังนี้

รูปแบบ

ชื่ออาร์เรย์.pop();

ตัวอย่างที่ 26

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  console.log("ก่อนลบ:", language);
   // ลบข้อมูลท้ายอาร์เรย์
3  let removedItem = language.pop();
4  console.log("ข้อมูลที่ถูกลบ:", removedItem);
5  console.log("หลังลบ:", language);
6  console.log("ความยาวใหม่:", language.length);
   // ลบต่อเนื่องหลายครั้ง
7  let removed1 = language.pop();
8  let removed2 = language.pop();
9  console.log("ลบครั้งที่ 2:", removed1);
10 console.log("ลบครั้งที่ 3:", removed2);
11 console.log("อาร์เรย์ที่เหลือ:", language);
```

จากตัวอย่างที่ 26 ในบรรทัดที่ 1 เป็นการประกาศอาร์เรย์ language ที่มีข้อมูลเริ่มต้น 3 รายการ บรรทัดที่ 3 ใช้เมธอด pop() ลบข้อมูลรายการสุดท้ายซึ่งเป็นข้อความคำว่า *JavaScript* โดยมีการประกาศตัวแปรชื่อ removedItem เพื่อเก็บค่าที่ถูกลบทำให้สามารถนำข้อมูลที่ถูกลบมาใช้ใหม่ได้ และในบรรทัดที่ 7-8 เป็นการลบข้อมูลในอาร์เรย์โดยแต่ละครั้งจะลบรายการสุดท้ายออกไป การลบข้อมูลนี้จะเกิดขึ้นกับอาร์เรย์ language ทุกครั้งเช่นเดียวกับเมธอด push() แต่การคืนค่าจะต่างกันโดยเมธอด pop() จะคืนค่าเป็นข้อมูลที่ถูกลบ ส่วนเมธอด push() จะให้ค่าเป็นจำนวนข้อมูลในอาร์เรย์

3) การเพิ่มข้อมูลต้นอาร์เรย์ด้วย unshift()

เมธอด unshift() ใช้เพิ่มข้อมูลใหม่ลงต้นอาร์เรย์ สามารถเพิ่มได้หลายรายการในครั้งเดียว และจะเพิ่มความยาวของอาร์เรย์โดยอัตโนมัติ เมธอดนี้จะคืนค่าเป็นความยาวใหม่ของอาร์เรย์เช่นเดียวกับ push() มีรูปแบบและตัวอย่างดังนี้

รูปแบบ

ชื่ออาร์เรย์.unshift(ข้อมูล_1, ข้อมูล_2, ..., ข้อมูล_n);

ตัวอย่างที่ 27

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  console.log("ก่อนเพิ่ม:", language);
   // เพิ่มรายการเดียวที่ต้นอาร์เรย์
3  let newLength1 = language.unshift("XML");
```

```
4  console.log("หลังเพิ่ม XML:", language);
5  console.log("ความยาวใหม่:", newLength1);
   // เพิ่มหลายรายการพร้อมกันที่ต้นอาร์เรย์
6  let newLength2 = language.unshift("JSON", "YAML", "TOML");
7  console.log("หลังเพิ่มหลายรายการ:", language);
8  console.log("ความยาวสุดท้าย:", newLength2);
```

จากตัวอย่างที่ 27 ในบรรทัดที่ 1 เป็นการประกาศอาร์เรย์ language ที่มีข้อมูลเริ่มต้น 3 รายการ บรรทัดที่ 3 ใช้เมธอด unshift() เพิ่มข้อความคำว่า *XML* ไว้ที่ต้นอาร์เรย์ทำให้ *XML* กลายเป็นรายการแรก และในบรรทัดที่ 6 แสดงการเพิ่มข้อมูลหลายรายการพร้อมกัน ได้แก่ *JSON*, *YAML* และ *TOML* ที่ต้นอาร์เรย์ โดยรายการที่เพิ่มครั้งหลังจะอยู่หน้ารายการที่เพิ่มก่อนหน้าซึ่งจะได้ผลลัพธ์เป็น ['JSON', 'YAML', 'TOML', 'XML', 'HTML', 'CSS', 'JavaScript'] ทั้งนี้ การเปลี่ยนเพิ่มข้อมูลจะเกิดขึ้นกับอาร์เรย์เดิมคือ language และเมธอดจะคืนค่าความยาวใหม่ของอาร์เรย์ ซึ่งสามารถประกาศตัวแปรมาจัดเก็บค่านี้เพื่อนำไปใช้งานในส่วนได้

4) การลบข้อมูลต้นอาร์เรย์ด้วย shift()

เมธอด shift() ใช้ลบข้อมูลรายการแรกออกจากอาร์เรย์ เมธอดนี้จะคืนค่าเป็นข้อมูลที่ถูกลบออก หากอาร์เรย์ว่างเปล่าจะคืนค่า undefined และจะลดความยาวของอาร์เรย์โดยอัตโนมัติ

รูปแบบ

ชื่ออาร์เรย์.unshift(ข้อมูล_1, ข้อมูล_2, ..., ข้อมูล_n);

ตัวอย่างที่ 28

```
1  let language = ["HTML", "CSS", "JavaScript"];
2  console.log("ก่อนลบ:", language);
   // ลบข้อมูลต้นอาร์เรย์
3  let removedItem = language.shift();
4  console.log("ข้อมูลที่ถูกลบ:", removedItem);
5  console.log("หลังลบ:", language);
6  console.log("ความยาวใหม่:", language.length);
   // ลบต่อเนื่องหลายครั้ง
7  let removed1 = language.shift();
8  let removed2 = language.shift();
9  console.log("ลบครั้งที่ 2:", removed1);
10 console.log("ลบครั้งที่ 3:", removed2);
11 console.log("อาร์เรย์ที่เหลือ:", language);
```

จากตัวอย่างที่ 28 ในบรรทัดที่ 1 เป็นการประกาศอาร์เรย์ language ที่มีข้อมูลเริ่มต้น 3 รายการ บรรทัดที่ 3 ใช้เมธอด shift() ลบข้อมูลรายการแรกซึ่งเป็นข้อความคำว่า *HTML* โดยมีการประกาศตัวแปรชื่อ removedItem เพื่อเก็บค่าที่ถูกลบ และในบรรทัดที่ 7-8 เป็นการลบข้อมูลในอาร์เรย์จำนวน 2 ครั้งต่อเนื่องกัน โดยแต่ละครั้งจะลบรายการแรกออกไปจากอาร์เรย์ การลบข้อมูลนี้จะเกิดขึ้นกับอาร์เรย์ language ทุกครั้ง เช่นเดียวกับเมธอดอื่นๆ แต่การคืนค่าจะเหมือนกับเมธอด pop() ที่คืนค่าเป็นข้อมูลที่ถูกลบ หลังจากที่ดำเนินการตามคำสั่งครบทุกคำสั่งแล้วอาร์เรย์ language จะเป็นอาร์เรย์ว่าง ซึ่งไม่มีค่าใดๆ อยู่ในอาร์เรย์

3.4.2 คุณสมบัติ length

คุณสมบัติ length เป็นคุณสมบัติพื้นฐานของอาร์เรย์ที่ใช้ในการทราบจำนวนข้อมูลในอาร์เรย์ มีความสำคัญในการตรวจสอบข้อมูล การวนซ้ำ และการจัดการอาร์เรย์ในรูปแบบต่าง ๆ โดยคุณสมบัติ length จะคืนค่าเป็นตัวเลขที่แสดงจำนวนข้อมูลในอาร์เรย์ การใช้งานไม่ต้องใส่เครื่องหมายวงเล็บเหมือนเมธอดเนื่องจากไม่ใช่ฟังก์ชัน

รูปแบบ

ชื่ออาร์เรย์.length

ตัวอย่างที่ 29

```
1   let language = ["HTML", "CSS", "JavaScript"];
2   console.log("อาร์เรย์:", language);
3   console.log("จำนวนข้อมูล:", language.length);
    // การตรวจสอบอาร์เรย์ว่างเปล่า
4   let empty = [ ];
5   console.log("อาร์เรย์ว่าง:", empty);
6   console.log("จำนวนข้อมูลในอาร์เรย์ว่าง:", empty.length);
    // การใช้ในเงื่อนไข
7   if (language.length > 0) {
8       console.log("อาร์เรย์มีข้อมูล", language.length, "รายการ");
9   } else {
10      console.log("อาร์เรย์ว่างเปล่า");
11  }
    // การใช้ใน for loop
12  console.log("แสดงข้อมูลทั้งหมด:");
13  for (let i = 0; i < language.length; i++) {
14      console.log("ตำแหน่งที่", i, ":", language[i]);
15  }
```

จากตัวอย่างที่ 29 ในบรรทัดที่ 1-3 แสดงการใช้ length กับอาร์เรย์ language ที่มีข้อมูล 3 รายการ โดยจะทำการแสดงข้อมูลในอาร์เรย์และจำนวนข้อมูลในอาร์เรย์ language ที่ได้จากคุณสมบัติ length ข้อมูลที่แสดงจะเป็น ['HTML', 'CSS', 'JavaScript'] และจำนวนของข้อมูลในอาร์เรย์จะเป็น 3

บรรทัดที่ 4-6 แสดงการใช้กับอาร์เรย์ว่างเปล่า โดยเริ่มจากการสร้างอาร์เรย์ว่างเปล่าชื่อ empty จากนั้นทำการแสดงข้อมูลในอาร์เรย์ซึ่งจะได้เป็น [ ] และแสดงจำนวนข้อมูลในอาร์เรย์ empty ที่ได้จากคุณสมบัติ length ซึ่งมีค่าเป็น 0

บรรทัดที่ 7-11 แสดงการใช้คุณสมบัติ length ของอาร์เรย์ language ในเงื่อนไข if เพื่อตรวจสอบว่าอาร์เรย์มีข้อมูลหรือไม่ หากค่าจากคุณสมบัติ length มีค่ามากกว่า 0 จะมีการแสดงข้อความพร้อมจำนวนข้อมูลที่ได้จากคุณสมบัติ length อาร์เรย์ language เช่นกัน แต่ถ้ามีค่าเป็น 0 หรือน้อยกว่า จะแสดงข้อความว่า *อาร์เรย์ว่างเปล่า* ซึ่งในกรณีตัวอย่างจะแสดงผลลัพธ์เป็น *อาร์เรย์มีข้อมูล 3 รายการ*

บรรทัดที่ 12-15 แสดงการใช้คุณสมบัติ length ของอาร์เรย์ language เพื่อเป็นเงื่อนไขในการตรวจสอบการวนซ้ำในลูป for เพื่อนำค่าในอาร์เรย์ตำแหน่งดัชนีต่าง ๆ ตั้งแต่ 0 ถึง 2 มาแสดง

4. การจัดการเอกสารเว็บด้วย JavaScript

จากหลักการพื้นฐานของ JavaScript ในหัวข้อก่อนหน้านี้ เป็นพื้นฐานสำคัญของ ที่จะนำมาประยุกต์ใช้ร่วมกับภาษา HTML เพื่อเพิ่มความสามารถการทำงานของส่วนหน้าของเว็บแอปพลิเคชัน ทั้งในส่วนของการตอบสนองต่อการกระทำของผู้ใช้ การเปลี่ยนแปลงเนื้อหาของหน้าเว็บแบบ การตรวจสอบความถูกต้องของข้อมูลในฟอร์ม และสร้างประสบการณ์การใช้งานที่มีปฏิสัมพันธ์มากขึ้น

4.1 รูปแบบการใช้งานภาษา JavaScript ในเอกสารเว็บ

ภาษา JavaScript สามารถนำมาใช้งานในเอกสารเว็บได้หลากหลายรูปแบบ โดยแต่ละรูปแบบมีความเหมาะสมในการใช้งานที่ไม่เหมือนกัน การเข้าใจรูปแบบการใช้งานเหล่านี้จะช่วยให้ผู้พัฒนาสามารถเลือกวิธีการที่เหมาะสมกับสถานการณ์ต่าง ๆ ได้ โดยรูปแบบการใช้งาน JavaScript ในเอกสารเว็บสามารถแบ่งออกเป็น 3 วิธี ได้แก่ การใช้แบบแทรกในแท็ก การใช้แบบภายในและการใช้แบบภายนอก

4.1.1 การใช้ภาษา JavaScript แบบแทรกในแท็ก

การใช้ภาษา JavaScript แบบแทรกในแท็ก (Inline JavaScript) เป็นวิธีการเขียนชุดคำสั่งภาษา JavaScript ลงในแอตทริบิวต์ของแท็ก HTML โดยตรง วิธีการนี้เหมาะสำหรับคำสั่งที่สั้นและเฉพาะเจาะจง โดยทั่วไปจะใช้กับการจัดการเหตุการณ์ เช่น การคลิก การส่งแบบฟอร์ม หรือการเปลี่ยนแปลงข้อมูล รูปแบบการใช้งานลักษณะนี้มีดังนี้

รูปแบบ

<แท็ก แอตทริบิวต์เหตุการณ์="คำสั่ง JavaScript">

จากรูปแบบการใช้งานข้างต้น การใช้งานภาษา JavaScript แบบแทรกในแท็กจะเขียนเป็นแอตทริบิวต์ของแท็กที่ต้องการให้ชุดคำสั่งของภาษา JavaScript ทำงาน โดยแอตทริบิวต์ที่ใช้จะเป็นแอตทริบิวต์ที่ตอบสนองต่อเหตุการณ์ในลักษณะต่าง ๆ ที่เกิดขึ้นกับแท็กนั้น (Event Attribute) เช่น เหตุการณ์การคลิก การพิมพ์ หรือการเปลี่ยนแปลงข้อมูล เป็นต้น

แอตทริบิวต์เหตุการณ์ (Event Attributes) เป็นแอตทริบิวต์พิเศษในภาษา HTML ที่ใช้สำหรับกำหนดคำสั่งภาษา JavaScript ให้ทำงานเมื่อเกิดเหตุการณ์ใดเหตุการณ์หนึ่งขึ้น แอตทริบิวต์เหล่านี้ช่วยให้เว็บแอปพลิเคชันสามารถตอบสนองต่อการกระทำของผู้ใช้ได้อย่างเหมาะสม แอตทริบิวต์เหตุการณ์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชันแสดงดังตารางที่ 10-1

ตารางที่ 10-1 แอตทริบิวต์เหตุการณ์ของภาษา HTML

แอตทริบิวต์	เหตุการณ์ที่เกิดขึ้น	อิลิเมนต์ที่ใช้ได้	ตัวอย่างการใช้
onclick	เมื่อมีการคลิกเมาส์	ปุ่ม, ลิงก์, ภาพ, ข้อความ	การส่งแบบฟอร์ม, เปิดหน้าต่าง
onchange	เมื่อมีการเปลี่ยนแปลงค่า	ช่องป้อนข้อมูล, รายการเลือก	ตรวจสอบข้อมูล, คำนวณผลลัพธ์
onkeyup	เมื่อปล่อยปุ่มคีย์บอร์ด	ช่องป้อนข้อมูล	ค้นหาแบบเรียลไทม์, แปลงข้อความ
onmouseover	เมื่อเมาส์เลื่อนผ่าน	ปุ่ม, ภาพ, ข้อความ, ลิงก์	แสดงคำแนะนำ, เปลี่ยนสี
onmouseout	เมื่อเมาส์เลื่อนออก	ปุ่ม, ภาพ, ข้อความ, ลิงก์	ซ่อนคำแนะนำ, คืนสีเดิม
onsubmit	เมื่อส่งแบบฟอร์ม	แบบฟอร์ม	ตรวจสอบข้อมูลก่อนส่ง
onload	เมื่อโหลดอิลิเมนต์เสร็จ	หน้าเว็บ, ภาพ	เริ่มต้นการทำงาน, แสดงข้อความ

การเลือกใช้แอตทริบิวต์เหตุการณ์ขึ้นอยู่กับลักษณะการโต้ตอบที่ต้องการ และประเภทของอิลิเมนต์ โดยควรพิจารณาความเหมาะสมของเหตุการณ์กับพฤติกรรมของผู้ใช้ เช่น การใช้ onclick สำหรับปุ่มหรือการใช้ onchange สำหรับช่องป้อนข้อมูล เป็นต้น

ค่าของแอตทริบิวต์เหตุการณ์จะเป็นชุดคำสั่งของ JavaScript ที่ต้องการให้ดำเนินการเมื่อเหตุการณ์นั้นเกิดขึ้น ซึ่งรูปแบบการเขียนชุดคำสั่งภาษา JavaScript ก็จะเป็นไปตามข้อกำหนดและอิลิเมนต์ของการเขียนภาษา JavaScript ดังที่ได้กล่าวถึงในบทที่ 9 ทั้งนี้การเขียนประโยคคำสั่ง ภาษา JavaScript ในแท็กเป็น

ลักษณะการเขียนคำสั่งในบรรทัดเดียวเรียงต่อกัน จึงต้องใช้เครื่องหมาย ; ปิดท้ายประโยคคำสั่ง ตัวอย่างของการใช้ภาษา JavaScript แบบแทรกในแท็กมีดังนี้

ตัวอย่างที่ 30

```
1  <h1>ตัวอย่างการใช้ JavaScript แบบ Inline</h1>
2  <button onclick="var message = 'สวัสดีครับ!'; console.log(message);">
3      คลิกเพื่อแสดงข้อความในคอนโซล
4  </button>
5  <button onclick="var x = 5; var y = 3; console.log('ผลบวก: ' + (x + y));">
6      คำนวณเลข
7  </button>
8  <button onclick="for(var i = 1; i <= 3; i++) { console.log('ครั้งที่ ' + i); }">
9      ทำซ้ำ 3 ครั้ง
10 </button>
```

ตัวอย่างที่ 30 เป็นเอกสารเว็บที่เขียนขึ้นตามโครงสร้างมาตรฐานภาษา HTML ดังที่ได้กล่าวถึงในบทที่ 5 โดยแสดงเฉพาะส่วนเนื้อหาของอิลิเมนต์ body ซึ่งประกอบด้วยอิลิเมนต์หัวเรื่องระดับ 1 และปุ่มอีก 3 ปุ่ม แต่ละปุ่มจะมีข้อความบนปุ่มเป็น *คลิกเพื่อแสดงข้อความในคอนโซล คำนวณเลข* และ *ทำซ้ำ 3 ครั้ง* ตามลำดับ ทั้งนี้ แต่ละปุ่มได้มีการกำหนดแอตทริบิวต์เหตุการณ์ onclick เพื่อเป็นตัวกำหนดว่าคำสั่งภาษา JavaScript จะถูกดำเนินการเมื่อมีการคลิกที่อิลิเมนต์นั้นๆ ดังแสดงในบรรทัดที่ 2, 5 และ 8 ตามลำดับ โดยรายละเอียดคำสั่ง JavaScript ที่จะทำงานในแต่ละปุ่มมีดังนี้

- ปุ่ม คลิกเพื่อแสดงข้อความในคอนโซล มีการประกาศตัวแปรชื่อ message ที่เก็บข้อความคำว่า สวัสดีครับ! และใช้คำสั่ง console.log() เพื่อแสดงค่าของตัวแปร ดังกล่าวในคอนโซลของเว็บเบราว์เซอร์
- ปุ่ม คำนวณเลข มีการประกาศตัวแปร x และ y พร้อมกำหนดค่าเป็นตัวเลข 5 และ 3 ตามลำดับ จากนั้นแสดงผลบวกของตัวแปรทั้งสองในคอนโซลของเว็บเบราว์เซอร์
- ปุ่ม ทำซ้ำ 3 ครั้ง มีการใช้คำสั่งวนซ้ำ for เพื่อวนซ้ำจำนวน 3 ครั้ง โดยกำหนดตัวแปรนับจำนวนให้มีค่าตั้งแต่ 1-3 โดยจะแสดงข้อความคำว่า ครั้งที่ ตามด้วยค่าของตัวแปนับจำนวนดังกล่าวในคอนโซลของเว็บเบราว์เซอร์

การใช้ภาษา JavaScript แบบเขียนในแท็กมีข้อดีคือใช้งานง่ายและรวดเร็ว เหมาะสำหรับคำสั่งที่ไม่ซับซ้อน อย่างไรก็ตาม ข้อเสียคือทำให้รหัสภาษา HTML ไม่เป็นระเบียบยากต่อการบำรุงรักษา และไม่สามารถนำไปใช้ซ้ำในหน้าอื่นได้

4.1.2 การใช้ภาษา JavaScript แบบภายใน

การใช้ภาษา JavaScript แบบภายใน (Internal JavaScript) เป็นวิธีการเขียนชุดเขียนคำสั่งภาษา JavaScript รวมไว้ในไฟล์เอกสารเว็บ โดยเขียนคำสั่งทั้งหมดไว้ภายในอิลิเมนต์ script ซึ่งจะอยู่ภายในอิลิเมนต์ head หรือ body ก็ได้ แต่โดยทั่วไป จะวางไว้ก่อนแท็กปิด </body> เพื่อให้เว็บเบราว์เซอร์สร้างโครงสร้าง DOM ของเอกสารเว็บให้เสร็จสมบูรณ์ก่อนที่ JavaScript จะเริ่มทำงาน ซึ่งช่วยลดปัญหาการเข้าถึงอิลิเมนต์ที่ยังไม่ถูกโหลด โดยมีรูปแบบและตัวอย่างดังนี้

รูปแบบ

```
<script>
    คำสั่งภาษา JavaScript;
</script>
```

ตัวอย่างที่ 31

```
1  <!DOCTYPE html>
2  <html lang="th">
3  <head>
4  <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>การใช้ JavaScript แบบภายใน</title>
7      <script>
8          console.log("อิลิเมนต์ยังไม่ได้โหลด");
9          console.log(document.getElementById("content").innerHTML);
10         for (var i = 1; i <= 3; i++) {
11             console.log("การทำงาน JavaScript แบบภายใน ครั้งที่ " + i);
12         }
13     </script>
14 </head>
15 <body>
16     <h1>ตัวอย่างการใช้ JavaScript แบบภายใน</h1>
17     <p  id="content">เนื้อหาของอิลิเมนต์ p</p>
18     <button onclick="greeting()" id="age">ทักทาย</button>
19     <script>
20         console.log("อิลิเมนต์โหลดเสร็จแล้ว");
```

```
21              console.log(document.getElementById("content").innerHTML);
22              function greeting() {
23                      console.log("สวัสดี Programmer");
24              }
25          </script>
26  </body>
27  </html>
```

จากตัวอย่างที่ 31 แสดงการใช้ JavaScript แบบภายในที่แยกชุดคำสั่งภาษา JavaScript ออกจากแท็กและสร้างเป็นฟังก์ชันเพื่อเรียกใช้ผ่านเหตุการณ์ที่เกิดขึ้นกับอิลิเมนต์ นอกจากนี้ยังแสดงให้เห็นผลกระทบที่เกิดขึ้นจากตำแหน่งการวางอิลิเมนต์ script ต่อการประมวลผลของเว็บเบราว์เซอร์ โดยในตัวอย่างมีการอิลิเมนต์ script ไว้ใน 2 ตำแหน่งคือ ภายในอิลิเมนต์ head (บรรทัดที่ 7-13) และภายในอิลิเมนต์ body ก่อนแท็กปิด </body> (บรรทัดที่ 19-25) โดยมีกลไกการทำงานมีรายละเอียดดังนี้

- เว็บเบราว์เซอร์จะทำการอ่านรหัสในเอกสารเว็บจากบนลงล่างจากซ้ายไปขวาตามกลไกการทำงานของเว็บเบราว์เซอร์ที่ได้กล่าวถึงในบทที่ 1 เมื่อพบส่วนที่เป็นภาษา JavaScript ในบรรทัดที่ 7-13 ก็จะทำการส่งให้ JavaScript Engine ประมวลผล ซึ่งคำสั่งในบรรทัดที่ 8 จะแสดงข้อความว่า อิลิเมนต์ยังไม่ได้โหลด เพื่อชี้ให้เห็นว่าเว็บเบราว์เซอร์ยังไม่ได้ประมวลผลถึงส่วนที่อยู่ภายในอิลิเมนต์ body
- คำสั่งในบรรทัดที่ 9 เป็นการแสดงเนื้อหาที่อยู่ในอิลิเมนต์ที่มี id="content" ซึ่งขณะนั้นยังไม่มีอิลิเมนต์ดังกล่าวในโครงสร้าง DOM เนื่องจากเว็บเบราว์เซอร์ยังไม่ได้ประมวลผลถึงอิลิเมนต์นั้น จึงทำให้แสดงข้อผิดพลาด TypeError ในส่วนคอนโซลของเว็บเบราว์เซอร์ ซึ่งข้อผิดพลาดที่เกิดขึ้นไม่ได้หยุดการทำงานของ JavaScript Engine จึงทำการอ่านและประมวลผลในคำสั่งบรรทัดที่ 10-12 ต่อไปโดยเป็นการวนรอบแสดงข้อความจำนวน 3 รอบ ซึ่งการทำงานของคำสั่งในส่วนนี้ไม่พบข้อผิดพลาดใดๆ เนื่องจากไม่ได้อ้างถึงอิลิเมนต์ของเอกสารเว็บ
- คำสั่งภาษา JavaScript ในตำแหน่งที่สอง (บรรทัดที่ 19-25) จะทำงานหลังจากที่อิลิเมนต์ทั้งหมดในเอกสารได้ถูกสร้างขึ้นในโครงสร้าง DOM แล้ว ดังนั้น การเข้าถึงอิลิเมนต์ที่มี id="content" ในบรรทัดที่ 21 จึงดำเนินการได้สำเร็จ เนื่องจากอิลิเมนต์ดังกล่าวได้ถูกสร้างขึ้นแล้วในบรรทัดที่ 17 โดยแสดงเป็นข้อความคำว่า เนื้อหาของอิลิเมนต์ p นอกจากนี้ ยังมีการสร้างฟังก์ชัน greeting() (บรรทัดที่ 22-24) ซึ่งจะถูกเรียกเมื่อมีการคลิกปุ่ม ทักทาย ซึ่งในแท็กของปุ่ม (บรรทัดที่ 18) ได้กำหนดแอตทริบิวต์เหตุการณ์ onclick โดยให้เรียกใช้ฟังก์ชัน greeting เมื่อเกิดการคลิกที่ปุ่ม การแยกคำสั่งออกจากแท็กและจัดกลุ่มคำสั่งในรูปแบบฟังก์ชันช่วยให้เอกสารเว็บมีความเป็นระเบียบมากขึ้นและสามารถนำไปใช้ซ้ำในอิลิเมนต์อื่นๆ ที่อยู่ในเอกสารเว็บเดียวกันได้

จากตัวอย่างจะพบว่าการใช้ JavaScript แบบภายในมีข้อดีคือจัดระเบียบได้ง่ายกว่าแบบแทรกลงในแท็ก และสามารถสร้างฟังก์ชันที่ซับซ้อนได้ อย่างไรก็ตาม ยังมีส่วนของภาษา JavaScript ที่ทำให้โครงสร้างภาษา HTML ดูไม่เป็นระเบียบอยู่บ้างและไม่สามารถนำชุดคำสั่งไปใช้ซ้ำในหน้าเว็บอื่นได้

4.1.3 การใช้ภาษา JavaScript แบบภายนอก

การใช้ภาษา JavaScript แบบภายนอก (External JavaScript) เป็นวิธีการเขียนชุดคำสั่งภาษา JavaScript โดยแยกส่วนคำสั่ง JavaScript ออกจากโครงสร้างเอกสารเว็บ และจัดเก็บไว้ในไฟล์แยกต่างหากที่มีนามสกุลเป็น .js ส่งผลให้เอกสารเว็บมีความเป็นระเบียบ ง่ายต่อการดูแลรักษาและสามารถนำชุดคำสั่งไปใช้ซ้ำในหน้าเว็บหลายหน้าได้ วิธีการนี้เหมาะสมสำหรับการพัฒนาเว็บแอปพลิเคชันที่มีขนาดใหญ่หรือมีหลายหน้าที่ใช้ชุดคำสั่ง JavaScript เดียวกัน ซึ่งสอดคล้องกับแนวคิดการแยกความรับผิดชอบเช่นเดียวกับการใช้ CSS แบบภายนอก การใช้ภาษา JavaScript แบบภายนอกมีขั้นตอนการดำเนินการดังนี้

ขั้นตอนที่ 1) สร้างไฟล์ใหม่ที่มีนามสกุล .js เช่น script.js เพื่อบันทึกชุดคำสั่ง JavaScript ไว้ในไฟล์ที่สร้างใหม่นี้ โดยภายในไฟล์ JavaScript ไม่ต้องมีแท็ก <script> หรือแท็กของภาษา HTML อื่นใด

ขั้นตอนที่ 2) เขียนชุดคำสั่งภาษา JavaScript ตามอิลิเมนต์ของภาษาในไฟล์ที่ได้สร้างไว้ในขั้นตอนก่อนหน้านี้

ขั้นตอนที่ 3) เชื่อมโยงไฟล์กับเอกสารเว็บ โดยในเอกสารเว็บที่ต้องการใช้งานให้ใช้อิลิเมนต์ script พร้อมกับแอตทริบิวต์ src เพื่อระบุตำแหน่งของไฟล์ JavaScript ที่สร้างขึ้น ทั้งนี้ ตำแหน่งของการอิลิเมนต์ script ส่งผลต่อลำดับการประมวลผลดังที่ได้กล่าวถึงมาแล้วการใช้งานภาษา JavaScript แบบภายใน หากเป็นฟังก์ชันที่ไม่ได้มีการอ้างอิงอิลิเมนต์ของเอกสารเว็บสามารถวางไว้ในส่วนหัวของเอกสารได้ แต่หากเป็นคำสั่งที่ต้องอ้างอิงถึงอิลิเมนต์ในเอกสารเว็บควรวางไว้ก่อนแท็กปิด </body> เพื่อให้อิลิเมนต์ทั้งหมดถูกสร้างในโครงสร้าง DOM ก่อน รูปแบบการเชื่อมโยงไฟล์กับเอกสารเว็บมีดังนี้

รูปแบบ

```
<script src="ชื่อไฟล์.js"></script>
```

นอกจากข้อดีของการใช้ภาษา JavaScript แบบภายนอกที่ได้กล่าวไว้ข้างต้น ยังมีข้อควรพิจารณาเกี่ยวกับเวลาในการโหลดไฟล์ กล่าวคือ เมื่อเว็บเบราว์เซอร์โหลดเอกสารเว็บ จะต้องโหลดไฟล์องค์ประกอบอื่น ๆ ร่วมด้วย เช่น ไฟล์รูปภาพ ไฟล์ CSS และไฟล์ JavaScript ภายนอกที่เชื่อมโยงเข้ามา ซึ่งอาจส่งผลให้เวลาในการโหลดหน้าเว็บโดยรวมเพิ่มขึ้นเล็กน้อย โดยเฉพาะหากมีการเชื่อมโยงไฟล์ JavaScript ภายนอกหลายไฟล์หรือมีขนาดใหญ่ ดังนั้น ผู้พัฒนาควรพิจารณาความเหมาะสมระหว่างการจัดโครงสร้างรหัสคำสั่งให้มีความเป็นระบบ กับประสิทธิภาพในการโหลดหน้าเว็บให้สมดุลกัน

4.2 กล่องโต้ตอบ

การใช้กล่องโต้ตอบ (Dialog Box) เป็นวิธีการพื้นฐานในการสื่อสารระหว่างเว็บแอปพลิเคชันกับผู้ใช้ ภาษา JavaScript มีกล่องโต้ตอบที่พร้อมใช้งานที่ช่วยให้สามารถแสดงการแจ้งเตือน การรับข้อมูลจากผู้ใช้ และขอการยืนยันในการดำเนินการ โดยกล่องโต้ตอบเหล่านี้จะปรากฏเป็นหน้าต่างแยกต่างหากจากหน้าเว็บหลัก และผู้ใช้จำเป็นต้องตอบสนองต่อกล่องโต้ตอบก่อนจึงจะสามารถใช้งานส่วนอื่นของหน้าเว็บได้

4.2.1 ฟังก์ชัน alert()

ฟังก์ชัน alert() เป็นฟังก์ชันพื้นฐานที่ใช้แสดงกล่องโต้ตอบแบบข้อความเดียว (Single Message Box) เพื่อแจ้งเตือนหรือให้ข้อมูลแก่ผู้ใช้ โดยกล่องโต้ตอบจะมีปุ่ม ตกลง หรือ OK เพียงปุ่มเดียว เมื่อผู้ใช้คลิกปุ่มดังกล่าวกล่องโต้ตอบจะปิดลง และการทำงานของโปรแกรมจะดำเนินต่อไป ฟังก์ชันนี้เหมาะสำหรับกรณีที่ต้องการแจ้งผลลัพธ์ที่ไม่ต้องการให้ผู้ใช้โต้ตอบกลับ เช่น การแจ้งว่าบันทึกข้อมูลสำเร็จ หรือแสดงข้อผิดพลาดจากการกรอกข้อมูลผิดพลาด เป็นต้น รูปแบบ ตัวอย่างและผลลัพธ์ของการใช้งานของฟังก์ชัน alert( ) มีดังนี้

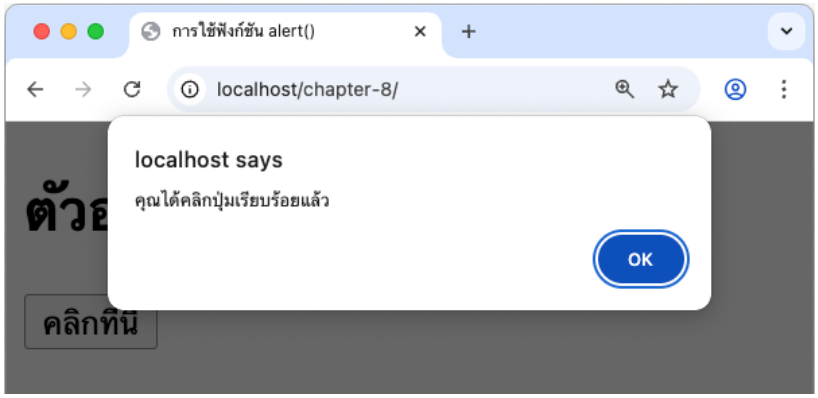
รูปแบบ

alert(ข้อความที่ต้องการแสดง);

ตัวอย่างที่ 32

```
1  //รหัสคำสั่งต่อไปนี้อยู่ในอิลิเมนต์ body ตามโครงสร้างมาตรฐานของภาษา HTML
2  <h2>ตัวอย่างการใช้ alert()</h2>
3  <button onclick="alert('คุณได้คลิกปุ่มเรียบร้อยแล้ว')">คลิกที่นี่</button>
```

ผลลัพธ์ของตัวอย่างที่ 32



จากตัวอย่างที่ 32 เป็นการสร้างเอกสารเว็บที่ประกอบด้วยปุ่มหนึ่งปุ่ม และเมื่อผู้ใช้คลิกปุ่ม จะเกิดเหตุการณ์ onclick ซึ่งเรียกใช้ฟังก์ชัน alert() โดยแสดงข้อความ “คุณได้คลิกปุ่มเรียบร้อยแล้ว” ในกล่อง

โต้ตอบแบบแจ้งเตือน ผู้ใช้จะต้องคลิกปุ่ม “ตกลง” เพื่อปิดกล่องดังกล่าว ก่อนจะสามารถโต้ตอบกับอิลิเมนต์อื่นในหน้าเว็บได้

4.2.2 ฟังก์ชัน prompt()

ฟังก์ชัน prompt() ใช้สำหรับแสดงกล่องโต้ตอบเพื่อรับข้อมูลจากผู้ใช้ โดยจะแสดงข้อความคำถามและช่องกรอกข้อมูลภายในหน้าต่างแยกต่างหาก ผู้ใช้สามารถพิมพ์ข้อมูลลงไปในช่องกรอก และกดปุ่มตกลง (OK) หรือยกเลิก (Cancel) ได้ หากผู้ใช้กดตกลง ค่าที่กรอกจะถูกส่งกลับมาให้โปรแกรมในรูปแบบข้อมูลชนิดข้อความ หากกดยกเลิก หรือปิดหน้าต่างโดยไม่กรอกข้อมูล จะได้ค่ากลับเป็น null

ลักษณะสำคัญของฟังก์ชัน prompt() คือข้อมูลที่ได้รับจะอยู่ในรูปแบบข้อความเสมอ แม้ว่าผู้ใช้จะพิมพ์ตัวเลข เช่น 123 ก็จะถือว่าเป็นข้อความ หากต้องการนำไปคำนวณทางคณิตศาสตร์ควรแปลงค่าโดยใช้ฟังก์ชัน Number() หรือ parseInt() ก่อน ฟังก์ชัน prompt() รับพารามิเตอร์สองตัว คือข้อความคำถามที่แสดงให้ผู้ใช้เห็น และค่าเริ่มต้นที่แสดงในช่องกรอกข้อมูล

รูปแบบ

```
let = prompt("ข้อความคำถาม", "ค่าที่เริ่มต้นแสดงในช่องกรอกข้อมูล");
```

ตัวอย่างที่ 33

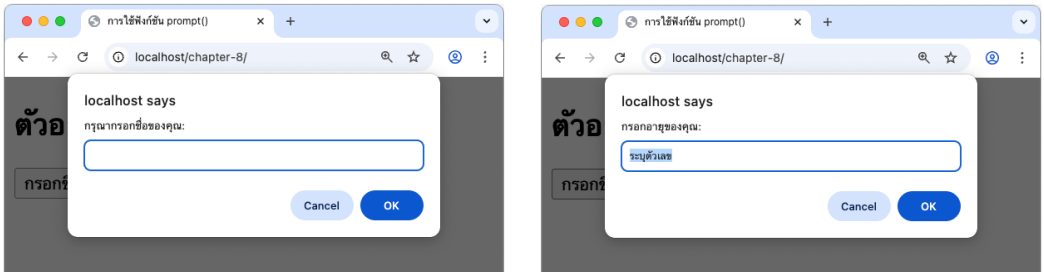
HTML

```
<h2>ตัวอย่างการใช้ prompt()</h2>
<button onclick="greeting();">กรอกชื่อของคุณ</button>
<button onclick="birthYear();">กรอกอายุของคุณ</button>
<script  src="dialog.js">
```

JavaScript: dialog.js

```
function greeting() {
    let name = prompt('กรุณากรอกชื่อของคุณ:', '');
    console.log('ชื่อของคุณคือ: ' + name);
}
function birthYear() {
    let year = prompt('กรอกอายุของคุณ:', 'ระบุตัวเลข');
    year = 2568-parseInt(year);
    console.log('ปีพ.ศ.เกิดของคุณคือ: ' + year);
}
```

ผลลัพธ์ของตัวอย่างที่ 33



จากตัวอย่างที่ 33 แสดงการใช้งานฟังก์ชัน prompt() ในสองรูปแบบ และใช้ JavaScript แบบภายนอกโดยเขียนไว้ในไฟล์ชื่อ dialog.js

ฟังก์ชัน greeting() ในบรรทัดที่ 1-4 ใช้ฟังก์ชัน prompt() เพื่อรับชื่อจากผู้ใช้ และแสดงผลลัพธ์ในคอนโซล ส่วนฟังก์ชัน birthYear() ในบรรทัด 5-9 ใช้ฟังก์ชัน prompt() เพื่อรับอายุจากผู้ใช้ จากนั้นแปลงข้อความที่ได้รับเป็นตัวเลขด้วยฟังก์ชัน parseInt() และคำนวณปีเกิดโดยใช้ปี พ.ศ. 2568 หักลบด้วยอายุที่ได้รับ การใช้งานนี้แสดงให้เห็นความสำคัญของการแปลงชนิดข้อมูลเมื่อต้องการนำข้อมูลจาก prompt() ไปใช้ในการคำนวณ

4.2.3 ฟังก์ชัน confirm()

ฟังก์ชัน confirm() เป็นฟังก์ชันพื้นฐานของกล่องโต้ตอบที่ใช้สำหรับขอการยืนยันจากผู้ใช้ โดยแสดงกล่องโต้ตอบที่มีข้อความคำถามพร้อมปุ่มสองปุ่มคือ ตกลง (OK) และ ยกเลิก (Cancel) ทำให้ผู้ใช้สามารถเลือกยืนยันหรือปฏิเสธการดำเนินการได้ ฟังก์ชัน confirm() จะคืนค่าผลลัพธ์เป็นชนิดข้อมูล Boolean โดยคืนค่า true เมื่อผู้ใช้คลิกปุ่ม ตกลง และคืนค่า false เมื่อผู้ใช้คลิกปุ่ม ยกเลิก หรือกดปุ่ม Esc เพื่อปิดกล่องโต้ตอบ

ลักษณะการทำงานนี้ทำให้ฟังก์ชัน confirm() เหมาะสำหรับการตรวจสอบการตัดสินใจของผู้ใช้ก่อนดำเนินการที่สำคัญ เช่น การลบข้อมูล การออกจากระบบ การยกเลิกการดำเนินการ หรือการแก้ไขข้อมูลที่มีผลกระทบสูง เป็นต้น การใช้ฟังก์ชัน confirm() ช่วยสร้างความมั่นใจให้กับผู้ใช้และลดความเสี่ยงจากการดำเนินการที่ไม่ได้ตั้งใจ

รูปแบบ

```
let ตัวแปร = confirm("ข้อความที่ต้องการให้ผู้ใช้ยืนยัน");
```

ตัวอย่างที่ 34

HTML

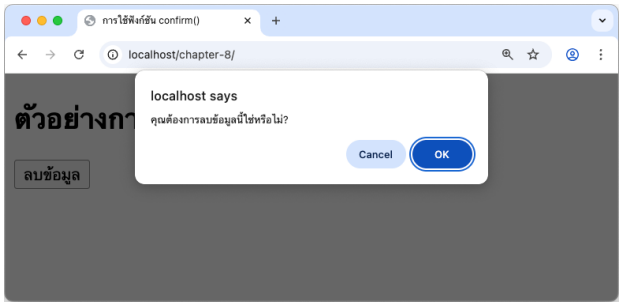
```
1  <h2>ตัวอย่างการใช้ confirm()</h2>
2  <button onclick="deleteData();">ลบข้อมูล</button>
```

```
3   <button onclick="logout();">ออกจากระบบ</button>
4   <script  src="dialog.js">
```

JavaScript: dialog.js

```
1   function deleteData() {
2       var deleteConfirm = confirm("คุณต้องการลบข้อมูลนี้ใช่หรือไม่?");
3       if (deleteConfirm) {
4           console.log("ดำเนินการลบข้อมูลเรียบร้อย");
5       } else {
6       console.log("ยกเลิกการลบข้อมูล");
7       }
8   }
9   function logout() {
10      var logoutConfirm = confirm("คุณต้องการออกจากระบบใช่หรือไม่?");
11      if (logoutConfirm) {
12          console.log("ออกจากระบบเรียบร้อย");
13      }
14  }
```

ผลลัพธ์ของตัวอย่างที่ 34



จากตัวอย่างที่ 34 แสดงการใช้ฟังก์ชัน confirm() ในสองสถานการณ์ที่แตกต่างกัน โดยแยกส่วนภาษา JavaScript ออกเป็นไฟล์แยกชื่อ dialog.js

ฟังก์ชัน deleteData() ในบรรทัดที่ 1-8 ใช้ฟังก์ชัน confirm() เพื่อขอยืนยันการลบข้อมูล โดยจัดเก็บค่าผลลัพธ์ไว้ในตัวแปร deleteConfirm จากนั้นใช้โครงสร้างการตัดสินใจแบบ if-else เพื่อแสดงข้อความที่แตกต่างกันตามการตัดสินใจของผู้ใช้ ส่วนฟังก์ชัน logout() ในบรรทัดที่ 9-14 ใช้ฟังก์ชัน confirm() เพื่อยืนยัน

การออกจากระบบ โดยดำเนินการต่อเฉพาะเมื่อผู้ใช้คลิก *ตกลง* เท่านั้น การใช้งานนี้แสดงให้เห็นความยืดหยุ่นของฟังก์ชัน confirm() ในการสร้างการโต้ตอบที่เหมาะสมกับบริบทของแต่ละการทำงาน

4.3 การจัดการ DOM

DOM (Document Object Model) เป็นโครงสร้างข้อมูลที่แทนเอกสาร HTML ในรูปแบบของต้นไม้ เว็บเบราว์เซอร์สร้างขึ้นเมื่อโหลดหน้าเว็บ โดย DOM ทำหน้าที่เป็นสะพานเชื่อมระหว่างเอกสาร HTML กับภาษา JavaScript ทำให้สามารถเข้าถึง แก้ไข หรือจัดการอิลิเมนต์ในหน้าเว็บได้แบบไดนามิก ดังที่ได้กล่าวที่ในบทที่ 1 โดยในโครงสร้าง DOM แต่ละแท็ก HTML จะถูกแทนด้วยโหนด (Node) ที่มีความสัมพันธ์แบบพ่อ-ลูก (Parent-Child) และพี่น้อง (Sibling) กัน โดยอิลิเมนต์ html เป็น Root Node หรือโหนดราก ส่วนอิลิเมนต์ อื่นๆ เช่น head, body, div และ p เป็น จะเป็นโหนดลูกและโหนดหลาน ตามลำดับ การจัดโครงสร้างแบบนี้ทำให้ JavaScript สามารถนำทางไปยังอิลิเมนต์ใดก็ได้ในเอกสารได้อย่างเป็นระบบ

4.3.1 การเข้าถึงอิลิเมนต์ในเอกสาร

การเข้าถึงและจัดการอิลิเมนต์ในเอกสารเว็บด้วย JavaScript ถือเป็นกลไกพื้นฐานสำคัญที่ช่วยให้สามารถโต้ตอบและปรับเปลี่ยนข้อมูลภายในเอกสารเว็บแบบพลวัต เช่น การเปลี่ยนข้อความ การเปลี่ยนสี การซ่อนหรือแสดงอิลิเมนต์บางอย่าง หรือแม้แต่การเพิ่มอิลิเมนต์ใหม่ เป็นต้น เมื่อเว็บเบราว์เซอร์ประมวลผลเอกสาร HTML จะสร้างโครงสร้าง DOM Tree ซึ่งทำให้ JavaScript สามารถเข้าถึงและจัดการอิลิเมนต์ได้ผ่านฟังก์ชันเฉพาะที่ออกแบบมาสำหรับงานนี้โดยเฉพาะ JavaScript มีฟังก์ชันมาตรฐานที่ใช้ในการเข้าถึงอิลิเมนต์ในเอกสารที่ควรเข้าใจ ได้แก่ การเข้าถึงผ่าน ID การเข้าถึงผ่านคลาส และการเข้าถึงผ่านชื่อแท็ก

1) ฟังก์ชัน getElementById()

ฟังก์ชัน getElementById() เป็นฟังก์ชันพื้นฐานที่ใช้ในการเข้าถึงอิลิเมนต์ผ่านแอตทริบิวต์ id ซึ่งในหลักการของภาษา HTML นั้น แอตทริบิวต์ id ของแต่ละอิลิเมนต์ต้องมีค่าที่ไม่ซ้ำกันในเอกสารเดียวกัน ทำให้ฟังก์ชันนี้สามารถค้นหาและเข้าถึงอิลิเมนต์ได้อย่างเฉพาะเจาะจง

รูปแบบ

```
document.getElementById("id ของอิลิเมนต์ที่ต้องการเข้าถึง")
```

ตัวอย่างที่ 35

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5  </head>
6  <body>
7      <h1 id="main-title">หัวเรื่องหลัก</h1>
```

```
8       <p id="content">เนื้อหาของหน้าเว็บ</p>
9       <script>
10          var title = document.getElementById("main-title");
11          var paragraph = document.getElementById("content");
12          console.log(title.textContent); // แสดง: หัวเรื่องหลัก
13          console.log(paragraph.textContent); // แสดง: เนื้อหาของหน้าเว็บ
14      </script>
15  </body>
16  </html>
```

จากตัวอย่างที่ 35 ฟังก์ชัน getElementById() ได้ถูกเรียกใช้เพื่อเข้าถึงอิลิเมนต์ที่มี id="main-title" และ id="content" โดยผลลัพธ์ที่ได้จะเป็นออบเจกต์ที่แทนอิลิเมนต์นั้นๆ ซึ่งสามารถนำไปใช้ในการจัดการต่อไปได้

2) ฟังก์ชัน getElementsByClassName()

ฟังก์ชัน getElementsByClassName() ใช้สำหรับเข้าถึงอิลิเมนต์ที่มีคลาสเดียวกัน เนื่องจากอิลิเมนต์หลายตัวสามารถมีค่าคลาสเดียวกันได้ ผลลัพธ์ที่ได้จึงเป็นกลุ่มของอิลิเมนต์ในรูปแบบ HTMLCollection ซึ่ง HTMLCollection เป็นออบเจกต์พิเศษที่มีลักษณะคล้ายกับอาร์เรย์ตามที่ได้เรียนรู้ในหัวข้อที่ 3 โดยสามารถเข้าถึงสมาชิกแต่ละตัวได้ผ่านดัชนีและมีคุณสมบัติ length แต่ไม่สามารถใช้เมธอดของอาร์เรย์ เช่น push() หรือ pop() ได้โดยตรง

รูปแบบ

document.getElementsByClassName("ชื่อคลาสของกลุ่มอิลิเมนต์ที่ต้องการเข้าถึง")

ตัวอย่างที่ 36

```
1   <!DOCTYPE html>
2   <html>
3   <head>
4       <meta charset="utf-8">
5   </head>
6   <body>
7       <p class="highlight">ข้อความที่ 1</p>
8       <p class="highlight">ข้อความที่ 2</p>
9       <div class="highlight">เนื้อหาในกล่อง</div>
```

```
10        <script>
11            var elements = document.getElementsByClassName("highlight");
12            console.log(elements.length); // แสดง: 3
13            console.log(elements[0].textContent); // แสดง: ข้อความที่ 1
14            console.log(elements[1].textContent); // แสดง: ข้อความที่ 2
15        </script>
16    </body>
17    </html>
```

จากตัวอย่างที่ 36 ฟังก์ชันจะค้นหาอิลิเมนต์ทั้งหมดที่มี class="highlight" และส่งคืนเป็น HTMLCollection ซึ่งสามารถเข้าถึงแต่ละอิลิเมนต์ได้ผ่านดัชนี โดยเริ่มต้นจากเลข 0

3) ฟังก์ชัน getElementsByTagName()

ฟังก์ชัน getElementsByTagName() ใช้สำหรับเข้าถึงอิลิเมนต์ทั้งหมดที่มีชื่อแท็กเดียวกัน เช่น อิลิเมนต์ p ทั้งหมด หรืออิลิเมนต์ div ทั้งหมดในเอกสาร ผลลัพธ์ที่ได้เป็น HTMLCollection เช่นเดียวกับฟังก์ชัน getElementsByClassName() ซึ่งมีลักษณะคล้ายอาร์เรย์แต่เป็นออบเจกต์พิเศษที่ออกแบบมาสำหรับการจัดเก็บอิลิเมนต์ DOM โดยเฉพาะ

รูปแบบ

```
document.getElementsByTagName("ชื่อแท็กที่ต้องการเข้าถึง")
```

ตัวอย่างที่ 37

```
1     <!DOCTYPE html>
2     <html>
3     <head>
4         <meta charset="utf-8">
5     </head>
6     <body>
7         <h2>หัวเรื่องย่อยที่ 1</h2>
8         <h2>หัวเรื่องย่อยที่ 2</h2>
9         <p>ย่อหน้าแรก</p>
10        <p>ย่อหน้าที่สอง</p>
11        <script>
12            var headings = document.getElementsByTagName("h2");
13            var paragraphs = document.getElementsByTagName("p");
```

```
14              console.log("จำนวนหัวเรื่อง h2: " + headings.length); // แสดง: 2
15              console.log("จำนวนย่อหน้า p: " + paragraphs.length); // แสดง: 2
16              console.log(headings[0].textContent); // แสดง: หัวเรื่องย่อยที่ 1
17              console.log(paragraphs[1].textContent); // แสดง: ย่อหน้าที่สอง
19          </script>
20  </body>
21  </html>
```

จากตัวอย่างที่ 37 ฟังก์ชันจะค้นหาอิลิเมนต์ตามชื่อแท็กที่ระบุ และส่งคืนเป็น HTMLCollection ที่บรรจุอิลิเมนต์ทั้งหมดที่ตรงกับเงื่อนไข การใช้งานฟังก์ชันนี้เหมาะสำหรับกรณีที่ต้องการจัดการอิลิเมนต์ชนิดเดียวกันทั้งหมดในเอกสาร

การเลือกใช้ฟังก์ชันเหล่านี้ขึ้นอยู่กับลักษณะของงานและวิธีการระบุอิลิเมนต์เป้าหมาย โดยทั่วไปแล้วฟังก์ชัน getElementById() จะมีความเหมาะสมในการใช้งาน เนื่องจากค้นหาเพียงอิลิเมนต์เดียว ในขณะที่อีกสองฟังก์ชันจะมีประโยชน์เมื่อต้องการทำงานกับกลุ่มของอิลิเมนต์

4.3.2 การจัดการเนื้อหาและแอตทริบิวต์

หลังจากที่ทราบวิธีการเข้าถึงอิลิเมนต์ในเอกสารแล้ว ขั้นตอนต่อไปคือการจัดการเนื้อหาและแอตทริบิวต์ของอิลิเมนต์เหล่านั้น เพื่อปรับเปลี่ยนการแสดงผลหรือพฤติกรรมของหน้าเว็บให้เป็นแบบพลวัต JavaScript มีคุณสมบัติและเมธอดหลากหลายที่ช่วยในการจัดการเนื้อหาและแอตทริบิวต์ของอิลิเมนต์ได้อย่างมีประสิทธิภาพ

1) การจัดการเนื้อหาของอิลิเมนต์

การจัดการเนื้อหาของอิลิเมนต์สามารถทำได้หลายรูปแบบ โดยมีคุณสมบัติที่สำคัญที่ควรทราบ ได้แก่ textContent และ innerHTML ซึ่งแต่ละคุณสมบัติมีลักษณะการใช้งานที่แตกต่างกัน

- **คุณสมบัติ textContent**

คุณสมบัติ textContent ใช้สำหรับการอ่านหรือกำหนดเนื้อหาข้อความภายในอิลิเมนต์ โดยจะแสดงเฉพาะข้อความธรรมดาและไม่รวมแท็ก HTML ใดๆ เป็นวิธีที่ปลอดภัยในการจัดการข้อความ เนื่องจากป้องกันการแทรกรหัสคำสั่งที่ไม่พึงประสงค์

รูปแบบ	
1	**ฟังก์ชันการเข้าถึงอิลิเมนต์( ).textContent**
2	**ออบเจกต์ของอิลิเมนต์.textContent**

ตัวอย่างที่ 38

HTML

```
1  <h1 id="title">หัวเรื่องเดิม</h1>
2  <p id="description">คำอธิบายเดิม</p>
3  <button onclick="changeText()">เปลี่ยนข้อความ</button>
4  <script  src="content.js"></script>
```

JavaScript: content.js

```
1  function changeText() {
2      var title = document.getElementById("title");
3      var desc = document.getElementById("description");
4      console.log(title.textContent); // แสดง: หัวเรื่องเดิม
       // เปลี่ยนเนื้อหา
5      title.textContent = "หัวเรื่องใหม่";
6      desc.textContent = "คำอธิบายที่ได้รับการอัปเดต";
7  }
```

จากตัวอย่างที่ 38 การใช้ textContent จะแสดงและรับเฉพาะข้อความธรรมดาเท่านั้น หากมีการใส่แท็ก HTML ลงไป เช่น title.textContent = "หัวเรื่องใหม่" แท็ก จะแสดงเป็นข้อความธรรมดาแทนการจัดรูปแบบ ทำให้ปลอดภัยจากการแทรกรหัสคำสั่งที่ไม่พึงประสงค์ เมื่อคลิกปุ่ม *เปลี่ยนข้อความ* ผู้ใช้จะเห็นการเปลี่ยนแปลงของเนื้อหาทันที

- **คุณสมบัติ innerHTML**

คุณสมบัติ innerHTML ใช้สำหรับการอ่านหรือกำหนดเนื้อหา HTML ภายในอิลิเมนต์ สามารถใส่แท็ก HTML ได้และเว็บเบราว์เซอร์จะแปลงแท็กเหล่านั้นให้เป็นอิลิเมนต์จริง อย่างไรก็ตาม ต้องใช้ความระมัดระวังเนื่องจากอาจเกิดช่องโหว่ด้านความปลอดภัยหากข้อมูลมาจากผู้ใช้โดยตรง

รูปแบบ

1 **ฟังก์ชันการเข้าถึงอิลิเมนต์( ).innerHTML**

2 **ออบเจกต์ของอิลิเมนต์.innerHTML**

ตัวอย่างที่ 39

HTML

```
1  <div id="content">เนื้อหาเดิม</div>
2  <button onclick="changeHTML()">เปลี่ยนเนื้อหา HTML</button>
```

```
3   <script  src="inner.js"></script>
```

JavaScript: inner.js

```
1   function changeHTML() {
2       var content = document.getElementById("content");
        // แสดงเนื้อหา HTML ปัจจุบัน
3       console.log(content.innerHTML); // แสดง: เนื้อหาเดิม
        // เปลี่ยนเนื้อหาเป็น HTML
4       content.innerHTML = "<strong>เนื้อหาใหม่</strong> พร้อม <em>การจัดรูปแบบ</em>";
5   }
```

จากตัวอย่างที่ 39 การใช้ innerHTML สามารถแทรกแท็ก HTML ได้และเว็บเบราว์เซอร์จะแปลงให้เป็นการจัดรูปแบบจริง เช่น จะทำให้ข้อความหนา และ จะทำให้ข้อความเอียง อย่างไรก็ตาม หากข้อมูลมาจากผู้ใช้โดยตรง อาจมีความเสี่ยงจากการแทรกรหัสคำสั่ง JavaScript ที่เป็นอันตราย เมื่อคลิกปุ่ม *เปลี่ยนเนื้อหา HTML* ผู้ใช้จะเห็นการเปลี่ยนแปลงและการจัดรูปแบบทันที

2) การจัดการแอตทริบิวต์ของอิลิเมนต์

การจัดการแอตทริบิวต์ของอิลิเมนต์เป็นสิ่งสำคัญในการควบคุมพฤติกรรมและลักษณะการแสดงผลของอิลิเมนต์ JavaScript มีเมธอดหลักสองตัวสำหรับการทำงานกับแอตทริบิวต์ ได้แก่ getAttribute() และ setAttribute()

- **เมธอด getAttribute()**

เมธอด getAttribute() ใช้สำหรับการอ่านค่าของแอตทริบิวต์ที่ระบุ หากแอตทริบิวต์ที่ต้องการไม่มีอยู่ในอิลิเมนต์ จะคืนค่า null

รูปแบบ

```
1   ฟังก์ชันการเข้าถึงอิลิเมนต์( ).getAttribute("ชื่อแอตทริบิวต์")
2   ออบเจกต์ของอิลิเมนต์.getAttribute("ชื่อแอตทริบิวต์")
```

ตัวอย่างที่ 40

HTML

```
1   <img id="product" src="product-default.jpg" alt="สินค้าของเรา" width="200">
2   <br>
3   <button onclick="showImageInfo()">แสดงข้อมูลภาพ</button>
4   <script  src="get_attribute.js"></script>
```

JavaScript: get_attribute.js

```
1  function showImageInfo() {
2      var image = document.getElementById("product");
3      var info = "ข้อมูลภาพสินค้า:\n";
4      info += "ไฟล์: " + image.getAttribute("src") + "\n";
5      info += "คำอธิบาย: " + image.getAttribute("alt") + "\n";
6      info += "ความกว้าง: " + image.getAttribute("width") + " พิกเซล";
7      alert(info);
8  }
```

จากตัวอย่างที่ 40 เมื่อคลิกปุ่ม *แสดงข้อมูลภาพ* ฟังก์ชัน showImageInfo() จะใช้เมธอด getAttribute() เพื่อดึงค่าแอตทริบิวต์ src, alt และ width ของภาพมาแสดงในหน้าต่างแจ้งเตือน ทำให้เห็นการทำงานของเมธอดในการอ่านค่าแอตทริบิวต์ได้อย่างชัดเจน

- **เมธอด setAttribute()**

เมธอด setAttribute() ใช้สำหรับการกำหนดหรือเปลี่ยนแปลงค่าของแอตทริบิวต์ หากแอตทริบิวต์ที่ระบุไม่มีอยู่เดิม จะสร้างแอตทริบิวต์ใหม่ขึ้นมา

รูปแบบ

```
1  ฟังก์ชันการเข้าถึงอิลิเมนต์( ).setAttribute("ชื่อแอตทริบิวต์", "ค่าของแอตทริบิวต์")
2  ออบเจกต์ของอิลิเมนต์.setAttribute("ชื่อแอตทริบิวต์", "ค่าของแอตทริบิวต์")
```

ตัวอย่างที่ 41

HTML

```
1  <img id="banner" src="banner-summer.jpg" alt="ป้ายฤดูร้อน" width="250">
2  <br>
3  <button onclick="changeToWinter()">เปลี่ยนเป็นฤดูหนาว</button>
4  <script  src="set_attribute.js"></script>
```

JavaScript: set_attribute.js

```
1  function changeToWinter() {
2      var banner = document.getElementById("banner");
3      banner.setAttribute("src", "banner-winter.jpg");
4      banner.setAttribute("alt", "แบนเนอร์ฤดูหนาว");
5      banner.setAttribute("width", "300");
```

```
6       alert("เปลี่ยนแบนเนอร์เป็นฤดูหนาวแล้ว!");
7   }
```

จากตัวอย่างที่ 41 เมื่อคลิกปุ่ม *เปลี่ยนเป็นฤดูหนาว* ฟังก์ชัน changeToWinter() จะใช้เมธอด setAttribute() เพื่อเปลี่ยนค่าแอตทริบิวต์ src, alt และ width ของภาพแบนเนอร์ ทำให้เห็นการทำงานของเมธอดในการเปลี่ยนแปลงค่าแอตทริบิวต์ได้

4.4 การจัดการเหตุการณ์

เหตุการณ์ในบริบทของเว็บแอปพลิเคชันหมายถึงการกระทำหรือเหตุการณ์ที่เกิดขึ้นในเว็บเบราว์เซอร์ ซึ่งอาจเกิดจากการกระทำของผู้ใช้ เช่น การคลิกเมาส์ การกดแป้นพิมพ์ หรือการเลื่อนหน้าเว็บ หรืออาจเกิดจากระบบเอง เช่น การโหลดหน้าเว็บเสร็จสมบูรณ์ การเปลี่ยนขนาดหน้าต่าง หรือการหมดเวลาของตัวจับเวลา เป็นต้น เหตุการณ์เหล่านี้เป็นสิ่งที่เกิดขึ้นตลอดเวลาขณะที่ผู้ใช้โต้ตอบกับหน้าเว็บ แต่จะมีผลต่อการทำงานของเว็บแอปพลิเคชันก็ต่อเมื่อมีการกำหนดฟังก์ชันจัดการเหตุการณ์ เพื่อระบุว่าต้องการให้ทำอะไรเมื่อเหตุการณ์นั้นเกิดขึ้น

4.4.1 เหตุการณ์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน

เหตุการณ์ที่เกิดจากการกระทำของผู้ใช้มีหลายประเภทที่ควรเข้าใจเพื่อนำไปใช้ในการพัฒนาเว็บแอปพลิเคชันอย่างเหมาะสม โดยเหตุการณ์หลักที่พบบ่อยแสดงดังตารางที่ 10-2

ตารางที่ 10-2 ชื่อเหตุการณ์ที่พบบ่อย

ชื่อเหตุการณ์	คำอธิบาย	การใช้งานทั่วไป
click	เกิดขึ้นเมื่อผู้ใช้คลิกเมาส์บนอิลิเมนต์	คลิกปุ่ม ลิงก์ รูปภาพ
mouseover	เกิดขึ้นเมื่อผู้ใช้เลื่อนเมาส์อยู่เหนืออิลิเมนต์	สร้างเอฟเฟกต์เมื่อวางเมาส์
mouseout	เกิดขึ้นเมื่อผู้ใช้เลื่อนเมาส์ออกจากอิลิเมนต์	คืนสภาพเดิมเมื่อเมาส์ออก
keydown	เกิดขึ้นเมื่อผู้ใช้กดแป้นพิมพ์ลง	ตรวจสอบการป้อนข้อมูล ปุ่มลัด
keyup	เกิดขึ้นเมื่อผู้ใช้ปล่อยแป้นพิมพ์	ตรวจสอบเมื่อเสร็จสิ้นการกด
change	เกิดขึ้นเมื่อค่าในอินพุตมีการเปลี่ยนแปลง	ตรวจสอบข้อมูลในแบบฟอร์ม
submit	เกิดขึ้นเมื่อมีการส่งแบบฟอร์ม	ตรวจสอบก่อนส่งข้อมูล
load	เกิดขึ้นเมื่อหน้าเว็บโหลดเสร็จสมบูรณ์	เริ่มต้นการทำงานของหน้าเว็บ

4.4.2 การใช้เมธอด addEventListener()

การใช้เมธอด addEventListener() เพื่อจัดการเหตุการณ์เป็นวิธีการที่มีความยืดหยุ่นและมีประสิทธิภาพสูงกว่าการใช้แอตทริบิวต์เหตุการณ์ในแท็ก HTML เช่น onclick หรือ onmouseover เป็นต้น เมธอดนี้ช่วยให้สามารถแยกการจัดการเหตุการณ์ออกจากโครงสร้าง HTML ได้

รูปแบบ

ออบเจกต์ของอิลิเมนต์.addEventListener("ชื่อเหตุการณ์", ฟังก์ชันจัดการเหตุการณ์)

ตัวอย่างที่ 42

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4      <button id="myButton">คลิกเพื่อทดลอง</button>
5      <p id="message">ข้อความจะแสดงที่นี่</p>
6      <script>
7          var button = document.getElementById("myButton");
8          var message = document.getElementById("message");
9          button.addEventListener("click", function () {
10             message.textContent = "คุณได้คลิกปุ่มแล้ว!";
11         });
12     </script>
13 </body>
14 </html>
```

จากตัวอย่างที่ 42 เมื่อผู้ใช้คลิกปุ่ม *คลิกเพื่อทดลอง* เหตุการณ์ click จะเกิดขึ้น และฟังก์ชันที่กำหนดไว้จะทำงาน โดยเปลี่ยนข้อความในอิลิเมนต์ p เป็น *คุณได้คลิกปุ่มแล้ว!* การใช้ addEventListener() ทำให้การจัดการเหตุการณ์มีความเป็นระเบียบและสามารถบำรุงรักษารหัสคำสั่งได้ง่าย

4.5 การจัดการแบบฟอร์ม

การจัดการแบบฟอร์มด้วย JavaScript เป็นทักษะสำคัญในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากเป็นกลไกหลักในการรับข้อมูลจากผู้ใช้และประมวลผลข้อมูลดังกล่าวก่อนส่งไปยังฝั่งผู้ให้บริการ JavaScript ช่วยให้สามารถตรวจสอบความถูกต้องของข้อมูล แสดงข้อความแจ้งเตือน และควบคุมพฤติกรรมการส่งข้อมูลได้อย่างยืดหยุ่น ซึ่งนำไปสู่ประสบการณ์ผู้ใช้ที่ดีและลดปัญหาข้อมูลผิดพลาดที่อาจเกิดขึ้น

4.5.1 การเข้าถึงอิลิเมนต์ฟอร์มและการดึงค่าข้อมูล

การจัดการแบบฟอร์มเริ่มต้นจากการเข้าถึงอิลิเมนต์ภายในฟอร์มและการดึงค่าข้อมูลจากอินพุตประเภทต่าง ๆ JavaScript ตัวอย่างต่อไปนี้เป็นการประยุกต์ใช้การเข้าถึงอิลิเมนต์ดังที่ได้นำเสนอในหัวข้อก่อนหน้านี้ ดังนี้

ตัวอย่างที่ 43

HTML

```
1  <form>
2      <input type="text" id="username" placeholder="ชื่อผู้ใช้">
3      <select id="status">
4          <option value="active">ใช้งาน</option>
5          <option value="inactive">ไม่ใช้งาน</option>
6      </select>
7      <input type="checkbox" id="newsletter"> รับข่าวสาร
8      <button type="button" onclick="showData()">แสดงข้อมูล</button>
9  </form>
11 <div id="result"></div>
12 <script src="form-example.js"></script>
```

JavaScript: form-example.js

```
1  function showData() {
2      let name = document.getElementById('username').value;
3      let status = document.getElementById('status').value;
4      let news = document.getElementById('newsletter').checked;
5      document.getElementById('result').innerHTML =
           `ชื่อ: ${name}, สถานะ: ${status}, รับข่าว: ${news}`;
6  }
```

จากตัวอย่างที่ 43 แสดงการเข้าถึงข้อมูลจากอินพุตประเภทต่าง ๆ โดยใช้ getElementById() และคุณสมบัติ value สำหรับ input และ select ส่วน checkbox ใช้คุณสมบัติ checked ที่ได้ค่า true/false โดยแยกรหัสคำสั่ง JavaScript ไว้ในไฟล์ภายนอกชื่อ form-example.js เมื่อผู้ใช้กรอกข้อมูลในส่วนต่าง ๆ ของฟอร์มแล้วคลิกปุ่ม *แสดงข้อมูล* ระบบจะเรียกใช้ฟังก์ชัน showData() ซึ่งจะดึงค่าข้อมูลจากอิลิเมนต์แต่ละตัวผ่านคุณสมบัติ value และ checked จากนั้นนำข้อมูลทั้งหมดมาแสดงผลในรูปแบบข้อความภายในส่วน result โดยไม่มีการตรวจสอบความถูกต้องของข้อมูล

4.5.2 การตรวจสอบความถูกต้องของข้อมูลฝั่งผู้ใช้

การตรวจสอบข้อมูลฝั่งผู้ใช้ (Client-side Validation) เป็นขั้นตอนสำคัญที่ช่วยให้ผู้ใช้ทราบถึงข้อผิดพลาดของข้อมูลก่อนที่จะส่งไปยังฝั่งผู้ให้บริการ การตรวจสอบที่พบบ่อยประกอบด้วย การตรวจสอบข้อมูลที่จำเป็น การตรวจสอบรูปแบบข้อมูล และการตรวจสอบความยาวของข้อมูล เป็นต้น การแสดงข้อความแจ้งเตือนจะ

ช่วยให้ผู้ใช้แก้ไขข้อมูลได้ก่อนที่จะนำไปประมวลผล ซึ่งสอดคล้องกับแนวคิดการทำให้ข้อมูลคุณภาพก่อนทำการประมวลผลที่ได้กล่าวถึงในบทที่ 2 ตัวอย่างต่อไปนี้เป็นการประยุกต์ใช้ความรู้เกี่ยวกับภาษา JavaScript เพื่อใช้ในการตรวจสอบความถูกต้องของข้อมูล

ตัวอย่างที่ 44

HTML

```
1  <form>
2      <input type="text" id="fullname" placeholder="ชื่อ-สกุล">
3      <span id="name-error" style="color: red; font-size: 12px;"></span><br>
4      <input type="email" id="email" placeholder="อีเมล">
5      <span id="email-error" style="color: red; font-size: 12px;"></span><br>
6      <button type="button" onclick="validateForm()">ตรวจสอบ</button>
7  </form>
8  <div id="result"></div>
9  <script src="validation.js"></script>
```

JavaScript: form-example.js

```
1  function validateForm() {
2      let isValid = true;
       // ตรวจสอบชื่อ
3      let name = document.getElementById('fullname').value.trim();
4      let nameError = document.getElementById('name-error');
5      if (name === '') {
6          nameError.textContent = 'กรุณากรอกชื่อ-สกุล';
7          isValid = false;
8      } else {
9          nameError.textContent = '';
10     }
       // ตรวจสอบอีเมล
11     let email = document.getElementById('email').value.trim();
12     let emailError = document.getElementById('email-error');
13     if (email === '') {
14         emailError.textContent = 'กรุณากรอกอีเมล';
15         isValid = false;
```

```
16          } else if (email.indexOf('@') === -1) {
17              emailError.textContent = 'อีเมลต้องมีเครื่องหมาย @';
18              isValid = false;
19          } else {
20              emailError.textContent = '';
21          }
            // แสดงผล
22          let result = document.getElementById('result');
23          if (isValid) {
24              result.innerHTML = '<span style="color: green;">ข้อมูลถูกต้อง</span>';
25          } else {
26              result.innerHTML = '<span style="color: red;">กรุณาแก้ไขข้อมูล</span>';
27          }
28      }
```

จากตัวอย่างที่ 44 แสดงการตรวจสอบข้อมูลฟอร์มโดยตรวจสอบข้อมูลที่จำเป็น (Required) และรูปแบบอีเมลด้วยวิธีการพื้นฐาน เช่น การตรวจสอบว่าอีเมลมีเครื่องหมาย @ หรือไม่ เมื่อข้อมูลไม่ถูกต้อง จะแสดงข้อความแจ้งเตือนในส่วนผิดพลาดโดยแยกภาษา JavaScript ไว้ในไฟล์ validation.js

เมื่อผู้ใช้กรอกข้อมูลในช่องชื่อ-สกุล และอีเมล แล้วคลิกปุ่มตรวจสอบ จะมีการเรียกใช้ฟังก์ชัน validateForm() ซึ่งจะตรวจสอบว่าช่องชื่อ-สกุลไม่ว่างเปล่า และตรวจสอบอีเมลว่าต้องมีเครื่องหมาย @ หากข้อมูลใดไม่ถูกต้อง จะแสดงข้อความแจ้งเตือนสีแดงใต้ช่องที่มีปัญหานั้นๆ และแสดงสถานะเป็นข้อความว่า *กรุณาแก้ไขข้อมูล* ในส่วนผลลัพธ์ หากข้อมูลถูกต้องทั้งหมด จะแสดงข้อความ *ข้อมูลถูกต้อง* เป็นสีเขียว

4.5.3 การจัดการเหตุการณ์ฟอร์มและการป้องกันการส่งข้อมูล

การจัดการเหตุการณ์ที่เกี่ยวข้องกับฟอร์มและการควบคุมการส่งข้อมูลเป็นส่วนสำคัญของการพัฒนาเว็บแอปพลิเคชัน การใช้เหตุการณ์ submit ร่วมกับการป้องกันการส่งข้อมูลเมื่อข้อมูลไม่ถูกต้องจะช่วยให้เพิ่มคุณภาพในการคัดกรองข้อมูล นอกจากนี้การใช้เหตุการณ์อื่นๆ เช่น input หรือ change ยังช่วยให้สามารถตรวจสอบข้อมูลแบบเรียลไทม์ได้ จากตัวอย่างต่อไปนี้เป็นการประยุกต์ใช้ความรู้เกี่ยวกับภาษา JavaScript เพื่อใช้ในการจัดการเหตุการณ์ฟอร์มและการป้องกันการส่งข้อมูล และการใช้เมธอด .preventDefault( ) ซึ่งเป็นเมธอดของออบเจกต์ Event ที่ใช้ ยับยั้งพฤติกรรมปกติของ HTML อิลิเมนต์ ที่จะเกิดขึ้นเมื่อมีเหตุการณ์บางอย่าง เช่น การส่งฟอร์ม การคลิกลิงก์ หรือการลากวาง เป็นต้น

ตัวอย่างที่ 45

HTML

```
1  <form id="loginForm">
2      <input type="text" id="username" placeholder="ชื่อผู้ใช้" required>
3      <span id="username-error" style="color: red; font-size: 12px;"></span><br>
4      <input type="password" id="password" placeholder="รหัสผ่าน" required>
5      <span id="password-error" style="color: red; font-size: 12px;"></span><br>
6      <button type="submit">เข้าสู่ระบบ</button>
7  </form>
8  <div id="result"></div>
9  <script src="form-handler.js"></script>
```

JavaScript: form-handler.js

```
   // เพิ่ม event listener ให้กับฟอร์ม
1  document.getElementById('loginForm').addEventListener('submit', function (e) {
2      e.preventDefault(); // ป้องกันการส่งฟอร์มโดยอัตโนมัติ
3      let username = document.getElementById('username').value.trim();
4      let password = document.getElementById('password').value;
5      let isValid = true;
       // ตรวจสอบชื่อผู้ใช้
6      if (username.length < 3) {
7          userError = document.getElementById('username-error')
8          userError.textContent = 'ชื่อผู้ใช้ต้องมีอย่างน้อย 3 ตัวอักษร';
9          isValid = false;
10     } else {
11         document.getElementById('username-error').textContent = '';
12     }
13     // ตรวจสอบรหัสผ่าน
14     if (password.length < 6) {
15         passwordError = document.getElementById('password-error');
16         passwordError.textContent = 'รหัสผ่านต้องมีอย่างน้อย 6 ตัวอักษร';
17         isValid = false;
18     } else {
```

```
19              document.getElementById('password-error').textContent = '';
20          }
21          // แสดงผลการตรวจสอบ
22          if (isValid) {
23              result=document.getElementById('result');
24              result.innerHTML = '<span style="color: green;">เข้าสู่ระบบสำเร็จ</span>';
25          }
26  });
    // ตรวจสอบแบบเรียลไทม์
27  document.getElementById('username').addEventListener('input', function () {
28          if (this.value.length >= 3) {
29              document.getElementById('username-error').textContent = '';
30          }
31  });
```

จากตัวอย่างที่ 45 แสดงการจัดการเหตุการณ์ submit และการป้องกันการส่งข้อมูลด้วย preventDefault() เมื่อข้อมูลไม่ถูกต้อง รวมถึงการตรวจสอบแบบเรียลไทม์ด้วยเหตุการณ์ input โดยแยกรหัสคำสั่ง JavaScript ไว้ในไฟล์ form-handler.js

ตัวอย่างนี้ใช้ปุ่มประเภท submit ซึ่งจะส่งฟอร์มไปยังปลายทางโดยอัตโนมัติเมื่อถูกคลิก อย่างไรก็ตาม การใช้คำสั่ง preventDefault() ในบรรทัดที่ 2 ของไฟล์ form-handler.js ซึ่งมีหน้าที่หยุดพฤติกรรมปกติของเหตุการณ์ submit ทำให้ฟอร์มไม่ส่งข้อมูลไปยังเครื่องผู้ให้บริการเว็บโดยอัตโนมัติ แต่จะให้ JavaScript เป็นผู้ควบคุมการทำงานแทน การใช้วิธีนี้ช่วยให้สามารถตรวจสอบข้อมูลก่อนส่งได้ และป้องกันการส่งข้อมูลที่ไม่ถูกต้องไปยังฝั่งผู้ให้บริการ หากต้องการให้ฟอร์มส่งข้อมูลจริง สามารถเพิ่มคำสั่ง this.submit() ในส่วนท้ายของฟังก์ชันเมื่อข้อมูลถูกต้องทั้งหมด

เมื่อผู้ใช้กรอกชื่อผู้ใช้และรหัสผ่านแล้วคลิกปุ่ม เข้าสู่ระบบ ระบบจะดักจับเหตุการณ์ submit และหยุดการส่งฟอร์มด้วย preventDefault() จากนั้นตรวจสอบว่าชื่อผู้ใช้มีอย่างน้อย 3 ตัวอักษรและรหัสผ่านมีอย่างน้อย 6 ตัวอักษร หากข้อมูลไม่ครบเงื่อนไข จะแสดงข้อความแจ้งเตือนใต้ช่องที่มีปัญหา หากข้อมูลถูกต้อง จะแสดงข้อความ เข้าสู่ระบบสำเร็จ นอกจากนี้ยังมีการตรวจสอบแบบเรียลไทม์ที่จะลบข้อความแจ้งเตือนทันทีเมื่อผู้ใช้พิมพ์ชื่อผู้ใช้ครบ 3 ตัวอักษร

การจัดการแบบฟอร์มด้วย JavaScript ที่นำเสนอในหัวข้อนี้เป็นพื้นฐานสำคัญในการพัฒนาเว็บแอปพลิเคชันที่มีปฏิสัมพันธ์กับผู้ใช้ ความสามารถในการเข้าถึงข้อมูล ตรวจสอบความถูกต้อง และควบคุมการส่งข้อมูลจะช่วยให้เว็บแอปพลิเคชันมีความน่าเชื่อถือและให้ประสบการณ์ที่ดีแก่ผู้ใช้

5. สรุปท้ายบท

ฟังก์ชันใน JavaScript ทำหน้าที่เป็นกลไกสำคัญในการจัดกลุ่มรหัสคำสั่งเพื่อสร้างการทำงานที่เป็นระบบและรองรับการเรียกใช้งานซ้ำในหลายบริบทของโปรแกรม ช่วยให้รหัสคำสั่งมีความกระชับ อ่านง่าย และบำรุงรักษาได้สะดวกยิ่งขึ้น การใช้ฟังก์ชันช่วยลดความซ้ำซ้อนของรหัส (Redundancy) และสนับสนุนแนวคิดการเขียนโปรแกรมแบบแยกส่วน (Modular Programming) ทำให้การพัฒนาเว็บแอปพลิเคชันขนาดใหญ่มีความยืดหยุ่นต่อการปรับแก้ไขและขยายความสามารถเพิ่มเติม ฟังก์ชันใน JavaScript มีรูปแบบที่หลากหลาย ได้แก่ Function Declaration ซึ่งเป็นรูปแบบมาตรฐานที่ตัวแปลภาษา JavaScript (Engine) จะประมวลผลก่อนการเรียกใช้งาน (Hoisting), Function Expression ที่กำหนดฟังก์ชันให้กับตัวแปร และ Arrow Function ที่เป็นรูปแบบกระชับที่เหมาะกับการกำหนดฟังก์ชันแบบไม่ระบุชื่อในกรณีการส่งต่อฟังก์ชันเป็นพารามิเตอร์ การเข้าใจกลไกการทำงานของฟังก์ชัน การจัดการพารามิเตอร์และค่าที่ส่งกลับ รวมถึงการประยุกต์ใช้ฟังก์ชันแบบไม่ระบุชื่อ (Anonymous Function) และฟังก์ชันเรียกกลับ (Callback Function) ถือเป็นส่วนสำคัญในการสร้างตรรกะการทำงานที่ยืดหยุ่น ช่วยให้นักพัฒนาสามารถกำหนดพฤติกรรมที่ซับซ้อนและตอบสนองต่อเหตุการณ์ผู้ใช้ได้

ออบเจกต์ถือเป็นโครงสร้างข้อมูลหลักใน JavaScript ที่รองรับการจัดเก็บและเข้าถึงข้อมูลในรูปแบบคู่คีย์-ค่า (Key-Value Pair) โดยสามารถเก็บทั้งข้อมูลเชิงเดี่ยว เช่น ตัวเลขหรือข้อความ และข้อมูลเชิงซ้อน เช่น ฟังก์ชันหรืออาร์เรย์ภายในคุณสมบัติของออบเจกต์ ออบเจกต์ใน JavaScript มีความยืดหยุ่นสูงและสามารถสร้างได้หลายวิธี เช่น Object Literal ที่เหมาะกับการสร้างออบเจกต์ขนาดเล็กอย่างรวดเร็ว, Constructor Function ที่ใช้สร้างออบเจกต์หลายตัวที่มีโครงสร้างเหมือนกัน และการใช้ Class ที่แนะนำใน ECMAScript 2015 (ES6) เพื่อสนับสนุนแนวคิดเชิงวัตถุ (Object-Oriented Programming: OOP) การเข้าถึงคุณสมบัติของออบเจกต์ทำได้ทั้งผ่าน Dot Notation เช่น object.property ซึ่งอ่านง่าย และ Bracket Notation เช่น object['property'] ที่เหมาะสมเมื่อชื่อคุณสมบัติเป็นตัวแปรหรือมีอักขระพิเศษ การเพิ่ม ลบ หรือปรับเปลี่ยนเมธอดของออบเจกต์ช่วยให้สามารถควบคุมพฤติกรรมและการทำงานของโปรแกรมได้อย่างแม่นยำ ซึ่งเป็นพื้นฐานสำคัญสำหรับการสร้างเว็บแอปพลิเคชันที่มีโครงสร้างข้อมูลซับซ้อน

อาร์เรย์เป็นโครงสร้างข้อมูลที่มีความยืดหยุ่นสูง ช่วยให้สามารถจัดเก็บชุดข้อมูลหลายรายการภายในตัวแปรเดียวและเข้าถึงสมาชิกแต่ละตัวผ่านดัชนี (Index) ที่เริ่มจาก 0 การจัดการข้อมูลด้วยอาร์เรย์จึงเป็นวิธีที่สะดวกและมีประสิทธิภาพสำหรับการทำงานกับข้อมูลขนาดใหญ่หรือข้อมูลที่มีความสัมพันธ์เชิงลำดับ เมธอดสำคัญของอาร์เรย์ เช่น push() สำหรับเพิ่มข้อมูลท้ายอาร์เรย์, pop() สำหรับลบข้อมูลจากท้ายอาร์เรย์, shift() และ unshift() สำหรับจัดการข้อมูลที่จุดเริ่มต้นของอาร์เรย์ รวมถึงคุณสมบัติ length ที่ช่วยระบุขนาดของอาร์เรย์ ทำให้นักพัฒนาสามารถควบคุมการประมวลผลข้อมูลได้อย่างมีประสิทธิภาพ นอกจากนี้อาร์เรย์ยังรองรับการซ้อนอาร์เรย์ย่อย (Nested Array) และการใช้เมธอดระดับสูง เช่น map(), filter(), และ reduce() เพื่อประมวลผลข้อมูลในเชิงฟังก์ชัน ทำให้การพัฒนาเว็บแอปพลิเคชันที่มีตรรกะซับซ้อนมีความยืดหยุ่นและง่ายต่อการบำรุงรักษา

การจัดการเอกสารเว็บด้วย JavaScript ผ่านกลไก DOM (Document Object Model) ช่วยให้นักพัฒนาสามารถเข้าถึงและเปลี่ยนแปลงอิลิเมนต์ภายในหน้าเว็บได้แบบเรียลไทม์ การใช้ textContent และ innerHTML

ช่วยในการปรับเปลี่ยนเนื้อหา ทั้งในลักษณะข้อความล้วนและเนื้อหา HTML แบบโครงสร้าง ขณะที่เมธอด getAttribute() และ setAttribute() รองรับการอ่านและแก้ไขแอตทริบิวต์ของอิลิเมนต์ ทำให้สามารถควบคุมพฤติกรรมและลักษณะการแสดงผลได้อย่างละเอียด นอกจากนี้กลไก Event Handling ที่ใช้ฟังก์ชัน addEventListener() เพื่อจัดการกับเหตุการณ์ เช่น การคลิกหรือการกรอกข้อมูลในฟอร์ม ช่วยให้สามารถกำหนดการตอบสนองต่อการกระทำของผู้ใช้ได้อย่างยืดหยุ่น การจัดการฟอร์ม เช่น การตรวจสอบความถูกต้องของข้อมูลก่อนการส่ง (Validation) และการยับยั้งการส่งข้อมูลแบบปกติ (Prevent Default Submission) เป็นแนวทางที่สำคัญในการเพิ่มความปลอดภัยและความถูกต้องของข้อมูลที่รับจากผู้ใช้ ซึ่งมีบทบาทสำคัญต่อการพัฒนาเว็บแอปพลิเคชันที่ต้องจัดการข้อมูลจำนวนมากและต้องการความน่าเชื่อถือในการประมวลผล

6. กิจกรรมท้ายบท

6.1 จงอธิบายบทบาทของฟังก์ชันในภาษา JavaScript พร้อมทั้งแสดงตัวอย่างการประกาศฟังก์ชันทั้ง 3 รูปแบบที่ศึกษาในบทนี้ ได้แก่ Function Declaration, Function Expression และ Arrow Function พร้อมระบุความแตกต่างระหว่างรูปแบบเหล่านั้น

6.2 พารามิเตอร์และค่าที่ส่งกลับจากฟังก์ชันมีความสำคัญอย่างไรในการพัฒนาเว็บแอปพลิเคชัน? จงอธิบายและแสดงตัวอย่างฟังก์ชันที่รับพารามิเตอร์และส่งค่ากลับ

6.3 อธิบายความแตกต่างระหว่างออบเจกต์กับอาร์เรย์ในภาษา JavaScript ในด้านโครงสร้างการจัดเก็บข้อมูลและวิธีการเข้าถึง พร้อมยกตัวอย่างการประกาศและเข้าถึงข้อมูลในแต่ละแบบ

6.4 DOM (Document Object Model) มีความสำคัญอย่างไรต่อการพัฒนาเว็บแอปพลิเคชันแบบโต้ตอบ? จงอธิบายกลไกที่ JavaScript ใช้ค้นหาและปรับเปลี่ยนอิลิเมนต์ผ่าน DOM พอสังเขป

6.5 เหตุการณ์ (Event) ใน JavaScript หมายถึงอะไร? จงยกตัวอย่างเหตุการณ์ที่พบบ่อยในการพัฒนาเว็บแอปพลิเคชัน พร้อมแสดงรูปแบบการเขียน Event Handler ที่ใช้กับอิลิเมนต์ของหน้าเว็บอย่างน้อย 2 รูปแบบ

6.6 จงเขียนเอกสารเว็บที่ประกอบด้วยฟอร์มรับข้อมูลชื่อและอายุ โดยมีข้อกำหนดดังนี้

6.6.1 เมื่อผู้ใช้กดปุ่มส่ง ให้ฟังก์ชัน JavaScript ตรวจสอบว่ากรอกข้อมูลครบหรือไม่ หากไม่ครบให้แสดงข้อความแจ้งเตือนใน DOM โดยไม่ใช้ alert()

6.6.2 หากข้อมูลครบ ให้แสดงข้อความต้อนรับในรูปแบบ "สวัสดีคุณ [ชื่อ] อายุ [อายุ] ปี" ในพื้นที่แสดงผลบนหน้าเว็บ โดยให้ปรากฏอยู่ท้ายฟอร์ม

6.6.3 บันทึกไฟล์เป็น form-validate.html แล้วทดสอบในเว็บเบราว์เซอร์



Backend: ภาษาจัดการข้อมูลฝั่งผู้ให้บริการ



ภาษา PHP



การจัดการข้อมูลจากผู้ใช้ด้วย PHP



ภาษา PHP

บทที่ 11 มุ่งเน้นการนำเสนอพื้นฐานของภาษา PHP ซึ่งเป็นภาษาสคริปต์ฝั่งผู้ให้บริการที่มีบทบาทสำคัญในการพัฒนาเว็บแอปพลิเคชัน โดยผู้อ่านจะได้เรียนรู้โครงสร้างภาษา กลไกการทำงาน และรูปแบบการเขียนคำสั่งที่จำเป็นสำหรับการประมวลผลข้อมูล การจัดการตรรกะ และการควบคุมการทำงาน เพื่อเป็นพื้นฐานสำหรับการพัฒนาเว็บแอปพลิเคชันโดยเฉพาะการรับค่าจากฟอร์มและการทำงานร่วมกับฐานข้อมูลในบทถัดไป หัวข้อหลักในบทนี้ประกอบด้วย

1. ความรู้เบื้องต้นเกี่ยวกับภาษา PHP
2. การสร้างไฟล์และไวยากรณ์ของภาษา PHP
3. ตัวแปรและชนิดข้อมูล
4. การแสดงผลด้วยฟังก์ชัน echo และ print
5. โครงสร้างการควบคุม
6. ฟังก์ชันและอาร์เรย์
7. สรุปท้ายบท
8. กิจกรรมท้ายบท

1. ความรู้เบื้องต้นเกี่ยวกับภาษา PHP

ภาษา PHP เป็นหนึ่งในภาษาสคริปต์ฝั่งผู้ให้บริการที่ได้รับความนิยมในการพัฒนาเว็บแอปพลิเคชันในวงกว้าง ด้วยความที่ภาษา PHP มีรูปแบบการเขียนบางส่วนที่คล้ายคลึงกับภาษา JavaScript อันเนื่องมาจากการได้รับอิทธิพลของภาษาต้นแบบของภาษาทั้งสอง จึงทำให้สามารถนำความรู้จากการเขียนภาษา JavaScript มาปรับใช้ในการเขียนภาษา PHP ได้เช่นกัน ในหัวข้อนี้จะนำเสนอให้ทราบถึงความหมาย บทบาท พัฒนาการและมาตรฐานที่เกี่ยวข้อง ดังนี้

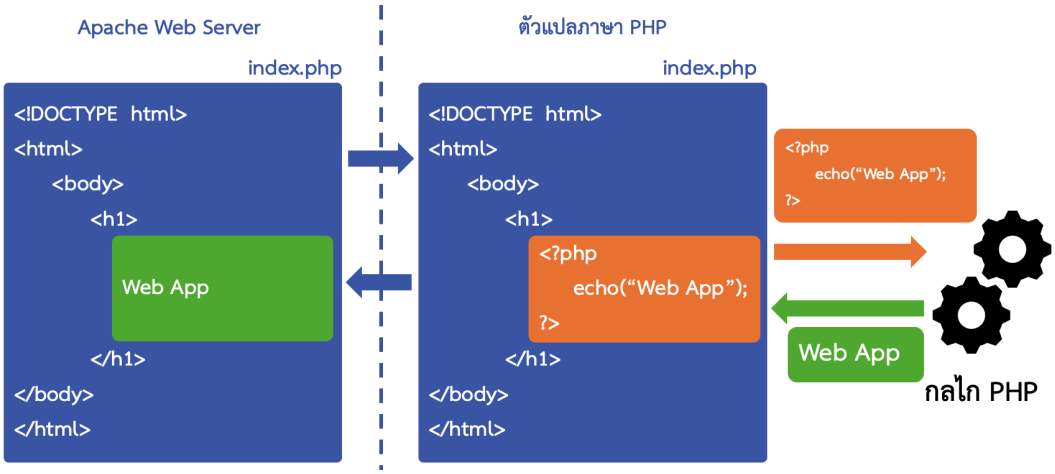
1.1 ความหมายและบทบาทของภาษา PHP

PHP เป็นคำย่อแบบเรียกซ้ำ (Recursive Acronym) ของคำว่า PHP: Hypertext Preprocessor เป็นภาษาคอมพิวเตอร์ที่จัดเป็นภาษาสคริปต์ที่เปิดเผยรหัส (Open Source Scripting Language) ที่ได้รับความนิยมอย่างกว้างขวาง โดยเฉพาะในการพัฒนาเว็บ ซึ่งรหัสคำสั่งของ PHP สามารถเขียนร่วมในเอกสารเว็บได้ (The PHP Documentation Group, n.d.)

ในบริบทของการพัฒนาเว็บแอปพลิเคชัน ภาษา PHP ทำหน้าที่เป็นสะพานเชื่อมระหว่างส่วนแสดงผลและฐานข้อมูล โดยรับข้อมูลจากผู้ใช้ผ่านแบบฟอร์มที่สร้างด้วยภาษาสำหรับเทคโนโลยีส่วนหน้า จากนั้นนำมาประมวลผลตามตรรกะทางธุรกิจ เรียกใช้หรือบันทึกข้อมูลในฐานข้อมูล และนำเสนอผลลัพธ์กลับสู่ผู้ใช้ในรูปแบบหน้าเว็บ กระบวนการเหล่านี้เกิดขึ้นในเครื่องผู้ให้บริการเว็บ ก่อนที่ผลลัพธ์จะถูกส่งไปยังเว็บเบราว์เซอร์ของผู้ใช้ตามกลไกการทำงานของเว็บแอปพลิเคชันที่ได้อธิบายไว้ในบทที่ 1

กลไกการแปลภาษาของ PHP มีลักษณะคล้ายกับภาษา JavaScript เนื่องจากอยู่ในกลุ่มภาษาสคริปต์เช่นกัน โดยรหัสคำสั่งจะถูกประมวลผลทีละบรรทัดในขณะที่มีการเรียกใช้งานจริง ไม่จำเป็นต้องคอมไพล์เป็นไฟล์ลักษณะอื่น อย่างไรก็ตาม PHP อาศัยตัวแปลภาษาที่ติดตั้งในเครื่องผู้ให้บริการเว็บ เช่น Zend Engine ซึ่งจะทำงานร่วมกับซอฟต์แวร์ให้บริการเว็บอย่าง Apache หรือ Nginx

หน้าที่หลักของซอฟต์แวร์ผู้ให้บริการเว็บในบริบทการทำงานร่วมกับตัวแปลภาษา PHP คือการจำแนกไฟล์ที่ผู้ใช้ร้องขอ หากไฟล์ที่ร้องขอมีนามสกุลเป็น .php (หรือนามสกุลอื่นตามการตั้งค่าของซอฟต์แวร์ผู้ให้บริการเว็บที่กำหนดให้จะต้องส่งต่อไปยังตัวแปลภาษา PHP) ซอฟต์แวร์ผู้ให้บริการเว็บก็จะไฟล์ดังกล่าวซึ่งภายไฟล์จะมีรหัสคำสั่งภาษา PHP ไปยังตัวแปลภาษา PHP ก่อนเพื่อทำการแปลและประมวลผลรหัสคำสั่ง เมื่อตัวแปลภาษาทำการประมวลผลเสร็จแล้ว จะส่งผลลัพธ์กลับคืนไปยังซอฟต์แวร์ผู้ให้บริการเว็บในรูปแบบโครงสร้าง HTML (หรือรูปแบบอื่นๆ เช่น JSON และ XML เป็นต้น ทั้งนี้จะขึ้นอยู่กับคำสั่งที่อยู่ภายในไฟล์ PHP ว่าต้องการผลลัพธ์ในรูปแบบใด แต่ค่าเริ่มของรูปแบบผลลัพธ์จะเป็นโครงสร้าง HTML) เมื่อผู้ให้บริการเว็บได้รับผลลัพธ์ที่ผ่านการประมวลผลของตัวแปลภาษา PHP แล้ว ก็จะส่งกลับไปยังโปรแกรมเว็บเบราว์เซอร์ของผู้ใช้เพื่อทำการประมวลผลโครงสร้างภาษา HTML ต่อไป ผู้ใช้จะไม่พบรหัส PHP ใด ๆ ที่อยู่ในไฟล์ต้นฉบับเลย กลไกการทำงานร่วมกันระหว่างซอฟต์แวร์ผู้ให้บริการเว็บและตัวแปลภาษา PHP ที่กล่าวถึงข้างต้น แสดงดังภาพที่ 11.1



ภาพที่ 11.1 กลไกการทำงานร่วมกันระหว่างซอฟต์แวร์ผู้ให้บริการเว็บและตัวแปลภาษา PHP

จุดเด่นของภาษา PHP อยู่ที่ความเรียบง่ายของภาษาที่เอื้อต่อการเรียนรู้ โครงสร้างภาษาที่ไม่ซับซ้อน รองรับทั้งรูปแบบการเขียนแบบขั้นตอน (Procedural) และแบบเชิงวัตถุ (Object-Oriented) อีกทั้งยังสามารถใช้งานร่วมกับระบบฐานข้อมูลได้หลากหลายประเภท มีไลบรารีและเฟรมเวิร์กจำนวนมากที่ช่วยอำนวยความสะดวกในการพัฒนา เช่น Laravel, Symfony และ Yii2 เป็นต้น นอกจากนี้ PHP ยังมีชุมชนผู้ใช้และเอกสารประกอบจำนวนมากที่จะช่วยสนับสนุนการเรียนรู้ได้เป็นอย่างดี

1.2 พัฒนาการและมาตรฐานของภาษา PHP

ภาษา PHP มีจุดเริ่มต้นในปี พ.ศ. 2538 โดย Rasmus Lerdorf นักพัฒนาชาวเดนมาร์ก-แคนาดา ซึ่งสร้างภาษานี้ขึ้นมาเพื่อใช้ในการพัฒนาเว็บเพจส่วนตัว โดยเริ่มจากชุดของสคริปต์ภาษา C ที่เรียกว่า Personal Home Page/Forms Interpreter หรือ PHP/FI ซึ่งมีความสามารถในการประมวลผลแบบฟอร์มและเชื่อมต่อกับฐานข้อมูลเบื้องต้น ซึ่งคุณสมบัติบางอย่างที่มาจากชุดสคริปต์ของภาษา C ได้ถ่ายทอดมายังการเขียนในภาษา PHP เช่น การใช้เครื่องหมายจุลภาค (;) ปิดท้ายประโยคคำสั่ง การใช้เครื่องหมายปีกกา ({ }) สำหรับจัดกลุ่มคำสั่ง รวมทั้งรูปแบบการประกาศตัวแปรที่คล้ายคลึงกัน (The PHP Documentation Group, n.d.; Zend Technologies, 2024)

การพัฒนาของภาษา PHP มีวิวัฒนาการสำคัญในหลายเวอร์ชัน โดย PHP 2.0 ซึ่งเปิดตัวในปี พ.ศ. 2540 เป็นเวอร์ชันแรกที่เผยแพร่สู่สาธารณะอย่างเป็นทางการ ต่อมาในปี พ.ศ. 2541 PHP 3.0 ได้รับการพัฒนาใหม่ทั้งหมดโดย Andi Gutmans และ Zeev Suraski พร้อมเปลี่ยนชื่อภาษาเป็น “PHP: Hypertext Preprocessor” ดังที่รู้จักกันในปัจจุบัน ในปี พ.ศ. 2543 ได้เปิดตัว PHP 4.0 โดยมีการนำ Zend Engine รุ่นแรกมาใช้เพื่อเพิ่มประสิทธิภาพการประมวลผล

ในปี พ.ศ. 2547 PHP 5.0 ได้รับการปรับปรุงครั้งใหญ่ โดยพัฒนา Zend Engine 2.0 ให้สามารถรองรับการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ได้อย่างมีประสิทธิภาพมากยิ่งขึ้น ต่อมาใน

ปี พ.ศ. 2558 PHP 7.0 ได้เปิดตัวพร้อม Zend Engine 3.0 ซึ่งปรับปรุงกลไกการจัดการหน่วยความจำใหม่ทั้งหมด และเพิ่มการสนับสนุนการกำหนดชนิดข้อมูล (Type Declarations) ที่ช่วยให้การเขียนรหัสคำสั่งมีความแม่นยำมากขึ้น

ในปี พ.ศ. 2563 PHP 8.0 ได้เพิ่มกลไก Just-In-Time (JIT) Compilation ที่ช่วยเพิ่มประสิทธิภาพในการประมวลผล พร้อมทั้งเพิ่มคุณลักษณะใหม่ของภาษา เช่น Union Types, Named Arguments และ Match Expression ซึ่งช่วยเพิ่มความยืดหยุ่นและความปลอดภัยในการพัฒนาเว็บแอปพลิเคชัน

การเรียกรุ่นของ PHP ใช้หมายเลขเวอร์ชันแบบ Major.Minor.Patch เช่น PHP 8.2.15 โดยการเปลี่ยน Major Version เกิดขึ้นเมื่อมีการเปลี่ยนแปลงใหญ่ที่อาจไม่สามารถย้อนกลับได้ ส่วน Minor Version เพิ่มขึ้นเมื่อมีคุณสมบัติใหม่ และ Patch Version สำหรับการแก้ไขข้อผิดพลาด

มาตรฐานและแนวปฏิบัติการเขียนโปรแกรม PHP อยู่ภายใต้การกำกับดูแลของ PHP Framework Interop Group (PHP-FIG) โดยเผยแพร่เป็น PHP Standards Recommendation (PSR) เพื่อให้เกิดความเข้ากันได้ระหว่างไลบรารีและเฟรมเวิร์กต่าง ๆ ปัจจุบันมีข้อกำหนด PSR หลายฉบับที่ครอบคลุมด้านต่าง ๆ ของการพัฒนา แต่ข้อกำหนดที่เป็นพื้นฐานสำคัญสำหรับผู้เริ่มเรียนรู้ ประกอบด้วย PSR-1 (Basic Coding Standard) ที่กำหนดกฎเกณฑ์พื้นฐานการในการเขียนรหัสคำสั่ง เช่น การใช้แท็ก PHP การตั้งชื่อคลาส เป็นต้น PSR-12 (Extended Coding Style) ที่กำหนดรูปแบบการเขียนรหัสคำสั่งให้สม่ำเสมอ ครอบคลุมเรื่องการย่อหน้า การใช้เครื่องหมายวงเล็บ และการจัดเรียงรหัสคำสั่ง (PHP-FIG, n.d.)

1.3 ความสามารถของภาษา PHP

ภาษา PHP ได้รับการออกแบบให้มีความยืดหยุ่นและครอบคลุมความสามารถที่จำเป็นต่อการพัฒนาเว็บแอปพลิเคชันแบบพลวัต โดยสามารถประมวลผลในฝั่งผู้ให้บริการเพื่อจัดการข้อมูลและสร้างเนื้อหาตามบริบทของผู้ใช้งาน ความสามารถสำคัญของภาษา PHP มีดังต่อไปนี้

1.3.1 ความสามารถในการสร้างเนื้อหาเว็บแบบพลวัต

ภาษา PHP สามารถสร้างและแสดงผลหน้าเว็บที่มีเนื้อหาเปลี่ยนแปลงได้ เช่น การแสดงชื่อผู้ใช้หลังเข้าสู่ระบบ หรือการเปลี่ยนเนื้อหาหน้าเว็บตามคำค้น เป็นต้น

1.3.2 ความสามารถในการจัดการไฟล์บนเครื่องผู้ให้บริการ

ภาษา PHP สามารถอ่าน เขียน และลบไฟล์ที่อยู่บนเครื่องผู้ให้บริการได้ ซึ่งมีประโยชน์ในการจัดการไฟล์ เช่น การอัปโหลดไฟล์เอกสาร การอัปโหลดรูปภาพ เป็นต้น

1.3.3 ความสามารถในการรับข้อมูลจากแบบฟอร์ม

ภาษา PHP สามารถรับข้อมูลที่ผู้ใช้กรอกผ่านแบบฟอร์มที่สร้างจากภาษา HTML และนำไปประมวลผล เช่น การลงทะเบียน การเข้าสู่ระบบ หรือการค้นหาข้อมูล เป็นต้น

1.3.4 ส่งและรับคุกกี้

ภาษา PHP รองรับการจัดการคุกกี้ ซึ่งเป็นกลไกในการจัดเก็บข้อมูลขนาดเล็กที่เกี่ยวข้องกับการใช้งานเว็บไซต์ เช่น การจดจำการเข้าสู่ระบบ หรือการตั้งค่าภาษาในการแสดงผลหน้าเว็บ

1.3.5 ติดต่อและจัดการฐานข้อมูล (Database Interaction)

ภาษา PHP สามารถเพิ่ม ลบ หรือแก้ไขข้อมูลในฐานข้อมูลได้ โดยรองรับการเชื่อมต่อกับฐานข้อมูลที่หลากหลาย เช่น MySQL, PostgreSQL และ SQLite เป็นต้น

1.3.6 ควบคุมการเข้าถึงของผู้ใช้ (User Access Control)

ภาษา PHP สามารถสร้างระบบตรวจสอบสิทธิ์ในการใช้งาน เช่น การกำหนดบทบาทของผู้ใช้ การเข้าสู่ระบบ และการจำกัดสิทธิ์เข้าถึงหน้าเว็บบางส่วน

1.3.7 การเข้ารหัสข้อมูล (Data Encryption)

ภาษา PHP มีฟังก์ชันสำหรับการเข้ารหัสข้อมูลเพื่อเพิ่มความปลอดภัยในการจัดการข้อมูลสำคัญ เช่น การเข้ารหัสข้อมูลรหัสผ่านก่อนการบันทึกในฐานข้อมูล เป็นต้น

2. การสร้างไฟล์และไวยากรณ์ของภาษา PHP

ก่อนการเขียนรหัสคำสั่งในภาษา PHP ผู้พัฒนาจำเป็นต้องจัดเตรียมเครื่องมือที่ใช้ในการประมวลผลคำสั่งในฝั่งผู้ให้บริการ โดยต้องติดตั้งกลไกผู้ให้บริการเว็บภายในเครื่อง ซึ่งประกอบด้วยตัวแปลภาษา PHP และโปรแกรมให้บริการเว็บ เช่น Apache หรือ Nginx เป็นต้น โดยในหนังสือเล่มนี้ ได้เสนอแนวทางการเตรียมสภาพแวดล้อมไว้ 2 แนวทาง ได้แก่ การใช้โปรแกรมสำเร็จรูปอย่าง XAMPP ที่กล่าวไว้ในบทที่ 3 และการใช้ Docker Desktop ที่อธิบายในบทที่ 4 ไม่ว่าผู้พัฒนาจะเลือกวิธีใด ไฟล์ PHP ที่เขียนขึ้นจะต้องถูกจัดวางไว้ในโฟลเดอร์ที่กำหนดให้เป็น Document Root เพื่อให้สามารถเข้าถึงผ่านเว็บเบราว์เซอร์ได้อย่างถูกต้อง

สำหรับเนื้อหาในหัวข้อนี้จะอธิบายถึงการสร้างไฟล์และไวยากรณ์ของภาษา PHP เพื่อให้เขียนเว็บแอปพลิเคชันด้วยภาษา PHP ได้ถูกต้อง ลดข้อผิดพลาดที่อาจเกิดขึ้นจากการเขียนรหัสคำสั่งที่ไม่เป็นไปตามไวยากรณ์ของภาษา

2.1 การสร้างไฟล์ภาษา PHP

ข้อกำหนดที่เกี่ยวข้องกับการสร้างไฟล์ของภาษา PHP ทั้งที่เป็นแนวปฏิบัติและข้อกำหนดตามมาตรฐาน PSR-1 และ PSR-12 มีดังนี้

2.1.1 นามสกุลของไฟล์

ในการเขียนรหัสคำสั่งภาษา PHP สามารถดำเนินการได้ในสองรูปแบบหลัก ได้แก่ การเขียนร่วมกับโครงสร้างเอกสาร HTML และการเขียนเฉพาะรหัสคำสั่ง PHP เพียงอย่างเดียว โดยไม่ว่าจะเลือกเขียนในรูปแบบใด ไฟล์ที่สร้างขึ้นจะต้องบันทึกด้วยนามสกุล .php เพื่อให้เครื่องผู้ให้บริการเว็บสามารถระบุและประมวลผลเนื้อหาของไฟล์ด้วยตัวแปลภาษา PHP ได้อย่างถูกต้อง แม้ว่าในทางเทคนิคจะสามารถตั้งนามสกุลเป็นอย่างอื่นได้ แต่การใช้งานในลักษณะดังกล่าวต้องมีการปรับแต่งค่าการทำงานของโปรแกรมให้บริการเว็บเพิ่มเติม ซึ่งอาจทำให้เกิดความซับซ้อนและไม่เหมาะสมกับผู้เริ่มต้นใช้งานภาษา PHP

2.1.2 การเข้ารหัสอักขระ

ตามมาตรฐาน PSR-1 ได้กำหนดไว้ว่าไฟล์ PHP ทุกไฟล์จะต้องบันทึกโดยใช้รหัสอักขระแบบ UTF-8 โดยไม่มี BOM (UTF-8 without BOM: Byte Order Mark) เท่านั้น ทั้งนี้ หากใช้รหัสอักขระอื่นโดยเฉพาะรหัสอักขระที่มี BOM อาจทำให้เกิดปัญหาในการประมวลผล การส่งข้อมูลหรือการแสดงผลบางอย่างได้ การตั้งค่ารหัสอักขระนี้สามารถกำหนดได้จากโปรแกรมเขียนรหัสคำสั่งที่ผู้พัฒนาใช้งาน

2.2 ไวยากรณ์ของภาษา PHP

ข้อกำหนดทางไวยากรณ์ของภาษา PHP ที่เป็นไปตามมาตรฐาน PSR-1 และ PSR-12 รวมทั้งแนวปฏิบัติอื่นๆ ที่สำคัญมีดังต่อไปนี้

2.2.1 การกำหนดแท็กเปิดและแท็กปิดของภาษา PHP

การเขียนรหัสคำสั่ง PHP จะต้องอยู่ภายในขอบเขตที่กำหนดโดย แท็กเปิด (Opening Tag) และแท็กปิด (Closing) ของภาษา PHP เพื่อให้ตัวแปลภาษาทำงานได้อย่างถูกต้อง โดยรูปแบบแท็กตามปกติเป็นดังนี้

แท็กเปิด	<?php	แท็กปิด	?>

ตัวแปลภาษา PHP จะทำการอ่านและประมวลผลเฉพาะเนื้อหาที่อยู่ระหว่างแท็ก <?php และ ?> เท่านั้น ส่วนเนื้อหานอกแท็กจะถูกมองว่าเป็นข้อความธรรมดาหรือ HTML ดังนั้นการเว้นวรรคหรือขึ้นบรรทัดใหม่หลังจากแท็กเปิด <?php จึงมีความสำคัญ เพื่อให้เกิดการแบ่งคำ (Token Separation) ได้อย่างถูกต้อง หากไม่มีช่องว่างอาจเกิดข้อผิดพลาดทางไวยากรณ์ (Syntax Error)

1) ข้อกำหนดของการเขียนคำสั่งภาษา PHP ร่วมกับเอกสารเว็บ

ในการเขียนรหัสคำสั่งภาษา PHP ร่วมกับเอกสารเว็บที่มีโครงสร้างของภาษา HTML จะต้องแทรกคำสั่ง PHP ไว้ภายในแท็กเปิด <?php และแท็กปิด ?> ดังที่กล่าวไว้ก่อนหน้านี้ โดยสามารถวางรหัส PHP ไว้ในตำแหน่งใดของเอกสารก็ได้ ทั้งภายในหรือภายนอกโครงสร้างภาษา HTML ขึ้นอยู่กับวัตถุประสงค์ของการประมวลผลข้อมูลหรือการแสดงผลของเว็บแอปพลิเคชัน โดยรูปแบบการเขียนแท็กเปิดและแท็กปิดของภาษา HTML สามารถเขียนได้ในสองรูปแบบดังนี้

2) การละแท็กปิดในไฟล์ PHP

ในกรณีที่ไฟล์ประกอบด้วยรหัสคำสั่งของภาษา PHP เพียงอย่างเดียว โดยไม่มีการแทรกเนื้อหา HTML หรือข้อความอื่น ๆ การละเว้นการเขียนแท็กปิด ?> ที่ส่วนท้ายของไฟล์ถือเป็นแนวปฏิบัติที่แนะนำตามมาตรฐาน PSR ทั้งนี้เพื่อหลีกเลี่ยงปัญหาที่อาจเกิดจากช่องว่าง (Whitespace) หรือบรรทัดว่างที่แทรกโดยไม่ตั้งใจหลังแท็กปิด

2.2.2 การเขียนประโยคคำสั่ง (Statement Syntax)

การเขียนประโยคคำสั่งในภาษา PHP จำเป็นต้องปิดท้ายประโยคด้วยเครื่องหมายอัฒภาค (;) เสมอ โดยสามารถเขียนได้ทั้งแบบหลายบรรทัดและแบบบรรทัดเดียว ดังรูปแบบต่อไปนี้

รูปแบบที่ 1

```
<?php
    คำสั่งที่ 1;
    คำสั่งที่ ...;
?>
```

รูปแบบที่ 2

```
<?php  คำสั่งที่ 1;  ?>
```

ในรูปแบบที่ 1 เป็นการเขียนรหัสแบบแยกบรรทัด โดยแท็กเปิด <?php และแท็กปิด ?> จะอยู่คนละบรรทัดกับคำสั่ง PHP และแต่ละคำสั่งต้องสิ้นสุดด้วยเครื่องหมายอัฒภาค เพื่อให้ตัวแปลภาษาสามารถประมวลผลได้ถูกต้อง

ส่วนรูปแบบที่ 2 เป็นการเขียนคำสั่งในบรรทัดเดียว โดยรวมแท็กเปิด คำสั่ง และแท็กปิดไว้ในบรรทัดเดียวกัน ซึ่งสามารถทำงานได้เช่นเดียวกัน แต่ต้องเว้นวรรคระหว่างแท็กกับคำสั่งเพื่อให้เกิดการแบ่งคำอย่างเหมาะสม อย่างไรก็ตาม การเขียนหลายประโยคคำสั่งในบรรทัดเดียว เช่น <?php คำสั่ง; คำสั่ง; ...; คำสั่ง; ?> แม้จะไม่ทำให้เกิดข้อผิดพลาดทางไวยากรณ์ แต่ขัดต่อแนวทางการจัดรูปแบบรหัสที่แนะนำในมาตรฐาน PSR-12 ซึ่งระบุว่า "ต้องไม่มีประโยคคำสั่งมากกว่าหนึ่งคำสั่งในหนึ่งบรรทัด (There MUST NOT be more than one statement per line)" เพื่อความสะดวกในการอ่าน วิเคราะห์ บำรุงรักษาและการทำงานร่วมกัน

ข้อควรระวังอีกประการหนึ่ง คือ หากมีการลืมใส่เครื่องหมายอัฒภาคที่จุดสิ้นสุดของประโยคคำสั่ง โดยเฉพาะเมื่อเขียนรหัสภาษา PHP ร่วมกับโครงสร้าง HTML อาจส่งผลให้เกิดข้อผิดพลาดขณะประมวลผล ซึ่งอาจแสดงเป็นหน้าว่างหรือข้อความผิดพลาด (Error Message) บนเว็บเบราว์เซอร์

2.2.3 การเขียนบรรทัดหมายเหตุ (Comments)

การเขียนบรรทัดทัดหมายเหตุหรือคอมเมนต์ (Comments) มีวัตถุประสงค์หลักเพื่อช่วยให้รหัสคำสั่งอ่านเข้าใจได้ง่ายขึ้น โดยเฉพาะในการอธิบายการทำงานของประโยคคำสั่ง เงื่อนไขหรือโครงสร้างที่ซับซ้อน รวมถึงใช้ในการปิดกั้นบางส่วนของรหัสเพื่อการทดสอบโปรแกรม ภาษา PHP มีรูปแบบการเขียนคำอธิบาย 3 รูปแบบ ดังนี้

รูปแบบที่ 1

```
// ข้อความหรือประโยคคำสั่งที่ต้องการทำเป็นบรรทัดหมายเหตุ
```

รูปแบบที่ 2

```
# ข้อความหรือประโยคคำสั่งที่ต้องการทำเป็นบรรทัดหมายเหตุ
```

จากรูปแบบการเขียนบรรทัดหมายเหตุรูปแบบที่ 1 และ 2 เป็นการเขียนบรรทัดหมายเหตุแบบบรรทัดเดียวแต่ใช้เครื่องหมายต่างกันคือ // และ # ซึ่งจะทำให้ข้อความที่อยู่ด้านหลังเครื่องหมายดังกล่าวในบรรทัดเดียวกันเป็นข้อความหมายเหตุที่ตัวแปลภาษาจะไม่ทำการประมวลผล

รูปแบบที่ 3

```
/*
ข้อความหรือประโยคคำสั่งที่ต้องการทำเป็นบรรทัดหมายเหตุบรรทัดที่ 1
ข้อความหรือประโยคคำสั่งที่ต้องการทำเป็นบรรทัดหมายเหตุบรรทัดที่ ...
*/
```

รูปแบบที่ 3 เป็นการเขียนบรรทัดหมายเหตุแบบหลายบรรทัดซึ่งมีการกำหนดจุดเริ่มต้นด้วย /* และจุดสิ้นสุดด้วย */ ทั้งนี้ข้อควรระหว่างในการเขียนบรรทัดคำสั่งแบบหลายบรรทัดคือการเขียนบรรทัดคำสั่งแบบหลายบรรทัดซ้อนกัน (Nested Comment) เพราะอาจทำให้ตัวแปลภาษา PHP แปลผิดพลาดหรือหยุดทำงานได้

2.2.4 การเขียนคำสำคัญ คำสั่งและฟังก์ชันในภาษา PHP

คำสำคัญในภาษา PHP เช่น if, else, return, function, true และ false เป็นต้น จะต้องเขียนด้วยตัวพิมพ์เล็ก (Lowercase) ตามข้อกำหนดของ PSR-12 ถึงแม้ว่าคำสำคัญเหล่านี้จะไม่มีความแตกต่างในด้านตัวพิมพ์หรือเรียกว่าไม่ไวต่อตัวพิมพ์ (Case-Insensitive) กล่าวคือ การพิมพ์ IF และ if ต่างก็สามารถใช้งานได้ไม่มีความแตกต่างกันในการทำงาน แต่เพื่อให้รหัสคำสั่งอ่านง่ายและสอดคล้องกันทั้งระบบจึงควรเขียนด้วยตัวอักษรพิมพ์เล็กทั้งหมด

นอกจากนี้ชื่อฟังก์ชันและชื่อคำสั่งต่างก็ไม่มีความแตกต่างด้านตัวพิมพ์ด้วยเช่นกัน ยกเว้นชื่อตัวแปรที่มีความไวต่อตัวพิมพ์ (Case-Sensitive) กล่าวคือ ถึงแม้จะพิมพ์ชื่อเดียวกันแต่ใช้ตัวอักษรพิมพ์เล็กพิมพ์ใหญ่ต่างกันจะถือว่าเป็นตัวแปรคนละตัวกัน เช่น $Nok กับ $NOK เป็นตัวแปรคนละตัวกัน

2.3 นิพจน์และตัวดำเนินการ

นิพจน์และตัวดำเนินเป็นหน่วยพื้นฐานของประโยคคำสั่งที่ใช้ในการคำนวณ เปรียบเทียบ หรือควบคุมค่าของตัวแปร เพื่อให้โปรแกรมสามารถตอบสนองตามตรรกะที่กำหนดได้อย่างถูกต้อง

2.3.1 นิพจน์

นิยามของนิพจน์ในภาษา PHP มีหลักการเดียวกันกับที่ได้นำเสนอไว้ในบทที่ 9 หัวข้อ 2.2.1 เรื่องนิพจน์ของภาษา JavaScript กล่าวคือ นิพจน์หมายถึงหน่วยคำสั่งที่สามารถประเมินค่าได้ ซึ่งอาจ

ประกอบด้วยตัวแปร ค่าคงที่ ตัวดำเนินการ หรือฟังก์ชัน โดยมีวัตถุประสงค์ในการใช้งานเพื่อให้ได้ค่าผลลัพธ์ที่สามารถนำไปใช้ในการกำหนดค่า แสดงผล หรือเงื่อนไขอื่น ๆ ในโปรแกรม ประเภทของนิพจน์ในภาษา PHP มีดังนี้

1) นิพจน์เชิงคำนวณ

นิพจน์เชิงคำนวณเป็นนิพจน์ที่ใช้ตัวดำเนินการทางคณิตศาสตร์ในการคำนวณค่าต่าง ๆ เช่น การบวก ลบ คูณ หาร และให้ผลลัพธ์เป็นตัวเลข

2) นิพจน์เปรียบเทียบ

นิพจน์เปรียบเทียบเป็นนิพจน์ที่ใช้เปรียบเทียบความสัมพันธ์ระหว่างค่าสองค่า เช่น เท่ากัน มากกว่า น้อยกว่า และให้ผลลัพธ์เป็นค่าตรรกะ

3) นิพจน์ตรรกะ

นิพจน์ตรรกะเป็นนิพจน์ที่ใช้ตัวดำเนินการทางตรรกะเพื่อรวมหรือปรับเปลี่ยนเงื่อนไขต่าง ๆ เช่น AND, OR, NOT และให้ผลลัพธ์เป็นค่าตรรกะ

4) นิพจน์กำหนดค่า

นิพจน์กำหนดค่าเป็นนิพจน์ที่ใช้กำหนดค่าให้กับตัวแปร ทั้งการกำหนดค่าธรรมดาและการกำหนดค่าพร้อมคำนวณ เช่น +=, -=, *= เป็นต้น

5) นิพจน์ข้อความ

นิพจน์ข้อความเป็นนิพจน์ที่ใช้ตัวดำเนินการต่อข้อความ (.) เพื่อรวมข้อความหลายส่วนเข้าด้วยกันเป็นข้อความเดียว

6) นิพจน์เรียกใช้ฟังก์ชัน

นิพจน์เรียกใช้ฟังก์ชันเป็นนิพจน์ที่เกิดจากการเรียกใช้ฟังก์ชันและได้รับค่าผลลัพธ์จากการทำงานของฟังก์ชันนั้น

2.3.2 ตัวดำเนินการ

ตัวดำเนินการ คือสัญลักษณ์หรือคำสำคัญที่ใช้ในการดำเนินการกับข้อมูล เช่น การคำนวณทางคณิตศาสตร์ การเปรียบเทียบค่า การกำหนดค่า และการดำเนินการทางตรรกะ ตัวดำเนินการจะทำงานร่วมกับตัวถูกดำเนินการ (Operand) ซึ่งอาจเป็นค่าคงที่ ตัวแปร หรือผลลัพธ์จากนิพจน์อื่น ภาษา PHP มีตัวดำเนินการหลายประเภทตามลักษณะการใช้งาน ดังนี้

1) ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการทางคณิตศาสตร์ใช้สำหรับการคำนวณทางคณิตศาสตร์พื้นฐาน ตัวดำเนินการทางคณิตศาสตร์ในภาษา PHP แสดงดังตารางที่ 11-1

ตารางที่ 11-1 ตัวดำเนินการทางคณิตศาสตร์ในภาษา PHP

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ผลลัพธ์
+	บวก	$a + $b	ผลบวกของ $a และ $b

ตารางที่ 11-1 (ต่อ)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ผลลัพธ์
-	ลบ	$a - $b	ผลต่างของ $a และ $b
*	คูณ	$a * $b	ผลคูณของ $a และ $b
/	หาร	$a / $b	ผลหารของ $a และ $b
%	หารเอาเศษ (Modulo)	$a % $b	เศษจากการหาร $a ด้วย $b
**	ยกกำลัง	$a ** $b	$a ยกกำลัง $b

2) ตัวดำเนินการเปรียบเทียบ

ตัวดำเนินการเปรียบเทียบใช้สำหรับการเปรียบเทียบค่าระหว่างตัวแปรหรือค่าคงที่ โดยผลลัพธ์จะเป็นค่าตรรกะ คือ true หรือ false ตัวดำเนินการเปรียบเทียบในภาษา PHP แสดงดังตารางที่ 11-2

ตารางที่ 11-2 ตัวดำเนินการเปรียบเทียบในภาษา PHP

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
==	เท่ากับ (ไม่พิจารณาชนิดข้อมูล)	$a == $b
===	เท่ากับ (พิจารณาชนิดข้อมูล)	$a === $b
!= หรือ <>	ไม่เท่ากับ (ไม่พิจารณาชนิดข้อมูล)	$a != $b
!==	ไม่เท่ากับ (พิจารณาชนิดข้อมูล)	$a !== $b
<	น้อยกว่า	$a < $b
>	มากกว่า	$a > $b
<=	น้อยกว่าหรือเท่ากับ	$a <= $b
>=	มากกว่าหรือเท่ากับ	$a >= $b

3) ตัวดำเนินการทางตรรกะ

ตัวดำเนินการทางตรรกะใช้สำหรับการรวมหรือปรับเปลี่ยนเงื่อนไขทางตรรกะมักใช้ร่วมกับตัวดำเนินการเปรียบเทียบ ตัวดำเนินการทางตรรกะในภาษา PHP แสดงดังตารางที่ 11-3

ตารางที่ 11-3 ตัวดำเนินการทางตรรกะในภาษา PHP

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ผลลัพธ์เป็น true เมื่อ
&& หรือ and	และ (AND)	$a && $b	ทั้ง $a และ $b เป็น true
|| หรือ or	หรือ (OR)	$a || $b	$a หรือ $b อย่างน้อยหนึ่งตัวเป็น true
! หรือ not	ไม่ (NOT)	!$a	$a เป็น false

ตารางที่ 11-3 (ต่อ)

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	ผลลัพธ์เป็น true เมื่อ
xor	หรือพิเศษ (XOR)	$a xor $b	$a และ $b มีค่าต่างกัน

4) ตัวดำเนินการกำหนดค่า

ตัวดำเนินการกำหนดค่าใช้สำหรับการกำหนดค่าให้กับตัวแปร รวมถึงการดำเนินการและกำหนดค่าในขั้นตอนเดียว ตัวดำเนินการกำหนดค่าในภาษา PHP แสดงตารางที่ 11-4

ตารางที่ 11-4 ตัวดำเนินการกำหนดค่าในภาษา PHP

ตัวดำเนินการ	ความหมาย	ตัวอย่าง	เทียบเท่ากับ
=	กำหนดค่า	$a = $b	$a = $b
+=	บวกแล้วกำหนดค่า	$a += $b	$a = $a + $b
-=	ลบแล้วกำหนดค่า	$a -= $b	$a = $a - $b
*=	คูณแล้วกำหนดค่า	$a *= $b	$a = $a * $b
/=	หารแล้วกำหนดค่า	$a /= $b	$a = $a / $b
%=	หารเอาเศษแล้วกำหนดค่า	$a %= $b	$a = $a % $b
.=	ต่อข้อความแล้วกำหนดค่า	$a .= $b	$a = $a . $b
.=	ต่อข้อความแล้วกำหนดค่า	$a .= $b	$a = $a . $b

2.4 ตัวดำเนินการเพิ่ม/ลดค่า (Increment/Decrement Operators)

ตัวดำเนินการเพิ่ม/ลดค่า เป็นตัวดำเนินการที่ใช้สำหรับการเพิ่มค่าหรือลดค่าของตัวแปรทีละ 1 หน่วย โดยเป็นรูปแบบของการดำเนินการแบบย่อซึ่งนิยมใช้ในโครงสร้างควบคุมแบบวนซ้ำ และสถานการณ์ที่ต้องการเพิ่มหรือลดค่าของตัวแปรอย่างต่อเนื่อง ตัวดำเนินการประเภทนี้แบ่งออกเป็น 2 กลุ่ม คือ ตัวดำเนินการเพิ่มค่า (Increment) และตัวดำเนินการลดค่า (Decrement) ซึ่งแต่ละกลุ่มสามารถใช้งานได้ทั้งแบบนำหน้าหรือตามหลังตัวแปรก็ได้ แต่ผลลัพธ์ของแต่ละแบบจะส่งผลต่อลำดับการทำงานไม่เหมือนกัน ดังตารางที่ 11-5

ตารางที่ 11-5 ตัวดำเนินการเพิ่ม/ลดค่าในภาษา PHP

ตัวดำเนินการ	ความหมาย	ตัวอย่าง
++$a	เพิ่มค่าก่อนใช้ (Pre-increment)	++$a
$a++	เพิ่มค่าหลังใช้ (Post-increment)	$a++
--$a	ลดค่าก่อนใช้ (Pre-decrement)	--$a
$a--	ลดค่าหลังใช้ (Post-decrement)	$a--

3. ตัวแปรและชนิดข้อมูล

ในการเขียนโปรแกรมด้วยภาษาประมวลผลต่าง ๆ ตัวแปรและชนิดของข้อมูลมีบทบาทสำคัญอย่างยิ่งในการจัดเก็บ จัดการ และประมวลผลข้อมูลที่ใช้ภายในโปรแกรม โดยทั่วไปแล้ว ภาษา PHP เป็นภาษาที่ใช้การกำหนดชนิดข้อมูลพลวัตลักษณะเดียวกันกับภาษา JavaScript กล่าวคือ ไม่จำเป็นต้องระบุชนิดข้อมูลของตัวแปรอย่างชัดเจนขณะประกาศตัวแปร ซึ่งตัวแปลภาษา PHP จะตรวจสอบและกำหนดชนิดของข้อมูลให้โดยอัตโนมัติตามค่าที่ถูกกำหนดให้กับตัวแปรในขณะทำงาน

3.1 ตัวแปรในภาษา PHP

ตัวแปรในภาษา PHP เป็นองค์ประกอบสำคัญที่ใช้ในการเก็บข้อมูลเพื่อประมวลผลในภายหลัง โดยข้อมูลที่เก็บไว้ในตัวแปรสามารถเปลี่ยนแปลงค่าได้ตลอดระยะเวลาการทำงานของโปรแกรม ลักษณะเด่นของตัวแปรในภาษา PHP คือสามารถใช้งานได้โดยไม่ต้องประกาศชนิดข้อมูลอย่างชัดเจนล่วงหน้า ซึ่งเป็นคุณสมบัติของภาษาที่กำหนดชนิดข้อมูลแบบพลวัต

3.1.1 การสร้างและกำหนดค่าให้ตัวแปร

การสร้างตัวแปรในภาษา PHP ไม่ต้องใช้คำสงวนหรือคำสั่งเฉพาะในการประกาศตัวแปรเหมือนในภาษา JavaScript ที่จะต้องใช้คำว่า var, let, หรือ const การสร้างตัวแปรในภาษา PHP จะใช้เครื่องหมายดอลลาร์ ($) ตามด้วยชื่อตัวแปรที่ต้องการ และใช้เครื่องหมายเท่ากับ (=) ในการกำหนดค่า โดยมีรูปแบบดังต่อไปนี้

รูปแบบ

$ชื่อตัวแปร = ค่า;

ตัวอย่างที่ 1

```
1  $name = "Nok";
2  $age = 25;
3  $department;
```

จากตัวอย่างที่ 1 แสดงการสร้างตัวแปรจำนวน 3 ตัว โดยตัวแปรตัวแรกคือ $name มีการกำหนดค่าชนิดข้อความที่มีค่าเป็น *Nok* ให้กับตัวแปร ตัวแปรตัวที่ 2 คือ $age มีการกำหนดค่าเป็นตัวเลขที่มีค่าเป็น 25 ให้กับตัวแปร และตัวแปรตัวที่ 3 มีการสร้างตัวแปรขึ้นมาแต่ไม่ได้กำหนดค่าใดๆ ให้ ในกรณีนี้ ค่าในตัวแปรจะมีค่าเป็น null หมายความว่า มีการสร้างตัวแปร แต่ไม่มีค่าใดๆ

3.1.2 ข้อกำหนดการตั้งชื่อตัวแปร

จากการสร้างตัวแปรข้างต้น ชื่อตัวแปรต้องตั้งขึ้นตามกฎการตั้งชื่อของภาษา PHP ซึ่งมีข้อกำหนดดังนี้

- ชื่อตัวแปรต้องเริ่มต้นด้วยเครื่องหมาย $ เสมอ

- ตัวอักษรตัวแรกหลังเครื่องหมาย $ ต้องเป็นตัวอักษร a-z หรือ A-Z หรือขีดล่าง
- ตัวอักษรที่เหลือหลังจากตัวอักษรตัวแรกสามารถเป็นตัวอักษร ตัวเลข 0-9 หรือขีดล่าง
- ชื่อตัวแปรมีลักษณะไวต่อตัวพิมพ์ใหญ่-เล็ก
- ไม่สามารถใช้คำสงวน (Reserved Words) ของภาษา PHP

ตัวอย่างของชื่อที่ถูกต้องตามข้อกำหนดของการตั้งชื่อตัวแปร เช่น $vat $price และ $firstname เป็นต้น ส่วนตัวอย่างของการตั้งชื่อตัวแปรที่ไม่เป็นไปตามข้อกำหนด เช่น $5students ไม่ถูกต้อง เนื่องจากขึ้นต้นด้วยตัวเลขซึ่งตัวอักษรตัวแรกจะต้องเป็นตัวอักษร a-z หรือ A-Z หรือขีดล่าง (_) หรือ $first name เป็นชื่อที่ไม่ถูกต้องเนื่องจากมีช่องว่างในชื่อซึ่งไม่เป็นไปตามข้อกำหนดที่ว่า ตัวอักษรที่เหลือหลังจากตัวอักษรตัวแรกสามารถเป็นตัวอักษร ตัวเลข 0-9 หรือขีดล่างได้ เป็นต้น

ถึงแม้ว่า PSR จะไม่มีข้อกำหนดในการตั้งชื่อตัวแปร แต่แนวปฏิบัติที่ดีในการตั้งชื่อตัวแปรในทุกภาษาที่มีเหมือนกันคือการตั้งชื่อให้สอดคล้องกับข้อมูลที่จัดเก็บ และในผู้เขียนภาษา PHP ส่วนใหญ่มักใช้รูปแบบการตั้งชื่อแบบ $camelCase หรือ $snake_case

3.2 ชนิดของข้อมูลในภาษา PHP

ชนิดของข้อมูล (Data Type) ในภาษา PHP เป็นแนวคิดพื้นฐานที่ระบุลักษณะและคุณสมบัติของข้อมูลที่จัดเก็บในตัวแปร โดยมีความสำคัญต่อการประมวลผลและการทำงานของ ตัวแปลภาษา PHP ในการพิจารณาว่าสามารถดำเนินการใดกับข้อมูลนั้นได้บ้าง เช่น การคำนวณทางคณิตศาสตร์ การเปรียบเทียบค่า การแปลงประเภท หรือการแสดงผล เป็นต้น รวมทั้งมีความสัมพันธ์กับการประกาศตัวแปรดังที่ได้กล่าวถึงในหัวข้อก่อนหน้านี้

3.2.1 การจัดการชนิดข้อมูลในภาษา PHP

ภาษา PHP เป็นภาษาที่ใช้ระบบกำหนดชนิดข้อมูลแบบไม่ตายตัว (Dynamic Type System) เช่นเดียวกับภาษา JavaScript ซึ่งแตกต่างจากภาษาที่ใช้ระบบชนิดข้อมูลแบบตายตัว (Static Type System) อย่างเช่น Java หรือ C++ เป็นต้น

ภาษา PHP ใช้การแปลงชนิดข้อมูลแบบอ่อน (Weak Typing หรือ Type Juggling) กล่าวคือ ภาษา PHP สามารถแปลงชนิดของข้อมูลโดยอัตโนมัติเมื่อนำไปใช้ในบริบทที่ต่างกัน ตัวอย่างเช่น นิพจน์ "123" + 456 ประกอบด้วยค่าคงที่ชนิดข้อความและตัวเลข แต่ใช้ตัวดำเนินการบวก ในกรณีนี้ PHP Engine จะแปลงข้อความให้เป็นตัวเลขและดำเนินการคำนวณ ทำให้ได้ผลลัพธ์เป็น 579 เป็นต้น

ตัวอย่างที่ 2

```
1  <?php
2      $value = "100";
3      echo $value + 50;
4      echo gettype($value);
5  ?>
```

จากตัวอย่างที่ 2 แสดงให้เห็นถึงลักษณะการแปลงชนิดข้อมูลอัตโนมัติของภาษา PHP โดยในบรรทัดที่ 6 มีการกำหนดค่าข้อความ *100* ให้กับตัวแปร $value ส่วนบรรทัดที่ 3 เมื่อนำตัวแปรนี้มาคำนวณกับตัวเลข 50 ตัวแปลภาษา PHP จะแปลงข้อความให้เป็นตัวเลขชั่วคราวเพื่อการคำนวณและแสดงผล 150 แต่ชนิดข้อมูลดั้งเดิมของตัวแปรไม่เปลี่ยนแปลง ซึ่งทำการตรวจสอบชนิดข้อมูลโดยใช้ฟังก์ชัน gettype() ในบรรทัดที่ 4 โดยฟังก์ชันดังกล่าวจะให้ผลลัพธ์เป็นชื่อชนิดของข้อมูล ซึ่งผลลัพธ์ที่ได้ในตัวอย่างเป็น string

3.2.2 ข้อมูลประเภทสเกลาร์

ข้อมูลประเภทสเกลาร์ (Scalar Data Types) เป็นข้อมูลที่เก็บค่าเดี่ยว (Single Value) โดยตรงในหน่วยความจำ และมีลักษณะการจัดการแบบผ่านค่า (Pass-By-Value) กล่าวคือ เมื่อนำค่าไปกำหนดให้ตัวแปรอื่นหรือส่งเป็นพารามิเตอร์ให้ฟังก์ชัน จะมีการสำเนาค่าจริงไปยังตำแหน่งใหม่ ไม่ใช่การอ้างอิงถึงตำแหน่งเดียวกัน แบ่งเป็น 5 ชนิด ดังนี้

1) บูลีน

บูลีน (Boolean) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บค่าความจริง โดยมีค่าได้เพียง 2 ค่าเท่านั้น คือ true และ false

2) จำนวนเต็ม

จำนวนเต็ม (Integer) คือชนิดข้อมูลที่ใช้แทนจำนวนเต็มบวก ลบ หรือศูนย์ โดยไม่มีจุดทศนิยม ในภาษา PHP ขนาดของจำนวนเต็มขึ้นอยู่กับแพลตฟอร์มของระบบ โดยทั่วไปในระบบ 32 บิต จะสามารถเก็บค่าได้ในช่วงประมาณ -2,147,483,648 ถึง 2,147,483,647 ส่วนในระบบ 64 บิต จะเก็บค่าได้ในช่วงที่กว้างกว่า

3) จำนวนทศนิยม

จำนวนทศนิยม (Float) หรือที่เรียกอีกชื่อหนึ่งว่า double คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บจำนวนที่มีจุดทศนิยม โดย PHP ใช้มาตรฐาน IEEE 754 แบบเลขทศนิยมความแม่นยำสองเท่า (Double-Precision Floating Point) ในการจัดเก็บข้อมูล ทำให้สามารถเก็บค่าที่มีความแม่นยำสูงและช่วงค่าที่กว้าง จำนวนทศนิยมสามารถเขียนในรูปแบบต่าง ๆ ได้ เช่น รูปแบบทศนิยมธรรมดา และรูปแบบเลขยกกำลัง (Scientific Notation) เป็นต้น ข้อมูลชนิดนี้เหมาะสำหรับการคำนวณทางคณิตศาสตร์ที่ต้องการความแม่นยำ การคำนวณทางวิทยาศาสตร์ และการประมวลผลข้อมูลทางการเงิน

4) ข้อความ

ข้อความ (String) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บข้อความหรือลำดับของอักขระ เช่น ตัวอักษร ตัวเลข สัญลักษณ์ และช่องว่าง โดย PHP ใช้การเข้ารหัสแบบ UTF-8 เป็นค่าเริ่มต้น ทำให้สามารถรองรับอักขระจากหลากหลายภาษา รวมถึงภาษาไทยและสัญลักษณ์พิเศษ

ในภาษา PHP มีวิธีการสร้างข้อความหลายรูปแบบ โดยแต่ละรูปแบบมีลักษณะการทำงานที่แตกต่างกัน ประกอบด้วย การใช้เครื่องหมายคำพูดเดี่ยว (Single Quote) การใช้เครื่องหมายคำพูดคู่ (Double Quote) และการใช้ Heredoc และ Nowdoc สำหรับข้อความยาวหลายบรรทัด

3.2.3 ข้อมูลประเภทรวม

ข้อมูลประเภทรวม (Compound Data Types) เป็นข้อมูลที่มีโครงสร้างซับซ้อนกว่าข้อมูลประเภทสเกลาร์ โดยสามารถจัดเก็บข้อมูลหลายค่าหรือมีโครงสร้างที่ซับซ้อนได้ ตัวแปรจะเก็บการอ้างอิง (Reference) หรือที่อยู่ในหน่วยความจำที่ชี้ไปยังตำแหน่งจริงของข้อมูลแทนการเก็บค่าจริงโดยตรง

1) อาร์เรย์

อาร์เรย์ (Array) คือชนิดข้อมูลที่ใช้สำหรับจัดเก็บข้อมูลหลายค่าในตัวแปรเดียว โดยแต่ละค่าจะมีดัชนี (Index) หรือคีย์ (Key) เป็นตัวระบุตำแหน่ง ในภาษา PHP อาร์เรย์มีความยืดหยุ่นสูง สามารถเก็บข้อมูลต่างชนิดกันได้ในอาร์เรย์เดียวกัน และสามารถใช้ทั้งดัชนีตัวเลขและข้อความเป็นคีย์ได้

อาร์เรย์ใน PHP แบ่งออกเป็น 3 ประเภทหลัก คือ อาร์เรย์แบบมีดัชนีตัวเลข (Indexed Array) เป็นอาร์เรย์ที่ใช้ตัวเลขจำนวนเต็มเริ่มต้นที่ 0 เป็นดัชนีในการเข้าถึงค่าที่จัดเก็บภายในอาร์เรย์ อาร์เรย์แบบเชื่อมโยง (Associative Array) ที่ใช้ข้อความแทนตัวเลขในการเป็นคีย์ เพื่อเข้าถึงค่าที่เก็บไว้ และ อาร์เรย์หลายมิติ (Multidimensional Array) เป็นอาร์เรย์ที่มีการซ้อนอาร์เรย์ภายในอีกชั้นหนึ่งหรือมากกว่า เพื่อจัดเก็บข้อมูลที่มีโครงสร้างซับซ้อน เช่น ตาราง หรือข้อมูลเชิงโครงสร้าง เป็นต้น

2) ออบเจกต์

ออบเจกต์ (Object) คือชนิดข้อมูลที่สร้างขึ้นจากคลาส (Class) ซึ่งเป็นแม่แบบสำหรับการสร้างออบเจกต์ ออบเจกต์ประกอบด้วยคุณสมบัติ (Properties) สำหรับเก็บข้อมูล และเมธอด (Methods) สำหรับดำเนินการกับข้อมูลนั้น แนวคิดนี้เป็นพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ (Object-Oriented Programming) ซึ่งมีความสำคัญอย่างยิ่งในการใช้งาน Laravel Framework

3.2.4 ข้อมูลประเภทพิเศษ

ข้อมูลประเภทพิเศษในภาษา PHP มีดังต่อไปนี้

1) ค่าว่าง

ค่าว่าง (NULL) คือชนิดข้อมูลพิเศษที่ใช้สำหรับแสดงถึงค่าว่างที่ถูกกำหนดขึ้นอย่างตั้งใจ หรือตัวแปรที่ยังไม่ได้รับการกำหนดค่า ค่า NULL มีค่าเพียงค่าเดียวคือ null (ไม่สนใจตัวพิมพ์ใหญ่-เล็ก) และมักใช้ในการเริ่มต้นตัวแปรที่จะเก็บออบเจกต์ การรีเซ็ตค่าตัวแปร หรือการแสดงให้เห็นว่าไม่มีข้อมูลในขณะนั้น

2) ทรัพยากร

ทรัพยากร (Resource) คือชนิดข้อมูลพิเศษที่ใช้อ้างอิงถึงทรัพยากรภายนอก เช่น การเชื่อมต่อฐานข้อมูล การเปิดไฟล์ การเชื่อมต่อเครือข่าย หรือการประมวลผลรูปภาพ เป็นต้น ทรัพยากรไม่สามารถสร้างขึ้นโดยตรงจากรหัสคำสั่งภาษา PHP ได้ แต่จะถูกสร้างและคืนค่าโดยฟังก์ชันเฉพาะ

3.3 การตรวจสอบชนิดข้อมูล

การตรวจสอบชนิดข้อมูลเป็นเทคนิคสำคัญในการพัฒนาเว็บแอปพลิเคชันด้วย PHP โดยเฉพาะเมื่อต้องการตรวจสอบความถูกต้องของข้อมูลหรือการดำเนินการที่แตกต่างกันตามชนิดของข้อมูล PHP มีฟังก์ชันสำหรับการตรวจสอบชนิดข้อมูล แสดงดังตารางต่อไปนี้

ตารางที่ 11-6 รายการฟังก์ชันสำหรับการตรวจสอบข้อมูล

ฟังก์ชันตรวจสอบ	รายละเอียด
is_int() หรือ is_integer()	ตรวจสอบว่าข้อมูลเป็นจำนวนเต็มหรือไม่
is_float() หรือ is_double()	ตรวจสอบว่าข้อมูลเป็นจำนวนจริงหรือไม่
is_string()	ตรวจสอบว่าข้อมูลเป็นข้อความหรือไม่
is_bool()	ตรวจสอบว่าข้อมูลเป็นค่าความจริง (true/false) หรือไม่
is_null()	ตรวจสอบว่าข้อมูลเป็น null หรือไม่
is_array()	ตรวจสอบว่าข้อมูลเป็นอาร์เรย์หรือไม่
is_object()	ตรวจสอบว่าข้อมูลเป็นอ็อบเจกต์หรือไม่
is_numeric()	ตรวจสอบว่าข้อมูลสามารถแปลงเป็นตัวเลขได้หรือไม่
is_callable()	ตรวจสอบว่าข้อมูลสามารถเรียกใช้งานเป็นฟังก์ชันได้หรือไม่

จากตารางที่ 11-6 เป็นแสดงรายการฟังก์ชันสำหรับการตรวจสอบข้อมูล โดยผลลัพธ์ของฟังก์ชันจะคืนค่าเป็นบูลีน

4. การแสดงผลด้วยฟังก์ชัน echo และ print

ในการพัฒนาเว็บแอปพลิเคชันด้วยภาษา PHP การแสดงผลลัพธ์จากการประมวลผลของภาษา PHP โดยจะนำไปแสดงไว้ในเอกสารเว็บในตำแหน่งที่มีฟังก์ชันการแสดงผลของภาษา PHP อยู่ ฟังก์ชันที่มีหน้าที่แสดงผลลัพธ์ลักษณะนี้ ได้แก่ echo และ print ซึ่งทั้งสองฟังก์ชันมีหน้าที่คล้ายกันในการแสดงผลลัพธ์ แต่มีความแตกต่างในรายละเอียดการใช้งาน

4.1 ฟังก์ชัน echo

ฟังก์ชัน echo เป็นฟังก์ชันหลักที่ใช้สำหรับแสดงผลข้อมูลในตัวแปรหรือข้อความในเอกสารเว็บ โดยจะทำให้ตำแหน่งที่ฟังก์ชันนี้อยู่แทนที่ด้วยผลลัพธ์หลังจากที่ทำการประมวลผลภาษา PHP เสร็จแล้ว ฟังก์ชัน echo สามารถแสดงผลได้ทั้งข้อความ ค่าตัวแปร และการแสดงผลร่วมกันระหว่างข้อความกับตัวแปร มีรูปแบบการใช้งานดังนี้

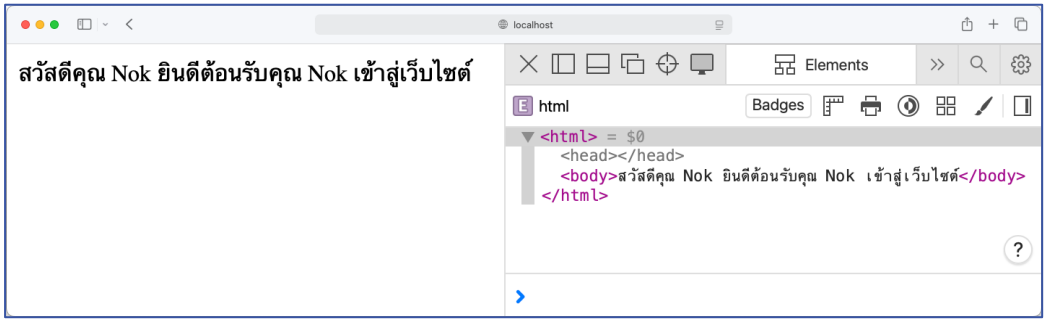
รูปแบบ

```
1  echo "ข้อความ";
2  echo  $ตัวแปร;
3  echo  "ข้อความ1" , "ข้อความ2";
4  echo  "ข้อความ" , ตัวแปร;
5  echo  "ข้อความ $ตัวแปร";
```

ตัวอย่างที่ 3 ไฟล์ index.php

```
1  <?php
2      $name = "Nok";
3      echo "สวัสดีคุณ ";
4      echo $name;
5      echo " ยินดีต้อนรับคุณ ", $name, " เข้าสู่เว็บไซต์";
```

ผลลัพธ์ของตัวอย่างที่ 3



จากตัวอย่างที่ 3 ในบรรทัดที่ 3 เป็นการแสดงผลข้อความโดยตรง ส่วนบรรทัดที่ 4 เป็นการแสดงผลค่าของตัวแปร $name และบรรทัดที่ 5 แสดงการใช้ฟังก์ชัน echo ในการแสดงผลร่วมกันระหว่างข้อความและตัวแปร โดยใช้เครื่องหมาย , ในการแยกค่าที่ต้องการนำมาแสดงร่วมกัน ผลลัพธ์ที่จะส่งกลับไปยังเว็บเบราว์เซอร์จะเป็นโครงสร้าง HTML ซึ่งเป็นกลไกการทำงานของภาษา PHP ที่ได้กล่าวถึงตอนต้น ถึงแม้ว่าจะมีแค่ส่วนคำสั่งภาษา PHP ก็ตาม โดยจากตัวอย่างที่ 1 จะทำให้ข้อความทั้งหมดแสดงต่อกันในส่วน body โดยแสดงเป็น *สวัสดีคุณ Nok ยินดีต้อนรับคุณ Nok เข้าสู่เว็บไซต์* เนื่องจากเว็บเบราว์เซอร์ไม่พบแท็กสำหรับการขึ้นบรรทัดใหม่

ตัวอย่างที่ 4 ต่อไปนี้ ใช้ฟังก์ชัน echo แสดงผลเช่นกัน แต่ข้อความที่แสดงมีการใช้แท็กของภาษา HTML เป็นส่วนของข้อความ รวมทั้งมีการแสดงผลร่วมกับข้อความ

ตัวอย่างที่ 4 ไฟล์ index-2.php

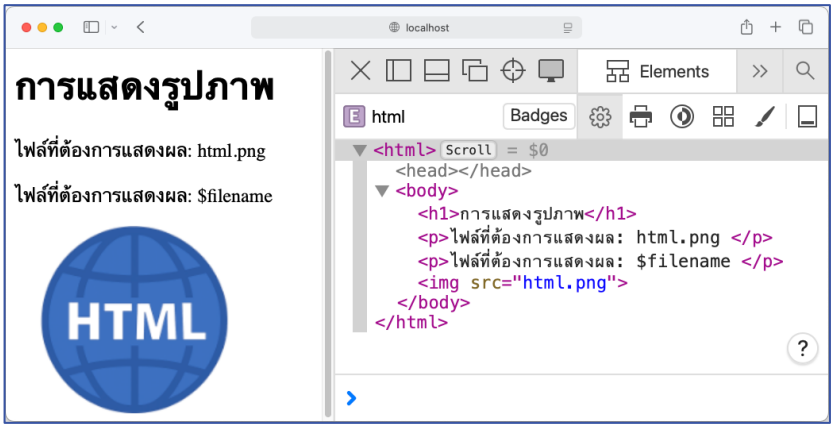
```
1  <?php
2      $filename = "html.png";
3      echo "<h1>การแสดงรูปภาพ</h1>";
4      echo "<p>ไฟล์ที่ต้องการแสดงผล: $filename </p>";
5      echo '<p>ไฟล์ที่ต้องการแสดงผล: $filename </p>';
6      echo "<img src='$filename'>";
```

จากรหัสคำสั่งในตัวอย่างที่ 4 บรรทัดที่ 2 เป็นการกำหนดค่าข้อความ "html.png" ให้กับตัวแปร $filename ซึ่งจะถูกนำไปใช้ในคำสั่งถัดไป บรรทัดที่ 3 เป็นการแสดงข้อความ <h1>การแสดงรูปภาพ</h1> โดยแม้ว่าจะมีการใส่แท็ก HTML อยู่ภายในค่าที่ส่งออกจาก echo แต่สำหรับตัวแปลภาษา PHP แล้วจะมองข้อความดังกล่าวเป็นเพียงข้อความเท่านั้น และจะไม่ดำเนินการตีความภายในตัว PHP เอง

ในบรรทัดที่ 4 และ 5 มีความแตกต่างกันอย่างสำคัญที่ควรสังเกต บรรทัดที่ 4 ใช้เครื่องหมายอัญประกาศคู่ (" ") ตัวแปร $filename จะถูกแปลและแทนค่าด้วย "html.png" ภายในข้อความ

ในบรรทัดที่ 5 ใช้เครื่องหมายอัญประกาศเดี่ยว (' '): ตัวแปร $filename จะไม่ถูกแปล แต่จะแสดงเป็นข้อความ $filename ตามที่ปรากฏ

ผลลัพธ์ของตัวอย่างที่ 4



จากผลลัพธ์ของตัวอย่างที่ 4 ซึ่งแสดงภาพการทำงานของรหัสคำสั่ง PHP บนเว็บเบราว์เซอร์ จะเห็นได้ว่าข้อมูลที่แสดงอยู่ในส่วนของเอกสาร HTML เป็นผลลัพธ์ที่ได้จากการประมวลผลของคำสั่ง echo() ทั้งหมด โดยเฉพาะในแต่ละบรรทัดมีลักษณะการใช้งานที่แตกต่างกันอย่างชัดเจนและส่งผลต่อผลลัพธ์บนหน้าเว็บ

- บรรทัดที่ 4 echo " ไฟล์ที่ต้องการแสดงผล: $filename "; เป็นการใช้เครื่องหมายอัญประกาศคู่ ทำให้ตัวแปร $filename ถูกประเมินและแทนค่าด้วย "html.png" ก่อนจะแสดงผลรวมกับข้อความที่เหลือบนหน้าเว็บ
- บรรทัดที่ 5 echo ' ไฟล์ที่ต้องการแสดงผล: $filename '; ใช้เครื่องหมายอัญประกาศเดี่ยวซึ่งตัวแปร $filename จะไม่ถูกประเมินค่า แต่จะแสดงผลตามข้อความต้นฉบับทั้งหมด ดังนั้นในผลลัพธ์ของบรรทัดนี้จะเห็นข้อความ $filename แสดงอยู่ตามเดิม
- บรรทัดที่ 6 echo "<img src='$filename'>"; เป็นการใช้เครื่องหมายอัญประกาศคู่ล้อมรอบข้อความ และอัญประกาศเดี่ยวล้อมรอบค่าของแอตทริบิวต์ src ซึ่งช่วยให้สามารถประเมินค่า $filename ได้อย่าง

ถูกต้อง ส่งผลให้ผลลัพธ์ที่ผู้ใช้ได้รับบนเว็บเบราว์เซอร์คือ <img src="html.png"> ซึ่งกลไกของเบราว์เซอร์จะใช้ค่านี้ในการแสดงรูปภาพตามตำแหน่งที่กำหนด

จากภาพผลลัพธ์ที่แสดงในเบราว์เซอร์จะพบว่า ข้อมูลทั้งข้อความและภาพที่แสดงอยู่ในส่วน <body> ล้วนเกิดจากกระบวนการตีความรหัส PHP บนฝั่งผู้ให้บริการเว็บ ก่อนจะส่งเอกสาร HTML ที่สมบูรณ์กลับมายังเว็บเบราว์เซอร์ ซึ่งกลไกนี้ถือเป็นหลักสำคัญของการพัฒนาเว็บแอปพลิเคชันด้วยภาษา PHP ในฐานะภาษาฝั่งผู้ให้บริการ

4.2 ฟังก์ชัน print

ฟังก์ชัน print เป็นอีกฟังก์ชันหนึ่งที่ใช้สำหรับแสดงผลข้อมูลมีหน้าที่คล้ายกับ echo แต่มีข้อจำกัดบางประการ คำสั่ง print สามารถแสดงผลได้เพียงหนึ่งพารามิเตอร์ในแต่ละครั้ง และมีการคืนค่าเป็นตัวเลขจำนวนเต็มที่มีค่าเป็น 1 เสมอ ซึ่งทำให้สามารถใช้ในนิพจน์ได้ แต่การทำงานอาจใช้เวลามากกว่าฟังก์ชัน echo เล็กน้อย รูปแบบและตัวอย่างการใช้งานฟังก์ชัน print มีดังนี้

รูปแบบ

```
1  print "ข้อความ";
2  print  $ตัวแปร;
3  print  "ข้อความ $ตัวแปร_1 $ตัวแปร_2  ... $ตัวแปร_n";
```

ตัวอย่างที่ 5 ไฟล์ index-3.php

```
1  <?php
2        $name = "Nok";
3        print "สวัสดีคุณ ";
4        print $name;
5        echo " ยินดีต้อนรับคุณ ", $name, " เข้าสู่เว็บไซต์";
```

จากตัวอย่างที่ 5 การใช้งานฟังก์ชัน print เพื่อแสดงผลข้อความหรือตัวแปร มีลักษณะคล้ายกันกับการใช้ฟังก์ชัน echo แต่ในบรรทัดที่ 5 ของตัวอย่างที่ 3 จะไม่สามารถใช้งานในรูปแบบดังกล่าวได้ เนื่องจามีค่าพารามิเตอร์จำนวน 3 ค่า คือ ข้อความ " ยินดีต้อนรับคุณ " , ตัวแปร $name และ ข้อความ " เข้าสู่เว็บไซต์" ซึ่งจะทำให้เกิดข้อผิดพลาด Parse error หากต้องการแสดงผลในลักษณะดังกล่าวจะต้องใช้รูปแบบที่ 3 ที่รวมข้อความและตัวแปรไว้ในเครื่องหมายอัญประกาศคู่

4.3 การประยุกต์ใช้คำสั่งแสดงผลร่วมกับแนวคิดการแยกส่วนการทำงาน

การเขียนรหัสคำสั่งภาษา PHP ลงในเอกสาร HTML หรือที่เรียกว่า Embedded PHP เป็นแนวทางที่ได้รับความนิยมในการพัฒนาเว็บแอปพลิเคชัน โดยสะท้อนแนวคิดการแยกส่วนการทำงานระหว่าง ส่วนจัดการข้อมูล ซึ่งดำเนินการในฝั่งผู้ให้บริการเว็บ (Backend) และ ส่วนแสดงผล ซึ่งดำเนินการในฝั่งผู้ใช้งาน (Frontend)

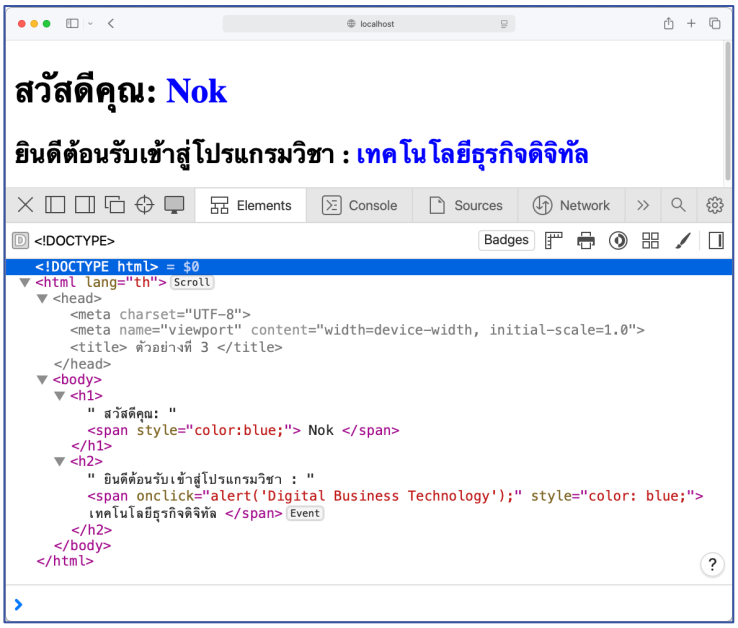
ได้อย่างชัดเจน โดยภาษาสำหรับโครงสร้างและการนำเสนอ เช่น HTML CSS และ JavaScript ทำหน้าที่เป็นแม่แบบการแสดงผล (Template) ในขณะที่ภาษา PHP ทำหน้าที่ประมวลผล จัดเตรียม และส่งข้อมูลไปแสดงในตำแหน่งต่าง ๆ ของแม่แบบเหล่านั้น ตัวอย่างและผลลัพธ์การทำงานต่อไปนี้จะแสดงให้เห็นถึงบทบาทการทำงานของแต่ละส่วน ดังนี้

ตัวอย่างที่ 6 ไฟล์ index-4.php

```
<?php
    $title = "ตัวอย่างที่ 3";
    $name = "Nok";
    $department_th = "เทคโนโลยีธุรกิจดิจิทัล";
    $department_en = "Digital Business Technology";
?>
<!DOCTYPE html>
<html lang="th">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>
        <?phpecho $title;  ?>
    </title>
</head>
<body>
    <h1>
        สวัสดีคุณ:
        <span  style="color:blue;">
        <?php  echo $name;  ?>
        </span>
    </h1>
    <h2>
        ยินดีต้อนรับเข้าสู่โปรแกรมวิชา :
        <span style="color: blue;"
        onclick="alert('<?php  echo $department_en; ?>');">
```

```
26              <?php  echo $department_th;     ?>
27              </span>
28          </h2>
29      </body>
30      </html>
```

ผลลัพธ์ของตัวอย่างที่ 6



จากตัวอย่างที่ 6 และผลลัพธ์การแสดงผลในเว็บเบราว์เซอร์ สามารถอธิบายการทำงานจำแนกตามบทบาทของแต่ละส่วนดังนี้

4.3.1 ส่วนการทำงานของ PHP

ในบรรทัดที่ 2–4 ของตัวอย่างที่ 6 เป็นการประกาศตัวแปร $title, $name, $department_th และ $department_en พร้อมทั้งกำหนดค่าข้อมูลให้กับแต่ละตัวแปร โดยตัวแปรเหล่านี้จะถูกใช้ในส่วนต่าง ๆ ของเอกสาร HTML เพื่อแสดงผลลัพธ์ตามค่าที่กำหนดไว้ จุดสำคัญคือ PHP สามารถเขียนแทรกไว้ในตำแหน่งใดของโครงสร้าง HTML ก็ได้ โดยไม่จำกัดว่าจะต้องอยู่ในส่วน body หรือส่วน head ตัวแปรที่ประกาศไว้ในไฟล์เดียวกันตามลักษณะในตัวอย่างที่ 4 จะสามารถนำไปใช้ซ้ำได้ในทุกส่วนของเอกสาร HTML ไม่ว่าจะแสดงภายใน <title>, <h1>, หรือ ก็ตาม

การแสดงผลของค่าตัวแปรทำได้ด้วยคำสั่ง echo ซึ่งในตัวอย่างใช้รูปแบบการเขียนคำสั่งแบบบรรทัดเดียว เช่น <?php echo $name; ?> โดยเหมาะสำหรับกรณีที่มีเพียงคำสั่งเดียวในแต่ละตำแหน่ง อย่างไรก็ตามสามารถเขียนโครงสร้างภาษา PHP แบบหลายบรรทัดก็ได้เพื่อเอื้ออำนวยต่อการทำความเข้าใจรหัสคำสั่ง

ส่วนที่น่าสังเกตอีกจุดคือในบรรทัดของ <span onclick="alert('<?php echo $department_en; ?>');"> มีการใช้ PHP เพื่อนำค่าในตัวแปร $department_en ของภาษา PHP ไปเป็นค่าพารามิเตอร์ให้กับคำสั่ง alert() ของ JavaScript ซึ่งเป็นตัวอย่างของการที่ PHP ทำหน้าที่จัดเตรียมข้อมูลเพื่อให้ JavaScript ใช้ในการโต้ตอบกับผู้ใช้งาน

4.3.2 ส่วนการทำงานของภาษา HTML

ภาษา HTML ทำหน้าที่เป็นโครงสร้างหลักของเอกสารเว็บ โดยกำหนดลำดับและตำแหน่งของเนื้อหาที่จะแสดงผล หากเขียนรหัสคำสั่งภาษา PHP โดยไม่มีการกำหนดโครงสร้าง HTML ดังเช่นตัวอย่างที่ 3 และ 4 ที่ใช้เพียงคำสั่งภาษา PHP เพียงอย่างเดียว ตัวแปลภาษา PHP จะทำการสร้างโครงสร้าง HTML ตามมาตรฐานและนำผลลัพธ์ที่ได้ของภาษา PHP แสดงเป็นเนื้อหาภายใน <body> ของเอกสาร HTML โดยปริยาย แต่ในตัวอย่างที่ 3 มีการกำหนดโครงสร้าง HTML อย่างครบถ้วนตั้งแต่ <!DOCTYPE html> จนถึง </html> ซึ่งทำหน้าที่เป็นแม่แบบการจัดวางตำแหน่งของเนื้อหา ตัวแปลภาษา PHP ก็จะใช้โครงสร้าง HTML ที่กำหนดนี้แทนการสร้างโครงสร้าง HTML ขึ้นมาเอง และจะนำผลลัพธ์ของฟังก์ชันแสดงผลอย่าง echo แสดงไว้ในตำแหน่งที่มีการใช้ฟังก์ชันแสดงผลนั้น (บรรทัดที่ 13, 20 และ 26) เช่น ในบรรทัดที่ 13 จะมีการนำค่าในตัวแปร $title ที่ได้ประกาศไว้ในบรรทัดที่ 2 มาใช้เป็นเนื้อหาของอิลิเมนต์ title เป็นต้น

4.3.3 ส่วนการทำงานของภาษา CSS

ภาษา CSS ที่ใช้ในตัวอย่างนี้แสดงอยู่ในรูปแบบเขียนลงในแท็ก โดยมีการกำหนดคุณสมบัติ color: blue; ให้กับอิลิเมนต์ (บรรทัดที่ 19 และ 25) โดยเนื้อหาที่อยู่ภายในอิลิเมนต์นี้มาจากผลลัพธ์ของฟังก์ชันแสดงผลของภาษา PHP (บรรทัดที่ 20 และ 26) ทำให้เนื้อหาของอิลิเมนต์แสดงผลเป็นไปตามชุดกฎของ CSS ความสัมพันธ์นี้แสดงให้เห็นว่า CSS ทำหน้าที่ควบคุมลักษณะการแสดงผลของอิลิเมนต์ HTML ซึ่งจะส่งผลต่อเนื้อหาที่สร้างมาจากภาษา PHP ด้วยเช่นกัน

4.3.4 ส่วนการทำงานของภาษา JavaScript

ภาษา JavaScript มีบทบาทในการควบคุมพฤติกรรมและตอบสนองต่อเหตุการณ์ที่เกิดขึ้นกับอิลิเมนต์ต่าง ๆ ในเอกสาร HTML โดยในตัวอย่างนี้ การใช้งานภาษา JavaScript ปรากฏในแอตทริบิวต์ onclick ของแท็ก (บรรทัดที่ 25) เมื่อผู้ใช้คลิกที่ข้อความ *เทคโนโลยีธุรกิจดิจิทัล* ภาษา JavaScript จะทำงานโดยแสดงหน้าต่างแจ้งเตือน ที่แสดงข้อความเป็นชื่อโปรแกรมวิชาภาษาอังกฤษ คือ *Digital Business Technology* ซึ่งเป็นค่าที่ได้มาจากฟังก์ชันการแสดงผลของภาษา PHP (echo $department_en;) กลไกนี้สะท้อนให้เห็นถึงการเชื่อมโยงกันระหว่างภาษา JavaScript ที่ทำงานในฝั่งผู้ใช้ในบทบาทการจัดการเหตุการณ์ที่เกิดขึ้นกับอิลิเมนต์และตอบสนองต่อเหตุการณ์นั้น และภาษา PHP ที่ทำงานในฝั่งผู้ให้บริการในบทบาทการจัดเตรียมข้อมูลให้กับภาษา JavaScript

5. โครงสร้างการควบคุม

การเขียนโปรแกรมด้วยภาษา PHP นั้น นอกจากการประกาศตัวแปรและการใช้ฟังก์ชันแสดงผลแล้ว การใช้โครงสร้างการถือเป็นองค์ประกอบสำคัญที่ทำให้โปรแกรมสามารถตัดสินใจและทำงานตามเงื่อนไขที่กำหนดได้ คำสั่งควบคุมจะช่วยให้โปรแกรมมีความยืดหยุ่นและสามารถตอบสนองต่อสถานการณ์ต่าง ๆ ได้อย่างเหมาะสม โดยภาษา PHP มีคำสั่งควบคุมหลักสองประเภท ได้แก่ โครงสร้างการตัดสินใจ (Decision Structures) และโครงสร้างการวนซ้ำ (Loop Structures)

สำหรับผู้อ่านที่ได้ศึกษาเนื้อหาในบทที่ 9 เกี่ยวกับภาษา JavaScript มาก่อน จะพบว่าโครงสร้างคำสั่งควบคุมของภาษา PHP มีความคล้ายคลึงกับ JavaScript ทั้งในด้านแนวคิดและรูปแบบทางไวยากรณ์ ความคล้ายคลึงนี้เกิดจากการที่ทั้งสองภาษาอยู่ในตระกูลภาษาที่มีลักษณะโครงสร้างคล้ายภาษา C (C-like Syntax) จึงมีการใช้เครื่องหมายปีกกา {} เพื่อครอบกลุ่มคำสั่ง การใช้เครื่องหมายเปรียบเทียบ (เช่น ==, !=, <, >) และโครงสร้างคำสั่งจำพวก if, else, switch, for, while, foreach ที่มีลักษณะการใช้งานใกล้เคียงกัน

5.1 โครงสร้างการตัดสินใจ

โครงสร้างการตัดสินใจ เป็นกลไกที่ช่วยให้โปรแกรมสามารถเลือกดำเนินการในส่วนใดส่วนหนึ่งขึ้นอยู่กับผลการประเมินเงื่อนไขที่กำหนด การใช้โครงสร้างการตัดสินใจทำให้โปรแกรมมีความสามารถในการประมวลผลตามบริบทและสถานการณ์ที่แตกต่างกัน ซึ่งเป็นพื้นฐานสำคัญในการสร้างเว็บแอปพลิเคชันที่สามารถตอบสนองต่อการโต้ตอบของผู้ใช้

5.1.1 โครงสร้างการตัดสินใจแบบทางเดียว

โครงสร้างการตัดสินใจแบบทางเดียวเป็นรูปแบบการควบคุมที่ง่ายที่สุด ซึ่งจะดำเนินการกลุ่มคำสั่งที่กำหนดเมื่อเงื่อนไขเป็นจริงและข้ามไปดำเนินการคำสั่งถัดไปเมื่อเงื่อนไขเป็นเท็จ รูปแบบของโครงสร้างการตัดสินใจแบบทางเดียวในภาษา PHP มีดังนี้

รูปแบบ

```
if (เงื่อนไข) {
    คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง;
    ...;
}
```

ตัวอย่างที่ 7

```
1  <?php
2  $score = 85;
3  if ($score >= 80) {
4      echo "ผลการเรียนดีมาก";
5  }
```

```
6    echo "จบการตรวจสอบ";
```

จากตัวอย่างที่ 7 บรรทัดที่ 2 เป็นการประกาศตัวแปร $score และกำหนดค่าเป็น 85 ในบรรทัดที่ 3 เป็นคำสั่งควบคุม if ที่ใช้นิพจน์เปรียบเทียบ $score >= 80 เมื่อทำการประเมินค่านิพจน์แล้วจะได้ค่าผลลัพธ์เป็น true เนื่องจาก 85 มากกว่าหรือเท่ากับ 80 จึงดำเนินการคำสั่งในบรรทัดที่ 4 ซึ่งเป็นการแสดงข้อความ *ผลการเรียนดีมาก* จากนั้นจึงดำเนินการต่อในบรรทัดที่ 6 แสดงข้อความ *จบการตรวจสอบ*

5.1.2 โครงสร้างการตัดสินใจแบบสองทาง

โครงสร้างการตัดสินใจแบบสองทางเป็นการขยายโครงสร้างการตัดสินใจแบบทางเดียวโดยเพิ่มทางเลือกสำหรับกรณีที่เงื่อนไขเป็นเท็จ ทำให้โปรแกรมสามารถกำหนดการทำงานที่แตกต่างกันสำหรับทั้งสองสถานการณ์ได้ รูปแบบของโครงสร้างการตัดสินใจแบบสองทางในภาษา PHP มีดังนี้

รูปแบบ

```
if (เงื่อนไข) {
    คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง;
    ...;
} else {
    คำสั่งที่จะทำงานเมื่อเงื่อนไขไม่เป็นจริง;
    ...;
}
```

ตัวอย่างที่ 8

```
1    <?php
2        $age = 16;
3        if ($age >= 18) {
4            echo "สามารถออกเสียงเลือกตั้งได้";
5        } else {
6            echo "ยังไม่สามารถออกเสียงเลือกตั้งได้";
7        }
8    ?>
```

จากตัวอย่างที่ 8 ตัวแปร $age มีค่าเป็น 16 เมื่อตรวจสอบเงื่อนไข $age >= 18 จะได้ผลลัพธ์เป็น false เนื่องจาก 16 ไม่มากกว่าหรือเท่ากับ 18 จึงข้ามไปดำเนินการในส่วน else ในบรรทัดที่ 5-7 และแสดงข้อความ *ยังไม่สามารถออกเสียงเลือกตั้งได้*

5.1.3 โครงสร้างการตัดสินใจแบบหลายเลือก

โครงสร้างการตัดสินใจแบบหลายเลือก เป็นการขยายโครงสร้างการตัดสินใจเพื่อรองรับการตรวจสอบหลายเงื่อนไขแบบเรียงลำดับ โดยใช้คำสั่ง if-elseif-else ทำให้โปรแกรมสามารถเลือกดำเนินการจากหลายทางเลือกได้

รูปแบบ

```
if (เงื่อนไขที่ 1) {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขเป็นจริง;
        ...;
} else if (เงื่อนไขที่ 2) {
        คำสั่งที่จะทำงานเมื่อเงื่อนไขที่ 2 เป็นจริง;
        ...;
} else {
        คำสั่งที่จะทำงานเมื่อไม่มีเงื่อนไขใดๆ ที่เป็นจริง;
        ...;
}
```

ตัวอย่างที่ 9

```
1   <?php
2       $grade = 75;
3       if ($grade >= 80) {
4           echo "เกรด A";
5       } elseif ($grade >= 70) {
6           echo "เกรด B";
7       } else {
8           echo "เกรด F";
9       }
10  ?>
```

จากตัวอย่างที่ 9 ตัวแปร $grade มีค่าเป็น 75 การตรวจสอบเงื่อนไขจะเริ่มจากเงื่อนไขแรก $grade >= 80 ซึ่งเป็น false จึงตรวจสอบเงื่อนไขถัดไป $grade >= 70 ซึ่งเป็น true จึงดำเนินการในบรรทัดที่ 6 แสดงข้อความ *เกรด B* และหยุดการตรวจสอบเงื่อนไขที่เหลือ

5.1.4 โครงสร้างการตัดสินใจแบบหลายกรณี

โครงสร้างการตัดสินใจแบบหลายกรณี (Switch Structure) เป็นทางเลือกของการใช้โครงสร้างการตัดสินใจแบบหลายเลือก (if-elseif-else) ในกรณีที่ต้องการตรวจสอบค่าของตัวแปรหรือนิพจน์เพียงค่าหนึ่งเทียบกับค่าคงที่หลายค่า โดยที่ค่าทั้งหมดต้องสามารถตรวจสอบได้ด้วยการเปรียบเทียบแบบเท่ากัน รูปแบบของโครงสร้างการตัดสินใจแบบหลายกรณีในภาษา PHP มีดังนี้

รูปแบบ

```
switch (ตัวแปรหรือนิพจน์) {
    case  ค่าคงที่_1:
        คำสั่งที่ต้องดำเนินการเมื่อตัวแปรหรือนิพิจน์ตรวจสอบมีค่าเท่ากับ ค่าคงที่_1;
        ...;
        break;
    case  ค่าคงที่_2:
        คำสั่งที่ต้องดำเนินการเมื่อตัวแปรหรือนิพิจน์ตรวจสอบมีค่าเท่ากับ ค่าคงที่_1;
        ...;
        break;
    default:
        คำสั่งที่ต้องดำเนินการเมื่อตัวแปรหรือนิพิจน์ตรวจสอบไม่ตรงกับค่าคงที่ใดๆ
        ...;
}
```

ส่วนประกอบของโครงสร้างการตัดสินใจแบบหลายกรณีประกอบด้วย คำสั่งควบคุม switch เป็นคำสั่งเริ่มต้นของโครงสร้าง และนิพจน์หรือตัวแปรที่เป็นค่าที่จะถูกนำมาเปรียบเทียบกับค่าใน case ต่าง ๆ คำสงวน case ใช้กำหนดค่าคงที่ที่จะเปรียบเทียบกับตัวแปรหรือนิพจน์ในคำสั่งควบคุม switch โดยจะมีเครื่องหมาย : หลังค่าคงที่เสมอ คำสงวน break ใช้สำหรับออกจากโครงสร้าง switch เพื่อป้องกันไม่ให้ดำเนินการในกรณีถัดไป และคำสงวน default กำหนดกลุ่มคำสั่งสำหรับกรณีที่ไม่มีกรณีใดตรงกัน ท้ายคำสงวนจะต้องมีเครื่องหมาย : เสมอ ทั้งนี้ส่วนนี้อาจไม่ระบุก็ได้

ตัวอย่างที่ 10

```
1   <?php
2       $day = "จันทร์";
3       switch ($day) {
4           case "จันทร์":
```

```
5                   echo "วันแรกของสัปดาห์";
6                   break;
7               case "ศุกร์":
8                   echo "วันสุดท้ายของสัปดาห์ทำงาน";
9                   break;
10              case "เสาร์":
11                  echo "วันแรกของวันหยุดสุดสัปดาห์";
12                  break;
13              case "อาทิตย์":
14                  echo "วันที่สองของวันหยุดสุดสัปดาห์";
15                  break;
16              default:
17                  echo "วันธรรมดา";
18          }
19  ?>
```

จากตัวอย่างที่ 10 ตัวแปร $day มีค่าเป็น *จันทร์* คำสั่ง switch จะเปรียบเทียบค่านี้กับแต่ละ case เมื่อพบว่าตรงกับ case *จันทร์* ในบรรทัดที่ 4 จะดำเนินการในบรรทัดที่ 5 แสดงข้อความ *วันแรกของสัปดาห์* และใช้คำสั่ง break ในบรรทัดที่ 6 เพื่อออกจากโครงสร้าง switch

5.2 โครงสร้างการวนซ้ำ

โครงสร้างการวนซ้ำเป็นรูปแบบการควบคุมการทำงานของโปรแกรมที่ทำให้สามารถดำเนินการชุดคำสั่งเดียวกันซ้ำหลายครั้งตามเงื่อนไขที่กำหนด ช่วยลดการเขียนรหัสคำสั่งซ้ำซ้อน เพิ่มประสิทธิภาพ และทำให้การจัดการข้อมูลจำนวนมากเป็นไปได้อย่างมีประสิทธิภาพ

5.2.1 โครงสร้างการวนซ้ำแบบ for

โครงสร้างการวนซ้ำแบบ for เป็นรูปแบบของการวนซ้ำที่ใช้เมื่อทราบจำนวนครั้งที่ต้องการให้โปรแกรมทำงานซ้ำอย่างแน่นอน รูปแบบของโครงสร้างการวนซ้ำแบบ for ในภาษา PHP มีดังนี้

รูปแบบ

```
for (นิพจน์กำหนดค่าเริ่มต้น; นิพจน์ตรวจสอบเงื่อนไข; นิพจน์เปลี่ยนแปลงค่าตัวแปรควบคุม) {
        คำสั่งที่ 1;
        ...;
}
```

ตัวอย่างที่ 11

```
1  <?php
2      echo "การนับจาก 1 ถึง 5:<br>";
3      for ($i = 1; $i <= 5; $i++) {
4          echo "ลำดับที่ " . $i . "<br>";
5      }
6      echo "จบการวนซ้ำ";
7  ?>
```

จากตัวอย่างที่ 11 บรรทัดที่ 3 เป็นคำสั่งควบคุมการวนซ้ำ for ที่มีการกำหนดค่าเริ่มต้นของตัวแปร $i เป็น 1 เงื่อนไขการวนซ้ำคือ $i <= 5 และการปรับปรุงตัวแปรด้วยการเพิ่มค่าทีละ 1 ($i++) ในรอบแรกของการวนซ้ำ ตัวแปร $i มีค่าเป็น 1 ซึ่งตรงตามเงื่อนไขการตรวจสอบที่จะต้องน้อยกว่าหรือเท่ากับ 5 จึงมีการทำงานในกลุ่มประโยคคำสั่งในบรรทัดที่ 4 ซึ่งเป็นการแสดงข้อความ *ลำดับที่* และค่าของตัวแปร $i คือ 1 จากนั้นจะมีการเพิ่มค่า $i เป็น 2 และตรวจสอบเงื่อนไขใหม่ กระบวนการนี้จะดำเนินต่อไปจนกระทั่ง $i มีค่าเป็น 6 ซึ่งไม่ตรงตามเงื่อนไข $i <= 5 การวนซ้ำจึงสิ้นสุดและดำเนินการคำสั่งในบรรทัดที่ 6 เพื่อแสดงข้อความ *จบการวนซ้ำ*

5.2.2 โครงสร้างการวนซ้ำแบบ while

โครงสร้างการวนซ้ำแบบ while เป็นรูปแบบการวนซ้ำที่เหมาะสำหรับกรณีที่ไม่ทราบจำนวนรอบการวนซ้ำที่แน่นอน และขึ้นอยู่กับผลของเงื่อนไขที่กำหนดไว้ การวนซ้ำแบบ while จะตรวจสอบเงื่อนไขก่อนการทำงาน หากเงื่อนไขเป็นจริงจึงจะดำเนินการในกลุ่มประโยคคำสั่ง รูปแบบของโครงสร้างการวนซ้ำแบบ while ในภาษา PHP มีดังนี้

รูปแบบ

```
while (นิพจน์ตรวจสอบเงื่อนไข) {
    คำสั่งที่ 1;
    ...;
}
```

ตัวอย่างที่ 12

```
1  <?php
2      $count = 1;
3      while ($count <= 3) {
4          echo "รอบที่ " . $count . "<br>";
5          $count++;
6      }
```

```
7       echo "สิ้นสุดการนับ";
8   ?>
```

จากตัวอย่างที่ 12 บรรทัดที่ 2 เป็นการประกาศตัวแปร $count และกำหนดค่าเริ่มต้นเป็น 1 ซึ่งทำก่อนเข้าสู่คำสั่ง while ในบรรทัดที่ 3 เป็นคำสั่ง while ที่ตรวจสอบเงื่อนไข $count <= 3 ในรอบแรก $count มีค่าเป็น 1 ซึ่งตรงตามเงื่อนไข ทำให้มีการดำเนินการกลุ่มประโยคคำสั่งบรรทัดที่ 4-5 ผลของคำสั่งในบรรทัดที่ 4 เป็นการแสดงข้อความ *รอบที่ 1* จากนั้นในบรรทัดที่ 5 จึงเพิ่มค่า $count ขึ้นอีก 1 แล้วจะวนกลับไปตรวจสอบเงื่อนไขใหม่ กระบวนการนี้จะดำเนินต่อไปจนกระทั่ง $count มีค่าเป็น 4 ซึ่งไม่ตรงตามเงื่อนไขที่กำหนดว่า $count <= 3 การวนซ้ำจึงสิ้นสุดและดำเนินการคำสั่งในบรรทัดที่ 7 ซึ่งเป็นการแสดงข้อความ *สิ้นสุดการนับ*

6. ฟังก์ชันและอาร์เรย์

ฟังก์ชันและอาร์เรย์เป็นองค์ประกอบสำคัญในการเขียนโปรแกรมในหลายๆ ภาษารวมถึงภาษา PHP โดยฟังก์ชันมีเป้าหมายในการจัดกลุ่มคำสั่งที่เกี่ยวข้องกันตามหน้าที่ เพื่อให้สามารถเรียกใช้งานซ้ำได้อย่างมีประสิทธิภาพ ลดความซ้ำซ้อนของรหัสคำสั่ง และเพิ่มความยืดหยุ่นในการพัฒนาโปรแกรม ขณะที่อาร์เรย์เป็นโครงสร้างข้อมูลที่ช่วยให้สามารถจัดเก็บข้อมูลหลายค่าไว้ในตัวแปรเดียวได้ โดยอ้างอิงแต่ละค่าด้วยตำแหน่งหรือคีย์ ซึ่งเหมาะสมอย่างยิ่งสำหรับการจัดการข้อมูลที่มีลักษณะเป็นกลุ่มหรือลำดับ

6.1 ฟังก์ชัน

ในการเขียนโปรแกรมด้วยภาษา PHP ฟังก์ชัน (Function) เป็นองค์ประกอบสำคัญที่ใช้ในการจัดกลุ่มคำสั่งที่มีลักษณะการทำงานเฉพาะเข้าไว้ด้วยกัน ทำให้สามารถเรียกใช้งานซ้ำได้หลายครั้งโดยไม่ต้องเขียนคำสั่งเดิมซ้ำ ๆ ฟังก์ชันจึงช่วยเพิ่มความเป็นระบบ ลดความซ้ำซ้อน และส่งเสริมหลักการเขียนโปรแกรมเชิงโครงสร้าง (Structured Programming)

6.1.1 การสร้างฟังก์ชัน

การสร้างฟังก์ชันในภาษา PHP มีรูปแบบคล้ายกับภาษา JavaScript ที่ผู้อ่านได้ศึกษาไปแล้วในบทที่ 9 โดยใช้คำสำคัญ function ตามด้วยชื่อฟังก์ชันและวงเล็บสำหรับรับพารามิเตอร์ ซึ่งภายในวงเล็บปีกกาจะเป็นคำสั่งที่ต้องการให้ฟังก์ชันประมวลผล โดยรูปแบบพื้นฐานมีดังนี้

รูปแบบ

```
function ชื่อฟังก์ชัน(พารามิเตอร์ 1, พารามิเตอร์ 2, ...) {
    คำสั่ง;
    return ค่าที่ส่งกลับ; // หากไม่มีการส่งค่ากลับสามารถละบรรทัดนี้ได้
}
```

ตัวอย่างที่ 13

```php
<?php
    function sayHello()
    {
        echo "สวัสดีครับ!";
    }
    function greetUser($name)
    {
        echo "สวัสดี " . $name . " ยินดีต้อนรับ!";
    }
    function addNumbers($a, $b)
    {
        $result = $a + $b;
        return $result;
    }
?>
```

จากตัวอย่างที่ 13 แสดงการสร้างฟังก์ชัน 3 ฟังก์ชัน ประกอบด้วย ฟังก์ชัน sayHello() เป็นฟังก์ชันที่ไม่รับพารามิเตอร์และไม่มีการคืนค่า ฟังก์ชัน greetUser() รับพารามิเตอร์หนึ่งตัวคือ $name และฟังก์ชัน addNumbers() รับพารามิเตอร์สองตัวและมีการคืนค่าผลลัพธ์ไปยังส่วนที่เรียกใช้ฟังก์ชันดังกล่าว เพื่อให้สามารถนำค่าดังกล่าวไปใช้งานต่อได้

6.1.2 การเรียกใช้ฟังก์ชัน

การเรียกใช้ฟังก์ชันทำได้โดยระบุชื่อฟังก์ชันตามด้วยวงเล็บ และส่งค่าอาร์กิวเมนต์หากฟังก์ชันต้องการพารามิเตอร์ การเรียกใช้ฟังก์ชันในภาษา PHP มีรูปแบบดังนี้

รูปแบบ

ชื่อฟังก์ชัน (พารามิเตอร์_1, พารามิเตอร์_2, ..., พารามิเตอร์_n);

ตัวอย่างที่ 14

```php
<?php
    // เรียกใช้ฟังก์ชันที่ไม่รับพารามิเตอร์
    sayHello();
    // เรียกใช้ฟังก์ชันที่รับพารามิเตอร์
```

```
3       greetUser("Nok");
        // เรียกใช้ฟังก์ชันที่คืนค่า
4       $sum = addNumbers(10, 20);
5       echo "ผลรวมคือ: " . $sum;
        // เรียกใช้และแสดงผลทันที
6       echo "5 + 3 = " . addNumbers(5, 3);
7   ?>
```

จากตัวอย่างที่ 14 เป็นการแสดงให้เห็นถึงกระบวนการเรียกใช้งานฟังก์ชันที่ได้ประกาศไว้ก่อนหน้านี้ในตัวอย่างที่ 13 โดยแบ่งรูปแบบการเรียกใช้ฟังก์ชันออกเป็น 4 ลักษณะ ได้แก่ ฟังก์ชันที่ไม่รับพารามิเตอร์ ฟังก์ชันที่รับพารามิเตอร์ ฟังก์ชันที่คืนค่า และฟังก์ชันที่คืนค่าและนำไปใช้งานต่อทันที

- บรรทัดที่ 1 ของรหัสคำสั่งมีการเรียกใช้ฟังก์ชัน sayHello() ซึ่งเป็นฟังก์ชันที่ไม่รับพารามิเตอร์ โดยเมื่อเรียกใช้งานจะมีผลให้แสดงข้อความว่า สวัสดีครับ!
- บรรทัดที่ 2 มีการเรียกใช้ฟังก์ชัน greetUser("Nok") ซึ่งเป็นฟังก์ชันที่รับพารามิเตอร์ชื่อ $name โดยการส่งค่าข้อความ Nok เข้าไปเป็นพารามิเตอร์ และแสดงผลข้อความว่า สวัสดี Nok ยินดีต้อนรับ! ซึ่งแสดงให้เห็นถึงการรับค่าภายนอกมาใช้ภายในฟังก์ชันได้
- บรรทัดที่ 3 มีการเรียกใช้ฟังก์ชัน addNumbers(10, 20) ซึ่งเป็นฟังก์ชันที่รับพารามิเตอร์ 2 ค่า และมีการคืนค่าผลลัพธ์กลับมา โดยค่าที่คืนกลับมาจะถูกเก็บไว้ในตัวแปร $sum และมีการใช้คำสั่ง echo แสดงผลลัพธ์ว่า “ผลรวมคือ: 30” ซึ่งเกิดจากการนำ 10 กับ 20 มาบวกกันภายในฟังก์ชัน
- บรรทัดที่ 4 เป็นการเรียกใช้ฟังก์ชัน addNumbers(5, 3) ซึ่งเป็นฟังก์ชันที่รับค่าพารามิเตอร์จำนวน 2 ค่า แล้วทำการประมวลผลผลรวมและคืนค่าผลลัพธ์กลับมา จากนั้นผลลัพธ์ที่ได้จะถูกนำไปต่อข้อความโดยใช้เครื่องหมายจุด (.) เพื่อรวมกับข้อความ 5 + 3 = ภายในคำสั่ง echo ทำให้เกิดการแสดงผลข้อความแบบรวมทันทีโดยไม่ต้องกำหนดตัวแปรเพื่อเก็บผลรวมไว้ก่อน ผลลัพธ์ที่ปรากฏบนเว็บเบราว์เซอร์จึงเป็นข้อความ 5 + 3 = 8 ซึ่งแสดงให้เห็นถึงความสามารถของภาษา PHP ในการเรียกใช้ฟังก์ชันที่คืนค่าและนำผลลัพธ์ไปใช้งานในตำแหน่งที่ต้องการได้

6.1.3 การกำหนดค่าเริ่มต้นให้พารามิเตอร์

ภาษา PHP อนุญาตให้ผู้พัฒนากำหนดค่าตั้งต้นให้กับพารามิเตอร์ของฟังก์ชันได้โดยตรงในขณะที่ประกาศฟังก์ชัน ซึ่งจะช่วยให้สามารถเรียกใช้งานฟังก์ชันโดยไม่จำเป็นต้องระบุค่าพารามิเตอร์บางตัวในทุกครั้งที่เรียกใช้ โดยค่าที่ไม่ได้ส่งเข้ามาจะใช้ค่าที่กำหนดไว้ล่วงหน้าแทน อย่างไรก็ตาม ข้อควรระวังคือ การเรียงลำดับพารามิเตอร์ในการสร้างฟังก์ชันจะต้องเรียงจากพารามิเตอร์ที่ไม่มีค่าเริ่มต้นก่อน หากมีการจัดลำดับพารามิเตอร์ผิดจะทำให้เกิดข้อผิดพลาด รูปแบบพร้อมตัวอย่างของการกำหนดค่าเริ่มต้นและการส่งค่าให้กับฟังก์ชันแบบทั่วไปมีดังนี้

รูปแบบ

```
function ชื่อฟังก์ชัน($พารามิเตอร์ที่ไม่มีค่าเริ่มต้น, $พารามิเตอร์2 = ค่าเริ่มต้น) {
    คำสั่ง;
}
```

ตัวอย่างที่ 15

```
1  <?php
2      function greet($name, $title= "คุณ") {
3          echo "สวัสดี $title $name";
4      }
5      greet("Nok");      // ไม่ส่งค่า $title
6  ?>
```

จากตัวอย่างที่ 15 ฟังก์ชัน greet() มีการกำหนดค่าเริ่มต้นให้กับพารามิเตอร์ $title เป็นคำว่า *คุณ* ซึ่งพารามิเตอร์นี้ใช้เก็บค่าคำหน้านำ สำหรับพารามิเตอร์ $name ใช้สำหรับรับค่าชื่อซึ่งไม่มีการกำหนดค่าเริ่มต้นให้ จึงต้องจัดลำดับไว้เป็นพารามิเตอร์แรกตามข้อกำหนด เมื่อเรียกใช้ฟังก์ชันโดยไม่ส่งคำนำหน้าไปให้ฟังก์ชันส่งแต่เพียงชื่อเท่านั้น ผลการทำงานจะแสดงผลเป็นข้อความว่า *สวัสดี คุณ Nok* แต่หากมีการส่งค่าคำนำหน้า เช่น greet("Nok","Mr."); ข้อความคำว่า *คุณ* ที่เป็นค่าเริ่มต้นของ $title ก็จะใช้เป็นค่าที่ส่งมานี้แทนคือ "Mr." ผลลัพธ์ที่ได้ก็จะเปลี่ยนเป็น *สวัสดี Mr. Nok*

ในตัวอย่างที่ 15 จะเห็นว่าการส่งค่าให้ฟังก์ชันจะต้องมีการส่งตามลำดับที่มีการประกาศไว้ แต่ใน PHP เวอร์ชัน 8 เป็นต้นมา มีการเพิ่มความสามารถในการส่งค่าให้กับฟังก์ชันแบบระบุชื่อ (Named Arguments) โดยอนุญาตให้ระบุชื่อพารามิเตอร์และกำหนดค่าให้กับพารามิเตอร์โดยไม่จำเป็นต้องเรียงลำดับตามที่ประกาศไว้ในฟังก์ชัน วิธีการนี้ช่วยลดโอกาสการเกิดข้อผิดพลาดจากการสลับลำดับพารามิเตอร์ โดยเฉพาะในกรณีที่มีพารามิเตอร์จำนวนมาก โดยรูปแบบการส่งค่าให้กับฟังก์ชันแบบระบุชื่อ เป็นดังนี้

รูปแบบ

```
ชื่อฟังก์ชัน(ชื่อพารามิเตอร์: ค่า);
```

ตัวอย่างที่ 16

```
1  greet(title: "Mr.", name: "Nok");
```

จากตัวอย่างที่ 16 เป็นการเรียกใช้ฟังก์ชันชื่อ greet() ที่ได้สร้างไว้ในตัวอย่างที่ 15 โดยส่งค่าพารามิเตอร์ด้วยการระบุชื่อ ประโยคคำสั่งนี้ระบุค่าพารามิเตอร์สองตัว ได้แก่ title และ name โดยกำหนดให้ title มีค่าเป็น *Mr.* และ name มีค่าเป็น *Nok* ลำดับของพารามิเตอร์ในขณะเรียกใช้งานนั้นไม่จำเป็นต้องตรงกับลำดับของการประกาศในฟังก์ชัน แต่มีข้อกำหนดว่า ชื่อที่ใช้ในการเรียกจะต้องตรงกันกับชื่อพารามิเตอร์ที่ประกาศ

ในฟังก์ชันทุกประกาศ กล่าวคือ หากมีการเรียกฟังก์ชันเป็น greet(Name: "Nok") จะเกิดข้อผิดพลาดเนื่องจากตัวพิมพ์ไม่ตรงกับตอนที่ประกาศไว้

6.1.4 การส่งค่ากลับจากฟังก์ชัน

การส่งค่ากลับจากฟังก์ชัน (Return Values) หมายถึง การใช้คำสั่ง return เพื่อส่งข้อมูลจากฟังก์ชันกลับไปยังตำแหน่งที่เรียกใช้ฟังก์ชันนั้น ค่าที่ส่งกลับสามารถเป็นได้ทั้งค่าคงที่ ตัวแปร หรือผลลัพธ์ของการคำนวณ เมื่อฟังก์ชันดำเนินการถึงคำสั่ง return ระบบจะหยุดการทำงานของฟังก์ชันทันที และส่งค่าดังกล่าวกลับไปยังผู้เรียกใช้งาน ซึ่งมักนำค่าที่ได้ไปจัดเก็บไว้ในตัวแปรหรือใช้งานต่อในกระบวนการต่าง ๆ ของโปรแกรม

ฟังก์ชันที่มีการส่งค่ากลับจะมีความยืดหยุ่นและสามารถนำกลับมาใช้งานซ้ำได้สะดวกกว่าฟังก์ชันที่ไม่มี return เนื่องจากสามารถควบคุมผลลัพธ์ที่ได้จากฟังก์ชันได้โดยตรง ในขณะที่ฟังก์ชันที่ไม่มีการส่งค่ากลับมักใช้เพื่อแสดงผลหรือกระทำบางอย่างโดยไม่ต้องการค่าตอบกลับ

รูปแบบ

```
function ชื่อฟังก์ชัน(พารามิเตอร์_1, ..., พารามิเตอร์_n) {
    คำสั่ง;
    ...;
    return ค่าที่ต้องการส่งกลับ;
}
```

ตัวอย่างที่ 17

```
1   <?php
2       function calculateAge($birthYear) {
3           $currentYear = 2568
4           $age = $currentYear - $birthYear;
5           return $age;
6       }
7       // เรียกใช้ฟังก์ชันและเก็บผลลัพธ์ไว้ในตัวแปร
8       $myAge = calculateAge(2538);
9       echo "อายุของ Nok คือ: " . $myAge . " ปี";
10  ?>
```

จากตัวอย่างที่ 17 ฟังก์ชัน calculateAge() ถูกสร้างขึ้นเพื่อรับค่าปีเกิดจากพารามิเตอร์ $birthYear และนำไปลบออกจากปีปัจจุบันซึ่งกำหนดเป็น 2568 จากนั้นส่งค่าที่ได้กลับผ่านคำสั่ง return เมื่อเรียกใช้ฟังก์ชันโดยส่งค่า 2538 เข้าไป ผลลัพธ์ที่ได้จะถูกนำไปเก็บในตัวแปร $myAge ซึ่งมีค่าเป็น 30 และแสดงข้อความว่า *อายุของ Nok คือ: 30 ปี* ตามค่าที่คำนวณได้

6.2 อาร์เรย์

ในภาษา PHP อาร์เรย์ (Array) เป็นโครงสร้างข้อมูลที่สามารถจัดเก็บข้อมูลหลายค่าไว้ในตัวแปรเดียว โดยสามารถเข้าถึงข้อมูลเหล่านั้นผ่านดัชนีหรือคีย์ที่กำหนด อาร์เรย์ใน PHP มีหลักการคล้ายคลึงกับอาร์เรย์ในภาษา JavaScript ที่ได้กล่าวถึงในบทก่อนหน้า ทั้งในด้านโครงสร้างพื้นฐานและการใช้งานร่วมกับลูปหรือฟังก์ชันต่าง ๆ

6.2.1 อาร์เรย์แบบดัชนีตัวเลข

อาร์เรย์แบบดัชนีตัวเลข (Indexed Array) เป็นโครงสร้างข้อมูลพื้นฐานในภาษา PHP ที่ใช้จัดเก็บข้อมูลหลายค่าในตัวแปรเดียว โดยค่าต่าง ๆ ภายในอาร์เรย์จะถูกเข้าถึงผ่านดัชนี ซึ่งเริ่มต้นจากเลข 0 ไปตามลำดับถัดไป อาร์เรย์ประเภทนี้มีลักษณะการทำงานคล้ายกับอาร์เรย์ในภาษา JavaScript และภาษาอื่น ๆ ที่อยู่ในตระกูล C-like

การสร้างอาร์เรย์แบบดัชนีตัวเลขในภาษา PHP สามารถทำได้ 2 รูปแบบ ได้แก่ การใช้คำสั่ง array() และการใช้เครื่องหมายวงเล็บเหลี่ยม [ ] ซึ่งเป็นรูปแบบที่ได้รับความนิยมมากขึ้นในเวอร์ชันใหม่ของภาษา PHP โดยสามารถเข้าถึงสมาชิกแต่ละตัวได้ด้วยการระบุดัชนีของข้อมูลในวงเล็บเหลี่ยมตามตัวอย่างต่อไปนี้

รูปแบบที่ 1

```
$ชื่อตัวแปร = array(ค่าที่1, ค่าที่2, ค่าที่3, ...);
```

รูปแบบที่ 2

```
$ชื่อตัวแปร = [ค่าที่1, ค่าที่2, ค่าที่3, ...];
```

ตัวอย่างที่ 18

```
1  <?php
2      $languages = array("HTML", "CSS", "JavaScript", "PHP");
3      echo $languages[0]; // แสดงผล: HTML
4      $colors = ["red", "green", "blue"];
5      echo $colors[2]; // แสดงผล: blue
6  ?>
```

จากตัวอย่างที่ 18 ตัวแปร $languages และ $colors ถูกกำหนดให้เก็บข้อมูลหลายค่า โดยใช้รูปแบบการสร้างอาร์เรย์แบบดัชนีตัวเลข ในกรณีของ $languages ใช้คำสั่ง array() ในการสร้าง และเมื่อเรียก $languages[0] จะได้ผลลัพธ์เป็นข้อความ *HTML* เนื่องจากดัชนีแรกของอาร์เรย์มีค่าดังกล่าว ส่วนตัวแปร $colors ใช้รูปแบบวงเล็บเหลี่ยม [ ] ซึ่งเป็นรูปแบบที่นิยมใช้ใน PHP เวอร์ชันใหม่ และเมื่อเรียก $colors[2] จะได้ผลลัพธ์เป็น *blue* ซึ่งเป็นข้อมูลที่อยู่ในลำดับที่สามของอาร์เรย์

6.2.2 อาร์เรย์แบบกำหนดชื่อดัชนี

อาร์เรย์แบบกำหนดชื่อดัชนี (Associative Array) เป็นโครงสร้างข้อมูลที่ใช้คู่ของชื่อดัชนี (Key) และค่า (Value) แทนการใช้ดัชนีตัวเลข โดยผู้พัฒนาสามารถกำหนดชื่อของดัชนีให้สื่อความหมายได้ตรงกับข้อมูลที่จัดเก็บ ซึ่งช่วยให้การเข้าถึงข้อมูลภายในอาร์เรย์ง่ายขึ้น

รูปแบบการสร้างอาร์เรย์ชนิดนี้มีลักษณะคล้ายกับออบเจกต์ในภาษา JavaScript หรือโครงสร้าง Map ในภาษาอื่น โดยสามารถสร้างได้โดยใช้คำสั่ง array() หรือวงเล็บเหลี่ยม [ ] ควบคู่กับการกำหนดค่าชื่อดัชนี (Key) และค่าของข้อมูล (Value) ด้วยเครื่องหมาย => โดยมีรูปแบบการสร้างอาร์เรย์และตัวอย่างการเข้าถึงอาร์เรย์ดังนี้

รูปแบบที่ 1

```
$ตัวแปร = array("ชื่อดัชนี1" => ค่าที่1, "ชื่อดัชนี2" => ค่าที่2, ...);
```

รูปแบบที่ 2

```
$ตัวแปร = ["ชื่อดัชนี1" => ค่าที่1, "ชื่อดัชนี2" => ค่าที่2, ...];
```

ตัวอย่างที่ 19

```
1  <?php
2      $student = array(
           "name" => "Nok",
           "department" => "เทคโนโลยีธุรกิจดิจิทัล",
           "year" => 3
       );
3      echo $student["name"]; // แสดงผล: Nok
4      echo $student["year"]; // แสดงผล: 3
5  ?>
```

ในตัวอย่างที่ 19 ตัวแปร $student เป็นอาร์เรย์แบบกำหนดชื่อดัชนีที่ใช้คู่ของ “name”, “department”, และ “year” เป็นชื่อดัชนีที่อ้างอิงถึงข้อมูลของนักศึกษา การเรียก $student["name"] จึงได้ค่าคือ “Nok” และ $student["year"] ได้ค่าคือ 3 การใช้อาร์เรย์ลักษณะนี้ช่วยให้รหัสคำสั่งมีความหมายมากขึ้น โดยเฉพาะเมื่อข้อมูลที่จัดเก็บเกี่ยวข้องกับลักษณะเชิงคุณลักษณะ (Attribute)

การใช้อาร์เรย์แบบกำหนดชื่อดัชนีจะมีบทบาทสำคัญอย่างมากในการทำงานร่วมกับฟอร์มข้อมูลจากฐานข้อมูล หรือการรับข้อมูลที่ส่งมาจากผู้ใช้ในรูปแบบของ key-value pair

6.2.3 อาร์เรย์แบบหลายมิติ

อาร์เรย์แบบหลายมิติ (Multidimensional Array) อาร์เรย์ที่มีอาร์เรย์อื่นเป็นสมาชิกภายในอีกชั้นหนึ่งหรือหลายชั้น โดยใช้สำหรับจัดเก็บข้อมูลที่มีลักษณะเป็นตาราง หรือข้อมูลที่มีโครงสร้างซ้อนกัน เช่น

รายชื่อพร้อมรหัสประจำตัว กลุ่มข้อมูลนักศึกษา กลุ่มสินค้ากับราคาหรือหมวดหมู่ เป็นต้น ซึ่งจะช่วยให้สามารถจัดการข้อมูลได้อย่างมีระบบและเป็นลำดับขั้น

ในภาษา PHP การสร้างอาร์เรย์แบบหลายมิตินั้นทำได้โดยการกำหนดอาร์เรย์ซ้อนในอาร์เรย์หลัก สามารถกำหนดได้ทั้งแบบใช้อาร์เรย์ดัชนีตัวเลข หรืออาร์เรย์แบบกำหนดชื่อดัชนี หรือผสมกันก็ได้ ทั้งนี้หลักการใช้งานอาร์เรย์แบบหลายมิติของ PHP คล้ายคลึงกับแนวคิดของภาษา JavaScript ทำให้สามารถเรียนรู้และประยุกต์ใช้ร่วมกันได้ง่าย

ตัวอย่างที่ 20

```
1   <?php
2       $students = [
                ["id" => "1001", "name" => "Nok", "major" => "DBT"],
                ["id" => "1002", "name" => "Ploy", "major" => "BIT"],
                ["id" => "1003", "name" => "Bank", "major" => "SE"]
3       ];
3       echo $students[0]["name"];  // แสดงผล: Nok
4       echo $students[2]["major"]; // แสดงผล: SE
5   ?>
```

จากตัวอย่างที่ 20 เป็นตัวอย่างโครงสร้างอาร์เรย์แบบ 2 มิติ ตัวแปร $students คืออาร์เรย์หลักที่มีสมาชิกแต่ละตัวเป็นอาร์เรย์แบบกำหนดชื่อดัชนีอีกชั้นหนึ่ง ทำให้สามารถเข้าถึงข้อมูลในรูปแบบตารางได้สะดวก เช่น $students[0]["name"] จะให้ค่าชื่อของนักศึกษาคนแรก ส่วน $students[2]["major"] จะให้ค่าหลักสูตรของนักศึกษาคนที่สาม

อาร์เรย์แบบหลายมิติมีประโยชน์มากในบริบทของการพัฒนาเว็บแอปพลิเคชัน เช่น การสร้างตารางแสดงข้อมูล การอ่านค่าจากฐานข้อมูล และการจัดการข้อมูลที่มีโครงสร้างซับซ้อน ทั้งนี้ควรระมัดระวังในการเข้าถึงข้อมูลแต่ละระดับเพื่อป้องกันข้อผิดพลาดที่อาจเกิดจากการเข้าถึงตำแหน่งที่ไม่ถูกต้อง

6.2.4 ฟังก์ชันสำคัญในการจัดการอาร์เรย์

ภาษา PHP มีฟังก์ชันสำเร็จรูป (Built-in Functions) สำหรับการจัดการอาร์เรย์หลากหลายรูปแบบ ซึ่งช่วยให้การเพิ่ม ลบ ตรวจสอบ และจัดเรียงข้อมูลภายในอาร์เรย์สามารถทำได้อย่างมีประสิทธิภาพ ฟังก์ชันเหล่านี้ช่วยลดความซับซ้อนของการเขียนโปรแกรม และสามารถนำไปประยุกต์ใช้ในหลากหลายสถานการณ์ของเว็บแอปพลิเคชัน

1) ฟังก์ชัน count()

ฟังก์ชัน count() ใช้กับค่าชนิดอาร์เรย์โดยตรง หรือตัวแปรที่เก็บข้อมูลชนิดอาร์เรย์ โดยจะคืนค่าเป็นจำนวนเต็มที่แสดงจำนวนขององค์ประกอบทั้งหมดที่อยู่ในอาร์เรย์ เหมาะสำหรับใช้ตรวจสอบว่าอาร์เรย์ว่ามีข้อมูลหรือไม่ หรือใช้ในการควบคุมการทำงานร่วมกับโครงสร้างควบคุมแบบวนซ้ำ มีรูปแบบดังนี้

รูปแบบ

count(อาร์เรย์หรือตัวแปรชนิดอาร์เรย์)

ตัวอย่างที่ 21

```
1  <?php
2      $products = ["Keyboard", "Mouse", "Monitor"];
3      echo "มีสินค้า " . count($products) . " รายการ";
4  ?>
```

จากตัวอย่างที่ 21 ตัวแปร $products เป็นอาร์เรย์ที่มีสมาชิกจำนวน 3 รายการ ฟังก์ชัน count($products) จะคืนค่า 3 และนำค่าดังกล่าวไปแสดงผลข้อความ *มีสินค้า 3 รายการ* บนหน้าเว็บ

2) ฟังก์ชัน array_push()

ฟังก์ชัน array_push() ใช้สำหรับ เพิ่มข้อมูลใหม่เข้าไปต่อท้ายอาร์เรย์ โดยสามารถเพิ่มได้ครั้งละหนึ่งค่าหรือหลายค่า ค่าที่เพิ่มจะถูกจัดให้อยู่ในลำดับถัดไปของอาร์เรย์โดยอัตโนมัติ เหมาะสำหรับการประยุกต์ใช้เมื่อมีรายการข้อมูลที่ต้องการจัดเก็บแบบเรียงลำดับตามลำดับเวลา หรือสถานการณ์ที่ต้องการเก็บรายการข้อมูลที่ผู้ใช้กรอกซ้ำได้เรื่อย ๆ

รูปแบบ

array_push(อาร์เรย์เป้าหมาย, ค่าที่จะเพิ่ม1, ค่าที่จะเพิ่ม2, ...);

ตัวอย่างที่ 22

```
1  <?php
2      $students = ["Nok", "Ann"];
3      array_push($students, "Kong", "Gaia");
4      print_r($students);
5  ?>
```

จากตัวอย่างที่ 22 ตัวแปร $students ถูกกำหนดให้เป็นอาร์เรย์ที่ประกอบด้วยค่าคงที่ 2 ค่า คือ "Nok" และ "Ann" จากนั้นใช้ฟังก์ชัน array_push() เพื่อเพิ่มค่า "Kong" และ "Gaia" ต่อท้ายอาร์เรย์เดิม ผลลัพธ์ที่ได้คือ $students จะมีสมาชิกทั้งหมด 4 ค่า โดยมีลำดับดังนี้ "Nok", "Ann", "Kong" และ "Gaia" การ

แสดงผลด้วย print_r() จะแสดงโครงสร้างของอาร์เรย์ทั้งหมดพร้อมดัชนี ซึ่งเหมาะสำหรับการตรวจสอบค่าภายในอาร์เรย์อย่างรวดเร็ว

3) ฟังก์ชัน in_array()

ฟังก์ชัน in_array() ใช้สำหรับตรวจสอบว่า ค่าหนึ่งค่ามีอยู่ในอาร์เรย์หรือไม่ โดยจะคืนค่า true เมื่อพบว่า มีค่าที่ตรงกันภายในอาร์เรย์ และคืนค่า false เมื่อไม่พบ ฟังก์ชันนี้เหมาะสำหรับใช้ตรวจสอบว่าข้อมูลที่ผู้ใช้ป้อนเข้ามาตรงกับค่าที่อนุญาตไว้หรือไม่ เช่น การตรวจสอบว่าค่าจังหวัดที่เลือกอยู่ในรายชื่อจังหวัดที่ระบบกำหนดไว้

รูปแบบ

in_array(ค่าที่ต้องการตรวจสอบ, อาร์เรย์เป้าหมาย);

ตัวอย่างที่ 23

```
1  <?php
2      $allowed = ["Web App", "Digital Business", "Multimedia"];
3      $input = "Web App";
4      if (in_array($input, $allowed)) {
5          echo "หมวดหมู่ที่เลือกได้รับอนุญาต";
6      }
7  ?>
```

ในตัวอย่างที่ 23 ตัวแปร $allowed คืออาร์เรย์ที่เก็บชื่อหมวดหมู่ที่อนุญาตให้เลือก ส่วน $input คือค่าที่ผู้ใช้ส่งเข้ามา หากค่าดังกล่าว ("Web App") มีอยู่ใน $allowed ฟังก์ชัน in_array() จะคืนค่า true ทำให้โปรแกรมแสดงข้อความว่า “หมวดหมู่ที่เลือกได้รับอนุญาต” ทั้งนี้การเปรียบเทียบค่าจะเป็นแบบ ค่าเท่ากัน (==) โดยไม่คำนึงถึงชนิดข้อมูล เว้นแต่จะระบุพารามิเตอร์ที่สามเป็น true เพื่อใช้การเปรียบเทียบแบบเข้มงวด (===)

4) ฟังก์ชัน array_merge()

ฟังก์ชัน array_merge() ใช้สำหรับ รวมอาร์เรย์สองชุดขึ้นไปเข้าด้วยกันเป็นอาร์เรย์ใหม่ โดยสมาชิกของอาร์เรย์ที่รวมจะถูกเรียงตามลำดับจากซ้ายไปขวา หากมีคีย์เป็นเลขดัชนี (index) ฟังก์ชันจะทำการจัดลำดับใหม่โดยอัตโนมัติ ส่วนกรณีที่เป็น associative array และมีคีย์ซ้ำกัน ค่าในอาร์เรย์ด้านขวาจะถูกใช้แทนที่ค่าเดิม

รูปแบบ

array_merge(อาร์เรย์1, อาร์เรย์2, ...);

ตัวอย่างที่ 24

```
1  <?php
2      $groupA = ["Nok", "Ann"];
3      $groupB = ["Kong", "Gaia"];
4      $allStudents = array_merge($groupA, $groupB);
5      print_r($allStudents);
6  ?>
```

จากตัวอย่างที่ 24 ตัวแปร $groupA และ $groupB เป็นอาร์เรย์ที่ประกอบด้วยชื่อของสมาชิกในแต่ละกลุ่ม เมื่อใช้ array_merge() เพื่อรวมอาร์เรย์ทั้งสองเข้าด้วยกันโดยเก็บไว้ในตัวแปร $allStudents จะได้อาร์เรย์ใหม่ที่มีสมาชิกทั้งหมดคือ "Nok", "Ann", "Kong" และ "Gaia" ตามลำดับ การแสดงผลด้วย print_r() จะแสดงสมาชิกในอาร์เรย์ใหม่โดยมีดัชนีเรียงลำดับใหม่เริ่มจาก 0 ถึง 3 ซึ่งเป็นค่าดัชนีของ $allStudents ที่ถูกจัดสรรโดยอัตโนมัติ

5) ฟังก์ชัน array_keys()

ฟังก์ชัน array_keys() ใช้สำหรับดึงเอาคีย์ทั้งหมดของอาร์เรย์ออกมาเป็นอาร์เรย์ใหม่ โดยเฉพาะกรณีที่อาร์เรย์มีโครงสร้างแบบกำหนดชื่อดัชนี (Associative Array) ซึ่งแต่ละรายการมีชื่อแทนค่า เช่น ['name' => 'Nok', 'email' => 'nok@example.com'] การใช้ array_keys() จะช่วยให้สามารถแยกรายการคีย์มาใช้งาน เช่น การตรวจสอบรายการช่องที่ผู้ใช้กรอกข้อมูล หรือสร้างตารางจากชื่อฟิลด์ในฐานข้อมูลอย่างเป็นระบบ

รูปแบบ

array_keys(อาร์เรย์);

ตัวอย่างที่ 25

```
1  <?php
2      $user = [
           "name" => "Nok",
           "email" => "nok@example.com",
           "age" => 22
3      ];

4      $keys = array_keys($user);
5      print_r($keys);
6  ?>
```

จากตัวอย่างที่ 25 ตัวแปร $user คืออาร์เรย์ที่มีการกำหนดชื่อดัชนีอย่างชัดเจน ในที่นี้ประกอบด้วย name, email, และ age เมื่อใช้ฟังก์ชัน array_keys() จะได้ผลลัพธ์เป็นอาร์เรย์ใหม่ที่ประกอบด้วยคีย์ทั้งสามนี้ ได้แก่ ["name", "email", "age"] ซึ่งสามารถนำไปใช้งานต่อ เช่น การสร้างฟอร์มโดยวนลูปจากชื่อคีย์หรือแสดงหัวตารางข้อมูลในเว็บแอปพลิเคชันอย่างเป็นระบบ

6) ฟังก์ชัน array_values()

ฟังก์ชัน array_values() ใช้สำหรับดึงค่าทั้งหมดในอาร์เรย์ออกมา โดยจะคืนค่าเป็นอาร์เรย์ที่มีเฉพาะค่าจากแต่ละสมาชิกเท่านั้น และจะจัดลำดับดัชนีใหม่เป็นเลขจำนวนเต็มตามลำดับเริ่มจาก 0 เสมอ เหมาะสำหรับการแยกเฉพาะค่าจากอาร์เรย์ที่มีชื่อดัชนี (Associative Array) หรือการเตรียมข้อมูลก่อนส่งออกหรือแสดงผลในรูปแบบที่ไม่ต้องการชื่อคีย์

รูปแบบ

array_values(อาร์เรย์);

ตัวอย่างที่ 26

```
1  <?php
2      $user = [
           "name" => "Nok",
           "email" => "nok@abc.com",
           "age" => 22
3      ];

4      $values = array_values($user);
5      print_r($values);
6  ?>
```

จากตัวอย่างที่ 26 ตัวแปร $user คืออาร์เรย์แบบกำหนดชื่อดัชนี ซึ่งมีค่าข้อมูลผู้ใช้แต่ละรายการ เช่น ชื่อ อีเมล และอายุ เมื่อใช้ฟังก์ชัน array_values() จะได้ผลลัพธ์เป็นอาร์เรย์ใหม่ที่ประกอบด้วยเฉพาะค่าของแต่ละรายการ ได้แก่ ["Nok", "nok@abc.com", 22] โดยไม่มีชื่อคีย์ติดมาด้วย ซึ่งสามารถนำไปใช้ในกรณีที่ต้องการจัดเก็บหรือแสดงผลข้อมูลโดยไม่ต้องอ้างอิงชื่อคีย์เดิม เช่น การสร้างแถวข้อมูลในตาราง HTML หรือการแปลงข้อมูลเป็น JSON

7) ฟังก์ชัน implode()

ฟังก์ชัน implode() ใช้สำหรับรวมค่าหลายค่าในอาร์เรย์ให้กลายเป็นข้อความเดียว โดยใช้ตัวคั่น (Separator) ที่กำหนด เช่น เครื่องหมายจุลภาค หรือเว้นวรรค เหมาะสำหรับกรณีที่ต้องการแปลงข้อมูล

จากอาร์เรย์ให้อยู่ในรูปแบบข้อความ เช่น การรวมชื่อหลายชื่อเพื่อแสดงผลในหน้าเว็บ หรือการเตรียมข้อมูลก่อนบันทึกลงฐานข้อมูล

รูปแบบ

implode(ตัวคั่น, อาร์เรย์);

ตัวอย่างที่ 27

```
1  <?php
2      $tags = ["PHP", "MySQL", "HTML", "CSS"];
3      $result = implode(", ", $tags);
4      echo $result;
5  ?>
```

จากตัวอย่างที่ 27 ตัวแปร $tags เป็นอาร์เรย์ที่เก็บคำสำคัญของเทคโนโลยีเว็บ เมื่อใช้ฟังก์ชัน implode() โดยระบุเครื่องหมายคั่นเป็นจุลภาคและเว้นวรรค (,) จะได้ผลลัพธ์เป็นข้อความ "PHP, MySQL, HTML, CSS" ซึ่งเหมาะสำหรับแสดงในรูปแบบข้อความธรรมดาหรือใช้เป็นค่าที่ส่งผ่านแบบฟอร์มในการแก้ไขข้อมูล

8) ฟังก์ชัน explode()

ฟังก์ชัน explode() ใช้สำหรับแยกข้อความออกเป็นอาร์เรย์โดยใช้ตัวคั่น (Delimiter) ที่กำหนด เช่น เครื่องหมายจุลภาค หรือเว้นวรรค เหมาะสำหรับกรณีที่ได้รับค่าข้อความจากผู้ใช้ที่มีลักษณะเป็นรายการ เช่น รายชื่อที่กรอกด้วยเครื่องหมายคั่น แล้วต้องการนำไปประมวลผลต่อในรูปแบบอาร์เรย์

รูปแบบ

explode(ตัวคั่น, ข้อความ);

ตัวอย่างที่ 28

```
1  <?php
2      $data = "PHP,MySQL,HTML,CSS";
3      $skills = explode(",", $data);
4      print_r($skills);
5  ?>
```

จากตัวอย่างที่ 29 ตัวแปร $data เก็บข้อความที่ประกอบด้วยชื่อทักษะต่าง ๆ คั่นด้วยเครื่องหมายจุลภาค (",") เมื่อใช้ฟังก์ชัน explode() จะได้ตัวแปร $skills ที่เป็นอาร์เรย์ซึ่งประกอบด้วยสมาชิก 4

ค่า ได้แก่ "PHP", "MySQL", "HTML", และ "CSS" ซึ่งสามารถนำไปใช้ในลูป foreach หรือประมวลผลทีละรายการต่อไป

7. สรุปท้ายบท

PHP คือภาษาโปรแกรมแบบฝั่งผู้ให้บริการที่ได้รับความนิยมในการสร้างเว็บแอปพลิเคชัน เนื่องจากมีลักษณะเป็นโอเพนซอร์ส ยืดหยุ่น และสามารถทำงานร่วมกับระบบจัดการฐานข้อมูลได้หลากหลาย นอกจากนี้ยังสามารถผสานรวมกับ HTML, CSS และ JavaScript ได้อย่างลงตัว ส่งผลให้ PHP เป็นภาษาหลักที่ใช้ในระบบเว็บจำนวนมากทั่วโลก ความสามารถในการทำงานร่วมกับระบบเครือข่าย โปรโตคอล และเว็บเซอร์วิสต่าง ๆ ทำให้ PHP เหมาะสมอย่างยิ่งต่อการพัฒนาเว็บแอปพลิเคชันที่ซับซ้อนและต้องการการตอบสนองแบบไดนามิก

การเขียนโปรแกรมด้วยภาษา PHP มีโครงสร้างที่เข้าใจง่าย เริ่มต้นจากการกำหนดแท็กเปิดและปิด เพื่อบ่งชี้ส่วนที่เป็นคำสั่ง PHP ภายในเอกสารเว็บ พร้อมด้วยกฎไวยากรณ์ เช่น การเขียนประโยคคำสั่ง การใช้บรรทัดหมายเหตุ และการใช้งานนิพจน์และตัวดำเนินการที่หลากหลาย เช่น คำนวณ เปรียบเทียบ และกำหนดค่า การทำความเข้าใจไวยากรณ์เหล่านี้มีความสำคัญต่อการสร้างรหัสคำสั่งที่ถูกต้องและสามารถทำงานได้ตามที่ตั้งใจไว้

แนวคิดเรื่องตัวแปรและชนิดของข้อมูลในภาษา PHP เป็นหัวใจสำคัญในการจัดเก็บและจัดการข้อมูลภายในระบบ โดย PHP มีการกำหนดชนิดข้อมูลแบบพลวัต ทำให้ไม่จำเป็นต้องระบุชนิดข้อมูลอย่างชัดเจนเมื่อประกาศตัวแปร อย่างไรก็ตาม ผู้พัฒนาควรเข้าใจชนิดข้อมูลพื้นฐาน เช่น จำนวนเต็ม ข้อความ บูลีน และอาร์เรย์ ตลอดจนการตรวจสอบชนิดข้อมูลด้วยฟังก์ชันในตัว เพื่อป้องกันข้อผิดพลาดที่อาจเกิดจากการใช้งานข้อมูลผิดประเภท

ฟังก์ชัน echo และ print เป็นเครื่องมือพื้นฐานในการแสดงผลลัพธ์บนเว็บเบราว์เซอร์ โดยสามารถแสดงข้อความหรือค่าตัวแปรได้โดยตรง ความสามารถในการแยกส่วนของคำสั่ง PHP ออกจาก HTML, CSS และ JavaScript ช่วยให้สามารถจัดการการประมวลผลและการนำเสนอได้อย่างเป็นระบบ ส่งผลให้รหัสคำสั่งอ่านง่ายและเหมาะสมกับแนวคิดของการออกแบบเว็บแอปพลิเคชันสมัยใหม่

การควบคุมทิศทางการทำงานของโปรแกรมทำได้ผ่านโครงสร้างการตัดสินใจและการวนซ้ำ ซึ่งช่วยให้โปรแกรมสามารถตอบสนองต่อเงื่อนไขต่าง ๆ และทำงานซ้ำตามที่ต้องการได้ โครงสร้างเหล่านี้ ได้แก่ if, if-else, switch, for, และ while ซึ่งล้วนเป็นกลไกสำคัญที่ทำให้ PHP สามารถควบคุมตรรกะของเว็บแอปพลิเคชันได้อย่างยืดหยุ่น

การสร้างฟังก์ชันและการใช้งานอาร์เรย์เป็นโครงสร้างข้อมูลที่สำคัญใน PHP โดยผู้พัฒนาสามารถสร้างฟังก์ชันของตนเองเพื่อแยกกระบวนการทำงานให้เป็นระบบ และใช้อาร์เรย์ในการจัดเก็บข้อมูลหลายค่าในตัวแปรเดียว ฟังก์ชันที่เกี่ยวข้องกับอาร์เรย์ เช่น count(), array_push(), implode(), explode() ฯลฯ ยังช่วยให้การจัดการข้อมูลเป็นไปอย่างมีประสิทธิภาพ ซึ่งมีความสำคัญโดยตรงต่อการรับข้อมูลจากฟอร์มและจัดเก็บข้อมูลในฐานข้อมูล

8. กิจกรรมท้ายบท

8.1 อธิบายความหมายและบทบาทของภาษา PHP ในการพัฒนาเว็บแอปพลิเคชัน พร้อมอธิบายกลไกการทำงานร่วมกันระหว่างซอฟต์แวร์ผู้ให้บริการเว็บและตัวแปลภาษา PHP ตั้งแต่รับคำร้องขอจากผู้ใช้จนถึงส่งผลลัพธ์กลับ

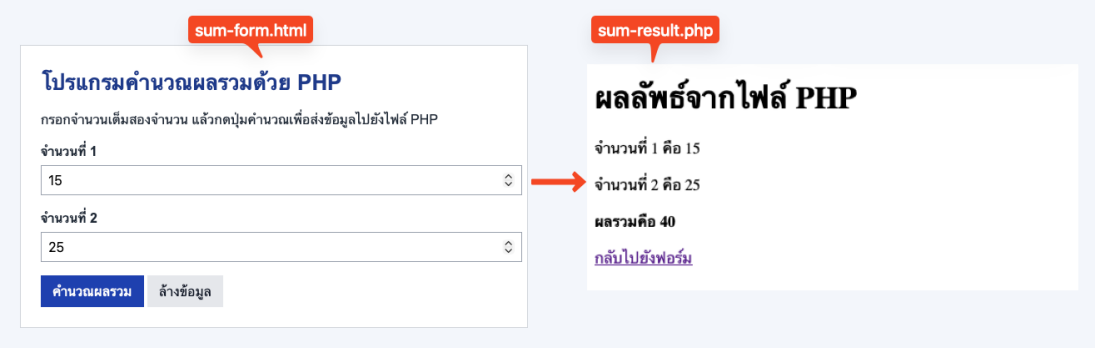
8.2 อธิบายข้อกำหนดที่เกี่ยวข้องกับการสร้างไฟล์ PHP และหลักการกำหนดโครงสร้างของไฟล์ PHP ที่เหมาะสมตามมาตรฐาน PSR-1 และ PSR-12 พร้อมยกตัวอย่างประกอบ

8.3 จงอธิบายแนวคิดของตัวแปรและชนิดข้อมูลในภาษา PHP โดยระบุรูปแบบการประกาศตัวแปร ชนิดข้อมูลพื้นฐานทั้งหมดที่ PHP รองรับ และตัวอย่างการใช้งานพอสังเขป

8.4 จงอธิบายแนวคิดของฟังก์ชันในภาษา PHP และยกตัวอย่างการสร้างฟังก์ชันที่รับพารามิเตอร์และส่งค่ากลับอย่างน้อย 1 ตัวอย่าง พร้อมอธิบายผลลัพธ์ที่เกิดขึ้น

8.5 จงอธิบายแนวคิดของอาร์เรย์ในภาษา PHP และเขียนรหัส PHP เพื่อสร้างอาร์เรย์เก็บรายชื่อสินค้าอย่างน้อย 3 รายการ จากนั้นใช้โครงสร้างการวนซ้ำเพื่อแสดงรายชื่อสินค้าแต่ละรายการด้วยฟังก์ชัน echo พร้อมอธิบายโครงสร้างรหัสที่ใช้

8.6 จงสร้างฟอร์ม HTML สำหรับรับค่าจำนวนเต็มสองจำนวน แล้วส่งข้อมูลไปยังไฟล์ PHP เพื่อคำนวณผลรวมและแสดงผลลัพธ์ออกทางเว็บเบราว์เซอร์ลักษณะดังภาพที่ 11.2 โดยมีข้อกำหนดดังนี้



ภาพที่ 11.2 ตัวอย่างฟอร์มที่สร้างด้วยภาษา HTML และผลลัพธ์จากการประมวลผลของภาษา PHP

8.6.1 สร้างไฟล์ sum-form.html เพื่อแสดงฟอร์มรับค่าจำนวนเต็มสองจำนวน โดยกำหนดชื่อช่องรับข้อมูลเป็น number1 และ number2

8.6.2 กำหนดให้ฟอร์มในไฟล์ sum-form.html ส่งข้อมูลด้วยวิธี POST ไปยังไฟล์ sum-result.php

8.6.3 สร้างไฟล์ sum-result.php เพื่อรับค่าจาก $_POST['number1'] และ $_POST['number2'] พร้อมตรวจสอบว่ามีการส่งค่ามาก่อนประมวลผล

8.6.4 เขียนฟังก์ชัน calculateSum($a, $b) เพื่อรับค่าจำนวนเต็มสองจำนวน แล้วส่งค่าผลรวมกลับ

8.6.5 เรียกใช้ฟังก์ชัน calculateSum() และแสดงค่าจำนวนที่รับมา รวมทั้งผลรวม ด้วยคำสั่ง echo

8.6.6 เปิดไฟล์ sum-form.html ผ่านเว็บเบราว์เซอร์ กรอกค่าตัวเลข ทดสอบการส่งข้อมูล และตรวจสอบผลลัพธ์ที่แสดงจากไฟล์ sum-result.php

8.6.7 ทดลองเปลี่ยนการส่งข้อมูลจากฟอร์มในไฟล์ sum-form.html ด้วยวิธี GET และเปลี่ยนการรับค่าในไฟล์ sum-result.php ให้ถูกต้อง

8.7 จงเขียนรหัส PHP ในไฟล์ชื่อ grade.php เพื่อสร้างอาร์เรย์ 2 มิติสำหรับเก็บข้อมูลนักศึกษา และแสดงผลการประเมินคะแนนออกทางเว็บเบราว์เซอร์ โดยมีข้อกำหนดดังนี้

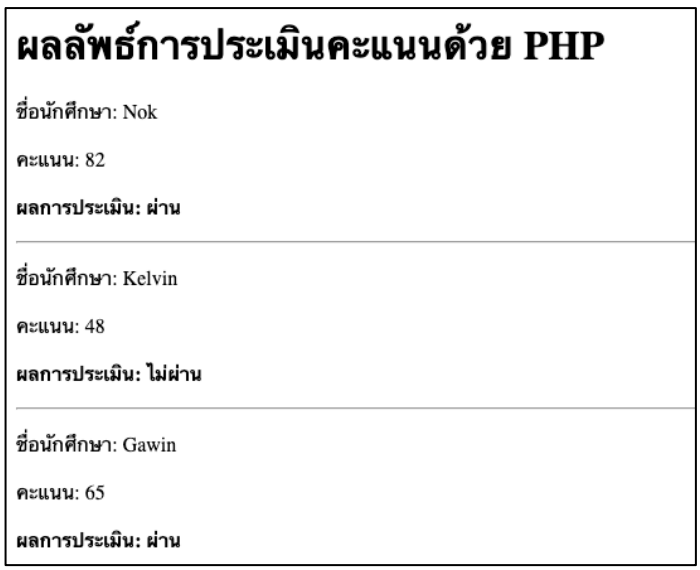
8.7.1 สร้างอาร์เรย์ชื่อ $students เป็นอาร์เรย์แบบดัชนีตัวเลข สำหรับเก็บรายการนักศึกษาอย่างน้อย 3 คน โดยสมาชิกแต่ละตัวของอาร์เรย์ $students เป็นอาร์เรย์แบบกำหนดชื่อดัชนี ซึ่งแทนข้อมูลของนักศึกษา 1 คน

8.7.2 อาร์เรย์แบบกำหนดชื่อดัชนีของนักศึกษาแต่ละคนต้องมีดัชนี name สำหรับเก็บชื่อนักศึกษาเป็นข้อมูลชนิดข้อความ และดัชนี score สำหรับเก็บคะแนนเป็นข้อมูลชนิดจำนวนเต็ม

8.7.3 ใช้โครงสร้างการวนซ้ำ foreach เพื่ออ่านข้อมูลนักศึกษาแต่ละคนจากอาร์เรย์ $students

8.7.4 ใช้คำสั่งเงื่อนไข if-else เพื่อตรวจสอบผลการประเมิน โดยกำหนดให้คะแนนตั้งแต่ 50 คะแนนขึ้นไปมีผลเป็น “ผ่าน” และคะแนนน้อยกว่า 50 คะแนนมีผลเป็น “ไม่ผ่าน”

8.7.5 ใช้ฟังก์ชัน echo เพื่อแสดงชื่อนักศึกษา คะแนน และผลการประเมินของนักศึกษาแต่ละคนออกทางเว็บเบราว์เซอร์ ผลลัพธ์ที่คาดหวังแสดงดังภาพที่ 11.3



ภาพที่ 11.3 ผลลัพธ์ที่คาดหวังในการแสดงข้อมูลจากอาร์เรย์สองมิติในภาษา PHP



การจัดการข้อมูลจากผู้ใช้ด้วย PHP

บทที่ 12 มุ่งเน้นการนำเสนอแนวทางการจัดการข้อมูลจากผู้ใช้ในเว็บแอปพลิเคชันด้วยภาษา PHP โดยผู้อ่านจะได้เรียนรู้กลไกการรับค่าจากฟอร์ม การตรวจสอบและการล้างข้อมูลเพื่อป้องกันความเสี่ยงจากข้อมูลไม่ปลอดภัย ตลอดจนกระบวนการติดต่อกับฐานข้อมูลด้วย PDO และการสร้างระบบ CRUD เบื้องต้น หัวข้อหลักในบทนี้ประกอบด้วย

1. การรับค่าจากฟอร์ม
2. การทำงานร่วมกับฐานข้อมูล
3. ระบบ CRUD
4. สรุปท้ายบท

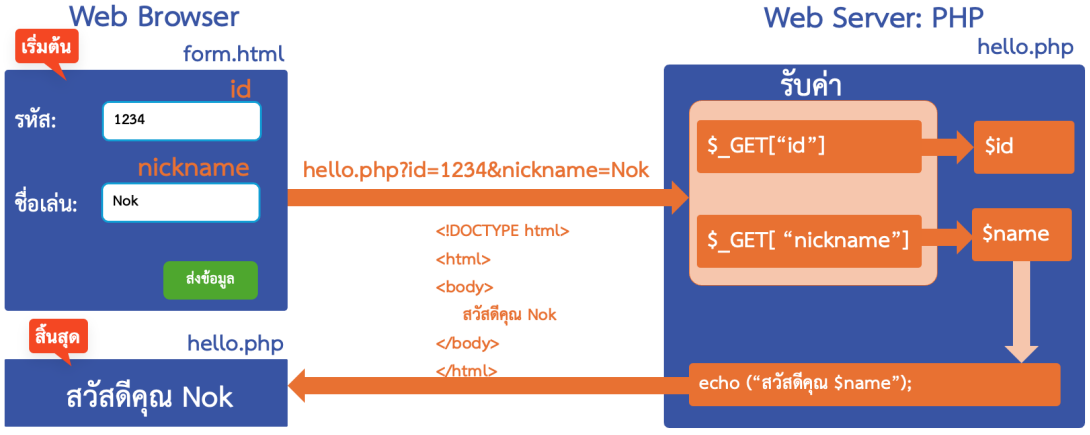
1. การรับค่าจากฟอร์ม

ในการพัฒนาเว็บแอปพลิเคชัน การรับข้อมูลจากผู้ใช้งานผ่านฟอร์มบนหน้าเว็บถือเป็นหนึ่งในกลไกสำคัญที่ทำให้ระบบสามารถโต้ตอบกับผู้ใช้งานได้อย่างมีประสิทธิภาพ โดยฟอร์มทำหน้าที่เป็นช่องทางให้ผู้ใช้กรอกข้อมูลที่จำเป็น เช่น ชื่อผู้ใช้ รหัสผ่าน อีเมล หมายเลขโทรศัพท์ ความคิดเห็น หรือแม้กระทั่งไฟล์แนบ แล้วส่งข้อมูลเหล่านี้ไปยังฝั่งผู้ให้บริการ (Backend) เพื่อทำการประมวลผล วิเคราะห์ ตรวจสอบความถูกต้อง หรือจัดเก็บในระบบฐานข้อมูลต่อไป

จากเนื้อหาของบทที่ 6 ที่ผ่านมา ผู้อ่านได้เรียนรู้เกี่ยวกับการสร้างฟอร์มด้วยแท็ก HTML เช่น <form>, <input>, <textarea>, <select> ไปแล้ว ซึ่งในหัวข้อนี้จะเน้นที่การรับข้อมูลจากฟอร์มในฝั่งของผู้ให้บริการโดยใช้ภาษา PHP ดังนี้

1.1 กลไกการรับค่าจากฟอร์ม

เพื่อให้เข้าใจกลไกการรับข้อมูลจากผู้ใช้งานได้อย่างเป็นระบบ หัวข้อนี้จะนำเสนอขั้นตอนการทำงานของฟอร์ม HTML ที่ส่งข้อมูลด้วยวิธี GET โดยจำลองสถานการณ์ตั้งแต่ผู้ใช้เปิดหน้าเว็บ กรอกข้อมูลในฟอร์ม และกดปุ่มส่งข้อมูล ไปจนถึงการที่ฝั่งผู้ให้บริการรับและประมวลผลข้อมูลด้วยภาษา PHP พร้อมทั้งแสดงผลลัพธ์กลับสู่ผู้ใช้ โดยใช้ภาพที่ 12.1 ประกอบการอธิบายดังนี้



ภาพที่ 12.1 กลไกการรับค่าจากฟอร์ม

ขั้นตอนที่ 1) ผู้ใช้ร้องขอหน้าเว็บชื่อ form.html จากเครื่องผู้ให้บริการเว็บ

กลไกการรับค่าจากฟอร์มจะเริ่มต้นจากการที่ผู้ใช้ร้องขอหน้าเว็บชื่อ form.html จากเครื่องผู้ให้บริการเว็บโดยเปิดผ่านเว็บเบราว์เซอร์ เมื่อผู้ใช้ได้รับไฟล์ form.html แสดงในเว็บเบราว์เซอร์แล้ว องค์ประกอบของฟอร์มที่ได้รับมีดังนี้

- action = "hello.php" หมายความว่า เมื่อผู้ใช้กดปุ่มส่งข้อมูล เว็บเบราว์เซอร์จะส่งข้อมูลกลับไปยังเครื่องผู้ให้บริการเว็บที่ส่งฟอร์มนี้มา

- method = "GET" หมายความว่า การส่งข้อมูลไปยังปลายทางจะส่งด้วยวิธี GET ซึ่งจะแสดงข้อมูลที่ส่งใน URL ด้วย
- ช่องกรอกรหัส มีการกำหนดแอตทริบิวต์ name="id" เพื่อใช้เป็นชื่ออ้างอิงในการรับค่าจากฟอร์มในฝั่งเครื่องผู้ให้บริการ
- ช่องกรอกชื่อเล่น มีการกำหนดแอตทริบิวต์ name="nickname" เพื่อใช้เป็นชื่ออ้างอิงในการรับค่าจากฟอร์มในฝั่งเครื่องผู้ให้บริการ

ขั้นตอนที่ 2) ผู้ใช้กรอกข้อมูลในฟอร์ม

ผู้ใช้ทำการกรอกข้อมูลในฟอร์มโดยกรอกในช่องรหัสมีค่าเป็น 1234 และช่องชื่อเล่นเป็น Nok

ขั้นตอนที่ 3) เว็บเบราว์เซอร์จัดรูปแบบข้อมูลและส่งไปยังเครื่องผู้ให้บริการเว็บ

เมื่อผู้ใช้คลิกปุ่มส่งข้อมูล เว็บเบราว์เซอร์จะรวบรวมค่าจากช่องกรอกข้อมูลในฟอร์ม โดยจัดรูปแบบให้อยู่ในลักษณะของข้อมูลพารามิเตอร์ที่แนบท้าย URL (Query String) ทั้งนี้เนื่องจากฟอร์มมีการกำหนดวิธีการส่งข้อมูลเป็นแบบ GET ข้อมูลจึงถูกรวมไว้ในส่วนท้ายของ URL โดยอยู่ในรูปของคู่ชื่อ–ค่า ที่สร้างจากแอตทริบิวต์ name ของช่องกรอกข้อมูลแต่ละรายการ จับคู่กับค่าที่ผู้ใช้กรอก เช่น id=1234 และ nickname=Nok

จากนั้นเว็บเบราว์เซอร์จะส่งคำขอ (HTTP Request) ไปยังปลายทางที่ระบุไว้ในแอตทริบิวต์ action ของฟอร์ม เช่น hello.php?id=1234&nickname=Nok ในกรณีที่ฟอร์มกำหนดวิธีการส่งข้อมูลเป็นแบบ POST แทนที่จะส่งพารามิเตอร์แนบท้าย URL ระบบจะจัดเก็บข้อมูลไว้ในเนื้อหาของคำขอ (Request Body) แล้วส่งไปยังปลายทางเช่นกัน โดยกระบวนการทำงานในภาพรวมยังคงคล้ายกัน ต่างกันเพียงตำแหน่งของข้อมูลและรูปแบบการเข้าถึงในฝั่งผู้ให้บริการเท่านั้น

ขั้นตอนที่ 4) เครื่องผู้ให้บริการเว็บได้รับข้อมูลและประมวลผลข้อมูล

เมื่อเครื่องผู้ให้บริการเว็บได้รับคำขอ โปรแกรมผู้ให้บริการเว็บจะทราบว่าผู้ใช้ต้องการเรียกดูไฟล์ชื่อ hello.php จึงทำการค้นหาไฟล์ในที่เก็บทรัพยากรของเว็บไซต์ (Document Root) เมื่อพบไฟล์ดังกล่าว ก็จะจัดส่งให้ตัวแปลภาษา PHP ตามกลไกที่ได้กล่าวถึงในตอนต้นของบทนี้ พร้อมกับพารามิเตอร์ของ URL ซึ่งจะถูกเก็บไว้ในตัวแปรพิเศษ $_GET (หากส่งด้วยวิธี POST จะเก็บไว้ในตัวแปรพิเศษ $_POST)

การจัดเก็บข้อมูลในตัวแปรพิเศษนั้นจะเป็นลักษณะอาร์เรย์แบบกำหนดชื่อดัชนีที่ใช้ชื่อแอตทริบิวต์ของช่องกรอกข้อมูลเป็นคีย์ และค่าของช่องกรอกนั้นเป็นค่าของอาร์เรย์ ในลักษณะ ['id'=>'1234', 'nickname'=>'Nok']

หากคำสั่งในภาษา PHP ต้องการใช้ข้อมูลจากฟอร์ม จะเข้าถึงจากตัวแปรพิเศษนั้นและใช้วิธีการดึงข้อมูลจากอาร์เรย์แบบกำหนดชื่อดังที่กล่าวถึงในหัวข้อที่ 6 จากภาพเป็นลักษณะคำสั่ง $name = $_GET["nickname"] หมายความว่า ให้นำค่าในตัวแปรพิเศษในตำแหน่งที่มีชื่อดัชนีเป็น nickname ไปกำหนดค่าให้กับตัวแปรชื่อ $name จากนั้นจะทำการแสดงผลโดยนำข้อความคำว่า "สวัสดีคุณ" ต่อด้วยค่าในตัวแปร $name

เมื่อประมวลผลแล้วจะได้เป็นข้อความ "สวัสดีคุณ Nok" ตัวแปลภาษา PHP จะส่งผลลัพธ์ที่มีข้อความดังกล่าวให้ปรากฏอยู่ในส่วน body ของ HTML ก่อนส่งกลับไปยังโปรแกรมผู้ให้บริการเว็บ

ขั้นตอนที่ 5) เครื่องผู้ให้บริการเว็บส่งผลลัพธ์กลับไปยังเว็บเบราว์เซอร์

เมื่อโปรแกรมผู้ให้บริการเว็บได้ผลลัพธ์จากการประมวลผลของตัวแปลภาษา PHP จะส่งผลนั้นกลับไปยังเว็บเบราว์เซอร์ (HTTP Response) และแสดงข้อความในหน้าเว็บเป็น "สวัสดีคุณ Nok"

ขั้นตอนเหล่านี้แสดงให้เห็นกลไกการรับข้อมูลจากฟอร์มที่ทำงานในลักษณะแบบ Client-Server ซึ่งฟอร์มทำหน้าที่เป็นส่วนติดต่อผู้ใช้และภาษา PHP ทำหน้าที่ประมวลผลข้อมูลเป็นเว็บแอปพลิเคชันที่สามารถรับและจัดการกับข้อมูลกับผู้ใชได้

1.2 การรับข้อมูลด้วย $_GET และ $_POST

จากกลไกรับค่าจากฟอร์มที่กล่าวถึงในหัวข้อก่อนหน้า ภาษา PHP มีตัวแปรพิเศษที่เรียกว่า Superglobal Variables สำหรับรับข้อมูลจากฟอร์ม HTML โดยตัวแปรที่พบบ่อยในการใช้งานเพื่อรับส่งข้อมูลจากฟอร์มคือ $_GET และ $_POST ซึ่งจะถูกเลือกใช้ตามค่า method ที่ระบุไว้ในแท็ก <form> ของ HTML ตามที่ได้อธิบายไว้ในบทที่ 6

1.2.1 การรับข้อมูลด้วยตัวแปร $_GET

ตัวแปร $_GET เป็นอาร์เรย์แบบกำหนดชื่อดัชนีที่เก็บข้อมูลที่ส่งมาด้วยวิธี GET โดยข้อมูลจะปรากฏใน URL และสามารถเข้าถึงได้ผ่านชื่อ name ที่กำหนดไว้ในฟอร์ม การใช้งานตัวแปร $_GET มีรูปแบบและตัวอย่างดังนี้

รูปแบบ

$_GET['ค่าของแอตทริบิวต์ name ของช่องกรอกข้อมูล']

ตัวอย่างที่ 1

HTML

```
1  <form action="search.php" method="GET">
2      <input type="text" name="keyword" placeholder="ค้นหาสินค้า">
3      <input type="submit" value="ค้นหา">
4  </form>
```

PHP: search.php

```
1  <?php
2      if (isset($_GET['keyword'])) {
3          $keyword = $_GET['keyword'];
4          echo "ผลการค้นหาสำหรับ: " . $keyword;
```

```
5       } else {
6           echo "กรุณาระบุคำค้นหา";
7       }
8   ?>
```

จากตัวอย่างที่ 1 แสดงการรับข้อมูลด้วยตัวแปร $_GET โดยในส่วน HTML บรรทัดที่ 1 กำหนด method="GET" ในแท็ก <form> และบรรทัดที่ 2 สร้างช่องกรอกข้อมูลด้วยชื่อ name="keyword"

ในส่วนของไฟล์ PHP (search.php) คำสั่งในบรรทัดที่ 2 ใช้ฟังก์ชัน isset() เพื่อตรวจสอบว่ามีการส่งค่าพารามิเตอร์ชื่อ keyword มายังเอกสารนี้หรือไม่ โดยฟังก์ชัน isset() จะคืนค่าเป็น true เมื่อพบว่ามีการกำหนดค่าจริงให้กับตัวแปรหรือคีย์นั้น และค่าดังกล่าวไม่เป็น null กล่าวคือ ในกรณีนี้จะตรวจสอบว่ามีการกำหนดค่าสำหรับคีย์ชื่อ keyword ภายในตัวแปร $_GET หรือไม่ หากเงื่อนไขเป็นจริง คำสั่งในบรรทัดที่ 3 จะทำการดึงค่าจาก $_GET['keyword'] มาเก็บไว้ในตัวแปร $keyword เพื่อใช้ในการประมวลผลในบรรทัดที่ 4 โดยจะใช้คำสั่ง echo เพื่อแสดงผลข้อความที่ต่อรวมกับค่าที่ผู้ใช้กรอกเข้ามา

เมื่อผู้ใช้กรอกข้อความ เช่น “โทรศัพท์” ลงในช่องค้นหา แล้วกดปุ่มส่งฟอร์ม เว็บเบราว์เซอร์จะสร้างคำขอในรูปแบบ URL เช่น search.php?keyword=โทรศัพท์ ซึ่งค่าพารามิเตอร์จะถูกแนบไปกับ URL และเมื่อ PHP ประมวลผลเสร็จจะส่งผลลัพธ์กลับไปแสดงข้อความว่า “ผลการค้นหาสำหรับ: โทรศัพท์” บนหน้าเว็บ

1.2.2 การรับข้อมูลด้วยตัวแปร $_POST

ตัวแปร $_POST เป็นอาร์เรย์แบบกำหนดชื่อดัชนีที่เก็บข้อมูลที่ส่งมาด้วยวิธี POST โดยข้อมูลจะไม่ปรากฏใน URL ทำให้เหมาะสำหรับข้อมูลที่เป็นความลับหรือข้อมูลจำนวนมาก การใช้งานตัวแปร $_POST มีรูปแบบและตัวอย่างการใช้ดังนี้

รูปแบบ

$_POST['ค่าของแอตทริบิวต์ name ของช่องกรอกข้อมูล']

ตัวอย่างที่ 2

HTML

```
1   <form action="login.php" method="POST">
2       <input type="text" name="username" placeholder="ชื่อผู้ใช้">
3       <input type="password" name="password" placeholder="รหัสผ่าน">
4       <input type="submit" value="เข้าสู่ระบบ">
5   </form>
```

PHP: login.php

```
1  <?php
2      if (isset($_POST['username']) && isset($_POST['password'])) {
3          $username = $_POST['username'];
4          $password = $_POST['password'];
5          echo "ชื่อผู้ใช้: " . $username . "<br>";
6          echo "รหัสผ่าน: " . str_repeat('*', strlen($password));
7      } else {
8          echo "กรุณากรอกข้อมูลให้ครบถ้วน";
9      }
10 ?>
```

จากตัวอย่างที่ 2 แสดงการรับข้อมูลจากฟอร์มด้วยตัวแปร $_POST โดยในส่วน HTML บรรทัดที่ 1 กำหนด method="POST" ภายในแท็ก <form> เพื่อระบุว่าเมื่อผู้ใช้ส่งข้อมูล ข้อมูลจะถูกส่งผ่านคำขอแบบ POST ซึ่งไม่มีการแนบข้อมูลไว้ใน URL ทำให้มีความปลอดภัยมากขึ้น

บรรทัดที่ 2 และ 3 ของ HTML เป็นช่องกรอกข้อมูลสำหรับชื่อผู้ใช้และรหัสผ่าน โดยกำหนดแอตทริบิวต์ name="username" และ name="password" ตามลำดับ เพื่อใช้เป็นคีย์ในการรับค่าจากฟอร์ม

ในส่วนของไฟล์ PHP (login.php) บรรทัดที่ 2 ใช้ฟังก์ชัน isset() เพื่อตรวจสอบว่ามีการส่งค่าพารามิเตอร์ชื่อ username และ password มาจากฟอร์มหรือไม่ หากมี บรรทัดที่ 3 และ 4 จะนำค่าดังกล่าวมาเก็บไว้ในตัวแปร $username และ $password ตามลำดับ

บรรทัดที่ 5 และ 6 ของไฟล์ PHP ใช้คำสั่ง echo เพื่อแสดงชื่อผู้ใช้และรหัสผ่าน โดยรหัสผ่านจะถูกแทนที่ด้วยสัญลักษณ์ * ตามความยาวของข้อความเพื่อความปลอดภัยเบื้องต้น

ข้อดีของการใช้ POST คือข้อมูลที่ส่งจะไม่ปรากฏใน URL ทำให้ปลอดภัยจากการถูกคัดลอกหรือถูกบันทึกไว้ในประวัติเบราว์เซอร์ และยังรองรับการส่งข้อมูลจำนวนมากหรือข้อมูลที่มีความซับซ้อนได้ดีกว่า GET เช่น ข้อความยาวหรือไฟล์ที่แนบจากผู้ใช้ เป็นต้น

1.3 การตรวจสอบและการล้างข้อมูล

การตรวจสอบและการล้างข้อมูล (Input Validation and Sanitization) เป็นขั้นตอนสำคัญที่นักพัฒนาเว็บแอปพลิเคชันต้องให้ความสำคัญเป็นอย่างมาก เนื่องจากข้อมูลที่ผู้ใช้ส่งมาผ่านฟอร์มอาจมีความเสี่ยงต่อความปลอดภัยของระบบ หรืออาจไม่ตรงตามรูปแบบที่ต้องการ การตรวจสอบและล้างข้อมูลจึงช่วยป้องกันการโจมตีและให้แน่ใจว่าข้อมูลที่ได้รับมีคุณภาพเป็นไปตามแนวทางด้านคุณภาพของข้อมูลในบทที่ 2 และความตระหนักถึงปลอดภัยของระบบสารสนเทศ

1.3.1 การตรวจสอบข้อมูล

การตรวจสอบข้อมูล (Validation) หมายถึง กระบวนการตรวจสอบความถูกต้อง ความสมบูรณ์และความสอดคล้องของข้อมูลที่ผู้ใช้งานป้อนผ่านฟอร์มก่อนที่จะนำไปประมวลผลหรือจัดเก็บ การตรวจสอบนี้มีเป้าหมายเพื่อให้แน่ใจว่าข้อมูลที่ได้รับมีคุณภาพเพียงพอและสอดคล้องกับเงื่อนไขที่ระบบกำหนด เช่น รูปแบบของอีเมล ความยาวของรหัสผ่าน หรือการกำหนดให้กรอกข้อมูลในช่องที่จำเป็น (Required Fields) เป็นต้น ในเชิงการเขียนโปรแกรมด้วยภาษา PHP การตรวจสอบข้อมูลมักใช้ โครงสร้างควบคุม (Control Structures) โดยเฉพาะคำสั่ง if...else หรือ switch ในการทดสอบเงื่อนไขเพื่อให้แน่ใจว่าข้อมูลตรงตามข้อกำหนดของระบบก่อนนำไปใช้ต่อ ฟังก์ชันตรวจสอบข้อมูลที่สำคัญในภาษา PHP มีดังนี้

1) ฟังก์ชัน empty()

ฟังก์ชัน empty() ใช้สำหรับตรวจสอบว่าตัวแปรมีค่าว่างเปล่าหรือไม่ โดยจะคืนค่า true หากค่าของตัวแปรตรงตามเงื่อนไขใดเงื่อนไขหนึ่งต่อไปนี้ ได้แก่ ไม่ถูกกำหนดค่า, มีค่าเป็น 0 (ทั้งจำนวนเต็มและสตริง "0"), เป็นสตริงว่าง (""), เป็น null, เป็น false, หรือเป็นอาร์เรย์ว่าง จึงเหมาะสำหรับใช้ตรวจสอบค่าที่ต้องมีการกรอกข้อมูลก่อนนำไปใช้งานต่อในระบบ

รูปแบบ	
empty(ตัวแปร)	
ตัวอย่างที่ 3	
1	$name = $_POST['name'];
2	if (empty($name)) {
3	echo "กรุณากรอกชื่อ";
4	}

จากตัวอย่างที่ 3 แสดงการตรวจสอบค่าของตัวแปร $name ซึ่งรับค่ามาจากฟอร์มผ่านเมธอด POST หากผู้ใช้ไม่กรอกชื่อหรือกรอกเป็นค่าว่าง เช่น สตริงว่าง ("") หรือ null ฟังก์ชัน empty() จะคืนค่า true ทำให้โปรแกรมแสดงข้อความว่า “กรุณากรอกชื่อ” เพื่อให้ผู้ใช้กรอกข้อมูลให้ครบถ้วนก่อนดำเนินการขั้นต่อไป

2) ฟังก์ชัน isset()

ฟังก์ชัน isset() ใช้สำหรับตรวจสอบว่าตัวแปรหรือองค์ประกอบของอาร์เรย์ได้รับการกำหนดค่าแล้วและไม่เป็น null หรือไม่ การใช้ isset() จึงมีบทบาทสำคัญในการตรวจสอบว่าข้อมูลที่ส่งจากฟอร์มมีการส่งชื่อเขตข้อมูล (ค่าของแอตทริบิวต์ name) มายังสคริปต์หรือไม่ ก่อนที่จะทำการเข้าถึงหรือใช้งานข้อมูลเหล่านั้น เพื่อป้องกันข้อผิดพลาดในการประมวลผล เช่น “undefined index”

รูปแบบ

empty(ตัวแปร)

ตัวอย่างที่ 4

```
1  if (isset($_POST['email'])) {
2          echo "มีการส่งอีเมลมาจากฟอร์ม";
3  }
```

จากตัวอย่างที่ 4 แสดงการตรวจสอบว่ามีการส่งข้อมูลชื่อ email มาจากฟอร์มหรือไม่ โดยใช้ isset($_POST['email']) ซึ่งจะให้ค่า true หากฟอร์มที่ส่งมามีการกำหนดช่องกรอกข้อมูลชื่อ email และผู้ใช้กรอกข้อมูลในช่องดังกล่าว เมื่อเงื่อนไขเป็นจริง โปรแกรมจะดำเนินการแสดงข้อความว่า “มีการส่งอีเมลมาจากฟอร์ม”

3) ฟังก์ชัน strlen()

ฟังก์ชัน strlen() ใช้สำหรับนับจำนวนอักขระในสตริงที่กำหนด ซึ่งมีประโยชน์อย่างยิ่งในการตรวจสอบความยาวของข้อมูลที่ผู้ใช้กรอกเข้ามา เช่น ชื่อผู้ใช้ รหัสผ่าน หรือข้อความ เพื่อให้เป็นไปตามข้อกำหนดของระบบ เช่น ต้องมีความยาวไม่น้อยกว่า 6 ตัวอักษร และไม่เกิน 50 ตัวอักษร เป็นต้น โดยค่าที่คืนจากฟังก์ชันนี้จะเป็นจำนวนเต็มแสดงจำนวนอักขระที่อยู่ในสตริงนั้น

รูปแบบ

strlen(ข้อความหรือตัวแปร)

ตัวอย่างที่ 5

```
1  $password = $_POST['password'];
2  if (strlen($password) < 6) {
3          echo "รหัสผ่านต้องมีอย่างน้อย 6 ตัวอักษร";
4  }
```

จากตัวอย่างที่ 5 กำหนดค่าให้ตัวแปร $password โดยนำค่าจากช่องกรอกข้อมูลของผู้ใช้ในฟอร์ม จากนั้นนำค่าที่ได้มาตรวจสอบด้วยฟังก์ชัน strlen() ว่ามีความยาวน้อยกว่า 6 ตัวอักษรหรือไม่ หากมีความยาวไม่ถึง ระบบจะแจ้งข้อความว่า “รหัสผ่านต้องมีอย่างน้อย 6 ตัวอักษร” เพื่อให้ผู้ใช้ปรับปรุงข้อมูลให้เป็นไปตามเงื่อนไขที่ระบบกำหนด

4) ฟังก์ชัน filter_var()

ฟังก์ชัน filter_var() เป็นฟังก์ชันสำเร็จรูปในภาษา PHP ที่ใช้สำหรับ ตรวจสอบหรือแปลงค่าของตัวแปรตามประเภทข้อมูลที่กำหนดไว้ล่วงหน้า โดยเฉพาะในกรณีที่ต้องการตรวจสอบรูปแบบข้อมูล เช่น

อีเมล หมายเลขจำนวนเต็ม ตัวเลขทศนิยม หรือ URL เป็นต้น โดยจะคืนค่าของตัวแปรเดิมหากตรวจสอบผ่าน หรือคืนค่า false หากไม่ผ่านการตรวจสอบ

รูปแบบ

filter_var(ค่าที่ต้องการตรวจสอบ, ประเภทตัวกรอง)

ประเภทตัวกรอง

FILTER_VALIDATE_EMAIL - ใช้ตรวจสอบว่าเป็นรูปแบบอีเมลที่ถูกต้องหรือไม่

FILTER_VALIDATE_INT - ใช้ตรวจสอบว่าเป็นจำนวนเต็มหรือไม่

FILTER_VALIDATE_FLOAT - ใช้ตรวจสอบว่าเป็นจำนวนจริงหรือไม่

FILTER_VALIDATE_URL - ใช้ตรวจสอบว่าเป็น URL ที่ถูกต้องหรือไม่

ตัวอย่างที่ 6

```
1  $email = $_POST['email'];
2  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
3          echo "รูปแบบอีเมลไม่ถูกต้อง";
4  }
```

จากตัวอย่างที่ 6 กำหนดค่าให้ตัวแปร $email โดยนำค่าจากช่องกรอกข้อมูลของผู้ใช้แล้วตรวจสอบค่าที่ได้รับด้วย filter_var() พร้อมใช้ตัวกรอง FILTER_VALIDATE_EMAIL หากรูปแบบไม่ตรงตามมาตรฐานของอีเมล ฟังก์ชันจะคืนค่า false และแสดงข้อความแจ้งเตือนให้ผู้ใช้กรอกอีเมลใหม่ให้ถูกต้อง

1.3.2 การล้างข้อมูล

การล้างข้อมูล (Data Sanitization) คือ กระบวนการทำความสะอาดหรือจัดรูปแบบข้อมูลที่ได้รับจากผู้ใช้ เพื่อกำจัดอักขระ ข้อมูล หรือโครงสร้างที่อาจเป็นอันตรายหรือไม่พึงประสงค์ออกไป โดยไม่ทำการปฏิเสธข้อมูลทั้งหมด กระบวนการนี้มีจุดมุ่งหมายเพื่อป้องกันความเสี่ยงจากการโจมตีระบบ เช่น การแทรกสคริปต์อันตราย (Cross-Site Scripting: XSS) หรือการบิดเบือนโครงสร้างคำสั่งในระบบที่ทำงานกับฐานข้อมูล นอกจากนี้การล้างข้อมูลยังช่วยให้ข้อมูลที่ได้รับอยู่ในรูปแบบที่เหมาะสมสำหรับการประมวลผลและแสดงผลในเว็บแอปพลิเคชัน ฟังก์ชันการล้างข้อมูลที่สำคัญในภาษา PHP มีดังนี้

1) ฟังก์ชัน trim()

ฟังก์ชัน trim() ในภาษา PHP ใช้สำหรับลบช่องว่าง (Whitespace Characters) ที่อยู่บริเวณ ต้น และท้ายของข้อความออกโดยไม่กระทบต่อข้อความส่วนกลาง ซึ่งรวมถึงช่องว่างปกติ (space) แท็บ (\t) การขึ้นบรรทัดใหม่ (\n) และอักขระพิเศษอื่น ๆ ที่ไม่จำเป็น การใช้ trim() เหมาะอย่างยิ่งสำหรับการทำความสะอาดค่าที่ได้รับจากผู้ใช้ก่อนนำไปจัดเก็บหรือประมวลผล เพราะผู้ใช้อาจป้อนอักขระเหล่านี้โดยไม่ตั้งใจ ซึ่งอาจส่งผลต่อความถูกต้องของข้อมูล เช่น การตรวจสอบรหัสผ่าน ชื่อผู้ใช้ หรืออีเมล เป็นต้น

รูปแบบ

trim(ข้อความหรือตัวแปร)

ตัวอย่างที่ 7

```
1  $name = "   Web App   ";
2  $cleanName = trim($name);
3  echo "ชื่อหลังล้างข้อมูล: " . $cleanName;
```

จากตัวอย่างที่ 26 ตัวแปร $name ได้รับค่าที่มีช่องว่างอยู่ก่อนและหลังข้อความ *Web App* ซึ่งอาจเกิดจากการป้อนข้อมูลผ่านฟอร์มในลักษณะที่ผู้ใช้กดเว้นวรรคโดยไม่ตั้งใจ เมื่อนำ $name เข้าสู่ฟังก์ชัน trim() ระบบจะลบช่องว่างที่ไม่จำเป็นออก ทำให้ผลลัพธ์ของตัวแปร $cleanName เป็นข้อความที่สะอาดและพร้อมใช้งานต่อไปในระบบ เช่น การเปรียบเทียบกับข้อมูลในฐานข้อมูลหรือการแสดงผลในหน้าเว็บ เป็นต้น หากต้องการลบช่องว่างเฉพาะด้านซ้ายหรือด้านขวาสามารถใช้ฟังก์ชัน ltrim() และ rtrim() ตามลำดับ ซึ่งทั้งสองฟังก์ชันมีการใช้งานเช่นเดียวกับฟังก์ชัน trim()

2) ฟังก์ชัน stripslashes()

ฟังก์ชัน stripslashes() เป็นฟังก์ชันที่ใช้สำหรับ ลบเครื่องหมาย backslash (\) ที่อยู่หน้าอักขระพิเศษ เช่น เครื่องหมายคำพูดเดี่ยว ('), เครื่องหมายคำพูดคู่ ("), หรือเครื่องหมายอื่น ๆ ที่อาจถูกใส่โดยอัตโนมัติเพื่อหลีกเลี่ยงการเกิดข้อผิดพลาดระหว่างการประมวลผลข้อความ ตัวอย่างเช่น "การพัฒนา \"Web App\"" จะถูกแปลงเป็น "การพัฒนา "Web App"" หลังจากใช้ stripslashes() ซึ่งช่วยให้ข้อความกลับมาอยู่ในรูปแบบที่เหมาะสมต่อการแสดงผลหรือการประมวลผลในขั้นตอนต่อไป

รูปแบบ

stripslashes(ข้อความหรือตัวแปร)

ตัวอย่างที่ 8

```
1  $input = "ชื่อของฉันคือ \"Nok\" และอายุ 25 ปี";
2  $clean = stripslashes($input);
3  echo "ก่อนล้าง: " . $input . "<br>";
4  echo "หลังล้าง: " . $clean;
```

จากตัวอย่างที่ 8 ตัวแปร $input เก็บข้อความที่มีเครื่องหมายคำพูดคู่ (") ซึ่งถูกเพิ่ม backslash เข้ามาโดยอัตโนมัติเพื่อให้สามารถใช้งานในข้อความได้อย่างถูกต้อง ฟังก์ชัน stripslashes() จะลบ backslash ออก ทำให้ได้ข้อความที่พร้อมสำหรับการแสดงผลในหน้าเว็บหรือการบันทึกข้อมูลในรูปแบบที่อ่านง่าย โดยไม่ทำให้รูปแบบของข้อความเพี้ยนจากที่ผู้ใช้งานตั้งใจ

3) ฟังก์ชัน htmlspecialchars()

ฟังก์ชัน htmlspecialchars() ใช้สำหรับแปลงอักขระพิเศษในข้อความ เช่น <, >, &, ", และ ' ให้เป็นรหัสแทนที่เรียกว่า HTML Entities เช่น <, >, &, ", และ ' ตามลำดับ เพื่อป้องกันการแสดงผลที่ผิดพลาดในเอกสาร HTML และเพื่อป้องกันช่องโหว่ด้านความปลอดภัย เช่น การโจมตีแบบ Cross-Site Scripting (XSS) เป็นต้น

การใช้ htmlspecialchars() ถือเป็นแนวทางสำคัญในการล้างข้อมูล (Sanitization) โดยเฉพาะเมื่อมีการนำข้อมูลจากผู้ใช้ไปแสดงในหน้าเว็บโดยตรง เช่น ข้อความความคิดเห็น ชื่อผู้ใช้งาน หรือข้อมูลจากฟอร์ม เพื่อไม่ให้เบราว์เซอร์ตีความข้อความเหล่านั้นเป็นคำสั่ง HTML หรือ JavaScript

รูปแบบ

htmlspecialchars(ข้อความ [, flags [, encoding [, double_encode]]])

พารามิเตอร์

ข้อความ คือข้อความต้นฉบับที่ต้องการแปลง

flags (ค่าปกติคือ ENT_QUOTES) กำหนดว่าอักขระใดจะถูกแปลง

encoding (ค่าปกติคือ UTF-8) ระบุชุดรหัสอักขระ

double_encode (ค่าปกติคือ true) หากตั้งเป็น false จะไม่แปลง HTML entities ที่ถูกแปลงไว้แล้ว

ตัวอย่างที่ 9

```
1  $comment = "<script>alert('XSS');</script>";
2  $safeComment = htmlspecialchars($comment);
3  echo "ความคิดเห็น: " . $safeComment;
```

จากตัวอย่างที่ 9 ตัวแปร $comment ได้รับข้อความที่ประกอบด้วยแท็ก <script> ซึ่งอาจเป็นอันตรายหากนำไปแสดงผลโดยไม่แปลงค่า ฟังก์ชัน htmlspecialchars() จะแปลงอักขระพิเศษเหล่านี้ เช่น < เป็น < และ > เป็น > ทำให้ข้อความไม่ถูกตีความเป็นคำสั่ง JavaScript โดยเว็บเบราว์เซอร์ แต่จะถูกแสดงเป็นข้อความธรรมดาซึ่งช่วยป้องกันการโจมตีแบบ XSS ได้

4) ฟังก์ชัน filter_var()

ฟังก์ชัน filter_var() ในภาษา PHP สามารถใช้งานได้ทั้งในกระบวนการตรวจสอบความถูกต้องของข้อมูล (Validation) และการล้างข้อมูล (Sanitization) โดยการล้างข้อมูลจะใช้ร่วมกับกลุ่มตัวกรอง FILTER_SANITIZE_* เพื่อกำจัดอักขระที่ไม่ปลอดภัยหรือไม่ต้องการออกจากข้อมูลที่ผู้ใช้ป้อนเข้ามา เช่น การลบแท็ก HTML, การลบสัญลักษณ์พิเศษ หรือการจัดรูปแบบข้อความให้อยู่ในรูปที่ปลอดภัยยิ่งขึ้นก่อนนำไปแสดงผลหรือจัดเก็บในระบบ

รูปแบบ

filter_var(ค่าที่ต้องการตรวจสอบ, ประเภทตัวกรอง)

ประเภทตัวกรอง

FILTER_SANITIZE_EMAIL - ใช้ลบอักขระที่ไม่เหมาะสมกับอีเมล

FILTER_SANITIZE_URL - ใช้ใช้ลบอักขระที่ไม่ปลอดภัยใน URL

FILTER_SANITIZE_NUMBER_INT - ใช้ลบอักขระที่ไม่ใช่ตัวเลขออกจากข้อความ

FILTER_SANITIZE_SPECIAL_CHARS - ใช้แปลงอักขระพิเศษให้เป็น HTML Entities

ตัวอย่างที่ 10

```
1  $email = "web<script>@example.com";
2  $url = "https://webapp.com/<script>";
3  $email = "web<script>@example.com";
4  $cleanEmail = filter_var($email, FILTER_SANITIZE_EMAIL);
5  $cleanUrl = filter_var($url, FILTER_SANITIZE_URL);
6  $cleanComment = filter_var($comment, FILTER_SANITIZE_SPECIAL_CHARS);
7  echo "อีเมลที่ล้างแล้ว: $cleanEmail<br>";
8  echo "URL ที่ล้างแล้ว: $cleanUrl<br>";
9  echo "ความคิดเห็นที่ล้างแล้ว: $cleanComment<br>";
```

จากตัวอย่างที่ 10 ตัวแปร $comment ได้รับข้อความที่ประกอบด้วยแท็ก <script> ซึ่งอาจเป็นอันตรายหากนำไปแสดงผลโดยไม่แปลงค่า ฟังก์ชัน filter_var() ร่วมกับ FILTER_SANITIZE_SPECIAL_CHARS จะแปลงอักขระพิเศษเหล่านี้ เช่น < ให้เป็น < และ > ให้เป็น > ทำให้ข้อความไม่ถูกตีความเป็นคำสั่ง JavaScript โดยเว็บเบราว์เซอร์ แต่จะถูกแสดงเป็นข้อความธรรมดา ซึ่งช่วยป้องกันการโจมตีแบบ XSS ได้อย่างมีประสิทธิภาพ ส่วนตัวแปร $email และ $url ถูกล้างให้เหลือเพียงอักขระที่ปลอดภัยตามประเภทของข้อมูลนั้นๆ

2. การทำงานร่วมกับฐานข้อมูล

การทำงานร่วมกับฐานข้อมูลเป็นหนึ่งในความสามารถที่สำคัญที่สุดของภาษา PHP ในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากเว็บแอปพลิเคชันส่วนใหญ่ต้องมีการจัดเก็บ เรียกใช้ และจัดการข้อมูลที่ได้จากผู้ใช้หรือระบบอื่นๆ ซึ่งฐานข้อมูลทำหน้าที่เป็นที่เก็บข้อมูลแบบถาวรที่สามารถเข้าถึงและจัดการได้อย่างมีประสิทธิภาพ

ภาษา PHP มีความสามารถในการทำงานร่วมกับระบบจัดการฐานข้อมูลหลายประเภท เช่น MySQL, PostgreSQL, SQLite, Microsoft SQL Server และ Oracle เป็นต้น โดยใช้กลไกการเชื่อมต่อที่เรียกว่า Database Extension ซึ่งทำหน้าที่เป็นตัวกลางระหว่างภาษา PHP กับระบบจัดการฐานข้อมูล ในบริบทของการ

พัฒนาเว็บแอปพลิเคชัน MySQL เป็นระบบจัดการฐานข้อมูลที่ได้รับความนิยมสูงเนื่องจากเป็นซอฟต์แวร์ฟรี มีประสิทธิภาพสูง และมีการสนับสนุนจากชุมชนนักพัฒนาอย่างต่อเนื่องดังที่ได้นำเสนอในบทที่ 4 การใช้เทคโนโลยีคอนเทนเนอร์ในการสร้างสภาพแวดล้อมการพัฒนาที่รวมการติดตั้ง MySQL เข้าไว้ด้วย

2.1 ขั้นตอนการทำงานร่วมกับฐานข้อมูล

ก่อนที่จะเข้าสู่รายละเอียดเชิงปฏิบัติในการเชื่อมต่อกับฐานข้อมูล สิ่งสำคัญคือการเข้าใจขั้นตอนโดยรวมของการทำงานร่วมกับฐานข้อมูลในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากการทำงานกับฐานข้อมูลไม่ได้เริ่มต้นจากการเขียนรหัสคำสั่ง PHP เพียงอย่างเดียว แต่ต้องมีการเตรียมการในหลายขั้นตอนเพื่อให้ได้ผลลัพธ์ที่ถูกต้อง การพัฒนาเว็บแอปพลิเคชันที่ทำงานร่วมกับฐานข้อมูลโดยทั่วไปจะมีขั้นตอนหลักดังนี้

ขั้นตอนที่ 1) การวิเคราะห์และออกแบบฐานข้อมูล

การวิเคราะห์และออกแบบฐานข้อมูลเป็นขั้นตอนในการศึกษาความต้องการของระบบ วิเคราะห์ความสัมพันธ์ของข้อมูล และออกแบบโครงสร้างฐานข้อมูลให้เหมาะสมกับการใช้งาน ซึ่งเป็นส่วนหนึ่งของกระบวนการพัฒนาระบบสารสนเทศที่ได้กล่าวถึงในบทที่ 2 ผลลัพธ์ที่ได้จากขั้นตอนนี้คือแผนผังความสัมพันธ์ของข้อมูล (Entity Relationship Diagram) และข้อกำหนดรายละเอียดของตารางที่จะใช้ในฐานข้อมูล

ขั้นตอนที่ 2) การสร้างฐานข้อมูลและตาราง

การสร้างฐานข้อมูลและตารางเป็นการนำผลการวิเคราะห์และออกแบบจากขั้นตอนแรกมาสร้างเป็นฐานข้อมูลจริงในระบบจัดการฐานข้อมูล ในกรณีของสภาพแวดล้อม LAMP Stack ที่นำเสนอในบทที่ 4 สามารถใช้เว็บแอปพลิเคชัน phpMyAdmin ในการสร้างฐานข้อมูลและตารางได้อย่างสะดวก

ขั้นตอนที่ 3) การเตรียมข้อมูลทดสอบ

การเตรียมข้อมูลทดสอบเป็นการใส่ข้อมูลตัวอย่างลงในตารางที่สร้างขึ้น เพื่อใช้ในการทดสอบการทำงานของเว็บแอปพลิเคชัน ข้อมูลทดสอบควรครอบคลุมสถานการณ์ที่อาจเกิดขึ้นในการใช้งานจริง

ขั้นตอนที่ 4) การเขียนรหัสคำสั่ง

การเขียนรหัสคำสั่งเป็นขั้นตอนของการเขียนรหัสคำสั่งภาษา PHP เพื่อเชื่อมต่อและจัดการข้อมูล และสร้างส่วนการทำงานเว็บแอปพลิเคชันให้มีความสามารถในการสร้างหรือบันทึกข้อมูล อ่าน แก้ไขและลบข้อมูล ที่เรียกว่าระบบ CRUD (Create-Read-Update-Delete)

ขั้นตอนที่ 5) การทดสอบและปรับปรุง

การทดสอบและปรับปรุงเป็นการทดสอบการทำงานของเว็บแอปพลิเคชันร่วมกับฐานข้อมูล และปรับปรุงทั้งโครงสร้างฐานข้อมูลและรหัสคำสั่ง PHP ให้มีประสิทธิภาพมากขึ้น

2.2 ประเภทของ Database Extension ใน PHP

PHP มี Database Extension สำหรับการทำงานกับ MySQL อยู่ 3 ประเภทหลัก ซึ่งแต่ละประเภทมีข้อดีและข้อจำกัดที่แตกต่างกัน การเลือกใช้ Extension ที่เหมาะสมจึงมีผลต่อประสิทธิภาพ ความปลอดภัย และความยืดหยุ่นของเว็บแอปพลิเคชัน

2.2.1 MySQL Extension

MySQL Extension เป็น Database Extension แรกที่ถูกพัฒนาขึ้นสำหรับ PHP และได้รับความนิยมในช่วงแรกของการพัฒนาเว็บด้วย PHP อย่างไรก็ตาม Extension นี้ได้ถูกยกเลิกการสนับสนุน (Deprecated) ตั้งแต่ PHP เวอร์ชัน 5.5.0 และถูกลบออกโดยสมบูรณ์ใน PHP เวอร์ชัน 7.0.0 เนื่องจากมีข้อจำกัดด้านความปลอดภัยและไม่สนับสนุนคุณสมบัติใหม่ๆ ของ MySQL การใช้ MySQL Extension ในปัจจุบันจึงไม่เหมาะสมและไม่ปลอดภัย

2.2.2 MySQLi Extension

MySQLi Extension (MySQL Improved) เป็น Database Extension ที่พัฒนาขึ้นเพื่อแทนที่ MySQL Extension โดยมีการปรับปรุงด้านประสิทธิภาพและความปลอดภัย รวมทั้งเพิ่มการสนับสนุนคุณสมบัติใหม่ของ MySQL เช่น Prepared Statements, Enhanced Debugging และ Multiple Statements เป็นต้น MySQLi Extension สนับสนุนการเขียนโปรแกรมทั้งแบบ Procedural และ Object-Oriented ทำให้มีความยืดหยุ่นในการใช้งาน อย่างไรก็ตาม MySQLi Extension ถูกออกแบบมาเพื่อใช้งานกับ MySQL เท่านั้น ไม่สามารถใช้กับระบบจัดการฐานข้อมูลอื่นได้

2.2.3 PDO Extension

PDO (PHP Data Objects) Extension เป็น Database Extension ที่ให้การเข้าถึงฐานข้อมูลแบบ Object-Oriented และรองรับระบบจัดการฐานข้อมูลหลายประเภท ไม่จำกัดเฉพาะ MySQL เท่านั้น PDO ใช้หลักการของ Database Abstraction Layer ซึ่งช่วยให้การเปลี่ยนแปลงระบบจัดการฐานข้อมูลทำได้ง่ายขึ้น โดยไม่ต้องเปลี่ยนแปลงโครงสร้างรหัสคำสั่งมากนัก นอกจากนี้ PDO ยังมีระบบความปลอดภัยที่แข็งแกร่ง โดยเฉพาะการสนับสนุน Prepared Statements ที่ช่วยป้องกัน SQL Injection ได้

จาก Database Extension ที่กล่าวถึงข้างต้น ในหนังสือเล่มนี้ได้เลือกใช้ PDO Extension ในการนำเสนอภายใต้หัวข้อการทำงานร่วมกับฐานข้อมูล โดยคำนึงถึงความเหมาะสมในการสาธิตกระบวนการเชื่อมต่อและจัดการข้อมูลผ่านภาษา PHP ทั้งนี้ เนื่องจาก PDO มีรูปแบบการใช้งานที่ชัดเจน สนับสนุนการทำงานร่วมกับระบบฐานข้อมูลได้หลากหลาย และมีโครงสร้างที่เอื้อต่อการเรียนรู้ในลักษณะเป็นระบบ อีกทั้งยังรองรับแนวทางป้องกันปัญหาด้านความปลอดภัย เช่น SQL Injection ได้อย่างเหมาะสม ซึ่งสอดคล้องกับเป้าหมายของหนังสือเล่มนี้ที่มุ่งเน้นการให้ความเข้าใจพื้นฐานและแนวทางปฏิบัติที่ดีในการพัฒนาเว็บแอปพลิเคชัน

2.3 การเตรียมความพร้อมก่อนการทำงานร่วมกับฐานข้อมูล

ก่อนที่จะเริ่มเขียนรหัสคำสั่งภาษา PHP เพื่อทำงานร่วมกับฐานข้อมูล ผู้อ่านควรจะเตรียมความพร้อมก่อนดำเนินการดังนี้

2.3.1 การตรวจสอบสภาพแวดล้อมการพัฒนา

ในการพัฒนาเว็บแอปพลิเคชันที่มีส่วนการทำงานร่วมกับระบบฐานข้อมูล นอกเหนือจากมีการติดตั้งกลไกการให้บริการเว็บแล้ว ยังต้องมีการติดตั้งระบบจัดการฐานข้อมูลและเครื่องมืออำนวยความสะดวกใน

การจัดการฐานข้อมูลผ่านเว็บแอปพลิเคชันอย่าง phpMyAdmin ทั้งนี้สภาพแวดล้อมดังกล่าวสามารถจัดเตรียมได้หลายวิธี ดังนี้

วิธีที่ 1) การติดตั้งด้วย XAMPP

การติดตั้งด้วย XAMPP ตามที่ได้นำเสนอในบทที่ 3 เหมาะสำหรับผู้ที่ต้องการความสะดวกในการติดตั้งและใช้งาน โดยสามารถเข้าถึง phpMyAdmin ได้จาก http://localhost/phpmyadmin

วิธีที่ 2) การติดตั้งด้วย Docker

การติดตั้งด้วย Docker ตามที่ได้นำเสนอในบทที่ 4 จะให้ความยืดหยุ่นมากกว่าและเป็นแนวทางที่ใช้ในอุตสาหกรรมซอฟต์แวร์ปัจจุบัน โดยสามารถเข้าถึง phpMyAdmin ได้จาก http://localhost:88

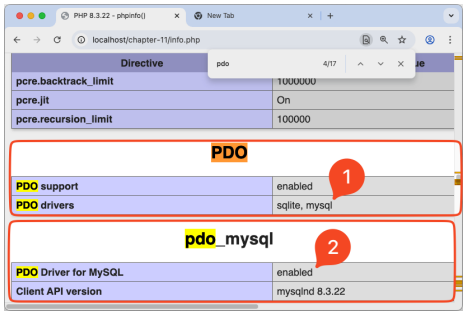
หากผู้อ่านยังไม่ได้ดำเนินการติดตั้งสภาพแวดล้อมการพัฒนา สามารถศึกษาเพิ่มเติมจากบทที่ 3 (สำหรับ XAMPP) หรือบทที่ 4 (สำหรับ Docker) ได้ ในบทนี้จะอ้างอิงจากการติดตั้งด้วย Docker ตามที่นำเสนอในบทที่ 4

2.3.2 การตรวจสอบการสนับสนุน PDO Extension

การตรวจสอบการสนับสนุน PDO Extension ของตัวแปลภาษา PHP สามารถทำได้หลายวิธี แต่วิธีที่ง่ายและรวดเร็วคือการใช้ฟังก์ชัน phpinfo() ซึ่งเป็นฟังก์ชันพร้อมใช้งานในการแสดงข้อมูลการกำหนดค่าทั้งหมดของ PHP รวมถึงส่วนเสริมที่ติดตั้งไว้ ในสภาพแวดล้อม LAMP Stack ที่สร้างขึ้นด้วย Docker ตามที่นำเสนอในบทที่ 4 โดยสร้างไฟล์ PHP ชื่อ info.php ไว้ในตำแหน่ง Document Root (c:\webapp\source\) พร้อมด้วยประโยคคำสั่งต่อไปนี้

ไฟล์	c:\webapp\source\info.php (หรือตำแหน่ง Document Root)
1	<?php
2	phpinfo();

เมื่อเรียกใช้ไฟล์นี้ผ่าน http://localhost/info.php จะปรากฏหน้าข้อมูลการกำหนดค่า PHP ให้ทำการค้นหา PDO (ใช้ฟังก์ชันการค้นหาในเว็บเบราว์เซอร์)

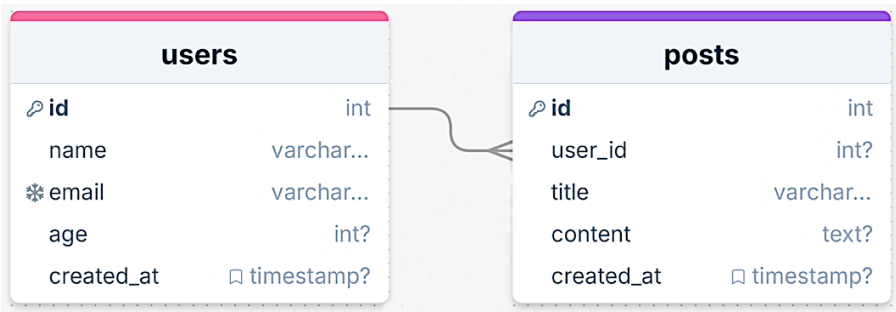


ภาพที่ 12.2 การตรวจสอบสถานะของการติดตั้ง PDO Extension สำหรับฐานข้อมูล MySQL

จากภาพที่ 12.2 หากสภาพแวดล้อมการพัฒนามีการติดตั้ง PDO Extension สำหรับการเชื่อมต่อกับฐานข้อมูล MySQL จะพบข้อมูลในส่วน PDO drivers ที่มีชื่อ mysql ดังแสดงในตำแหน่งที่ 1 ของภาพ และข้อมูล PDO Driver for MySQL มีค่าเป็น enabled ซึ่งหมายถึงเปิดใช้งานอยู่ หากทำการตรวจสอบแล้วไม่พบข้อมูลดังกล่าว สามารถทบทวนการติดตั้งในบทที่ 3 และ 4 อีกครั้ง หรือติดตามจากเว็บไซต์อย่างเป็นทางการของ XAMPP หรือ Docker Hub ตามวิธีที่เลือกใช้สร้างสภาพแวดล้อม

2.3.3 จัดเตรียมฐานข้อมูล

หลังจากที่ได้ตรวจสอบความพร้อมก่อนการทำงานร่วมกับฐานข้อมูลแล้ว ขั้นตอนต่อไปคือการสร้างฐานข้อมูล โดยใช้เว็บแอปพลิเคชัน phpMyadmin ตามที่ได้กล่าวถึงก่อนหน้า โดยฐานข้อมูลที่ใช้เป็นตัวอย่างการเรียนรู้นี้เป็นฐานข้อมูลตัวอย่างในการจัดการบทความอย่างง่ายชื่อ blog_system ที่ประกอบด้วยตารางผู้ใช้ (users) และตารางบทความ (posts) โดยทั้งสองตารางมีความสัมพันธ์กันในลักษณะ 1-to-many กล่าวคือ ผู้ใช้หนึ่งคนสามารถสร้างบทความได้หลายบทความ แต่ในบทความหนึ่งๆ จะมีเจ้าของบทความได้เพียงคนเดียว ความสัมพันธ์และรายการเขตข้อมูลของตารางแสดงดังภาพที่ 12.3



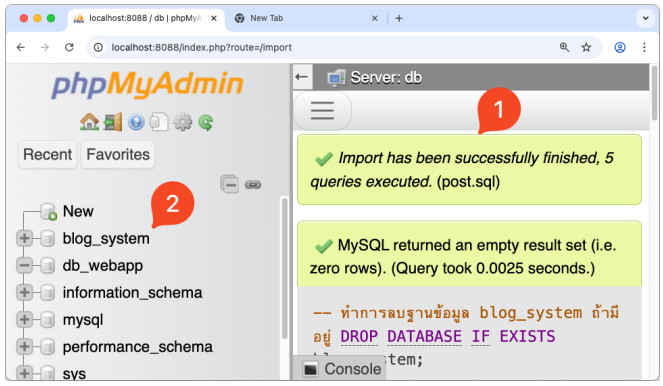
ภาพที่ 12.3 ตัวอย่างแผนภาพ ER-Diagram ของฐานข้อมูล blog_system

เพื่อสร้างฐานข้อมูลที่มีรายละเอียดข้างต้น ให้สร้างไฟล์ blogsystem.sql พร้อมรหัสคำสั่งต่อไป หลังจากนั้นให้นำเข้าไฟล์ดังกล่าวในเว็บแอปพลิเคชัน phpMyAdmin โดยหากนำเข้าและสร้างฐานข้อมูลได้สำเร็จจะแสดงข้อความผลการนำเข้าและชื่อฐานข้อมูลดังภาพที่ 12.4 หากมีข้อความแสดงข้อผิดพลาดหรือการแจ้งเตือน รวมทั้งไม่พบชื่อฐานข้อมูลในตำแหน่งดังภาพ ให้ตรวจสอบไฟล์ blogsystem.sql อีกครั้ง

ไฟล์ SQL c:\webapp\source\sql\blogsystem.sql

```
1  -- ทำการลบฐานข้อมูล blog_system ถ้ามีอยู่
2  DROP DATABASE IF EXISTS blog_system;
3  -- สร้างฐานข้อมูลและใช้ฐานข้อมูลนั้น
4  CREATE DATABASE blog_system CHARACTER SET utf8 COLLATE utf8_general_ci;
5  USE blog_system;
6  -- ตัวอย่างการสร้างตารางผู้ใช้
```

```
7   CREATE TABLE users (
8       id INT AUTO_INCREMENT PRIMARY KEY,
9       name VARCHAR(100) NOT NULL,
10      email VARCHAR(150) UNIQUE NOT NULL,
11      age INT,
12      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
13  );
14  -- ตัวอย่างการสร้างตารางโพสต์
15  CREATE TABLE posts (
16      id INT AUTO_INCREMENT PRIMARY KEY,
17      user_id INT,
18      title VARCHAR(200) NOT NULL,
19      content TEXT,
20      created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
21      FOREIGN KEY (user_id) REFERENCES users(id)
22  );
```



ภาพที่ 12.4 การแสดงสถานการณ์นำเข้าไฟล์ blogsystem.sql และสร้างฐานข้อมูลได้สำเร็จ

2.4 คำสั่งและฟังก์ชันสำคัญของภาษา PHP สำหรับการติดต่อกับฐานข้อมูลด้วย PDO

การทำงานกับฐานข้อมูลผ่าน PDO (PHP Data Objects) เป็นแนวทางที่ได้รับความนิยม เนื่องจากสามารถเชื่อมต่อกับระบบฐานข้อมูลได้หลายประเภทและสนับสนุนการใช้งาน Prepared Statements เพื่อป้องกันปัญหา SQL Injection ได้อย่างมีประสิทธิภาพ ในหัวข้อนี้จะนำเสนอฟังก์ชันหลักที่จำเป็นสำหรับการติดต่อกับฐานข้อมูลผ่าน PDO ได้แก่ การสร้างการเชื่อมต่อ การเตรียมคำสั่ง SQL และการดำเนินการคำสั่ง SQL เพื่อเพิ่ม แก้ไข ค้นหา และลบข้อมูล

2.4.1 การสร้าง PDO Object

การสร้าง PDO Object เป็นขั้นตอนสำคัญในการเริ่มต้นการติดต่อกับฐานข้อมูลผ่านภาษา PHP โดยใช้ Constructor ของคลาส PDO ซึ่งทำหน้าที่สร้างออบเจกต์ที่ใช้เป็นตัวกลางระหว่าง PHP และระบบจัดการฐานข้อมูล การสร้าง PDO Object จะคืนค่าเป็นออบเจกต์ที่สามารถใช้เรียกเมธอดต่าง ๆ เช่น prepare() และ query() ได้ หากไม่สามารถเชื่อมต่อได้จะเกิดข้อผิดพลาดแบบ Exception ซึ่งควรจัดการด้วยโครงสร้าง try-catch เพื่อความปลอดภัยของ รูปแบบและตัวอย่างของการสร้าง PDO Object มีดังนี้

รูปแบบ

```
$pdo = new PDO(DSN, USERNAME, PASSWORD);
```

จากรูปแบบและตัวอย่างของการสร้าง PDO Object มีพารามิเตอร์ที่สำคัญ 3 ค่า โดยแต่ละค่ามีรายละเอียดดังนี้

- DSN (Data Source Name) คือข้อความที่ใช้ระบุชนิดของระบบฐานข้อมูล (เช่น MySQL) ที่อยู่ของเซิร์ฟเวอร์ และชื่อฐานข้อมูลที่ต้องการเชื่อมต่อ โดยมีรูปแบบทั่วไปดังนี้

รูปแบบ DSN

```
"DBTYPE:host=HOSTNAME;dbname=DATABASE_NAME;charset=CHAR_ENCODE"
```

คำอธิบาย DSN

DBTYPE คือ ชนิดของฐานข้อมูลที่เชื่อมต่อ เช่น mysql ,sqlite และ pgsql เป็นต้น

HOSTNAME คือ ชื่อโฮสต์หรือ IP ของฐานข้อมูล กรณีที่อยู่เครื่องเดียวกับเครื่องผู้ให้บริการเว็บจะกำหนดเป็น localhost

DATABASE_NAME คือ ชื่อฐานข้อมูลที่ต้องการติดต่อ

CHAR_ENCODE คือรูปแบบชุดอักขระที่ใช้ในการแทนค่าในฐานข้อมูล เช่น utf8, utf8mb4

- USERNAME คือชื่อผู้ใช้ของระบบจัดการฐานข้อมูลที่มีสิทธิ์เข้าถึงฐานข้อมูลตามที่ระบุใน DSN
- PASSWORD คือรหัสผ่านของผู้ใช้ที่ระบุไว้ใน USERNAME

ตัวอย่างที่ 11

```
1  <?php
2      $pdo = new PDO("mysql:host=localhost;dbname=web_app_db;charset=utf8",
           "webapp_user", "webapp_user_pass");
3  ?>
```

จากตัวอย่างที่ 11 มีการสร้างตัวแปร $pdo ซึ่งเก็บอินสแตนซ์ของออบเจกต์ PDO สำหรับเชื่อมต่อกับฐานข้อมูลชื่อ web_app_db โดยกำหนดให้ใช้เซิร์ฟเวอร์ฐานข้อมูล MySQL ที่อยู่ในเครื่อง localhost และเข้ารหัสเป็น utf8 เพื่อรองรับภาษาไทยและตัวอักษรพิเศษ นอกจากนี้ยังระบุชื่อผู้ใช้และรหัสผ่านที่ตรงกับผู้ใช้งานที่มีสิทธิ์ในฐานข้อมูลนั้น หากไม่สามารถเชื่อมต่อได้ PHP จะส่งข้อผิดพลาดผ่าน Exception ซึ่งสามารถจัดการได้โดยการใช้โครงสร้าง try...catch ในหัวข้อถัดไป อนึ่ง ชื่อตัวแปร $pdo สามารถเปลี่ยนได้ แต่เพื่อให้สอดคล้องและต่อเนื่องในการอธิบายในฟังก์ชันอื่นๆ ซึ่งอาจต้องใช้ค่าที่ได้จากฟังก์ชันนี้ จึงกำหนดเป็น $pdo

2.4.2 การใช้โครงสร้าง try...catch

โครงสร้าง try...catch ใช้สำหรับจัดการข้อผิดพลาดที่อาจเกิดขึ้นระหว่างการทำงานของโปรแกรม โดยเฉพาะอย่างยิ่งเมื่อติดต่อกับฐานข้อมูล หากเกิดข้อผิดพลาด เช่น การเชื่อมต่อล้มเหลว คำสั่ง SQL ผิด หรือข้อมูลไม่ถูกต้อง PHP จะโยนข้อยกเว้น (Exception) ซึ่งหากไม่มีโครงสร้าง try...catch มารองรับ อาจทำให้โปรแกรมหยุดทำงานทันที โครงสร้างนี้ช่วยให้สามารถจัดการกับข้อผิดพลาดอย่างปลอดภัย แสดงข้อความที่เหมาะสม และป้องกันไม่ให้ข้อมูลภายในระบบรั่วไหลสู่ผู้ใช้

รูปแบบ

```
try {
    // คำสั่งที่อาจเกิดข้อผิดพลาด
} catch (Exception $e) {
    // การจัดการข้อผิดพลาด
}
```

ตัวอย่างที่ 12

```
1  try {
2      $pdo = new PDO("mysql:host=localhost;dbname=web_app_db;charset=utf8",
                "root", "rootpass");
3      $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
4      echo "เชื่อมต่อฐานข้อมูลสำเร็จ";
5  } catch (PDOException $e) {
6      echo "เกิดข้อผิดพลาด: " . $e->getMessage();
7  }
```

จากตัวอย่างที่ 12 การสร้าง PDO object ถูกวางไว้ในบล็อก try เพื่อจับข้อผิดพลาดที่อาจเกิดขึ้น เช่น ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง หากเกิดข้อผิดพลาด PHP จะโยน PDOException ซึ่งสามารถเข้าถึงได้ผ่านตัวแปร $e และใช้แสดงข้อความผิดพลาดโดยไม่หยุดการทำงานของโปรแกรมแบบกระทันหัน นอกจากนี้ยังสามารถบันทึกข้อผิดพลาดลง log ได้เพื่อการตรวจสอบในภายหลัง

2.4.3 ฟังก์ชัน prepare()

ฟังก์ชัน prepare() เป็นเมธอดของออบเจกต์ PDO ที่ใช้สำหรับเตรียมคำสั่ง SQL ก่อนนำไปดำเนินการจริงกับฐานข้อมูล โดยจะคืนค่าเป็นออบเจกต์ของคลาส PDOStatement ซึ่งสามารถใช้ร่วมกับฟังก์ชัน bindParam(), bindValue() และ execute() เพื่อดำเนินการกับข้อมูลได้อย่างปลอดภัยและมีประสิทธิภาพ การใช้ prepare() ช่วยลดความเสี่ยงจาก SQL Injection เนื่องจากค่าที่ผู้ใช้กรอกจะไม่ถูกรวมกับคำสั่ง SQL โดยตรง แต่จะถูกผูกไว้ผ่านตัวแปรแทนที่

รูปแบบ

```
$stmt = $pdo->prepare($sql);
```

จากรูปแบบของฟังก์ชัน prepare() ในส่วน $sql คือคำสั่ง SQL ที่ต้องการเตรียม อาจมี placeholder (?) หรือ named parameter (:name)

ตัวอย่างที่ 13

```
1  <?php
2      $sql = "INSERT INTO users (name, email) VALUES (:name, :email)";
3      $stmt = $pdo->prepare($sql);
4      $stmt->execute([
           ':name' => 'Nok',
           ':email' => 'nok@example.com'
5      ]);
6  ?>
```

จากตัวอย่างที่ 13 เป็นการใช้ฟังก์ชัน prepare() สำหรับเตรียมคำสั่ง SQL เพื่อเพิ่มข้อมูลผู้ใช้เข้าสู่ตาราง users โดยใช้ตัวแปรแทนที่ :name และ :email เพื่อความปลอดภัยและความยืดหยุ่นในการจัดการข้อมูล เมธอด prepare() จะคืนค่าเป็นออบเจกต์ $stmt จากนั้นเรียก execute() พร้อมส่งอาร์เรย์ที่ระบุค่าจริงของ :name และ :email ตามลำดับ ซึ่งจะทำให้ระบบสามารถแยกโครงสร้างคำสั่ง SQL ออกจากค่าที่ผู้ใช้ป้อนมา ลดความเสี่ยงจาก SQL Injection ได้อย่างมีประสิทธิภาพ อนึ่ง ชื่อตัวแปร $stmt สามารถเปลี่ยนได้ แต่เพื่อให้สอดคล้องและต่อเนื่องในการอธิบายในฟังก์ชันอื่นๆ ซึ่งอาจต้องใช้ค่าที่ได้จากฟังก์ชันนี้ จึงกำหนดเป็น $stmt

2.4.4 ฟังก์ชัน execute()

ฟังก์ชัน execute() เป็นเมธอดของออบเจกต์ PDOStatement ที่ใช้สำหรับดำเนินการคำสั่ง SQL ที่ได้เตรียมไว้ด้วย prepare() โดยสามารถส่งค่าที่จะถูกแทนที่ในตำแหน่งตัวแปร (เช่น :name, :email) ไป

พร้อมกับคำสั่งได้โดยตรง ฟังก์ชันนี้จะคืนค่า true หากคำสั่ง SQL ดำเนินการสำเร็จ และคืนค่า false หากเกิดข้อผิดพลาด เช่น ค่าซ้ำกับที่มีอยู่แล้วในฟิลด์แบบ UNIQUE

รูปแบบ

$pdoStatement->execute(อาร์เรย์ของค่าที่จะใช้แทนในคำสั่ง SQL);

ตัวอย่างที่ 14

```
1  $sql = "INSERT INTO users (name, email) VALUES (:name, :email)";
2  $stmt = $pdo->prepare($sql);
3  $result = $stmt->execute([
4          ':name' => 'Ann',
           ':email' => 'ann@example.com'
5  ]);
6  if ($result) {
7          echo "เพิ่มข้อมูลเรียบร้อยแล้ว";
8  } else {
9          echo "เกิดข้อผิดพลาดในการเพิ่มข้อมูล";
10 }
```

จากตัวอย่างที่ 14 เป็นการใช้ฟังก์ชัน execute() เพื่อเพิ่มข้อมูลลงในตาราง users โดยใช้ค่าจากอาร์เรย์ที่ระบุชื่อแทน (:name, :email) และแทนที่ด้วยข้อมูลของผู้ใช้ หากการดำเนินการสำเร็จจะได้รับข้อความยืนยันว่าข้อมูลถูกเพิ่มแล้ว ในทางกลับกันหากไม่สำเร็จ เช่น อีเมลซ้ำ จะมีข้อความแจ้งข้อผิดพลาด ช่วยให้ผู้ใช้หรือผู้พัฒนาเข้าใจผลลัพธ์ของคำสั่งได้ชัดเจน อนึ่ง ชื่อตัวแปร $stmt สามารถเปลี่ยนได้ แต่เพื่อให้สอดคล้องและต่อเนื่องในการอธิบายในฟังก์ชันอื่นๆ ซึ่งอาจต้องใช้ค่าที่ได้จากฟังก์ชันนี้ จึงกำหนดเป็น $pdoStatement

2.4.5 ฟังก์ชัน fetch()

ฟังก์ชัน fetch() ใช้สำหรับดึงแถวข้อมูลจากผลลัพธ์ของคำสั่ง SQL ที่ได้จากการดำเนินการกับคำสั่ง SELECT โดยเป็นเมธอดของออบเจกต์ PDOStatement ที่จะคืนค่าข้อมูลเพียงหนึ่งแถวต่อการเรียกใช้หนึ่งครั้ง หากไม่มีแถวข้อมูลเหลืออยู่ จะคืนค่า false ซึ่งสามารถนำไปใช้ร่วมกับโครงสร้างควบคุมเพื่อดึงข้อมูลทีละแถวจากชุดข้อมูลทั้งหมดได้

รูปแบบ

$pdoStatement->fetch(โหมดการคืนค่า);

โหมดการคืนค่า

PDO::FETCH_ASSOC — คืนค่าเป็นอาร์เรย์แบบกำหนดชื่อดัชนี (key เป็นชื่อคอลัมน์)

PDO::FETCH_NUM — คืนค่าเป็นอาร์เรย์แบบดัชนีตัวเลข

PDO::FETCH_BOTH — (ค่าเริ่มต้น) คืนค่าแบบทั้งชื่อคอลัมน์และตัวเลข

ตัวอย่างที่ 15

```
1  $sql = "SELECT * FROM users WHERE id = :id";
2  $stmt = $pdo->prepare($sql);
3  $stmt->execute([':id' => 1]);
4  $user = $stmt->fetch(PDO::FETCH_ASSOC);
5  if ($user) {
6          echo "ชื่อ: " . $user['name'] . "<br>";
7          echo "อีเมล: " . $user['email'];
8  } else {
9          echo "ไม่พบข้อมูลผู้ใช้";
10 }
```

จากตัวอย่างที่ 15 ฟังก์ชัน fetch() ถูกใช้เพื่อดึงข้อมูลของผู้ใช้ที่มี id เท่ากับ 1 โดยระบุให้คืนค่าเป็นอาร์เรย์ที่สามารถเข้าถึงได้ด้วยชื่อคอลัมน์ เช่น 'name' และ 'email' หากพบข้อมูลจะแสดงชื่อและอีเมล แต่ถ้าไม่พบจะมีข้อความแจ้งว่าไม่พบข้อมูล

2.4.6 ฟังก์ชัน fetchAll()

ฟังก์ชัน fetchAll() ใช้สำหรับดึงแถวข้อมูลทั้งหมดที่ได้จากคำสั่ง SQL ในครั้งเดียว โดยคืนค่าเป็นอาร์เรย์ที่มีหลายแถว ซึ่งแต่ละแถวจะอยู่ในรูปแบบอาร์เรย์ย่อย สามารถกำหนดโหมดการคืนค่าได้เช่นเดียวกับฟังก์ชัน fetch() เช่น PDO::FETCH_ASSOC เพื่อให้แต่ละแถวเป็นอาร์เรย์ที่มีคีย์เป็นชื่อคอลัมน์ การใช้ fetchAll() เหมาะกับสถานการณ์ที่จำนวนข้อมูลไม่มากนัก และต้องการโหลดข้อมูลทั้งหมดไว้ในตัวแปรก่อนนำไปประมวลผลต่อ

รูปแบบ

$pdoStatement->fetch(โหมดการคืนค่า);

ตัวอย่างที่ 16

```
1  $sql = "SELECT * FROM users";
2  $stmt = $pdo->prepare($sql);
3  $stmt->execute();
4  $users = $stmt->fetchAll(PDO::FETCH_ASSOC);
5  foreach ($users as $user) {
6          echo "ชื่อ: " . $user['name'] . "<br>";
```

```
7       echo "อีเมล: " . $user['email'] . "<hr>";
8   }
```

จากตัวอย่างที่ 16 ฟังก์ชัน fetchAll() ถูกใช้ดึงข้อมูลผู้ใช้ทั้งหมดจากฐานข้อมูลในครั้งเดียว โดยระบุให้คืนค่าทุกแถวเป็นอาร์เรย์แบบกำหนดชื่อดัชนี จากนั้นใช้คำสั่ง foreach เพื่อแสดงชื่อและอีเมลของผู้ใช้แต่ละราย

2.4.7 ฟังก์ชัน rowCount()

ฟังก์ชัน rowCount() ใช้สำหรับตรวจสอบจำนวนแถวที่ได้รับผลกระทบจากการดำเนินการ SQL เช่น การแทรก (INSERT), การปรับปรุง (UPDATE), การลบ (DELETE) หรือแม้แต่การค้นหา (SELECT) ในบางระบบฐานข้อมูล โดยฟังก์ชันนี้จะคืนค่าเป็นจำนวนเต็มที่ระบุจำนวนแถวที่ได้รับผลกระทบจากคำสั่ง SQL ที่ถูกดำเนินการผ่าน PDOStatement ซึ่งเป็นประโยชน์อย่างยิ่งในการตรวจสอบผลลัพธ์ของการดำเนินการว่ามีผลหรือไม่

รูปแบบ

```
$pdoStatement->rowCount();
```

ตัวอย่างที่ 17

```
1   $sql = "DELETE FROM users WHERE email = :email";
2   $stmt = $pdo->prepare($sql);
3   $stmt->execute([':email' => 'example@example.com']);
4   if ($stmt->rowCount() > 0) {
5       echo "ลบข้อมูลผู้ใช้สำเร็จ";
6   } else {
7       echo "ไม่พบข้อมูลที่ต้องการลบ";
8   }
```

จากตัวอย่างที่ 17 ฟังก์ชัน rowCount() ถูกใช้เพื่อตรวจสอบว่าการดำเนินการลบมีผลหรือไม่ โดยจะคืนค่ามากกว่า 0 หากมีแถวใดถูกลบออกจากฐานข้อมูล และจะคืนค่าเป็น 0 หากไม่พบข้อมูลที่ตรงตามเงื่อนไข ซึ่งช่วยให้สามารถแจ้งผลต่อผู้ใช้ได้อย่างถูกต้อง

3. ระบบ CRUD

การทำงานร่วมกับฐานข้อมูลในเว็บแอปพลิเคชันส่วนใหญ่ มักเกี่ยวข้องกับการดำเนินการข้อมูลในลักษณะของ CRUD ซึ่งเป็นตัวย่อที่หมายถึงการดำเนินการ 4 ประเภท ได้แก่ Create (การสร้างข้อมูล) Read (การอ่านหรือแสดงข้อมูล) Update (การปรับปรุงข้อมูล)และ Delete (การลบข้อมูล) โดยกระบวนการ CRUD เหล่านี้ถือ

เป็นพื้นฐานของการจัดการข้อมูลมักปรากฏอยู่ในทุกระบบสารสนเทศ เช่น ระบบจัดการสมาชิก ระบบสินค้า ระบบจองห้องพัก หรือระบบประกาศข่าว

3.1 ภาพรวมกระบวนการ CRUD ในเว็บแอปพลิเคชัน

ในบริบทของเว็บแอปพลิเคชันที่พัฒนาโดยใช้ภาษา PHP การดำเนินการ CRUD จะใช้โครงสร้างรหัสคำสั่ง (Script) ที่เชื่อมโยงกับคำสั่ง SQL ผ่าน PHP Data Objects (PDO) ซึ่งเป็นกลไกกลางที่ใช้ติดต่อกับระบบจัดการฐานข้อมูล เช่น MySQL ตามที่ได้นำเสนอไว้ในหัวข้อก่อนหน้า โดยในแต่ละกระบวนการ CRUD จะมีลักษณะการดำเนินงานที่สามารถสรุปได้ดังนี้

3.1.1 Create (การสร้างข้อมูลใหม่)

การสร้างข้อมูลใหม่ใช้ในการเพิ่มข้อมูลลงในตารางฐานข้อมูล เช่น การลงทะเบียนสมาชิกใหม่ โดยทั่วไปจะประกอบด้วยแบบฟอร์ม HTML สำหรับรับค่าจากผู้ใช้ และคำสั่ง INSERT INTO เพื่อเพิ่มข้อมูลเข้าสู่ตารางฐานข้อมูล

3.1.2 Read (การอ่านหรือแสดงข้อมูล)

การอ่านหรือแสดงข้อมูลใช้ในการดึงข้อมูลจากฐานข้อมูลมาแสดงบนหน้าเว็บ เช่น การแสดงรายชื่อสมาชิกทั้งหมด หรือรายละเอียดของบทความใดบทความหนึ่ง โดยใช้คำสั่ง SELECT ควบคู่กับการแสดงผลด้วย PHP

3.1.3 Update (การปรับปรุงข้อมูล)

การปรับปรุงข้อมูลใช้ในการแก้ไขข้อมูลเดิมในฐานข้อมูล เช่น การแก้ไขชื่อผู้ใช้หรืออีเมล โดยใช้คำสั่ง UPDATE ซึ่งต้องมีการระบุข้อมูลเป้าหมายที่ต้องการแก้ไขผ่าน WHERE

3.1.4 Delete (การลบข้อมูล)

การลบข้อมูลใช้ในการลบข้อมูลออกจากฐานข้อมูล เช่น การลบบัญชีผู้ใช้หรือบทความที่ไม่ต้องการ โดยใช้คำสั่ง DELETE FROM พร้อมเงื่อนไขที่ระบุข้อมูลที่จะลบอย่างชัดเจน

การสร้างระบบ CRUD ที่มีความสมบูรณ์จึงต้องพิจารณาทั้งการสร้างฟอร์ม การตรวจสอบข้อมูลที่รับเข้ามา และการจัดการข้อผิดพลาดที่อาจเกิดขึ้นในแต่ละขั้นตอน โดยรหัสคำสั่งภาษา PHP ที่เกี่ยวข้องควรเขียนให้เป็นระบบ แยกหน้าที่ของแต่ละส่วนไว้อย่างชัดเจน เพื่อให้ระบบสามารถขยายและดูแลรักษาได้ในระยะยาว

3.2 วงจรกระบวนการของ CRUD

การดำเนินการ CRUD ในเว็บแอปพลิเคชันมีขั้นตอนการทำงานโดยเริ่มตั้งแต่ผู้ใช้โต้ตอบกับส่วนติดต่อผู้ใช้จนกระทั่งข้อมูลถูกจัดเก็บหรือดึงมาจากฐานข้อมูล ดังนี้

ขั้นตอนที่ 1) การโต้ตอบของผู้ใช้กับส่วนติดต่อผู้ใช้

ผู้ใช้เข้ามาในเว็บแอปพลิเคชันและโต้ตอบกับส่วนติดต่อผู้ใช้ที่สร้างด้วยภาษา HTML เช่น การกรอกข้อมูลในฟอร์ม การคลิกปุ่มหรือการเลือกเมนู ข้อมูลที่ผู้ใช้กรอกจะถูกตรวจสอบเบื้องต้นด้วยภาษา JavaScript หากมีการใช้งาน

ขั้นตอนที่ 2) การส่งข้อมูลไปยังฝั่งผู้ให้บริการ

เมื่อผู้ใช้กดปุ่มส่งข้อมูล ข้อมูลจะถูกส่งไปยังไฟล์ PHP ที่ระบุในแอตทริบิวต์ action ของฟอร์ม โดยใช้วิธีการส่งข้อมูลที่กำหนดในแอตทริบิวต์ method ซึ่งอาจเป็น GET หรือ POST

ขั้นตอนที่ 3) การรับและตรวจสอบข้อมูลในฝั่งผู้ให้บริการ

ไฟล์ PHP จะรับข้อมูลจากผู้ใช้ผ่านตัวแปร $_GET หรือ $_POST จากนั้นจะมีการตรวจสอบและล้างข้อมูลโดยใช้ฟังก์ชันต่าง ๆ ที่ได้กล่าวถึงในหัวข้อ 1.3 เพื่อให้แน่ใจว่าข้อมูลมีคุณภาพและปลอดภัย

ขั้นตอนที่ 4) การเชื่อมต่อกับฐานข้อมูล

ไฟล์ PHP จะสร้างการเชื่อมต่อกับฐานข้อมูลโดยใช้ PDO พร้อมกับข้อมูลการเชื่อมต่อที่กำหนดไว้ หากการเชื่อมต่อล้มเหลว ระบบจะแสดงข้อความข้อผิดพลาด

ขั้นตอนที่ 5) การดำเนินการกับฐานข้อมูล

การดำเนินการกับฐานข้อมูลจะขึ้นอยู่กับประเภทของการดำเนินการ CRUD ไฟล์ PHP จะสร้างคำสั่ง SQL ที่เหมาะสม และใช้ Prepared Statement ในการส่งคำสั่งไปยังฐานข้อมูล เพื่อป้องกัน SQL Injection

ขั้นตอนที่ 6) การประมวลผลผลลัพธ์

ฐานข้อมูลจะส่งผลลัพธ์กลับมายังไฟล์ PHP ซึ่งอาจเป็นข้อมูลที่ดึงมา จำนวนแถวที่ได้รับผลกระทบ หรือข้อความข้อผิดพลาด

ขั้นตอนที่ 7) การแสดงผลลัพธ์

ไฟล์ PHP จะประมวลผลลัพธ์และสร้างเอกสาร HTML เพื่อแสดงผลให้ผู้ใช้เห็น เช่น ข้อความแจ้งว่าบันทึกข้อมูลสำเร็จ รายการข้อมูลที่ดึงมา หรือข้อความข้อผิดพลาด

3.3 โครงสร้างไฟล์และองค์ประกอบของระบบ CRUD

การสร้างระบบ CRUD ที่มีประสิทธิภาพต้องมีการจัดโครงสร้างไฟล์ที่เป็นระบบและแยกหน้าที่การทำงานของแต่ละไฟล์ให้ชัดเจน เพื่อให้ง่ายต่อการพัฒนา บำรุงรักษา และแก้ไขปัญหา การออกแบบโครงสร้างที่ดีจะช่วยให้ผู้พัฒนาสามารถเข้าใจและปรับปรุงระบบได้อย่างมีประสิทธิภาพ

3.3.1 ไฟล์หลักที่จำเป็นในระบบ CRUD มีดังนี้

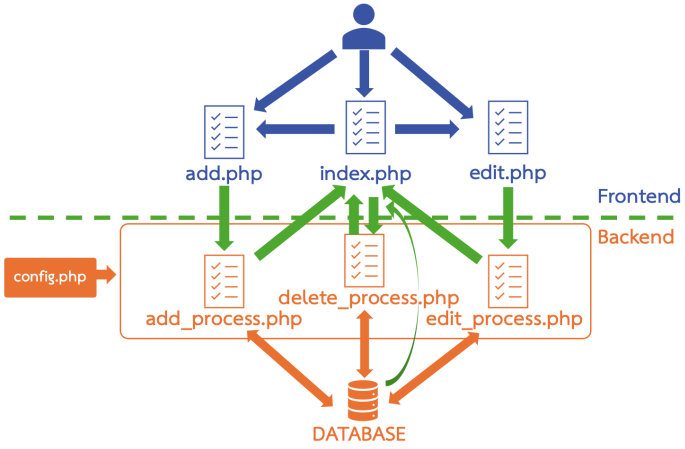
ระบบ CRUD เบื้องต้นจะประกอบด้วยไฟล์หลักที่จำเป็น 7 ไฟล์ ซึ่งแต่ละไฟล์จะมีหน้าที่และบทบาทในระบบ CRUD ที่เฉพาะเจาะจง ดังนี้

- config.php - ไฟล์การตั้งค่าการเชื่อมต่อฐานข้อมูล มีหน้าที่หลักคือเก็บข้อมูลการเชื่อมต่อกับฐานข้อมูล เช่น ชื่อเซิร์ฟเวอร์ ชื่อฐานข้อมูล ชื่อผู้ใช้ และรหัสผ่าน พร้อมทั้งสร้างการเชื่อมต่อ PDO Object ที่สามารถใช้งานร่วมกับไฟล์อื่นๆ ได้
- index.php - หน้าแสดงรายการข้อมูลทั้งหมด (Read) มีหน้าที่หลักคือแสดงรายการข้อมูลทั้งหมดในรูปแบบตาราง พร้อมลิงก์สำหรับแก้ไข และลบข้อมูลแต่ละรายการ นอกจากนี้ยังมีลิงก์สำหรับเพิ่มข้อมูลใหม่ บทบาทหลักในระบบ CRUD คือการอ่าน (Read) ข้อมูลจากฐานข้อมูล

- add.php - หน้าฟอร์มสำหรับเพิ่มข้อมูลใหม่ (Create) หน้าที่หลักคือแสดงฟอร์มสำหรับให้ผู้ใช้กรอกข้อมูลใหม่ ฟอร์มจะส่งข้อมูลไปยังไฟล์ add_process.php เมื่อผู้ใช้กดปุ่มบันทึก บทบาทหลักในระบบ CRUD คือการสร้าง (Create) ข้อมูลใหม่
- add_process.php - ไฟล์ประมวลผลการเพิ่มข้อมูล (Create) มีหน้าที่หลักคือรับข้อมูลจากฟอร์ม add.php ตรวจสอบความถูกต้อง และบันทึกลงฐานข้อมูล หลังจากบันทึกเสร็จจะเปลี่ยนเส้นทางไปยังหน้า index.php บทบาทหลักในระบบ CRUD คือการสร้าง (Create) ข้อมูลใหม่ลงในฐานข้อมูล
- edit.php - หน้าฟอร์มสำหรับแก้ไขข้อมูล (Update) มีหน้าที่หลักคือแสดงฟอร์มที่มีข้อมูลเดิมบรรจุไว้แล้ว เพื่อให้ผู้ใช้สามารถแก้ไขได้ ฟอร์มจะส่งข้อมูลไปยังไฟล์ edit_process.php เมื่อผู้ใช้กดปุ่มบันทึก บทบาทหลักในระบบ CRUD คือการแก้ไข (Update) ข้อมูล
- edit_process.php - ไฟล์ประมวลผลการแก้ไขข้อมูล (Update) มีหน้าที่หลักคือรับข้อมูลจากฟอร์ม edit.php ตรวจสอบความถูกต้อง และอัปเดตข้อมูลในฐานข้อมูล หลังจากอัปเดตเสร็จจะเปลี่ยนเส้นทางไปยังหน้า index.php บทบาทหลักในระบบ CRUD คือการแก้ไข (Update) ข้อมูลในฐานข้อมูล
- delete_process.php - ไฟล์ประมวลผลการลบข้อมูล (Delete) มีหน้าที่หลักคือรับพารามิเตอร์ ID ของข้อมูลที่ต้องการลบผ่าน URL เมื่อผู้ใช้คลิกลิงก์ "ลบ" จากหน้า index.php จากนั้นจะลบข้อมูลออกจากฐานข้อมูลและเปลี่ยนเส้นทางกลับไปยังหน้า index.php ทันที โดยไม่แสดงหน้าใดๆ ให้ผู้ใช้เห็น บทบาทหลักในระบบ CRUD คือการลบ (Delete) ข้อมูลออกจากฐานข้อมูล

3.3.2 ความสัมพันธ์ระหว่างไฟล์

ไฟล์ในระบบ CRUD มีความเชื่อมโยงกันในรูปแบบที่เป็นระบบ โดยมีการอ้างอิงและเรียกใช้งานร่วมกัน เพื่อให้เข้าใจความสัมพันธ์และการทำงานร่วมกันของไฟล์ต่าง ๆ ได้อย่างชัดเจน ภาพที่ 12.5แสดงโครงสร้างไฟล์และการไหลของข้อมูลระหว่างส่วน Frontend, Backend และ Database



ภาพที่ 12.5 ความสัมพันธ์ระหว่างไฟล์และการไหลของข้อมูลในระบบ CRUD

จากภาพที่ 12.5 การนำทางระหว่างหน้าในระบบจะเริ่มจากหน้า index.php ซึ่งเป็นหน้าหลักที่แสดงรายการข้อมูลทั้งหมด ผู้ใช้สามารถคลิกไปยัง add.php เพื่อเพิ่มข้อมูลใหม่ คลิกไปยัง edit.php เพื่อแก้ไขข้อมูล หรือคลิกลิงก์ลบที่อยู่ในหน้า index.php เพื่อเรียกไฟล์ delete_process.php ในการลบข้อมูล

เมื่อผู้ใช้กรอกข้อมูลในฟอร์ม add.php และกดปุ่มบันทึก ข้อมูลจะถูกส่งไปประมวลผลที่ add_process.php และเปลี่ยนเส้นทางกลับมาที่ index.php เช่นเดียวกับการแก้ไขข้อมูลที่จะผ่าน edit_process.php ก่อนกลับมาที่หน้าหลัก ส่วนการลบข้อมูลจะดำเนินการที่ delete_process.php และเปลี่ยนเส้นทางกลับไปที่ index.php ทันที

การใช้ไฟล์ config.php เป็นเทคนิคการเขียนรหัสคำสั่งที่นิยมใช้ในการพัฒนาเว็บแอปพลิเคชัน โดยนำคำสั่งการเชื่อมต่อกับฐานข้อมูลมาเขียนรวมไว้ในไฟล์เดียวแบบศูนย์กลาง ไฟล์ config.php จะถูกเรียกใช้โดยไฟล์อื่นๆ ทั้งหมดที่ต้องการเชื่อมต่อกับฐานข้อมูลด้วยคำสั่ง require_once 'config.php' เพื่อให้สามารถใช้ตัวแปร $pdo ในการดำเนินการกับฐานข้อมูล การใช้ไฟล์ config.php แยกต่างหากช่วยให้การจัดการข้อมูลการเชื่อมต่อฐานข้อมูลทำได้ง่าย และหากต้องการเปลี่ยนแปลงข้อมูลการเชื่อมต่อ เช่น ชื่อเซิร์ฟเวอร์ ชื่อฐานข้อมูล หรือรหัสผ่าน ก็สามารถแก้ไขได้ที่ไฟล์เดียวโดยไม่ต้องแก้ไขในทุกไฟล์ที่ใช้งาน

3.3.3 การออกแบบส่วนติดต่อผู้ใช้เบื้องต้น

ส่วนติดต่อผู้ใช้ของระบบ CRUD จะประกอบด้วยหน้าหลัก 3 หน้า ที่ผู้ใช้จะเห็นและโต้ตอบด้วย โดยแต่ละหน้าจะมีองค์ประกอบและการทำงานที่แตกต่างกันตามวัตถุประสงค์ของการใช้งาน

1) หน้าแสดงรายการ (index.php)

หน้าแสดงรายการ (index.php) เป็นหน้าหลักของระบบที่จะแสดงตารางข้อมูลทั้งหมดในฐานข้อมูล ผู้ใช้สามารถดูข้อมูลภาพรวมได้จากหน้านี้ พร้อมทั้งมีปุ่ม "เพิ่มข้อมูลใหม่" ที่ลิงก์ไปยัง add.php สำหรับเพิ่มข้อมูลใหม่ ในแต่ละแถวของตารางจะมีปุ่ม "แก้ไข" ที่ลิงก์ไปยัง edit.php พร้อมส่ง ID ของข้อมูลนั้น และปุ่ม "ลบ" ที่ลิงก์ไปยัง delete_process.php พร้อม ID เพื่อลบข้อมูลทันที

2) หน้าเพิ่มข้อมูล

หน้าเพิ่มข้อมูล (add.php) จะประกอบด้วยฟอร์มที่มีช่องกรอกข้อมูลตามฟิลด์ในฐานข้อมูล เพื่อให้ผู้ใช้สามารถกรอกข้อมูลใหม่ได้ ฟอร์มจะมีปุ่ม "บันทึก" สำหรับส่งข้อมูลไปยัง add_process.php เพื่อประมวลผลและบันทึกลงฐานข้อมูล นอกจากนี้ยังมีลิงก์ "ยกเลิก" เพื่อให้ผู้ใช้สามารถกลับไปยัง index.php ได้หากไม่ต้องการเพิ่มข้อมูล

3) หน้าแก้ไขข้อมูล

หน้าแก้ไขข้อมูล (edit.php) จะมีฟอร์มที่คล้ายกับหน้าเพิ่มข้อมูล แต่จะมีข้อมูลเดิมของรายการที่ต้องการแก้ไขแสดงอยู่ในช่องกรอกข้อมูลแล้ว ผู้ใช้สามารถแก้ไขข้อมูลตามต้องการและกดปุ่ม "บันทึก" เพื่อส่งข้อมูลไปยัง edit_process.php สำหรับอัปเดตข้อมูลในฐานข้อมูล เช่นเดียวกับหน้าเพิ่มข้อมูล หน้านี้ก็มีลิงก์ "ยกเลิก" เพื่อกลับไปยัง index.php โดยไม่บันทึกการเปลี่ยนแปลง

3.4 การสร้างการเชื่อมต่อฐานข้อมูล

จากการที่ได้ออกแบบโครงสร้างไฟล์และองค์ประกอบของระบบ CRUD ในหัวข้อก่อนหน้าไปแล้ว รวมทั้งได้กำหนดความสัมพันธ์ระหว่างไฟล์ และการออกแบบส่วนติดต่อผู้ใช้เบื้องต้น ในหัวข้อนี้จะเป็นการลงมือปฏิบัติจริงในการสร้างไฟล์ต่าง ๆ สำหรับระบบ CRUD ของการจัดการผู้ใช้ โดยเริ่มต้นจากการสร้างการเชื่อมต่อฐานข้อมูลซึ่งเป็นขั้นตอนแรกและสำคัญที่สุดในการทำงานร่วมกับฐานข้อมูล

การเชื่อมต่อที่ถูกต้องจะทำให้เว็บแอปพลิเคชันสามารถส่งคำสั่ง SQL ไปยังระบบจัดการฐานข้อมูลและรับผลลัพธ์กลับมาได้ ในส่วนนี้จะนำเสนอวิธีการสร้างการเชื่อมต่อฐานข้อมูลด้วย PDO Extension พร้อมทั้งรูปแบบการจัดการที่เหมาะสมสำหรับการพัฒนาเว็บแอปพลิเคชัน ตามแนวทางที่ได้วางแผนในการใช้ไฟล์ config.php เป็นศูนย์กลางการเชื่อมต่อ

3.4.1 การสร้างไฟล์ config.php

การสร้างไฟล์ config.php เป็นแนวทางปฏิบัติที่ดีในการพัฒนาเว็บแอปพลิเคชัน เนื่องจากช่วยให้การจัดการข้อมูลการเชื่อมต่อฐานข้อมูลเป็นไปอย่างเป็นระบบและง่ายต่อการดูแลรักษา ไฟล์นี้จะทำหน้าที่เป็นศูนย์กลางสำหรับข้อมูลการเชื่อมต่อ และสามารถถูกเรียกใช้โดยไฟล์อื่นๆ ในระบบ CRUD

ไฟล์ PHP **c:\webapp\source\config.php**

```php
<?php
    // ข้อมูลการเชื่อมต่อฐานข้อมูล
    $host = 'localhost';
    $dbname = 'blog_system';
    $username = 'root';
    $password = '';
    try {
        // สร้างการเชื่อมต่อ PDO
        $pdo = new PDO("mysql:host=$host;dbname=$dbname;charset=utf8",
            $username, $password);
        // กำหนดโหมดการแสดงข้อผิดพลาด
        $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        // กำหนดโหมดการดึงข้อมูล
        $pdo->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
    } catch (PDOException $e) {
        echo "การเชื่อมต่อฐานข้อมูลล้มเหลว: " . $e->getMessage();
        exit();
```

```
17      }
18  ?>
```

คำอธิบาย

- บรรทัดที่ 9 มีการสร้างวัตถุ PDO โดยกำหนดค่า DSN เป็น "mysql:host=$host;dbname=$dbname;charset=utf8" ซึ่งระบุชนิดฐานข้อมูลเป็น MySQL ที่อยู่ในเครื่องผู้ให้บริการเดียวกันนี้ กำหนดชื่อฐานข้อมูลคือ blog_system และกำหนดรูปแบบชุดอักขระเป็น UTF-8 เพื่อป้องกันปัญหาอักขระผิดเพี้ยน โดยเมื่อสร้าง PDO สำเร็จ จะสามารถใช้ตัวแปร $pdo ในการดำเนินการคำสั่ง SQL ต่อไป
- บรรทัดที่ 11 ใช้เมธอด $pdo->setAttribute(); เพื่อกำหนดให้ PDO แสดงข้อผิดพลาดในรูปแบบ Exception ซึ่งช่วยให้สามารถจัดการข้อผิดพลาดได้ผ่านโครงสร้าง try–catch อย่างเหมาะสม
- บรรทัดที่ 12 ใช้ $pdo->setAttribute(); เพื่อกำหนดให้ค่าที่ดึงจากฐานข้อมูลเมื่อใช้ฟังก์ชัน fetch() หรือ fetchAll() จะคืนมาในรูปแบบของอาร์เรย์แบบกำหนดชื่อคอลัมน์ (Associative Array) โดยไม่รวมค่าดัชนีตัวเลข ซึ่งช่วยให้รหัสคำสั่งอ่านง่ายและเข้าใจตรงตามชื่อฟิลด์ในฐานข้อมูล
- ในกรณีที่การเชื่อมต่อล้มเหลว เช่น ฐานข้อมูลไม่มีอยู่จริง หรือชื่อผู้ใช้และรหัสผ่านไม่ถูกต้อง บรรทัดในส่วน catch จะถูกเรียกใช้งาน (บรรทัดที่ 14-17) โดยแสดงข้อความข้อผิดพลาดด้วย $e->getMessage() และหยุดการทำงานของโปรแกรมทันทีด้วย exit(); เพื่อป้องกันไม่ให้โปรแกรมดำเนินการต่อโดยไม่มีการเชื่อมต่อที่ถูกต้องการทดสอบการเชื่อมต่อฐานข้อมูล

ข้อผิดพลาดที่อาจพบได้

- SQLSTATE[1049] Unknown database หมายถึง ฐานข้อมูล blog_system ยังไม่ถูกสร้าง แนวทางแก้ไขให้ตรวจสอบการสร้างฐานข้อมูลใน phpMyAdmin
- SQLSTATE[HY000] [1045] Access denied หมายถึง ชื่อผู้ใช้หรือรหัสผ่านไม่ถูกต้อง แนวทางแก้ไข: ตรวจสอบข้อมูลในบรรทัดที่ 4-5
- Connection refused หมายถึง เซิร์ฟเวอร์ MySQL ไม่ทำงาน แนวทางแก้ไขให้ตรวจสอบสถานะ Docker containers หรือ XAMPP

3.4.2 การทดสอบการเชื่อมต่อฐานข้อมูล

หลังจากสร้างไฟล์ config.php แล้ว ควรทำการทดสอบการเชื่อมต่อเพื่อให้แน่ใจว่าการกำหนดค่าต่าง ๆ ถูกต้อง การทดสอบสามารถทำได้โดยการสร้างไฟล์สำหรับการทดสอบการเชื่อมต่อ ดังนี้

ไฟล์ PHP **c:\webapp\source\test_connection.php**

```
1   <?php
2       require_once 'config.php';
```

```
3       echo "<h2>ทดสอบการเชื่อมต่อฐานข้อมูล</h2>";
4       try {
5           // ทดสอบการเรียกดูข้อมูลจากตาราง users
6           $stmt = $pdo->query("SELECT COUNT(*) as total FROM users");
7           $result = $stmt->fetch();
8           echo "<p>เชื่อมต่อฐานข้อมูลสำเร็จ</p>";
9           echo "<p>จำนวนผู้ใช้ในระบบ: " . $result['total'] . " คน</p>";
10      } catch (PDOException $e) {
11          echo "<p>เกิดข้อผิดพลาด: " . $e->getMessage() . "</p>";
12      }
```

คำอธิบาย

- บรรทัดที่ 2 ใช้คำสั่ง require_once เพื่อโหลดไฟล์ config.php ซึ่งจะทำให้สามารถใช้ตัวแปร $pdo ที่ได้จากการเชื่อมต่อฐานข้อมูลในไฟล์นี้ได้ การใช้ require_once จะป้องกันการโหลดไฟล์เดียวกันซ้ำ และหยุดการทำงานทันทีหากไฟล์ไม่พบ
- บรรทัดที่ 6 ใช้เมธอด $pdo->query() เพื่อส่งคำสั่ง SQL "SELECT COUNT(*) as total FROM users" ไปยังฐานข้อมูล ซึ่งจะนับจำนวนแถวทั้งหมดในตาราง users และส่งผลลัพธ์กลับมาเป็น PDOStatement object
- บรรทัดที่ 7 ใช้เมธอด $stmt->fetch() เพื่อดึงผลลัพธ์จากคำสั่ง SQL มาเก็บในตัวแปร $result ซึ่งจะได้ข้อมูลในรูปแบบ associative array ตามที่กำหนดไว้ในไฟล์ config.php
- บรรทัดที่ 8-9 แสดงผลลัพธ์การทดสอบ โดยแสดงข้อความยืนยันการเชื่อมต่อและจำนวนผู้ใช้ในระบบ โดยเข้าถึงค่าผ่าน $result['total'] ซึ่งสอดคล้องกับชื่อ alias ที่กำหนดในคำสั่ง SQL
- บรรทัดที่ 10-12 ในส่วน catch จะทำงานเมื่อเกิดข้อผิดพลาดใดๆ ในการทดสอบ โดยแสดงข้อความข้อผิดพลาดที่เกิดขึ้นผ่าน $e->getMessage()

การทดสอบ

- เข้าเว็บไซต์ด้วย URL locahost/host/test_connection.php

ผลลัพธ์ที่คาดหวัง

- แสดงข้อความ "เชื่อมต่อฐานข้อมูลสำเร็จ"
- แสดงจำนวนผู้ใช้ในระบบ (อาจเป็น 0 คนหากยังไม่มีข้อมูล)

ข้อผิดพลาดที่อาจพบได้

- Fatal error: require_once(): Failed opening required หมายถึง ไฟล์ config.php ไม่อยู่ในโฟลเดอร์เดียวกัน แนวทางแก้ไขให้ตรวจสอบตำแหน่งไฟล์หรือปรับ path ในบรรทัดที่ 2

- Table 'blog_system.users' doesn't exist หมายถึง ตาราง users ยังไม่ถูกสร้างหรือไม่มีอยู่ในฐานข้อมูล แนวทางแก้ไขให้ทำการตรวจสอบการนำเข้าไฟล์ SQL ใน phpMyAdmin
- Call to a member function query() on null หมายถึง การเชื่อมต่อฐานข้อมูลล้มเหลวในไฟล์ config.php แนวทางแก้ไขให้ทำการตรวจสอบข้อผิดพลาดในไฟล์ config.php ก่อน
- Parse error: syntax error หมายถึง มีข้อผิดพลาดในการเขียน syntax ของ PHP แนวทางแก้ไขให้ทำการตรวจสอบเครื่องหมาย semicolon และวงเล็บปีกกาในทุกบรรทัด

3.5 การสร้างไฟล์สำหรับระบบ CRUD

หลังจากที่ได้เตรียมการเชื่อมต่อฐานข้อมูลเรียบร้อยแล้ว ขั้นตอนต่อไปคือการสร้างระบบ CRUD โดยแต่ละการดำเนินการจะมีไฟล์และหน้าที่เฉพาะตามที่ได้อธิบายไว้ในหัวข้อ 3.3 ในส่วนนี้จะนำเสนอการสร้างแต่ละส่วนของระบบ CRUD อย่างละเอียด พร้อมตัวอย่างรหัสคำสั่งที่สามารถนำไปใช้งานได้จริง

3.5.1 เว็บหน้าหลัก

ไฟล์ index.php เป็นไฟล์เว็บหน้าหลักและเป็นโฮมเพจของระบบ CRUD ซึ่งทำหน้าที่เป็นจุดเริ่มต้นสำหรับผู้ใช้ในการเข้าถึงระบบ ตามความสัมพันธ์ของไฟล์ที่ได้อธิบายในหัวข้อ 3.3.2 ไฟล์นี้จะเป็นหน้าที่ผู้ใช้เห็นเป็นหน้าแรกและสามารถนำทางไปยังการดำเนินการอื่นๆ เช่น การเพิ่ม แก้ไข หรือลบข้อมูล หน้านี้จะใช้คำสั่ง SELECT ในการดึงข้อมูลและแสดงผลในรูปแบบตารางเพื่อแสดงข้อมูลจากฐานข้อมูล โดยมีรหัสคำสั่งดังนี้

ไฟล์ PHP **c:\webapp\source\index.php**

```
1  <?php
2  require_once 'config.php';
3  ?>
4  <!DOCTYPE html>
5  <html lang="th">
6  <head>
7      <meta charset="UTF-8">
8      <meta name="viewport" content="width=device-width, initial-scale=1.0">
9      <title>ระบบจัดการผู้ใช้</title>
10     <style>
11             table { border-collapse: collapse; width: 100%; }
12             th,td { border: 1px solid #ddd;        padding: 8px; text-align: left; }
13             th { background-color: #f2f2f2; }
14             .btn { padding: 5px 10px; text-decoration: none; margin: 2px; }
15             .btn-add { background-color: #4CAF50; color: white; }
```

```
16              .btn-edit { background-color: #2196F3; color: white; }
17              .btn-delete { background-color: #f44336; color: white; }
18      </style>
19  </head>
20  <body>
21      <h1>ระบบจัดการผู้ใช้</h1>
22      <a href="add.php" class="btn btn-add">เพิ่มผู้ใช้ใหม่</a>
23      <h2>รายการผู้ใช้</h2>
24      <?php
25      try {
26              // ดึงข้อมูลผู้ใช้ทั้งหมด
27              $stmt = $pdo->query("SELECT * FROM users ORDER BY created_at DESC");
28              $users = $stmt->fetchAll();
29              if (count($users) > 0) {
30                echo "<table>";
31                echo "<tr><th>ID</th><th>ชื่อ</th><th>อีเมล</th><th>อายุ</th><th>วันที่
                  สร้าง</th><th>การจัดการ</th></tr>";
32                foreach ($users as $user) {
33                      echo "<tr>";
34                      echo "<td>" . $user['id'] . "</td>";
35                      echo "<td>" . $user['name'] . "</td>";
36                      echo "<td>" . $user['email'] . "</td>";
37                      echo "<td>" . $user['age'] . "</td>";
38                      echo "<td>" . $user['created_at'] . "</td>";
39                      echo "<td>";
40                      echo "<a href='edit.php?id=" . $user['id'] . "' class='btn btn-edit'>แก้ไข
                        </a>";
41                      echo "<a href='delete_process.php?id=" . $user['id'] . "' class='btn btn-
                        delete' onclick='return confirm(\"คุณต้องการลบข้อมูลนี้หรือไม่?\")'>ลบ</a>";
42                      echo "</td>";
43                      echo "</tr>";
44                }
```

```
45                 echo "</table>";
46             } else {
47                 echo "<p>ไม่มีข้อมูลผู้ใช้ในระบบ</p>";
48             }
49     } catch (PDOException $e) {
50             echo "<p>เกิดข้อผิดพลาด: " . $e->getMessage() . "</p>";
51     }
52     ?>
53 </body>
54 </html>
```

คำอธิบาย

- บรรทัดที่ 2 ใช้คำสั่ง require_once เพื่อโหลดไฟล์ config.php ซึ่งจะทำให้สามารถใช้ตัวแปร $pdo ที่ได้จากการเชื่อมต่อฐานข้อมูลในไฟล์นี้ได้ การใช้ require_once จะป้องกันการโหลดไฟล์เดียวกันซ้ำ
- บรรทัดที่ 7-8 และ 10-18 เป็นส่วน HTML head ที่กำหนดข้อมูลเมตา ชื่อหน้าเว็บ และ CSS styles สำหรับการจัดรูปแบบตาราง ปุ่ม และลิงก์ โดยใช้ inline CSS
- บรรทัดที่ 22 เป็นการสร้างลิงก์ไปยังหน้า add.php สำหรับเพิ่มผู้ใช้ใหม่ ซึ่งเป็นจุดเชื่อมต่อไปยังการดำเนินการ Create ในระบบ CRUD
- บรรทัดที่ 27 ใช้เมธอด $pdo->query() เพื่อส่งคำสั่ง SQL "SELECT * FROM users ORDER BY created_at DESC" ไปยังฐานข้อมูล ซึ่งจะดึงข้อมูลผู้ใช้ทั้งหมดและเรียงลำดับตามวันที่สร้างจากรายการใหม่ไปรายการเก่า
- บรรทัดที่ 28 ใช้เมธอด fetchAll() เพื่อดึงข้อมูลทั้งหมดจากผลลัพธ์ของคำสั่ง SQL มาเก็บในตัวแปร $users ในรูปแบบ associative array
- บรรทัดที่ 29 ตรวจสอบจำนวนข้อมูลด้วยฟังก์ชัน count() หากมีข้อมูลจะแสดงตาราง หากไม่มีจะแสดงข้อความแจ้ง
- บรรทัดที่ 30-31 สร้างตารางและส่วนหัวตารางด้วย HTML โดยกำหนดคอลัมน์ เช่น ID ชื่อ อีเมล อายุ วันที่สร้าง และการจัดการ
- บรรทัดที่ 32-44 ใช้คำสั่งวนซ้ำ foreach เพื่อวนซ้ำแสดงข้อมูลผู้ใช้แต่ละคน โดยแต่ละแถวจะมีลิงก์แก้ไขที่ส่ง ID ไปยัง edit.php และลิงก์ลบที่ส่ง ID ไปยัง delete_process.php พร้อม JavaScript confirm เพื่อยืนยันการลบ

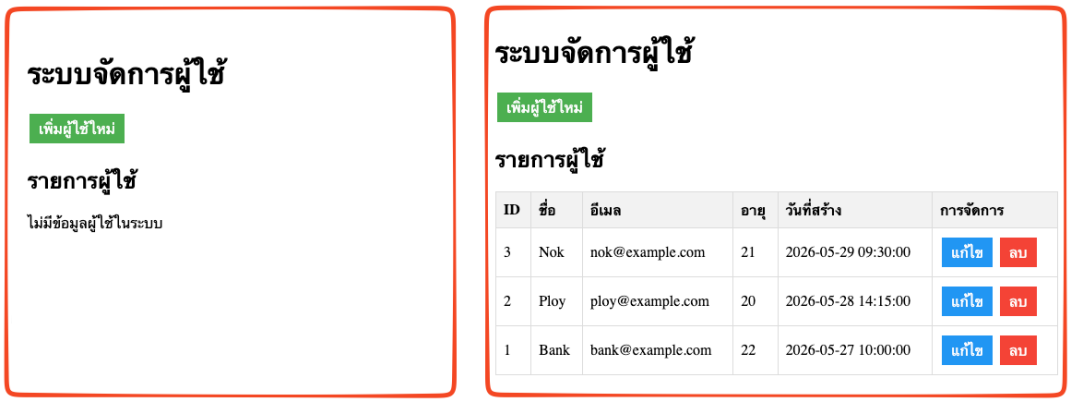
- บรรทัดที่ 49-51 ในส่วน catch จะทำงานเมื่อเกิดข้อผิดพลาดในการดึงข้อมูล โดยแสดงข้อความข้อผิดพลาดที่เกิดขึ้น

วิธีการทดสอบ

- เข้าถึงผ่าน http://localhost/index.php

ผลลัพธ์ที่คาดหวัง

- แสดงหน้าเว็บที่มีหัวเรื่อง "ระบบจัดการผู้ใช้"
- มีปุ่ม "เพิ่มผู้ใช้ใหม่" สีเขียว ที่คลิกแล้วไปหน้า add.php
- แสดงตารางข้อมูลผู้ใช้พร้อมปุ่มแก้ไขและลบ (หากมีข้อมูล) หรือข้อความ "ไม่มีข้อมูลผู้ใช้ในระบบ" (หากไม่มีข้อมูล)



ภาพที่ 12.6 ตัวอย่างผลลัพธ์ของหน้า index.php เมื่อเชื่อมต่อฐานข้อมูลสำเร็จกรณีที่มีและไม่มีข้อมูลในระบบ

ข้อผิดพลาดที่อาจพบได้

- Fatal error: require_once(): Failed opening required หมายถึง ไฟล์ config.php ไม่อยู่ในตำแหน่งที่ถูกต้อง แนวทางแก้ไขให้ตรวจสอบตำแหน่งไฟล์หรือปรับ path ในบรรทัดที่ 2
- Table 'blog_system.users' doesn't exist หมายถึง ตาราง users ยังไม่ถูกสร้างในฐานข้อมูล แนวทางแก้ไขให้ตรวจสอบการนำเข้าไฟล์ SQL ใน phpMyAdmin
- Call to a member function query() on null หมายถึง การเชื่อมต่อฐานข้อมูลล้มเหลว แนวทางแก้ไขให้ตรวจสอบไฟล์ config.php ว่าสามารถเชื่อมต่อได้หรือไม่
- Parse error: syntax error หมายถึง มีข้อผิดพลาดในการเขียน HTML หรือ PHP syntax แนวทางแก้ไขให้ตรวจสอบการปิดแท็ก HTML และเครื่องหมาย PHP
- Warning: count(): Parameter must be an array หมายถึง ตัวแปร $users ไม่ใช่ array แนวทางแก้ไขให้ตรวจสอบการทำงานของ fetchAll() และการเชื่อมต่อฐานข้อมูลผลลัพธ์ที่ถูกต้อง

3.5.2 หน้าเพิ่มข้อมูล

ไฟล์ add.php เป็นไฟล์ที่ทำหน้าที่แสดงฟอร์มสำหรับรับข้อมูลผู้ใช้ใหม่ ตามความสัมพันธ์ของไฟล์ที่ได้อธิบายในหัวข้อ 3.3.2 ไฟล์นี้จะรับการเรียกใช้จากลิงก์ *เพิ่มผู้ใช้ใหม่* ในหน้า index.php และจะส่งข้อมูลไปยัง add_process.php เมื่อผู้ใช้กดปุ่มบันทึก หน้านี้จะประกอบด้วยฟอร์ม HTML ที่มีช่องกรอกข้อมูล เช่น ชื่อ อีเมล และอายุ พร้อมการตรวจสอบข้อมูลเบื้องต้นด้วยแอตทริบิวต์ของ HTML ซึ่งมีรหัสคำสั่งดังนี้

ไฟล์ PHP **c:\webapp\source\add.php**

```
1  <!DOCTYPE html>
2  <html lang="th">
3  <head>
4      <meta charset="UTF-8">
5      <meta name="viewport" content="width=device-width, initial-scale=1.0">
6      <title>เพิ่มผู้ใช้ใหม่</title>
7  </head>
8  <body>
9      <h1>เพิ่มผู้ใช้ใหม่</h1>
10     <form action="add_process.php" method="POST">
11             <p>
12                 <label for="name">ชื่อ:</label><br>
13                 <input type="text" id="name" name="name" required>
14             </p>
15             <p>
16                 <label for="email">อีเมล:</label><br>
17                 <input type="email" id="email" name="email" required>
18             </p>
19             <p>
20                 <label for="age">อายุ:</label><br>
21                 <input type="number" id="age" name="age" min="1" max="120">
22             </p>
23             <p>
24                 <button type="submit">บันทึก</button>
25                 <a href="index.php">ยกเลิก</a>
```

```
26              </p>
27      </form>
28  </body>
29  </html>
```

คำอธิบาย

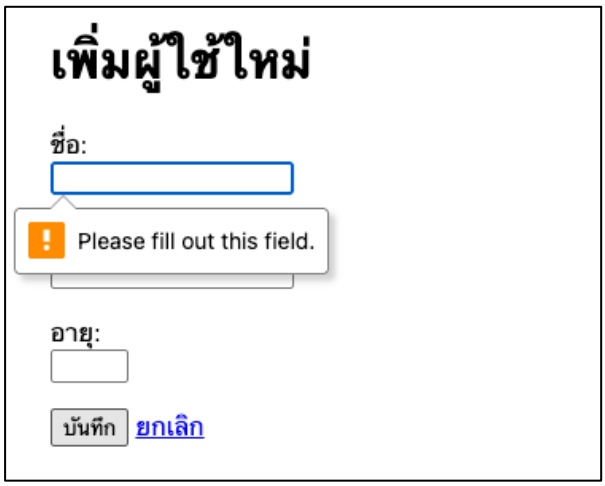
- บรรทัดที่ 3-7 เป็นส่วน HTML head ที่กำหนดข้อมูลเมตาและชื่อหน้าเว็บ โดยใช้โครงสร้าง HTML พื้นฐานโดยไม่ใช้ CSS ภายนอก
- บรรทัดที่ 10-27 เป็นการสร้างแท็ก form ที่กำหนด action="add_process.php" หมายความว่าเมื่อผู้ใช้กดปุ่มส่งข้อมูล ข้อมูลจะถูกส่งไปประมวลผลที่ไฟล์ add_process.php และกำหนด method="POST" เพื่อส่งข้อมูลในส่วนเนื้อหาของคำร้องขอ แทนการแนบข้อมูลไว้ใน URL เหมือนวิธี GET
- บรรทัดที่ 11-14 สร้างช่องกรอกชื่อด้วย input type="text" โดยกำหนด name="name" สำหรับการอ้างอิงในฝั่งเครื่องผู้ให้บริการ และ required เพื่อให้ผู้ใช้ต้องกรอกข้อมูลก่อนส่งฟอร์ม
- บรรทัดที่ 15-18 สร้างช่องกรอกอีเมลด้วย input type="email" ซึ่งจะมีการตรวจสอบรูปแบบอีเมลโดยอัตโนมัติของเบราว์เซอร์ และกำหนด required เพื่อให้เป็นข้อมูลที่จำเป็นต้องกรอก
- บรรทัดที่ 19-22 สร้างช่องกรอกอายุด้วย input type="number" โดยกำหนด min="1" max="120" เพื่อจำกัดค่าที่ผู้ใช้สามารถกรอกได้ในช่วงที่เหมาะสม และไม่กำหนด required เพื่อให้เป็นช่องที่อนุญาตให้เว้นว่างได้
- บรรทัดที่ 23-26 สร้างปุ่มบันทึกด้วย button type="submit" ซึ่งจะทำการส่งข้อมูลในฟอร์มไปยัง action ที่กำหนดไว้เมื่อผู้ใช้คลิก และสร้างลิงก์ยกเลิกที่จะพาผู้ใช้กลับไปยังหน้า index.php

วิธีการทดสอบ

- เข้าถึงผ่าน http://localhost/add.php หรือคลิกปุ่ม *เพิ่มผู้ใช้ใหม่* จากหน้า index.php

ผลลัพธ์ที่ถูกต้อง

- แสดงหน้าเว็บที่มีหัวเรื่อง *เพิ่มผู้ใช้ใหม่*
- มีฟอร์มที่มีช่องกรอก ชื่อ, อีเมล, อายุ
- ปุ่ม "บันทึก" และลิงก์ "ยกเลิก" ทำงานได้ถูกต้อง
- การตรวจสอบ required fields ทำงานเมื่อพยายามส่งฟอร์มโดยไม่กรอกข้อมูล



ภาพที่ 12.7 ตัวอย่างผลลัพธ์ของไฟล์ add.php สำหรับกรอกข้อมูลผู้ใช้ใหม่

ข้อผิดพลาดที่อาจพบได้

- ปุ่มบันทึกไม่ทำงาน อาจมีปัญหาเกี่ยวกับการเชื่อมต่อกับไฟล์ที่รับข้อมูลจากฟอร์ม แนวทางแก้ไขให้ตรวจสอบว่าไฟล์ add_process.php อยู่ในตำแหน่งที่ถูกต้อง
- การตรวจสอบ email ไม่ทำงาน หมายถึง เบราว์เซอร์ไม่รองรับ แนวทางแก้ไขทำได้โดยการใช้การสร้างฟังก์ชันตรวจสอบใน JavaScript ให้ทำการตรวจสอบค่าก่อนการส่งฟอร์ม
- Parse error: syntax error ... หมายถึง มีข้อผิดพลาดในโครงสร้างรหัส PHP เช่น ลืมเครื่องหมาย ; หรือปิด {} ไม่ครบคู่ แนวทางแก้ไขทำได้โดยตรวจสอบบรรทัดที่แสดงข้อผิดพลาดและใช้โปรแกรมพิมพ์รหัสคำสั่งที่มีความสามารถในการแสดงสีในบรรทัดที่ผิด (Syntax highlight) ช่วย

3.5.3 การประมวลผลการเพิ่มข้อมูล

ไฟล์ add_process.php เป็นไฟล์ที่ทำหน้าที่รับข้อมูลจากฟอร์ม add.php และดำเนินการบันทึกลงฐานข้อมูลตามความสัมพันธ์ของไฟล์ที่ได้อธิบายในหัวข้อ 3.3.2 ไฟล์นี้จะไม่แสดงหน้าเว็บให้ผู้ใช้เห็น แต่จะทำการประมวลผลข้อมูล ตรวจสอบความถูกต้อง บันทึกลงฐานข้อมูล และเปลี่ยนเส้นทางกลับไปยังหน้า index.php พร้อมข้อความแจ้งผลการดำเนินการ ซึ่งมีรหัสคำสั่งดังนี้

ไฟล์ PHP c:\webapp\source\add_process.php

```
1  <?php
2  require_once 'config.php';
3  if ($_POST) {
4      // รับและตรวจสอบข้อมูลจากฟอร์ม
5      $name = trim($_POST['name']);
6      $email = trim($_POST['email']);
```

```
7   if (!empty($_POST['age'])) {
8           $age = (int)$_POST['age'];
9   } else {
10          $age = null;
11  }
12  // ตรวจสอบข้อมูลพื้นฐาน
13  if (empty($name) || empty($email)) {
14          echo "<script>alert('กรุณากรอกชื่อและอีเมล'); history.back();</script>";
15          exit();
16  }
17  // ตรวจสอบรูปแบบอีเมล
18  if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
19          echo "<script>alert('รูปแบบอีเมลไม่ถูกต้อง'); history.back();</script>";
20          exit();
21  }
22  try {
23          // ตรวจสอบว่าอีเมลซ้ำหรือไม่
24          $stmt = $pdo->prepare("SELECT COUNT(*) FROM users WHERE email =
                :email");
25          $stmt->execute([':email' => $email]);
26          if ($stmt->fetchColumn() > 0) {
27                  echo "<script>alert('อีเมลนี้มีอยู่ในระบบแล้ว'); history.back();</script>";
28                  exit();
29          }
30          // เพิ่มข้อมูลใหม่
31          $stmt = $pdo->prepare("INSERT INTO users (name, email, age) VALUES
                (:name, :email, :age)");
32          $result = $stmt->execute([
                    ':name' => $name,
                    ':email' => $email,
                    ':age' => $age
33          ]);
```

```
34              if ($result) {
35                  echo "<script>alert('เพิ่มข้อมูลสำเร็จ'); " ,
36                      "window.location.href='index.php';</script>";
37              } else {
38                  echo "<script>alert('เกิดข้อผิดพลาดในการเพิ่มข้อมูล');" ,
                        "history.back();</script>";
39              }
40      } catch (PDOException $e) {
41              echo "<script>alert('เกิดข้อผิดพลาด: " . $e->getMessage() . "');",
                     history.back();</script>";
42      }
43  } else {
44      // กรณีที่ไม่ได้ส่งข้อมูลมา
45      header("Location: add.php");
46      exit();
47  }
```

คำอธิบาย

- บรรทัดที่ 2 ใช้คำสั่ง require_once เพื่อโหลดไฟล์ config.php ซึ่งจะทำให้สามารถใช้ตัวแปร $pdo ที่ได้จากการเชื่อมต่อฐานข้อมูลในไฟล์นี้ได้
- บรรทัดที่ 3 ตรวจสอบว่ามีการส่งข้อมูลด้วยวิธี POST หรือไม่ เพื่อป้องกันการเข้าถึงไฟล์โดยตรงผ่าน URL
- บรรทัดที่ 5-6 รับข้อมูลชื่อและอีเมลจากฟอร์มผ่านตัวแปร $_POST และใช้ฟังก์ชัน trim() เพื่อลบช่องว่างที่ไม่จำเป็น
- บรรทัดที่ 7-11 เป็นการตรวจสอบว่าช่องกรอกอายุเป็นค่าว่างหรือไม่ และจะมีการแปลงข้อมูลให้เป็นชนิดเลขจำนวนเต็มในกรณีที่มีการส่งค่ามาจากฟอร์ม และจะมีค่าเป็น null กรณีที่ไม่มีการส่งค่าจากฟอร์ม
- บรรทัดที่ 13-16 ตรวจสอบว่าข้อมูลที่จำเป็น (ชื่อและอีเมล) ถูกกรอกครบถ้วนหรือไม่ หากไม่ครบจะแสดง alert และให้ผู้ใช้กลับไปแก้ไขด้วย history.back()
- บรรทัดที่ 18-21 ตรวจสอบรูปแบบอีเมลด้วยฟังก์ชัน filter_var() และ FILTER_VALIDATE_EMAIL หากรูปแบบไม่ถูกต้องจะแสดงข้อความเตือน

- บรรทัดที่ 24-25 ตรวจสอบว่าอีเมลที่กรอกมีอยู่ในระบบแล้วหรือไม่ โดยใช้ prepared statement เพื่อช่วยลดความเสี่ยงจาก SQL Injection และใช้ fetchColumn() เพื่อดึงค่า COUNT มาตรวจสอบ
- บรรทัดที่ 26-29 หากอีเมลซ้ำจะแสดงข้อความแจ้งเตือนและให้ผู้ใช้กลับไปแก้ไข
- บรรทัดที่ 31-32 เป็นการเพิ่มข้อมูลลงฐานข้อมูลด้วย INSERT statement โดยใช้ prepared statement และ named parameters เพื่อป้องกัน SQL injection
- บรรทัดที่ 34-39 เป็นการตรวจสอบผลการเพิ่มข้อมูล หากสำเร็จจะแสดงข้อความด้วยหน้าต่างแจ้งเตือนและเปลี่ยนเส้นทางไปยัง index.php หากไม่สำเร็จจะแสดงข้อความแจ้งข้อผิดพลาด
- บรรทัดที่ 40-42 จะทำงานเมื่อเกิด PDOException โดยแสดงข้อความข้อผิดพลาดที่เกิดขึ้น
- บรรทัดที่ 43-47 หากไม่มีการส่งข้อมูลด้วย POST จะเปลี่ยนเส้นทางไปยังหน้า add.php

วิธีการทดสอบ

- กรอกข้อมูลในฟอร์มและกดปุ่มบันทึก

ผลลัพธ์ที่ถูกต้อง

- แสดงการแจ้งเตือน *เพิ่มข้อมูลสำเร็จ* และเปลี่ยนเส้นทางไปหน้า index.php
- ข้อมูลใหม่ปรากฏในตารางของหน้า index.php
- หากกรอกอีเมลซ้ำจะแสดงการแจ้งเตือน *อีเมลนี้มีอยู่ในระบบแล้ว*



ภาพที่ 12.8 ตัวอย่างผลลัพธ์ของไฟล์ add_process.php เมื่อเพิ่มข้อมูลสำเร็จและกรณีที่มีอีเมลซ้ำ

ข้อผิดพลาดที่อาจพบได้

- Fatal error: require_once(): Failed opening required หมายถึง ไฟล์ config.php ไม่อยู่ในตำแหน่งที่ถูกต้อง แนวทางแก้ไขให้ทำการตรวจสอบตำแหน่งไฟล์หรือปรับ path
- SQLSTATE[23000]: Integrity constraint violation หมายถึง มีข้อมูลซ้ำในฟิลด์ที่กำหนดเป็น UNIQUE แนวทางแก้ไขให้ทำการตรวจสอบการทำงานของการตรวจสอบอีเมลซ้ำ
- Call to a member function prepare() on null หมายถึง การเชื่อมต่อฐานข้อมูลล้มเหลว แนวทางแก้ไขให้ทำการตรวจสอบไฟล์ config.php

- Parse error: syntax error ... หมายถึง มีข้อผิดพลาดในโครงสร้างรหัส PHP เช่น ลืมเครื่องหมาย ; หรือปิด {} ไม่ครบคู่ แนวทางแก้ไขทำได้โดยตรวจสอบบรรทัดที่แสดงข้อผิดพลาดโดยใช้ editor ที่มี syntax highlight เพื่อแก้ไขให้ถูกต้อง

- Headers already sent หมายถึง มีการส่ง output ก่อนหน้า header() แนวทางแก้ไข: ตรวจสอบว่าไม่มีข้อความ ช่องว่าง หรือบรรทัดว่างก่อนแท็กเปิด <?php และก่อนเรียกใช้ header()

3.5.4 การแก้ไขข้อมูล

ไฟล์ edit.php ทำหน้าที่เป็นหน้าเว็บสำหรับแสดงฟอร์มแก้ไขข้อมูล โดยจะตรวจสอบค่า id ที่ส่งผ่าน URL เพื่อตรวจสอบว่ามีการระบุรหัสผู้ใช้ที่ต้องการแก้ไขหรือไม่ หากไม่พบ id ที่ถูกต้อง ระบบจะเปลี่ยนเส้นทางกลับไปยังหน้า index.php ทันที เมื่อพบ id ที่ถูกต้อง โปรแกรมจะดึงข้อมูลผู้ใช้จากฐานข้อมูลด้วยคำสั่ง SQL จากนั้นแสดงฟอร์มแก้ไขที่กรอกข้อมูลเดิมของผู้ใช้ให้อัตโนมัติ โดยในฟอร์มจะมีช่องกรอกชื่อ อีเมล และอายุ รวมถึงช่องแบบซ่อน (hidden) สำหรับส่งค่า id กลับไปยังไฟล์ edit_process.php เพื่อใช้ในการอัปเดตข้อมูลในฐานข้อมูล มีรหัสคำสั่งดังนี้

ไฟล์ PHP **c:\webapp\source\edit.php**

```
1  <?php
2  require_once 'config.php';
3  // ตรวจสอบว่ามี ID ส่งมาหรือไม่
4  if (!isset($_GET['id']) || !is_numeric($_GET['id'])) {
5      header("Location: index.php");
6      exit();
7  }
8  $id = (int)$_GET['id'];
9  try {
10     // ดึงข้อมูลผู้ใช้ที่ต้องการแก้ไข
11     $stmt = $pdo->prepare("SELECT * FROM users WHERE id = :id");
12     $stmt->execute([':id' => $id]);
13     $user = $stmt->fetch();
14     if (!$user) {
15             echo "<script>alert('ไม่พบข้อมูลผู้ใช้'); " ,
16                     "window.location.href='index.php';</script>";
17             exit();
18     }
```

```
19  } catch (PDOException $e) {
20      echo "<script>alert('เกิดข้อผิดพลาด: " . $e->getMessage() . "');",
                "window.location.href='index.php';</script>";
21      exit();
22  }
23  ?>
24  <!DOCTYPE html>
25  <html lang="th">
26  <head>
27      <meta charset="UTF-8">
28      <meta name="viewport" content="width=device-width, initial-scale=1.0">
29      <title>แก้ไขข้อมูลผู้ใช้</title>
30      <style>
31              .form-group {
32                      margin: 15px 0;
33              }
34              label {
35                      display: block;
36                      margin-bottom: 5px;
37              }
38              input,
39              textarea {
40                      width: 300px;
41                      padding: 8px;
42              }
43              .btn {
44                      padding: 10px 20px;
45                      margin: 5px;
46                      text-decoration: none;
47              }
48              .btn-save {
49                      background-color: #4CAF50;
```

```
                color: white;
                border: none;
            }
            .btn-cancel {
                background-color: gray;
                color: white;
            }
    </style>
</head>
<body>
    <h1>แก้ไขข้อมูลผู้ใช้</h1>
    <form action="edit_process.php" method="POST">
            <input type="hidden" name="id" value="<?php echo $user['id']; ?>">
            <div class="form-group">
                <label for="name">ชื่อ:</label>
                <input type="text" id="name" name="name" value="<?php echo
                    htmlspecialchars($user['name']); ?>" required>
            </div>
            <div class="form-group">
                <label for="email">อีเมล:</label>
                <input type="email" id="email" name="email" value="<?php echo
                    htmlspecialchars($user['email']); ?>" required>
            </div>
            <div class="form-group">
                <label for="age">อายุ:</label>
                <input type="number" id="age" name="age" value="<?php echo
                    htmlspecialchars($user['age']); ?>" min="1" max="120">
            </div>
            <button type="submit" class="btn btn-save">บันทึกการแก้ไข</button>
            <a href="index.php" class="btn btn-cancel">ยกเลิก</a>
    </form>
</body>
</html>
```

คำอธิบาย

- บรรทัดที่ 2: ใช้ require_once 'config.php'; เพื่อนำเข้าไฟล์กำหนดค่าการเชื่อมต่อฐานข้อมูล MySQL ซึ่งในไฟล์นี้กำหนดตัวแปร $pdo สำหรับการทำงานกับฐานข้อมูลผ่าน PDO
- บรรทัดที่ 4-7: ตรวจสอบว่ามีการส่งพารามิเตอร์ id ผ่าน URL หรือไม่ และตรวจสอบว่าค่าที่ส่งมาเป็นตัวเลข (is_numeric($_GET['id'])) เพื่อป้องกันการโจมตีด้วยการใส่ค่าที่ไม่ถูกต้อง (เช่น SQL injection หรือค่า string) ถ้าไม่ผ่านเงื่อนไขจะเปลี่ยนเส้นทางผู้ใช้กลับไปหน้า index.php และหยุดการทำงานด้วย exit()
- บรรทัดที่ 8: ทำการแปลงชนิดข้อมูล (type casting) ให้ค่าที่รับจาก URL เป็นเลขจำนวนเต็มเพื่อความปลอดภัยในการใช้งานในฐานข้อมูล
- บรรทัดที่ 11-13: สร้าง prepared statement ด้วย SELECT * FROM users WHERE id = :id เพื่อดึงข้อมูลผู้ใช้ตามรหัส id ที่ระบุ และป้องกัน SQL injection โดยใช้ named parameter :id จากนั้น execute statement และนำผลลัพธ์ที่ได้มาเก็บในตัวแปร $user ด้วย fetch()
- บรรทัดที่ 14-18 เป็นการตรวจสอบว่าผลลัพธ์จากการอ่านค่าในฐานข้อมูลมีผู้ใช้ตรงกับ id ที่กำหนดหรือไม่ ซึ่งเป็นการตรวจสอบจากผลที่ได้รับกลับมาจากการส่งคำสั่ง SQL หากผลของฟังก์ชัน fetch() เป็นค่าว่างจะแจ้งเตือนด้วยกล่องโต้ตอบของ JavaScript ว่า “ไม่พบข้อมูลผู้ใช้” และเปลี่ยนเส้นทางกลับไปยัง index.php เพื่อป้องกันการเข้าถึงหน้าแก้ไขโดยใช้ URL ที่ไม่ถูกต้อง
- บรรทัดที่ 9-22 ใช้โครงสร้าง try-catch จับข้อผิดพลาดที่อาจเกิดจากการทำงานของ PDO เช่น การเชื่อมต่อฐานข้อมูลล้มเหลว หรือ statement ผิดพลาด ในกรณีเกิดข้อผิดพลาด จะแสดงการแจ้งเตือนข้อความข้อผิดพลาดที่เกิดขึ้น และเปลี่ยนเส้นทางกลับไปยัง index.php เพื่อให้ผู้ใช้ไม่ติดอยู่ที่หน้าที่ล้มเหลว
- บรรทัดที่ 61-77 เป็นการสร้างฟอร์ม HTML ที่กำหนดให้ใช้ method="POST" และ action="edit_process.php" เพื่อส่งข้อมูลการแก้ไขไปยังไฟล์ประมวลผล
- บรรทัดที่ 62 มีการนำค่า id ของผู้ใช้ไปกำหนดค่าให้กับช่องกรอกข้อมูลแบบซ่อน (hidden input field) ซึ่งเป็นเทคนิคหนึ่งที่สำคัญในการส่งข้อมูลตัวระบุ (identifier) กลับไปยังฝั่งเครื่องผู้ให้บริการโดยไม่ต้องให้ผู้ใช้เห็นหรือแก้ไขค่าโดยตรง การใช้ input type="hidden" ช่วยให้โปรแกรมสามารถเก็บรหัสผู้ใช้ที่กำลังแก้ไขไว้ในฟอร์ม และเมื่อผู้ใช้ส่งข้อมูลไปยังไฟล์ edit_process.php ค่า id จะถูกรวมเป็นส่วนหนึ่งของข้อมูลที่ส่งผ่าน HTTP POST เพื่อใช้ในการระบุตำแหน่งข้อมูลเดิมในฐานข้อมูลที่ต้องการอัปเดต เทคนิคนี้มีความสำคัญอย่างยิ่งสำหรับการทำงานของระบบ CRUD เนื่องจากช่วยให้สามารถเชื่อมโยงระหว่างข้อมูลที่แก้ไขกับระเบียนเดิมในตารางฐานข้อมูลได้อย่างถูกต้องและปลอดภัย อีกทั้งยังลดความเสี่ยงจากข้อผิดพลาดที่อาจเกิดขึ้นหากใช้วิธีอ้างอิงผ่าน URL แบบ GET ซึ่งมักเปิดโอกาสให้ผู้ใช้หรือบุคคลภายนอกเปลี่ยนแปลงค่า id ได้โดยตรง

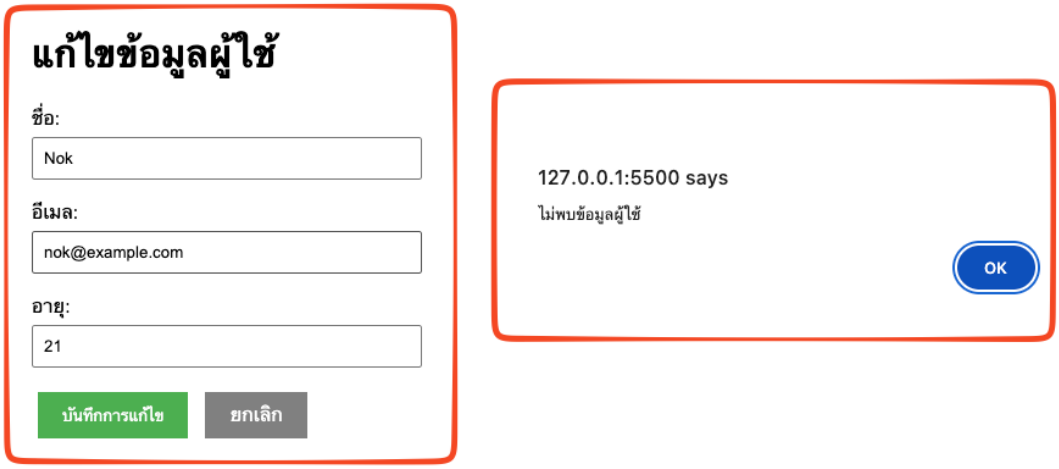
- แต่ละช่องกรอกข้อมูลมีการใช้ข้อมูลเดิมของผู้ใช้ที่ดึงมาจากฐานข้อมูล (value="<?php echo htmlspecialchars($user['...']); ?>") เพื่อให้ผู้ใช้แก้ไขข้อมูลที่ต้องการ และมีการใช้ htmlspecialchars() เพื่อป้องกันการแสดงผล HTML หรือ JavaScript แฝง (XSS: Cross-site scripting) ที่อาจแฝงอยู่ในข้อมูล

วิธีการทดสอบ

- เข้าถึงโดยการคลิกปุ่ม *แก้ไข* จากหน้า index.php เพื่อให้มีการส่งค่า id ไปยัง edit.php

ผลลัพธ์ที่ถูกต้อง

- แสดงหน้าเว็บที่มีหัวเรื่อง "แก้ไขข้อมูลผู้ใช้"
- มีฟอร์มที่มีช่องกรอก ชื่อ, อีเมล, อายุ พร้อมด้วยข้อมูลเดิมของผู้ใช้ที่ทำการแก้ไข
- ปุ่ม "บันทึก" และลิงก์ "ยกเลิก" ทำงานได้ถูกต้อง
- เมื่อคลิก “บันทึกการแก้ไข” ระบบจะส่งข้อมูลไป edit_process.php และเมื่อบันทึกสำเร็จต้อง มีการย้อนกลับไปที่ index.php
- ถ้า id ที่ส่งมาถูกต้องตามรูปแบบแต่ไม่มีข้อมูล ระบบต้องแสดงการแจ้งเตือน “ไม่พบข้อมูลผู้ใช้” และย้อนกลับหน้า index.php



ภาพที่ 12.9 ตัวอย่างผลลัพธ์ของไฟล์ edit.php กรณีพบข้อมูลและไม่พบข้อมูลผู้ใช้

ข้อผิดพลาดที่อาจพบได้

- Fatal error: require_once(): Failed opening required ‘config.php’ หมายถึงไฟล์ config.php ไม่อยู่ในตำแหน่งที่ถูกต้อง หรือไม่มีสิทธิ์ให้เว็บเซิร์ฟเวอร์เข้าถึงไฟล์ดังกล่าว แนวทางแก้ไขทำได้โดยการตรวจสอบตำแหน่งไฟล์ config.php ให้ถูกต้องตามลำดับโฟลเดอร์ที่อ้างอิงในรหัสคำสั่ง

- Call to a member function prepare() on null หมายถึง ตัวแปร $pdo ไม่มีการกำหนดค่า (null) เพราะเกิดข้อผิดพลาดในการโหลดไฟล์ config.php หรือไม่สามารถสร้างการเชื่อมต่อ PDO ได้ แนวทางแก้ไขทำได้โดยตรวจสอบให้แน่ใจว่าใน config.php มีการสร้างอ็อบเจกต์ PDO อย่างถูกต้อง และตรวจสอบข้อมูลการเชื่อมต่อฐานข้อมูล (host, username, password, database name)
- PDOException: SQLSTATE[42S02]: Base table or view not found: 1146 Table 'users' doesn't exist หมายถึง ตาราง users ที่อ้างอิงในคำสั่ง SQL ไม่ถูกสร้างขึ้นในฐานข้อมูล แนวทางแก้ไขทำได้โดยตรวจสอบฐานข้อมูลให้แน่ใจว่ามีตาราง users อยู่แล้ว และชื่อของตารางตรงกับที่ใช้ในคำสั่ง SQL
- ไม่พบข้อมูลผู้ใช้ หมายถึง ID ที่ส่งผ่าน URL ($_GET['id']) ไม่ตรงกับข้อมูลผู้ใช้ใด ๆ ในตาราง users ทำให้ตัวแปร $user เป็นค่า false แนวทางแก้ไขทำได้โดยตรวจสอบค่า ID ใน URL เช่น edit.php?id=1 ให้แน่ใจว่ามีระเบียนที่ตรงกับ ID นี้อยู่จริงในฐานข้อมูล
- Parse error: syntax error, unexpected ... หมายถึง มีข้อผิดพลาดในโครงสร้างรหัส PHP เช่น ลืมเครื่องหมาย ; หรือปิด {} ไม่ครบคู่ แนวทางแก้ไขทำได้โดยตรวจสอบบรรทัดที่แสดงข้อผิดพลาดโดยใช้ editor ที่มี syntax highlight เพื่อแก้ไขให้ถูกต้อง
- Headers already sent หมายถึง มีการแสดงผลออกทางหน้าจอ (เช่น echo หรือ whitespace) ก่อนการเรียกใช้ header("Location: ...") แนวทางแก้ไขทำได้โดยตรวจสอบให้แน่ใจว่าไม่มีการส่ง output ใด ๆ ก่อนเรียก header() โดยเฉพาะช่องว่างหรือบรรทัดว่างก่อน <?php ในไฟล์

3.5.5 การประมวลผลการแก้ไขข้อมูล

ไฟล์ edit_process.php เป็นไฟล์ประมวลผลที่ทำหน้าที่รับข้อมูลที่ผู้ใช้แก้ไขจากฟอร์มในหน้า edit.php และดำเนินการอัปเดตข้อมูลผู้ใช้ลงในฐานข้อมูล ตามความสัมพันธ์ของไฟล์ที่อธิบายในหัวข้อ 3.5.2 ไฟล์นี้ไม่ได้แสดงหน้าเว็บสำหรับผู้ใช้โดยตรง แต่จะประมวลผลข้อมูล ตรวจสอบความถูกต้อง และปรับปรุงข้อมูลในตารางฐานข้อมูล เมื่อการแก้ไขข้อมูลเสร็จสิ้น จะเปลี่ยนเส้นทางผู้ใช้กลับไปยังหน้า index.php พร้อมแจ้งผลการดำเนินการให้ทราบผ่านหน้าต่างแจ้งเตือนของ JavaScript โดยรหัสคำสั่งที่ใช้ในไฟล์นี้มีดังนี้:

ไฟล์ PHP c:\webapp\source\edit_process.php

```
1  <?php
2  require_once 'config.php';
3  if ($_POST) {
4      // รับและตรวจสอบข้อมูลจากฟอร์ม
5      $id = (int)$_POST['id'];
6      $name = trim($_POST['name']);
7      $email = trim($_POST['email']);
```

```
8       $age = !empty($_POST['age']) ? (int)$_POST['age'] : null;
9       // ตรวจสอบข้อมูลพื้นฐาน
10      if (empty($name) || empty($email) || $id <= 0) {
11              echo "<script>alert('กรุณากรอกข้อมูลให้ครบถ้วน'); history.back();</script>";
12              exit();
13      }
14      // ตรวจสอบรูปแบบอีเมล
15      if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
16              echo "<script>alert('รูปแบบอีเมลไม่ถูกต้อง'); history.back();</script>";
17              exit();
18      }
19      try {
20              // ตรวจสอบว่าอีเมลซ้ำกับคนอื่นหรือไม่
21              $stmt = $pdo->prepare("SELECT COUNT(*) FROM users WHERE email =
                        :email AND id != :id");
22              $stmt->execute([':email' => $email, ':id' => $id]);
23              if ($stmt->fetchColumn() > 0) {
24                      echo "<script>alert('อีเมลนี้มีคนอื่นใช้แล้ว'); history.back();</script>";
25                      exit();
26              }
27              // อัปเดตข้อมูล
28              $stmt = $pdo->prepare("UPDATE users SET name = :name, email = :email,
                        age = :age WHERE id = :id");
29              $result = $stmt->execute([
                        ':name' => $name,
                        ':email' => $email,
                        ':age' => $age,
                        ':id' => $id
30              ]);
31              if ($result && $stmt->rowCount() > 0) {
32                      echo "<script>alert('แก้ไขข้อมูลสำเร็จ'); ",
                              "window.location.href='index.php';</script>";
```

```
33              } else {
34                      echo "<script>alert('ไม่มีการเปลี่ยนแปลงข้อมูล'); ",
                            "window.location.href='index.php';</script>";
35              }
36      } catch (PDOException $e) {
37              echo "<script>alert('เกิดข้อผิดพลาด: " . $e->getMessage() . "');",
                        "history.back();</script>";
38      }
39  } else {
40      header("Location: index.php");
41      exit();
42  }
```

คำอธิบาย

- บรรทัดที่ 2: ใช้คำสั่ง require_once 'config.php'; เพื่อนำเข้าไฟล์กำหนดค่าการเชื่อมต่อฐานข้อมูล ซึ่งทำให้สามารถใช้ตัวแปร $pdo สำหรับทำงานกับฐานข้อมูลผ่าน PDO ได้
- บรรทัดที่ 3: ใช้คำสั่ง if ($_POST) เพื่อตรวจสอบว่ามีการส่งข้อมูลจากฟอร์มมาด้วยวิธี POST หรือไม่ หากมีข้อมูลส่งมา โปรแกรมจะเข้าสู่กระบวนการรับค่า ตรวจสอบ และอัปเดตข้อมูลผู้ใช้
- บรรทัดที่ 5–8: เป็นการรับค่าจากฟอร์ม ได้แก่ id, name, email และ age โดยค่า id ถูกแปลงเป็นจำนวนเต็มเพื่อใช้ระบุผู้ใช้ที่ต้องการแก้ไข ค่า name และ email ใช้ฟังก์ชัน trim() เพื่อลบช่องว่างที่ไม่จำเป็น ส่วนค่า age ใช้ตัวดำเนินการเงื่อนไขแบบย่อเพื่อตรวจสอบว่ามีการกรอกข้อมูลหรือไม่ หากมีข้อมูลจะถูกแปลงเป็นจำนวนเต็ม แต่ถ้าไม่มีข้อมูลจะกำหนดค่าเป็น null
- บรรทัดที่ 10–13: ใช้คำสั่งเงื่อนไขเพื่อตรวจสอบข้อมูลพื้นฐาน ได้แก่ ชื่อ อีเมล และรหัสผู้ใช้ หากชื่อหรืออีเมลว่าง หรือค่า id น้อยกว่าหรือเท่ากับ 0 โปรแกรมจะแสดงกล่องแจ้งเตือนว่า “กรุณากรอกข้อมูลให้ครบถ้วน” แล้วใช้ history.back() เพื่อย้อนกลับไปยังหน้าฟอร์มเดิม
- บรรทัดที่ 15–18: ใช้ฟังก์ชัน filter_var() ร่วมกับ FILTER_VALIDATE_EMAIL เพื่อตรวจสอบรูปแบบของอีเมล หากอีเมลไม่ถูกต้องตามรูปแบบ โปรแกรมจะแสดงกล่องแจ้งเตือนว่า “รูปแบบอีเมลไม่ถูกต้อง” แล้วย้อนกลับไปยังหน้าฟอร์มเดิม
- บรรทัดที่ 20–26 เป็นส่วนเริ่มต้นของโครงสร้าง try สำหรับประมวลผลกับฐานข้อมูล โดยเริ่มจากการตรวจสอบว่าอีเมลที่ผู้ใช้กรอกซ้ำกับผู้ใช้อื่นหรือไม่ ด้วยคำสั่ง SQL SELECT COUNT(*) FROM users WHERE email = :email AND id != :id ซึ่งมีเงื่อนไข id != :id เพื่อไม่นับอีเมลของผู้ใช้คนเดิม และใช้ prepared statement พร้อม named parameters เพื่อลดความเสี่ยงจาก SQL injection หากพบว่าอีเมล

ดังกล่าวมีผู้ใช้อื่นใช้อยู่แล้ว โปรแกรมจะแสดงกล่องแจ้งเตือนว่า “อีเมลนี้มีคนอื่นใช้แล้ว” และย้อนกลับไปยังหน้าฟอร์มเดิม

- บรรทัดที่ 27-30: เป็นการสร้างและประมวลผลคำสั่ง SQL สำหรับอัปเดตข้อมูลผู้ใช้ โดยใช้คำสั่ง UPDATE users SET name = :name, email = :email, age = :age WHERE id = :id ร่วมกับ prepared statement และ named parameters ได้แก่ :name, :email, :age และ :id เพื่อส่งค่าที่ต้องการแก้ไขไปยังฐานข้อมูลอย่างเป็นระบบ

- บรรทัดที่ 31-32: ใช้คำสั่ง if ($result && $stmt->rowCount() > 0) เพื่อตรวจสอบว่าคำสั่งอัปเดตทำงานสำเร็จและมีแถวข้อมูลถูกเปลี่ยนแปลงจริงหรือไม่ หากเงื่อนไขเป็นจริง โปรแกรมจะแสดงกล่องแจ้งเตือนว่า “แก้ไขข้อมูลสำเร็จ” และเปลี่ยนเส้นทางกลับไปยังหน้า index.php

- บรรทัดที่ 33-35: เป็นกรณีที่คำสั่งอัปเดตทำงานได้ แต่ไม่มีแถวข้อมูลใดเปลี่ยนแปลง เช่น ผู้ใช้กดบันทึกโดยไม่ได้แก้ไขข้อมูลเดิม โปรแกรมจะแสดงกล่องแจ้งเตือนว่า “ไม่มีการเปลี่ยนแปลงข้อมูล” และเปลี่ยนเส้นทางกลับไปยังหน้า index.php

- บรรทัดที่ 36-38: เป็นส่วน catch (PDOException $e) สำหรับจัดการข้อผิดพลาดที่เกิดจากการทำงานกับฐานข้อมูล หากเกิดข้อผิดพลาด โปรแกรมจะแสดงกล่องแจ้งเตือนพร้อมข้อความข้อผิดพลาด และใช้ history.back() เพื่อย้อนกลับไปยังหน้าก่อนหน้า

- บรรทัดที่ 39-42: เป็นกรณีที่ไม่มีการส่งข้อมูลด้วยวิธี POST เข้ามายังไฟล์นี้ โปรแกรมจะใช้คำสั่ง header("Location: index.php"); เพื่อเปลี่ยนเส้นทางกลับไปยังหน้า index.php และหยุดการทำงานด้วย exit();

วิธีการทดสอบ

- เข้าถึงหน้า edit.php โดยคลิกปุ่ม “แก้ไข” จากหน้า index.php เพื่อให้มีการส่งค่า id ของผู้ใช้ที่ต้องการแก้ไข

- แก้ไขข้อมูลชื่อ อีเมล หรืออายุ แล้วกดปุ่ม “บันทึกการแก้ไข”

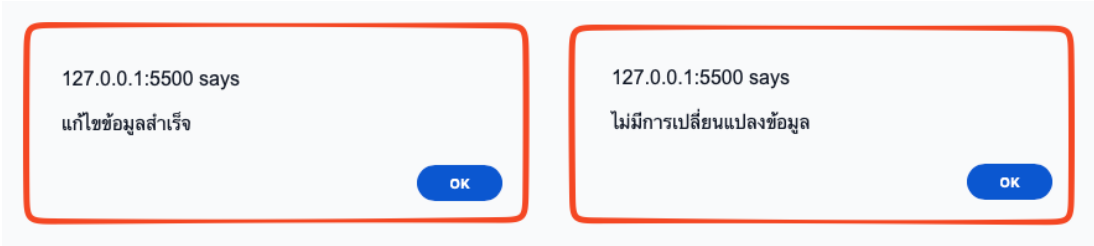
- ทดลองกรณีต่าง ๆ เช่น แก้ไขข้อมูลจริง กดบันทึกโดยไม่เปลี่ยนข้อมูล กรอกอีเมลผิดรูปแบบ หรือใช้อีเมลซ้ำกับผู้ใช้อื่น

ผลลัพธ์ที่ถูกต้อง

- หากแก้ไขข้อมูลสำเร็จ ระบบจะแสดงการแจ้งเตือนว่า “แก้ไขข้อมูลสำเร็จ” และเปลี่ยนเส้นทางกลับไปยังหน้า index.php

- หากไม่มีการเปลี่ยนแปลงข้อมูล ระบบจะแสดงการแจ้งเตือนว่า “ไม่มีการเปลี่ยนแปลงข้อมูล” และเปลี่ยนเส้นทางกลับไปยังหน้า index.php

- หากกรอกข้อมูลไม่ครบถ้วน ระบบจะแสดงการแจ้งเตือนว่า “กรุณากรอกข้อมูลให้ครบถ้วน” และย้อนกลับไปยังหน้าฟอร์มเดิม



ภาพที่ 12.10 ตัวอย่างผลลัพธ์ของไฟล์ edit_process.php กรณีแก้ไขข้อมูลสำเร็จและไม่มีการเปลี่ยนแปลงข้อมูล

- หากกรอกรูปแบบอีเมลไม่ถูกต้อง ระบบจะแสดงการแจ้งเตือนว่า “รูปแบบอีเมลไม่ถูกต้อง” และย้อนกลับไปยังหน้าฟอร์มเดิม
- หากใช้อีเมลซ้ำกับผู้ใช้อื่น ระบบจะแสดงการแจ้งเตือนว่า “อีเมลนี้มีคนอื่นใช้แล้ว” และย้อนกลับไปยังหน้าฟอร์มเดิม
- หากเข้าถึงไฟล์ edit_process.php โดยตรงโดยไม่มีการส่งข้อมูลด้วย POST ระบบจะเปลี่ยนเส้นทางกลับไปยังหน้า index.php

ข้อผิดพลาดที่อาจพบได้

- Fatal error: require_once(): Failed opening required 'config.php' หมายถึง ไฟล์ config.php ไม่อยู่ในตำแหน่งที่ถูกต้อง หรือเว็บเซิร์ฟเวอร์ไม่สามารถเข้าถึงไฟล์ดังกล่าวได้ แนวทางแก้ไขคือ ตรวจสอบตำแหน่งไฟล์และสิทธิ์การเข้าถึงไฟล์
- Call to a member function prepare() on null หมายถึง ตัวแปร $pdo ไม่มีค่า หรือไม่สามารถสร้างอ็อบเจกต์ PDO ได้ แนวทางแก้ไขคือ ตรวจสอบการเชื่อมต่อฐานข้อมูลในไฟล์ config.php
- PDOException: SQLSTATE[42S02]: Base table or view not found หมายถึง ไม่พบตาราง users ในฐานข้อมูล แนวทางแก้ไขคือ ตรวจสอบว่ามีการสร้างตาราง users แล้ว และใช้ชื่อฐานข้อมูลถูกต้อง
- SQLSTATE[23000]: Integrity constraint violation หมายถึง ข้อมูลที่บันทึกขัดกับข้อกำหนดของฐานข้อมูล เช่น มีอีเมลซ้ำในฟิลด์ที่กำหนดเป็น UNIQUE แนวทางแก้ไขคือ ตรวจสอบเงื่อนไขการตรวจสอบอีเมลซ้ำก่อนอัปเดตข้อมูล
- Parse error: syntax error ... หมายถึง มีข้อผิดพลาดทางไวยากรณ์ของ PHP เช่น ลืมเครื่องหมาย ; หรือปิดวงเล็บปีกกาไม่ครบคู่ แนวทางแก้ไขคือ ตรวจสอบบรรทัดที่แจ้งข้อผิดพลาดในโปรแกรมแก้ไขรหัสคำสั่ง
- Headers already sent หมายถึง มีการส่งผลลัพธ์ออกไปก่อนเรียกใช้ header() แนวทางแก้ไขคือ ตรวจสอบว่าไม่มีช่องว่างหรือข้อความใด ๆ ก่อนแท็กเปิด <?php และก่อนเรียกใช้ header()

3.5.6 การลบข้อมูล (Delete) - ไฟล์ประมวลผลการลบ

การลบข้อมูลเป็นการดำเนินการที่มีเพียงไฟล์ประมวลผลเดียว เนื่องจากการลบไม่ต้องการหน้าฟอร์ม แต่จะรับ ID ผ่าน URL และดำเนินการลบทันที ไฟล์ delete_process.php มีรหัสคำสั่งดังนี้

ไฟล์ PHP c:\webapp\source\delete_process.php

```
1  <?php
2  require_once 'config.php';
3  // ตรวจสอบว่ามี ID ส่งมาหรือไม่
4  if (!isset($_GET['id']) || !is_numeric($_GET['id'])) {
5      echo "<script>alert('ไม่พบข้อมูลที่ต้องการลบ'); ",
           "window.location.href='index.php';</script>";
6      exit();
7  }
8  $id = (int)$_GET['id'];
9  try {
10     // ตรวจสอบว่ามีข้อมูลผู้ใช้นี้อยู่จริงหรือไม่
11     $stmt = $pdo->prepare("SELECT COUNT(*) FROM users WHERE id = :id");
12     $stmt->execute([':id' => $id]);
13     if ($stmt->fetchColumn() == 0) {
14         echo "<script>alert('ไม่พบข้อมูลผู้ใช้ที่ต้องการลบ'); ",
               "window.location.href='index.php';</script>";
15         exit();
16     }
17     // ลบข้อมูลผู้ใช้
18     $stmt = $pdo->prepare("DELETE FROM users WHERE id = :id");
19     $result = $stmt->execute([':id' => $id]);
20     if ($result && $stmt->rowCount() > 0) {
21         echo "<script>alert('ลบข้อมูลสำเร็จ'); ",
               "window.location.href='index.php';</script>";
22     } else {
23         echo "<script>alert('เกิดข้อผิดพลาดในการลบข้อมูล'); ",
               "window.location.href='index.php';</script>";
24     }
```

```
25 } catch (PDOException $e) {
26     echo "<script>alert('เกิดข้อผิดพลาด: " . $e->getMessage() . "'); ",
27             "window.location.href='index.php';</script>";
28 }
```

คำอธิบาย

- บรรทัดที่ 2 ใช้ require_once 'config.php'; เพื่อนำเข้าไฟล์การกำหนดค่าการเชื่อมต่อฐานข้อมูล ซึ่งไฟล์นี้จะสร้างตัวแปร $pdo สำหรับทำงานกับฐานข้อมูลผ่าน PDO เพื่อให้สามารถเรียกใช้งานฐานข้อมูลได้อย่างปลอดภัยและง่ายดาย
- บรรทัดที่ 4–7 เป็นการตรวจสอบว่ามีการส่งพารามิเตอร์ id มาผ่าน URL หรือไม่ และตรวจสอบว่าค่าที่ส่งมานั้นเป็นตัวเลข (is_numeric($_GET['id'])) เพื่อป้องกันการโจมตีด้วยการส่งข้อมูลที่ไม่ถูกต้อง เช่น SQL injection หากไม่ผ่านเงื่อนไข จะแจ้งเตือน “ไม่พบข้อมูลที่ต้องการลบ” และเปลี่ยนเส้นทางกลับไปหน้า index.php
- บรรทัดที่ 8 เป็นการแปลงค่าพารามิเตอร์ id ที่ได้รับจาก URL ให้เป็นจำนวนเต็ม ((int)$_GET['id']) เพื่อความปลอดภัยในการนำไปใช้งานกับฐานข้อมูล
- บรรทัดที่ 11–16 เป็นการตรวจสอบว่ามีข้อมูลผู้ใช้ที่ต้องการลบอยู่จริงหรือไม่ โดยใช้ SELECT COUNT(*) ตรวจสอบว่ามีแถวในตาราง users ที่มีค่า id ตรงกับที่ส่งมาหรือไม่ หากไม่พบข้อมูล จะแจ้งเตือนและเปลี่ยนเส้นทางกลับไปหน้า index.php เพื่อป้องกันการลบข้อมูลที่ไม่ถูกต้อง
- บรรทัดที่ 16–20: หากพบข้อมูล จะทำการลบผู้ใช้ออกจากฐานข้อมูลด้วยคำสั่ง SQL DELETE FROM users WHERE id = :id โดยใช้ prepared statement และ named parameter :id เพื่อป้องกัน SQL injection จากนั้นตรวจสอบผลการทำงาน หากลบสำเร็จจะแจ้งว่า “ลบข้อมูลสำเร็จ” และเปลี่ยนเส้นทางกลับไปยัง index.php หากลบไม่สำเร็จจะแสดงข้อความข้อผิดพลาดและเปลี่ยนเส้นทางกลับไปยัง index.php
- บรรทัดที่ 9-28 เป็นการใช้โครงสร้าง try-catch เพื่อตรวจจับข้อผิดพลาดที่เกิดขึ้นขณะทำงาน เช่น การเชื่อมต่อฐานข้อมูลล้มเหลวหรือคำสั่ง SQL ผิดพลาด และแสดงข้อความข้อผิดพลาดให้ผู้ใช้ทราบ พร้อมเปลี่ยนเส้นทางกลับไปหน้า index.php เพื่อไม่ให้ผู้ใช้ติดอยู่ในหน้าลบข้อมูล

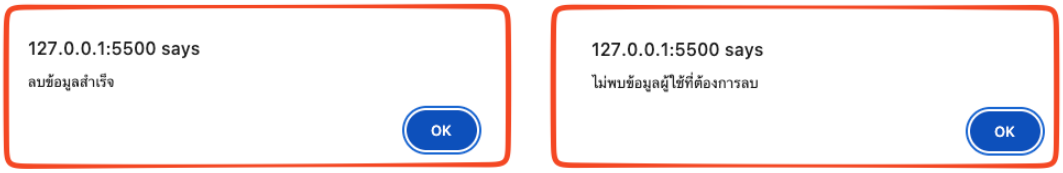
วิธีการทดสอบ

- เข้าถึงผ่าน http://localhost/index.php แล้วคลิกลบผู้ใช้ที่ต้องการทดสอบ

ผลลัพธ์ที่ถูกต้อง

- เมื่อค่า id ถูกต้องและมีข้อมูลอยู่ในฐานข้อมูล ระบบจะแจ้งเตือน “ลบข้อมูลสำเร็จ” และเปลี่ยนเส้นทางกลับไปหน้า index.php

- เมื่อค่า id ไม่ตรงกับข้อมูลใด ๆ ในฐานข้อมูล ระบบจะแจ้งเตือน “ไม่พบข้อมูลผู้ใช้ที่ต้องการลบ” และเปลี่ยนเส้นทางกลับไปหน้า index.php
- เมื่อไม่มีการส่งพารามิเตอร์ id หรือส่งค่าที่ไม่ใช่ตัวเลข ระบบจะแจ้งเตือน “ไม่พบข้อมูลที่ต้องการลบ” และเปลี่ยนเส้นทางกลับไปหน้า index.php



ภาพที่ 12.11 ตัวอย่างผลลัพธ์ของไฟล์ delete_process.php ในกรณีที่ลบข้อมูลสำเร็จและไม่สำเร็จ

ข้อผิดพลาดที่อาจพบ

- Fatal error: require_once(): Failed opening required ‘config.php’ หมายถึงไฟล์ config.php ไม่อยู่ในตำแหน่งที่ถูกต้องหรือเว็บเซิร์ฟเวอร์ไม่มีสิทธิ์เข้าถึงไฟล์ แนวทางแก้ไขทำได้โดยตรวจสอบตำแหน่งของไฟล์และกำหนดสิทธิ์การเข้าถึงให้ถูกต้อง
- Call to a member function prepare() on null หมายถึงตัวแปร $pdo ไม่ถูกกำหนดค่า เพราะเกิดข้อผิดพลาดในการโหลดไฟล์ config.php หรือเชื่อมต่อฐานข้อมูลไม่สำเร็จ แนวทางแก้ไขทำได้โดยตรวจสอบการสร้างอ็อบเจกต์ PDO ใน config.php และตรวจสอบการเชื่อมต่อฐานข้อมูลว่าข้อมูล host, username, password ถูกต้องหรือไม่
- PDOException: SQLSTATE[42S02]: Base table or view not found: 1146 Table ‘users’ doesn’t exist หมายถึงไม่มีตาราง users ในฐานข้อมูล แนวทางแก้ไขให้ทำการตรวจสอบให้แน่ใจว่าตาราง users ถูกสร้างในฐานข้อมูลแล้ว และชื่อถูกต้อง
- ไม่พบข้อมูลผู้ใช้ที่ต้องการลบ หมายถึง id ที่ส่งมาไม่ตรงกับข้อมูลใดในตาราง users แนวทางแก้ไขทำได้โดยตรวจสอบค่าพารามิเตอร์ id ใน URL ว่ามีข้อมูลอยู่จริงหรือไม่
- Parse error: syntax error, unexpected ... หมายถึงมีข้อผิดพลาดในโครงสร้างรหัสคำสั่ง PHP เช่น ลืม ; หรือ {} ไม่ครบคู่ แนวทางแก้ไขทำได้โดยตรวจสอบบรรทัดที่เกิดข้อผิดพลาดด้วย editor ที่มีระบบตรวจสอบ syntax
- Headers already sent หมายถึงมีการส่ง output (เช่น echo หรือช่องว่าง) ก่อนการเรียก header() แนวทางแก้ไขทำได้โดยตรวจสอบให้แน่ใจว่าไม่มีการส่ง output ใด ๆ ก่อนเรียกใช้ header() โดยเฉพาะบรรทัดว่างหรือช่องว่างก่อน <?php

4. สรุปท้ายบท

การจัดการข้อมูลจากผู้ใช้ในเว็บแอปพลิเคชันที่พัฒนาด้วยภาษา PHP เป็นกระบวนการสำคัญที่ช่วยให้ระบบสามารถรับ ส่ง และจัดเก็บข้อมูลได้อย่างปลอดภัยและมีประสิทธิภาพ การรับค่าจากฟอร์มโดยใช้ตัวแปรพิเศษ $_GET และ $_POST ทำหน้าที่เป็นกลไกหลักในการถ่ายโอนข้อมูลจากฝั่งผู้ใช้ไปยังฝั่งผู้ให้บริการ (Backend) ซึ่งข้อมูลที่ได้รับจำเป็นต้องผ่านการตรวจสอบและล้างข้อมูลอย่างละเอียดเพื่อป้องกันความเสี่ยงจากข้อมูลที่ไม่ปลอดภัย เช่น การโจมตีแบบ Cross-Site Scripting (XSS) หรือ SQL Injection ในบทนี้ได้อธิบายฟังก์ชันที่จำเป็นสำหรับการตรวจสอบข้อมูล เช่น empty(), isset(), strlen() และ filter_var() รวมถึงฟังก์ชันสำหรับการล้างข้อมูล เช่น trim(), stripslashes() และ htmlspecialchars() เพื่อให้ข้อมูลที่เข้าสู่ระบบมีความถูกต้องและปลอดภัยต่อการประมวลผล

ในการทำงานร่วมกับฐานข้อมูล PHP ใช้เทคโนโลยี PHP Data Objects (PDO) เป็นเครื่องมือมาตรฐานที่ช่วยเชื่อมต่อกับระบบจัดการฐานข้อมูลอย่างมีประสิทธิภาพ หัวข้อในบทนี้ได้กล่าวถึงวิธีการสร้างการเชื่อมต่อกับฐานข้อมูล การเตรียมคำสั่ง SQL ด้วย prepare() การดำเนินการคำสั่งด้วย execute() การดึงข้อมูลด้วย fetch() และ fetchAll() รวมถึงการตรวจสอบจำนวนแถวที่ได้รับผลกระทบด้วย rowCount() โดยใช้โครงสร้าง try...catch เพื่อจัดการข้อผิดพลาดที่อาจเกิดขึ้น ทำให้ระบบมีความเสถียรและปลอดภัยมากขึ้น

การจัดการข้อมูลจากผู้ใช้ในเว็บแอปพลิเคชันที่พัฒนาด้วยภาษา PHP เริ่มจากการรับค่าผ่านตัวแปรพิเศษ $_GET และ $_POST ซึ่งเป็นกลไกสำคัญที่ใช้ในการรับข้อมูลจากฟอร์ม HTML มายังฝั่งผู้ให้บริการ หลังจากได้รับข้อมูลแล้ว ผู้พัฒนาจำเป็นต้องตรวจสอบและล้างข้อมูลที่รับเข้ามา โดยใช้ฟังก์ชันพื้นฐานอย่าง empty(), isset(), strlen(), trim() และ filter_var() เพื่อช่วยลดความเสี่ยงจากข้อมูลที่ไม่สมบูรณ์หรืออาจเป็นอันตรายต่อระบบ

เมื่อข้อมูลผ่านการตรวจสอบแล้ว ระบบจะดำเนินการกับฐานข้อมูลโดยใช้ PHP Data Objects (PDO) ซึ่งเป็นเทคโนโลยีมาตรฐานที่รองรับการทำงานร่วมกับฐานข้อมูลหลายประเภท ภายในบทนี้ได้อธิบายการเชื่อมต่อกับฐานข้อมูล การเตรียมคำสั่ง SQL ด้วย prepare() การดำเนินการด้วย execute() การดึงข้อมูลด้วย fetch() และ fetchAll() ตลอดจนการนับแถวด้วย rowCount() พร้อมทั้งมีตัวอย่างประกอบเพื่อให้เห็นการทำงานที่เป็นระบบ โดยเชื่อมโยงการออกแบบฟอร์มในส่วนผู้ใช้กับการจัดการข้อมูลในฐานข้อมูล

ระบบ CRUD (Create, Read, Update, Delete) เป็นแนวคิดพื้นฐานที่สำคัญในการพัฒนาเว็บแอปพลิเคชันสำหรับการจัดการข้อมูลในระบบฐานข้อมูล โดยกระบวนการ Create หมายถึงการสร้างข้อมูลใหม่และบันทึกเข้าสู่ฐานข้อมูล กระบวนการ Read ใช้สำหรับดึงข้อมูลที่จัดเก็บไว้ออกมาแสดงผลเพื่อให้ผู้ใช้สามารถเรียกดูหรือตรวจสอบได้ กระบวนการ Update คือการแก้ไขหรือปรับปรุงข้อมูลเดิมที่มีอยู่แล้วให้ทันสมัยหรือถูกต้องตามความต้องการ ขณะที่กระบวนการ Delete เป็นการลบข้อมูลที่ไม่ต้องการออกจากระบบอย่างถาวร

5. กิจกรรมท้ายบท

5.1 อธิบายลำดับขั้นตอนของการรับค่าจากฟอร์ม HTML สู่การประมวลผลด้วยภาษา PHP โดยอธิบายการใช้ตัวแปร $_GET และ $_POST ว่ามีความแตกต่างกันอย่างไร พร้อมยกตัวอย่างคำสั่งที่เกี่ยวข้อง

5.2 จงอธิบายความแตกต่างระหว่างการตรวจสอบข้อมูลกับการล้างข้อมูล พร้อมยกตัวอย่างฟังก์ชันที่เหมาะสมในแต่ละกรณี ประกอบด้วย is_numeric(), htmlspecialchars(), trim() และ filter_var()

5.3 ให้อธิบายโครงสร้างคำสั่งในการเชื่อมต่อฐานข้อมูล MySQL ด้วย PDO พร้อมระบุบทบาทของ DSN, ชื่อผู้ใช้, รหัสผ่าน, การกำหนดโหมดแสดงข้อผิดพลาด และข้อดีของการใช้ PDO เมื่อเปรียบเทียบกับการใช้ mysqli โดยตรง

5.4 จงอธิบายความสัมพันธ์ระหว่างแนวคิด CRUD กับคำสั่ง SQL ที่ใช้ในการจัดการข้อมูลในเว็บแอปพลิเคชัน โดยระบุว่า Create, Read, Update และ Delete มีความหมายอย่างไร และสัมพันธ์กับคำสั่ง INSERT, SELECT, UPDATE และ DELETE ของ SQL อย่างไร พร้อมยกตัวอย่างสถานการณ์การใช้งานจากระบบจัดการผู้ใช้ในบทนี้

5.4.1 อธิบายความหมายของ Create และระบุว่าตรงกับคำสั่ง SQL ใดในระบบจัดการผู้ใช้

5.4.2 อธิบายความหมายของ Read และระบุว่าตรงกับคำสั่ง SQL ใดในระบบจัดการผู้ใช้

5.4.3 อธิบายความหมายของ Update และระบุว่าตรงกับคำสั่ง SQL ใดในระบบจัดการผู้ใช้

5.4.4 อธิบายความหมายของ Delete และระบุว่าตรงกับคำสั่ง SQL ใดในระบบจัดการผู้ใช้

5.5 จากโครงสร้างหน้าเว็บของระบบ CRUD เบื้องต้น จงอธิบายหน้าที่ของแต่ละไฟล์ ได้แก่ config.php, index.php, add.php, add_process.php, edit.php, edit_process.php และ delete_process.php และอธิบายกลไกการเชื่อมโยงระหว่างไฟล์เหล่านี้ โดยอ้างอิงจากกรณีศึกษาในบท

5.6 ทดลองเขียนระบบ CRUD สำหรับจัดการข้อมูลสมาชิก โดยใช้ฐานข้อมูล MySQL และภาษา PHP ผ่าน PDO พร้อมแสดงผลข้อมูลในรูปแบบตาราง HTML โดยระบบต้องประกอบด้วยไฟล์ config.php, index.php, add.php, add_process.php, edit.php, edit_process.php และ delete_process.php ดังนี้

5.6.1 สร้างฐานข้อมูลและตาราง members สำหรับเก็บข้อมูลสมาชิก โดยมีฟิลด์ id, name, email, tel และ address

5.6.2 สร้างไฟล์ config.php สำหรับกำหนดค่าการเชื่อมต่อฐานข้อมูล และสร้างอ็อบเจกต์ PDO เพื่อให้ไฟล์อื่นสามารถเรียกใช้งานฐานข้อมูลร่วมกันได้

5.6.3 สร้างไฟล์ index.php เพื่อดึงข้อมูลสมาชิกทั้งหมดจากฐานข้อมูล และแสดงผลในรูปแบบตาราง HTML พร้อมปุ่มหรือลิงก์สำหรับเพิ่ม แก้ไข และลบข้อมูล

5.6.4 สร้างไฟล์ add.php สำหรับแสดงฟอร์มเพิ่มสมาชิกใหม่ และสร้างไฟล์ add_process.php สำหรับรับค่าจากฟอร์ม ตรวจสอบข้อมูลเบื้องต้น และบันทึกข้อมูลด้วย Prepared Statements

5.6.5 สร้างไฟล์ edit.php สำหรับรับค่า id ของสมาชิก ดึงข้อมูลเดิมมาแสดงในฟอร์ม และสร้างไฟล์ edit_process.php สำหรับรับค่าที่แก้ไข ตรวจสอบข้อมูล และอัปเดตข้อมูลด้วย Prepared Statements

5.6.6 สร้างไฟล์ delete_process.php สำหรับรับค่า id ของสมาชิก ตรวจสอบว่ามีข้อมูลอยู่จริง และลบข้อมูลออกจากฐานข้อมูล พร้อมให้มีการยืนยันการลบก่อนดำเนินการ

5.6.7 ทดสอบระบบโดยเพิ่มข้อมูลสมาชิกอย่างน้อย 3 รายการ จากนั้นทดลองแสดงรายการ แก้ไขข้อมูล ลบข้อมูล และตรวจสอบว่าผลลัพธ์ในหน้า index.php เปลี่ยนแปลงถูกต้องตามการทำงานของระบบ

5.7 จงอธิบายความเสี่ยงของการโจมตีแบบ SQL Injection ในเว็บแอปพลิเคชัน โดยยกตัวอย่างรหัสคำสั่ง PHP ที่มีความเสี่ยงและรหัสคำสั่งที่ปรับปรุงให้ปลอดภัยโดยใช้ PDO Prepared Statements


การพัฒนาเว็บแอปพลิเคชัน
บรรณานุกรม
Web Application Development

บรรณานุกรม

จีราวุธ วารินทร์. (2566). โมเดิร์น JavaScript เก่งได้ใน 30 วัน ฉบับ Node.js + MongoDB. นนทบุรี: ไอดีซี.

เจษฎา โพธิ์จันทร์, สุนันท์ ธาติ, ณัฐชยา ปันชุม, และ สุดารัตน์ จันทิมา. (2565). การพัฒนาระบบสารสนเทศเพื่อการบริหารองค์การโดยใช้หลักการจัดการการไหลของสารสนเทศโลจิสติกส์ กรณีศึกษาคณะโลจิสติกส์และดิจิทัลซัพพลายเชน มหาวิทยาลัยนเรศวรกรณีศึกษาคณะโลจิสติกส์และดิจิทัลซัพพลายเชน มหาวิทยาลัยนเรศวร. *วารสารวิชาการเทคโนโลยีอุตสาหกรรมและวิศวกรรม มหาวิทยาลัยราชภัฏพิบูลสงคราม, 4*(2).

ฐานิตย์ อริยะกอง. (2568). การพัฒนาระบบสารสนเทศโรงพยาบาลสัตว์ โดย Power BI Desktop คณะสัตวแพทยศาสตร์ มหาวิทยาลัยขอนแก่น. *วารสารวิจัยศิลปวิทยาการลุ่มน้ำโขง, 33*(1), 1-17.

ณัฐโชติ พรหมฤกษ์. (2563). *Practical DevOps and Cloud Engineering.* นนทบุรี: ไอดีซีฯ.

ณัฐวุฒิ ทะนันไธสง, และ ณัฐพล วงษ์รัมย์. (2566). การพัฒนาเว็บแอปพลิเคชันระบบสารสนเทศภูมิศาสตร์สำหรับกลุ่มข้าวภูเขาไฟบ้านโคกเมือง ตำบลจรเข้มาก อำเภอประโคนชัย จังหวัดบุรีรัมย์. *สักทอง: วารสารวิทยาศาสตร์และเทคโนโลยี (สทวท.), 10*(1), 51-62.

ธนัชชา วีรวัฒนโยธิน. (13 มกราคม 2566). *LAMP คืออะไร ชุดซอฟต์แวร์พัฒนาเว็บไซต์ยอดนิยม.* เข้าถึงได้จาก โอเพ่นแลนด์สเคป: https://blog.openlandscape.cloud/what-is-lamp

ปภังกร อิ่นแก้ว. (2566). *วิทยาการข้อมูลเบื้องต้น.* เชียงใหม่: มหาวิทยาลัยเชียงใหม่.

ปานใจ ธารทัศนวงศ์. (2565). *Fundamental of Software Engineering&Digital Transformation.* นนทบุรี: ไอดีซีฯ.

พรรณี สวนเพลง. (2566). *ระบบสารสนเทศสำหรับการท่องเที่ยว.* กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

วันชนะ จูบรรจง, วิไรวรรณ แสนชะนะ, รุ่ง หมูล้อม, และ อภิชัย ซื่อสัตย์สกุลชัย. (2564). การส่งเสริมการท่องเที่ยวจังหวัดตากโดยใช้เทคโนโลยีเสมือนจริง. *วารสารวิทยาการและเทคโนโลยีสารสนเทศ (JIST), 11*(1), 56-64.

ศุภชัย สมพานิช. (2565). พัฒนา Web Application ด้วย JavaScript และ Node.js. กรุงเทพฯ: อินโฟเพรส.

ศุภนิดา โคตรชาดา, และ ดารณี เอื้อชนะจิต. (2566). คุณลักษณะของระบบสารสนเทศที่ดีและการยอมรับการใช้เทคโนโลยีของระบบสารสนเทศส่งผลต่อความสำเร็จของระบบบัญชีคอมพิวเตอร์ (e-LAAS) ขององค์กรปกครองส่วนท้องถิ่นในเขตปริมณฑล. *วารสารรัฐประศาสนศาสตร์ มหาวิทยาลัยราชภัฏสวนสุนันทา*, 422-435.

สาวิตรี จูเจี่ย, และ สุรศักดิ์ มังสิงห์. (2559). แนวคิดพื้นฐานของตัวชี้วัดคุณภาพสารสนเทศ. *Naresuan University Journal: Science and Technology, 24*(3), 1-13.

สุพรรณิการ์ ย่องซื่อ. (2563). *การพัฒนาเว็บไซต์เบื้องต้นด้วยภาษา HTML5 และ CSS.* กรุงเทพฯ: สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย.

สุวรรณี อัศวกุลชัย. (2561). *วิศวกรรมซอฟต์แวร์.* สำนักพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย.

อนันตพร อรุณฉาย, และ ฆัมภิชา ตันติสันติสม. (2567). ระบบสารสนเทศจำหน่ายน้ำดื่มบ้านเพชรไพรวัลย์. *สักทอง: วารสารวิทยาศาสตร์และเทคโนโลยี (สทวท.), 11*(2), 76-87.

อนุวัฒน์ พัฒนเชียร, และ อรญา สุวรรณโณ. (2566). การพัฒนาระบบสารสนเทศสำหรับการบริหารจัดการโครงการ คณะวิทยาการจัดการ มหาวิทยาลัยสงขลานครินทร์. *Journal of Information and Learning [JIL], 34*(3), 104-117.

อมีนา ฉายสุวรรณ, และ ชุมพล จันทร์ฉลอง. (2566). การพัฒนาเว็บแอปพลิเคชันพจนานุกรมคำศัพท์คอมพิวเตอร์และเทคโนโลยีสารสนเทศ. *วารสารวิจัยและพัฒนา วไลยอลงกรณ์ ในพระบรมราชูปถัมภ์, 18*(2), 29-44.

อรสา เตติวัฒน์. (2567). *ระบบสารสนเทศและองค์กร* (พิมพ์ครั้งที่ 2). นครสวรรค์: คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏนครสวรรค์.

อเล็กซ์ แทปส์ค็อต. (2567). *เว็บ 3 - Web 3 (จิรายุส ทรัพย์ศรีโสภา, ผู้แปล).* กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

โอภาส เอี่ยมสิริวงศ์. (2566). *การวิเคราะห์และออกแบบระบบ (ฉบับปรับปรุงเพิ่มเติม).* กรุงเทพฯ: ซีเอ็ดยูเคชั่น.

Al-Okaily, M. (2024). Assessing the effectiveness of accounting information systems in the era of COVID-19 pandemic. *VINE Journal of Information and Knowledge Management Systems, 54*(1), 157-175.

Amazon Web Services, Inc. (2023). *What is a web application?* เข้าถึงได้จาก Amazon Web Services (AWS): https://aws.amazon.com/what-is/web-application/

Amazon Web Services. (n.d.). *What is SDLC? - Software Development Lifecycle Explained*. Retrieved from AWS: https://aws.amazon.com/what-is/sdlc/?nc1=h_ls

Aziz, A. A., Haizan Nor, R. N., Jusoh, Y. Y., Wan Ab. Rahman, W. N., & Mohd Ali, N. (2024). Factors influencing information quality of information systems: A systematic literature review. *International Journal on Informatics Visualization, 8*(3-2), 1923-1931.

Berners-Lee, T. (n.d.). *The World Wide Web: Past, Present and Future*. Retrieved June 16, 2024, from W3: https://www.w3.org/People/Berners-Lee/1996/ppf.html

Bootstrap. (n.d.a). *About Bootstrap*. Retrieved June 1, 2024, from Bootstrap v5.3: https://getbootstrap.com/docs/5.3/about/overview/

Bootstrap. (n.d.b). *Get started with Bootstrap*. Retrieved June 1, 2024, from Bootstrap v5.3: https://getbootstrap.com/docs/5.3/getting-started/introduction/

Dennis, A., Wixom, B. H., & Roth, R. M. (2021). *Systems Analysis and Design* (8th ed.). Wiley.

Deno Land Inc. (2025, May 22). A brief history of JavaScript. Retrieved from Deno: https://deno.com/blog/history-of-javascript

DigitalOcean. (2024, July 23). *What is Containerization?* Retrieved from DigitalOcean: https://www.digitalocean.com/resources/articles/what-is-containerization

Doanh, N. K., Quynh, N. N., & Pham, T. L. (2022). Going organic or staying traditionalistic? The role of agriculture information system. *International Journal of Social Economics, 49*(10), 1458-1478.

Docker Inc. (n.d.a). *Docker Desktop*. Retrieved December 30, 2024, from Docker Docs: https://docs.docker.com/desktop/

Docker Inc. (n.d.b). *Multi-container applications*. Retrieved December 30, 2024, from Docker Docs: https://docs.docker.com/get-started/docker-concepts/running-containers/multi-container-applications/

Docker Inc. (n.d.c). *What is Docker?* Retrieved December 30, 2024, from Docker Docs: https://docs.docker.com/get-started/docker-overview/

Faradilla, A. (2025, July 27). *Web apps vs. websites: What are the key differences?* Retrieved from Hostinger: https://www.hostinger.com/tutorials/web-app-vs-website

Griffin, J. (2022, December 12). *HTML Charset Attribute*. Retrieved from The Webmaster: https://www.thewebmaster.com/html/attributes/charset/

Hamdat, A., Ceskakusumadewi, B., Samalam, A. G., Rizal, M., & Lawalata, I. L. (2024). The Impact of Management Information Systems on Decision-Making Efficiency. *Vifada Management and Digital Business, 1*(2), 56-74.

Hera, A., Al Rian, A., Faruque, M. O., Sizan, M. H., Khan, N. A., Rahaman, M. A., & Ali, M. J. (2024). Leveraging Information Systems for Strategic Management: Enhancing Decision-Making and Organizational Performance. *American Journal of Industrial and Business Management, 14*(8), 1045-1061.

IBM Cloud Education Team. (2022, June 6). *LAMP vs. MEAN: What's the Difference?* Retrieved from IBM: https://www.ibm.com/think/topics/lamp-vs-mean

IBM. (2023, November 1). *What is a virtual server?* Retrieved from IBM: https://www.ibm.com/think/topics/virtual-server

IBM. (2025, January 22). *Web application environment*. Retrieved from IBM Documentation: https://www.ibm.com/docs/en/db2-for-zos/13.0.0?topic=zos-web-application-environment

Ikechukwu, V. (2024, February 13). *The Best CSS Frameworks to Use in Your Projects*. Retrieved from freeCodeCamp.org: https://www.freecodecamp.org/news/best-css-frameworks-for-frontend-devs/#heading-types-of-css-frameworks

Ilya, G. (2014a, March 31). *Constructing the Object Model*. Retrieved from Web.Dev: https://web.dev/articles/critical-rendering-path/constructing-the-object-model#document_object_model_dom

Ilya, G. (2014b, March 31). *Render-tree Construction, Layout, and Paint*. Retrieved from Web.Dev: https://web.dev/articles/critical-rendering-path/render-tree-construction

Jayaprakash, A. (2024, September 16). *Docker Architecture: The components and processes - Part 1*. Retrieved from Blacksmith: https://www.blacksmith.sh/blog/docker-architecture-the-components-and-processes-part-1

Jones, E. (2023, May 30). *6 Pillars of Data Quality and How to Improve Your Data*. Retrieved from IBM: https://www.ibm.com/products/tutorials/6-pillars-of-data-quality-and-how-to-improve-your-data

Kendall, K. E., & Kendall, J. E. (2014). *Systems Analysis and Design* (9th ed.). Pearson.

Kirvan, P., Semilof, M., & Montgomery, J. (2024, September 11). What is a software stack? Retrieved from TechTarget: https://www.techtarget.com/searchapparchitecture/definition/software-stack

Kosaka, M. (2020, September 17). *Inside look at modern web browsers (part 1)* . Retrieved from Chrome Developers : https://developer.chrome.com/blog/inside-browser-part1#browser-architecture

Microsoft. (2025, January 22). *Windows and containers*. Retrieved from Microsoft Learns: https://learn.microsoft.com/en-us/virtualization/windowscontainers/about/

Monnikendam, M. (2020, November 30). *What Are Web Application Environments?* Retrieved from Lizard Global: https://www.lizard.global/blog/what-are-environments-in-web-and-app-development

Mozilla Contributors. (n.d.). *World Wide Web*. Retrieved February 20, 2025, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Glossary/World_Wide_Web

Mozilla Developer Network . (2025, February 25). *Critical rendering path – Web Performance Guide* . Retrieved from MDN Web Docs : https://developer.mozilla.org/en-US/docs/Web/Performance/Guides/Critical_rendering_path

Mozilla Developer Network. (2024, June 23). *CSS Color Values*. Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_colors/Color_values

Mozilla Developer Network. (2024, May 20). *<h1>–<h6>: The HTML Section Heading elements*. Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML/Reference/Elements/Heading_Elements

Mozilla Developer Network. (2025, April 10). *JavaScript technologies overview* . Retrieved from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/JavaScript_technologies_overview#javascript_implementations

Mozilla Developer Network. (2025, April 12). *Introduction to the server side*. Retrieved from MDN web docs: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/First_steps/Introduction

Mozilla Developer Network. (n.d.a). *Markup*. Retrieved December 30, 2024, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Glossary/Markup

Mozilla Developer Network. (n.d.b). *Understanding quirks and standards modes*. Retrieved December 30, 2024, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Web/HTML/Guides/Quirks_mode_and_standards_mode

Mozilla Developer Network. (n.d.c). *What is a URL?* Retrieved February 21, 2025, from MDN Web Docs: https://developer.mozilla.org/en-US/docs/Learn_web_development/Howto/Web_mechanics/What_is_a_URL

Nair, A. (2024, October 30). *Containers vs Virtual Machines: A Detailed Comparison for Developers*. Retrieved from DataCamp: https://www.datacamp.com/blog/containers-vs-virtual-machines

Nutanix. (2024, August 8). *What is Containerisation?* Retrieved from Nutanix UK: https://www.nutanix.com/en_gb/info/what-is-containerization

Pata , P., Manurung, B. A., Wardhana, E. H., Budiman, M., & Nataliawati. (2024, September). The Impact of Information Quality, Trust, Anxiety, and Health Consciousness on Intention to Buy Health Supplements in The Post Pandemic Era. *International Journal of Social and Management Studies, 6*(9), 3959-3967.

PHP-FIG. (n.d.). *PHP Standard Recommendations*. Retrieved October 30, 2024, from PHP-FIG: https://www.php-fig.org/psr/

Qudah, S. (2024, August 27). *Understanding Utility-First CSS (like Tailwind) vs. Component-Based CSS (like Bootstrap): Patterns, Advantages, and Teamwork Implications*. Retrieved from LinkedIn: https://www.linkedin.com/pulse/understanding-utility-first-css-like-tailwind-vs-bootstrap-qudah-21g9f

Ramotion. (2024a, March 4). *Web Application vs. Desktop Application: A Comparative Guide*. Retrieved from Web Design, UI/UX, Branding, and App Development Blog: https://www.ramotion.com/blog/web-application-vs-desktop-application

Ramotion. (2024b, July 3). *What is a web browser: Definition, types, and features*. Retrieved from https://www.ramotion.com/blog/what-is-web-browser/

Randal, A. (2020). The Ideal Versus the Real: Revisiting the History of Virtual Machines and Containers. *ACM Computing Surveys (CSUR), 53*(1), 1-31.

Red Hat. (2024, December 9). *What is virtualization?* Retrieved from Red Hat: https://www.redhat.com/en/topics/virtualization/what-is-virtualization

Roch, S., Fowler, J., Smith, B., & Bourgeois, D. (2022, February 24). Information Systems for Business and Beyond. Ontario, London.

Ross, J. (2025, March 10). *Chapter 9 CSS Frameworks*. Retrieved from Client-Side Web Development: https://info340.github.io/css-frameworks.html

Sadriddinovich, J. T., & Muhiddinovna, M. M. (2024). Web programming information. Sustainability of Education, Socio-Economic Science Theory, 2(19), 232–234.

Sheposh, R. (2024). *Garbage in, garbage out (GIGO)*. Retrieved from EBSCO: https://www.ebsco.com/research-starters/computer-science/garbage-garbage-out-gigo

Solida, A., & Ardiyansyah. (2024). Information Quality and Decisions of Online Drivers to Participate in BPJS Health Insurance. *International Journal of Multidisciplinary Research and Studies, 3*(1), 88-94.

Stack Overflow. (2024). *Technology*. Retrieved from 2024 Developer Survey: https://survey.stackoverflow.co/2024/technology#most-popular-technologies

Stoneman, E. (2020). *Learn Docker in a Month of Lunches.* New York: Manning.

Sturm, R., Pollard, C., & Craig, J. (2017). Chapter 7 - Managing Web-Based Applications. In *Application Performance Management (APM) in the Digital Enterprise* (pp. 83-93). Boston: Morgan Kaufmann.

Susnjara, S., & Smalley, I. (2024, May 20). *What is containerization?* Retrieved from IBM: https://www.ibm.com/think/topics/containerization

Susnjara, S., & Smalley, I. (2025, April 9). *What is virtualization?* Retrieved from IBM: https://www.ibm.com/think/topics/virtualization

W3C. (2024, April 11). *CSS Box Model Module Level 3*. Retrieved from https://www.w3.org/TR/css-box-3/

W3C. (2025, February 25). *CSS Snapshot 2024*. Retrieved from W3C Group Note: https://www.w3.org/TR/css/

W3Schools. (n.d.). *Responsive Web Design - The Viewport*. Retrieved December 30, 2024, from W3Schools: https://www.w3schools.com/css/css_rwd_viewport.asp

Web.Dev. (2024, March 31). *Objects*. Retrieved from Web.Dev: https://web.dev/learn/javascript/objects

WHATWG. (2024, June 16). *Introduction*. Retrieved from HTML Living Standard: https://html.spec.whatwg.org/multipage/introduction.html

WHATWG. (n.d.). *DOM Standard – Introduction to the DOM*. Retrieved from WHATWG: https://dom.spec.whatwg.org/#introduction-to-the-dom

World Wide Web Consortium. (2009, August 31). *Web Application Description Language (WADL) Submission* . Retrieved from W3C : https://www.w3.org/submissions/wadl/#x3-20001.1


การพัฒนาเว็บแอปพลิเคชัน
ดัชนีค้นคำ
Web Application Development

ดัชนี

ก

การพัฒนาเว็บแอปพลิเคชันส่วนหน้า, 22
การพัฒนาเว็บแอปพลิเคชันส่วนหลัง, 22

ค

เครื่องผู้ให้บริการเว็บ, 5
โครงสร้างต้นไม้ CSSOM, 14
โครงสร้างต้นไม้ DOM, 13
โครงสร้างต้นไม้การแสดงผล, 15

จ

จุดยึด. ดู *Anchor*

ช

ชื่อโดเมน. ดู *Domain Name*

ท

เทคโนโลยีสารสนเทศ, 35
แท็ก, 124
    แอตทริบิวต์, 125

น

แนวคิดการจำลองเสมือน, 82
    เทคโนโลยีคอนเทนเนอร์, 86
    เทคโนโลยีเครื่องเสมือน, 82

บ

    แบบจำลองกล่อง, 16

ร

ระบบ, 26
    องค์ประกอบของระบบ, 26
ระบบ CRUD, 443
ระบบสารสนเทศ, 32

ว

วงจรการพัฒนาระบบ, 44
เว็บ, 4
    องค์ประกอบ, 4
เว็บไซต์, 5
เว็บเบราว์เซอร์, 5
    ส่วนการแยกวิเคราะห์, 13
    ส่วนการแยกวิเคราะห์ CSS, 14
    ส่วนประกอบ, 12
เว็บแอปพลิเคชัน, 9

ส

สภาพแวดล้อมการใช้งานจริง, 65
สภาพแวดล้อมการพัฒนาเว็บแอปพลิเคชัน, 64
สภาพแวดล้อมเว็บแอปพลิเคชัน, 64
ส่วนประกอบพื้นฐานของเครื่องผู้ให้บริการเว็บ, 18
สารสนเทศ, 28
    องค์ประกอบของสารสนเทศ, 28

ห

หน้าต่างมุมมอง, 136
หน้าเว็บ, 4

อ

อิลิเมนต์, 127
เอกสารเว็บ, 4

ฮ

โฮมเพจ, 5

B

Back-end Development, 22

C

CRUD, 443

CSS, 188

- Bootstrap, 250
- CSS Framework, 249
- Flexbox, 238
- Grid, 243
- Selector, 195

D

Development Environment, 64

Docker, 97

- Docker Container, 101
- Docker Daemon, 101
- Docker Desktop, 97
- Docker Image, 101
- Docker Registry, 104

E

ECMAScript, 271

F

Front-end Development, 22

H

Homepage, 5

HTML, 122

- Element, 127

HTML Living Standard, 123

HyperText Markup Language. *See* HTML

I

Information, 28

Information System, 32

Information Technology, 35

J

JavaScript, 270

- Array, 339
- Function, 318
- Object, 324

L

LAMP Stack, 71

LEMP Stack, 71

M

MAMP Stack, 71

MEAN Stack, 71

P

PDO. *See* PHP Data Objects

PHP, 378

- Array, 410
- Function, 405

PHP Data Objects, 434

PHP Standard Recommendation, 380

Production Environment, 65

PSR. *See* PHP Standard Recommendation

S

System, 26

System Development Life Cycle, 44

T

Tag, 124

    Attribute, 125

U

Uniform Resource Locator. *See* URL

URL, 5

    Anchor, 8

    Domain Name, 6

    Parameters, 8

    Path to File, 7

    Port, 6

    Scheme, 6

V

Viewport, 136

Virtualization Approach, 82

    Container Technology, 86

    Virtual Machine Technology, 82

W

WAMP Stack, 71

Web, 4

Web Application, 9

Web Application Environment, 64

Web Browser, 5

Web Document, 4

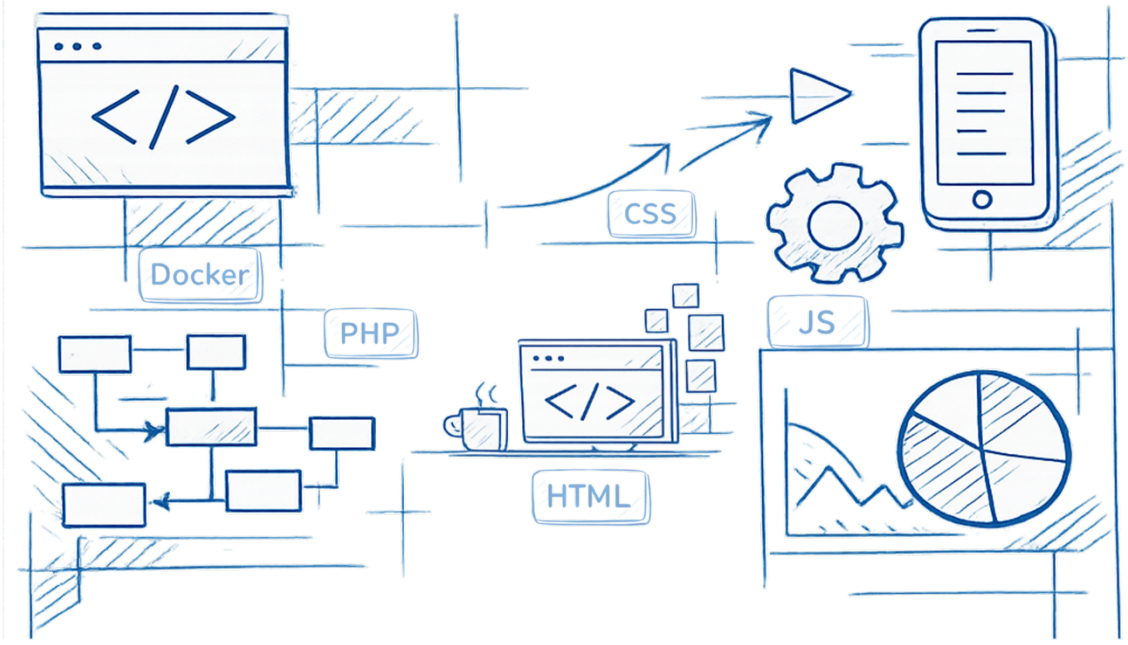
Web page, 4

Web Server, 5

Website, 5

การพัฒนาเว็บแอปพลิเคชัน

แนวคิดและหลักการ...สู่การปฏิบัติสำหรับผู้เริ่มต้น



ISBN 978-974-388-407-8

มหาวิทยาลัยราชภัฏกำแพงเพชร แม่สอด

ราคา 300 บาท