PHP Web Programming: Beginner

(Integrative-Generative AI Edition)

CONTENTS

Student Price Book Center

คำนำ

ในยุคดิจิทัลที่เว็บไซต์กลายเป็นเครื่องมือหลักของการสื่อสาร การบริการ และธุรกิจ การพัฒนาเว็บแอปพลิเคชันกลายเป็นทักษะที่จำเป็นสำหรับนักพัฒนา โปรแกรมเมอร์ และแม้แต่นักเรียนที่ต้องการเข้าสู่สายงานเทคโนโลยีสารสนเทศ ภาษา PHP (Hypertext Preprocessor) เป็นหนึ่งในภาษาที่ได้รับความนิยมและถูกใช้อย่างแพร่หลายในการพัฒนาเว็บไซต์ โดยเฉพาะอย่างยิ่งเว็บไซต์ที่ต้องมีการเชื่อมโยงกับฐานข้อมูลหรือระบบหลังบ้าน ด้วยเหตุนี้ หนังสือ *PHP Web Programming: Beginner* เล่มนี้จึงถูกเขียนขึ้นมาเพื่อเป็นแนวทางที่เป็นระบบและเข้าใจง่าย สำหรับผู้ที่เริ่มต้นเรียนรู้การเขียนโปรแกรมเว็บด้วยภาษา PHP

PHP เป็นภาษา Script ที่ทำงานฝั่งเซิร์ฟเวอร์ (Server-side scripting language) ซึ่งถูกพัฒนาขึ้นครั้งแรกเมื่อปี ค.ศ. 1995 และมีการพัฒนาต่อเนื่องจนกลายเป็นเครื่องมือมาตรฐานในโลกของเว็บแอปพลิเคชัน ด้วยโครงสร้างที่ยืดหยุ่น เรียนรู้ง่าย และสามารถเชื่อมโยงกับระบบฐานข้อมูลอย่าง MySQL ได้อย่างมีประสิทธิภาพ ทำให้ PHP ยังคงได้รับความนิยมอย่างต่อเนื่อง แม้ในปัจจุบันจะมีภาษาอื่น ๆ เช่น Python, Node.js หรือ Ruby เข้ามาแข่งขัน แต่ PHP ก็ยังคงมีฐานผู้ใช้ที่มั่นคง รวมถึง Framework ชั้นนำอย่าง Laravel, Symfony และ CMS อย่าง WordPress ที่ช่วยขับเคลื่อนเว็บนับล้านทั่วโลก

หนังสือเล่มนี้ถูกออกแบบมาอย่างเป็นระบบ เพื่อให้ผู้เรียนสามารถเริ่มต้นจากศูนย์ จนเข้าใจการทำงานของภาษา PHP ในระดับพื้นฐานได้อย่างลึกซึ้ง โดยเนื้อหาในแต่ละบทเรียงลำดับจากง่ายไปยาก พร้อมตัวอย่างที่สามารถทดลองได้จริง และแนะนำแนวทางการพัฒนาทักษะอย่างมีขั้นตอน

บทที่ 1 จะเริ่มต้นด้วยการปูพื้นฐานเกี่ยวกับ PHP และการตั้งค่าสิ่งแวดล้อมสำหรับการพัฒนา ไม่ว่าจะเป็นการติดตั้ง PHP ผ่านแพ็กเกจยอดนิยมอย่าง XAMPP, MAMP หรือ LAMP การตั้งค่าเว็บเซิร์ฟเวอร์ การสร้างไฟล์ PHP แรก และการใช้งาน PHP บนเบราว์เซอร์ พร้อมทั้งทำความเข้าใจโครงสร้างภาษา (syntax) และการทำงานของ PHP แบบ Server-side ซึ่งเป็นหัวใจของการเขียนเว็บแอปพลิเคชัน

บทที่ 2 จะพาผู้อ่านไปรู้จักกับตัวแปรและประเภทข้อมูลพื้นฐานใน PHP ซึ่งเป็นสิ่งสำคัญที่สุดของทุกภาษาการเขียนโปรแกรม ไม่ว่าจะเป็นตัวเลข สตริง ค่า boolean หรือ array พร้อมแสดงวิธีการกำหนดค่า การแสดงผล และการจัดการข้อมูลเบื้องต้น

บทที่ 3 มุ่งเน้นที่คำสั่งควบคุมโครงสร้างของโปรแกรม เช่น คำสั่งเงื่อนไข (if, else, switch) คำสั่งลูป (for, while, foreach) และการสร้างฟังก์ชันแบบง่ายๆ ซึ่งเป็นพื้นฐานสำคัญในการเขียนโปรแกรมที่สามารถควบคุมการทำงานตามตรรกะของผู้พัฒนา รวมถึงการจัดการข้อผิดพลาดอย่างง่าย ซึ่งจะช่วยให้ผู้อ่านสามารถพัฒนาโปรแกรมที่มีความเสถียรและปลอดภัยยิ่งขึ้น

บทที่ 4 กล่าวถึงการทำงานกับฟอร์มและการรับข้อมูลจากผู้ใช้ ซึ่งถือเป็นแกนกลางของเว็บแอปพลิเคชันในปัจจุบัน ผู้อ่านจะได้เรียนรู้การรับค่าจากฟอร์มผ่าน GET และ POST รวมถึงการใช้งานตัว

แปรพิเศษ (Superglobals) เช่น $_GET, $_POST, และ $_REQUEST ตลอดจนเทคนิคการตรวจสอบข้อมูลที่ได้รับมาจากผู้ใช้ ซึ่งเป็นจุดเริ่มต้นของการพัฒนาเว็บแบบโต้ตอบ (interactive web)

สิ่งที่ทำให้หนังสือเล่มนี้มีความโดดเด่น คือการนำเสนอเนื้อหาแบบ “เชิงลึก” ในแต่ละหัวข้อ เพื่อให้ผู้อ่านเข้าใจกลไกการทำงานจริง พร้อมทั้งมีการแนะนำเครื่องมือพัฒนา เช่น NetBeans IDE ซึ่งช่วยอำนวยความสะดวกในการพัฒนาโปรแกรม PHP ได้อย่างมืออาชีพ อีกทั้งยังมีตัวอย่างการบูรณาการความรู้เพื่อฝึกคิดเชิงวิเคราะห์และสามารถต่อยอดสู่ระดับที่สูงขึ้นได้

ผู้เขียนเชื่อมั่นว่า หนังสือ *PHP Web Programming: Beginner* เล่มนี้ จะเป็นจุดเริ่มต้นสำคัญสำหรับผู้อ่านทุกคน ไม่ว่าจะเป็นนักเรียน นิสิต นักศึกษา หรือบุคคลทั่วไปที่สนใจพัฒนาเว็บด้วย PHP โดยมีเป้าหมายให้ผู้อ่านสามารถเรียนรู้ได้ด้วยตนเอง และต่อยอดไปสู่ระดับกลางและระดับสูงต่อไปได้อย่างมั่นใจ

ขอขอบคุณผู้อ่านทุกท่านที่ให้เกียรติเลือกหนังสือเล่มนี้เป็นส่วนหนึ่งในการเรียนรู้ หากมีข้อเสนอแนะหรือคำติชม ผู้เขียนยินดีน้อมรับเพื่อนำไปปรับปรุงในฉบับถัดไปอย่างเต็มใจ

ด้วยรักและปรารถนาดี

ศูนย์หนังสือราคานักเรียน

สารบัญ

หน้า

บทที่ 1
รู้จักกับ PHP และการตั้งค่าสิ่งแวดล้อม
(Introduction to PHP and Environment Setting)

เนื้อหา

บทนำ

ในยุคที่เทคโนโลยีเว็บแอปพลิเคชันกลายเป็นหัวใจสำคัญของธุรกิจ การเข้าใจและใช้งานภาษา PHP อย่างมีประสิทธิภาพคือก้าวแรกที่สำคัญสำหรับนักพัฒนาเว็บทุกระดับ PHP หรือ “PHP: Hypertext Preprocessor” เป็นภาษาโปรแกรมฝั่งเซิร์ฟเวอร์ที่ได้รับความนิยมอย่างแพร่หลายในการสร้างเว็บไซต์แบบไดนามิก ด้วยโครงสร้างที่ยืดหยุ่น เขียนง่าย และรองรับการทำงานร่วมกับ HTML ได้อย่างกลมกลืน ทำให้ PHP เป็นภาษาหลักที่ใช้พัฒนาเว็บไซต์และระบบจัดการเนื้อหาชั้นนำ เช่น WordPress, Drupal และ Joomla

บทที่ 1 ของหนังสือเล่มนี้จะพาผู้อ่านเริ่มต้นจากจุดเริ่มต้นที่สำคัญที่สุด คือ การทำความเข้าใจว่า PHP คืออะไร และทำงานอย่างไรในบริบทของระบบฝั่งเซิร์ฟเวอร์ ผู้อ่านจะได้เรียนรู้แนวคิดของ server-side scripting ว่าทำไม PHP จึงสามารถตอบสนองต่อคำขอจากผู้ใช้ผ่านเว็บเบราว์เซอร์ได้อย่างยืดหยุ่น และมีประสิทธิภาพ

นอกจากแนวคิดเชิงทฤษฎีแล้ว ผู้อ่านยังจะได้เรียนรู้วิธีการติดตั้งสภาพแวดล้อมสำหรับการพัฒนา PHP ด้วยเครื่องมือยอดนิยมอย่าง XAMPP (สำหรับ Windows), MAMP (สำหรับ macOS) และ

LAMP (สำหรับ Linux) การตั้งค่าสิ่งแวดล้อมให้พร้อมใช้งานคือพื้นฐานสำคัญที่ช่วยให้สามารถพัฒนาและทดสอบโค้ดได้อย่างมีประสิทธิภาพในเครื่องของตนเอง

เมื่อเครื่องมือพร้อมใช้งาน บทเรียนจะต่อยอดสู่การเชื่อมโยง PHP เข้ากับเว็บเซิร์ฟเวอร์ที่เป็นที่นิยม เช่น Apache และ Nginx โดยอธิบายโครงสร้างการให้บริการของเซิร์ฟเวอร์ การจัดการพอร์ต และการตอบสนอง HTTP Request อย่างเป็นระบบ เพื่อให้ผู้อ่านเข้าใจภาพรวมของการทำงานแบบ Full-stack ได้อย่างชัดเจน

จากนั้น ผู้อ่านจะได้ลองสร้างไฟล์ PHP แรกและทดสอบการทำงานผ่านเว็บเบราว์เซอร์จริง ซึ่งเป็นประสบการณ์สำคัญที่จะปลดล็อกความรู้เชิงปฏิบัติ ทำให้เห็นผลลัพธ์การทำงานแบบไดนามิก เช่น การแสดงวันที่อัตโนมัติ การคำนวณ หรือการแสดงข้อความจากตัวแปร ที่เกิดจากโค้ด PHP ฝั่งเซิร์ฟเวอร์

ท้ายที่สุด บทนี้จะสรุปด้วยการปูพื้นฐานทางไวยากรณ์ (Syntax) ที่จำเป็นของ PHP ไม่ว่าจะเป็นการประกาศตัวแปร การใช้คำสั่ง echo, การแทรก PHP ลงใน HTML และการใช้คอมเมนต์ ผู้อ่านจะได้เรียนรู้สไตล์การเขียนโค้ดที่ชัดเจนและสามารถขยายต่อไปยังระดับที่ซับซ้อนมากขึ้นได้อย่างมั่นคง

บทนำนี้จึงไม่ใช่เพียงแค่การแนะนำภาษา PHP แต่คือการวางรากฐานที่แข็งแรงสำหรับเส้นทางของนักพัฒนาเว็บ ด้วยความเข้าใจทั้งเชิงแนวคิดและการปฏิบัติที่ผสมผสานกันอย่างลงตัว ผู้อ่านจะพร้อมก้าวเข้าสู่บทถัดไปได้อย่างมั่นใจและเป็นระบบ.

รู้จักกับ PHP และการตั้งค่าสิ่งแวดล้อม

หัวข้อที่ 1: PHP คืออะไร และทำงานอย่างไร (Server-side Scripting)

☐ PHP คืออะไร?

PHP (ย่อมาจาก *PHP: Hypertext Preprocessor*) คือภาษาโปรแกรมประเภท *Server-side Scripting* ที่ใช้สำหรับพัฒนาเว็บไซต์และเว็บแอปพลิเคชัน ซึ่งรันบนเซิร์ฟเวอร์ก่อนจะส่งผลลัพธ์ (HTML) กลับไปยังเว็บเบราว์เซอร์ของผู้ใช้

☐ จุดเด่นของ PHP

- ใช้งานง่าย เรียนรู้เร็ว
- ประสิทธิภาพดีบนเว็บ
- ทำงานร่วมกับ HTML ได้อย่างแนบเนียน
- รองรับฐานข้อมูลหลากหลาย เช่น MySQL, PostgreSQL
- มี Framework ยอดนิยม เช่น Laravel, Symfony

☐ การทำงานของ PHP (Request ⟶ Response)

1. ผู้ใช้เรียกหน้าเว็บผ่านเบราว์เซอร์ (เช่น http://localhost/index.php)
2. เว็บเซิร์ฟเวอร์ (Apache/Nginx) จะร้องขอให้ PHP interpreter ประมวลผลไฟล์ .php
3. PHP จะประมวลผลโค้ดฝั่งเซิร์ฟเวอร์ แล้วสร้าง HTML ออกมา

4. HTML ผลลัพธ์จะถูกส่งกลับไปยังเบราว์เซอร์ของผู้ใช้

☐ *ข้อสังเกต:* เบราว์เซอร์ **จะไม่เห็นโค้ด PHP เลย** เห็นแค่ผลลัพธ์ที่ PHP รันแล้วเท่านั้น

หัวข้อที่ 2: การติดตั้ง PHP (XAMPP, MAMP, LAMP)

☐ **วิธีติดตั้ง PHP ที่ง่ายที่สุดสำหรับผู้เริ่มต้นคือใช้ แพ็กเกจรวม (Bundle) เช่น:**

☐ **XAMPP (Windows/macOS/Linux)**

- ประกอบด้วย Apache, MySQL, PHP, phpMyAdmin
- เหมาะสำหรับผู้เริ่มต้นที่สุด
- ดาวน์โหลดได้จาก: https://www.apachefriends.org/

☐ **MAMP (สำหรับ macOS และ Windows)**

- เหมาะกับนักพัฒนา Mac
- รวม Apache, PHP, MySQL พร้อม GUI
- ดาวน์โหลด: https://www.mamp.info/

☐ **LAMP (Linux + Apache + MySQL + PHP)**

- เหมาะสำหรับระบบ Linux
- ใช้คำสั่งใน terminal เช่น:
- sudo apt update
- sudo apt install apache2 php mysql-server libapache2-mod-php

☐ **วิธีตรวจสอบว่า PHP ติดตั้งแล้วหรือไม่**

เปิด terminal / command prompt แล้วพิมพ์:

php -v

หากเห็นเวอร์ชัน เช่น PHP 8.3.1 แปลว่าพร้อมใช้งานแล้ว

หัวข้อที่ 3: การใช้งาน PHP กับเว็บเซิร์ฟเวอร์ (Apache, Nginx)

☐ **การทำงานร่วมกับ Apache (ผ่าน XAMPP/MAMP)**

- ไฟล์ .php ต้องอยู่ในโฟลเดอร์ที่ Apache ให้บริการ เช่น:
  - Windows (XAMPP): C:\xampp\htdocs\
  - macOS (MAMP): /Applications/MAMP/htdocs/
- เรียกใช้งานผ่านเบราว์เซอร์โดยพิมพ์:
- http://localhost/hello.php

☐ **ใช้งาน PHP CLI โดยไม่ใช้ Apache (ทางเลือก)**

PHP มี Web Server ในตัว (ตั้งแต่ PHP 5.4 ขึ้นไป) ใช้รันแบบง่ายๆ:

php -S localhost:8000

แล้วเปิดเบราว์เซอร์ไปที่ http://localhost:8000

หัวข้อที่ 4: สร้างไฟล์ PHP แรกและรันบนเบราว์เซอร์

☐ **ขั้นตอนการสร้าง**

1. เปิด Notepad / VS Code แล้วพิมพ์โค้ด:
2. <?php
3. echo "สวัสดีชาวโลก!";
4. ?>
5. บันทึกชื่อไฟล์ว่า hello.php แล้วเก็บไว้ที่ htdocs ของ XAMPP
6. เปิดเบราว์เซอร์แล้วเข้าลิงก์:
7. http://localhost/hello.php

☐ **คำอธิบายโค้ด**

```
<?php
echo "สวัสดีชาวโลก!";
?>
```

- <?php ... ?> คือแท็กเปิด-ปิดของ PHP
- echo คือคำสั่งแสดงข้อความไปยังเบราว์เซอร์

หัวข้อที่ 5: Syntax พื้นฐานของ PHP

☐ **การประกาศตัวแปร**

```
$name = "สมชาย";
$age = 25;
```

☐ **คำสั่งแสดงผล**

```
echo "ชื่อ: $name อายุ: $age";
```

☐ **คำสั่งเงื่อนไข**

```
if ($age >= 18) {
    echo "บรรลุนิติภาวะแล้ว";
} else {
    echo "ยังไม่บรรลุนิติภาวะ";
}
```

☐ **ลูป**

```
for ($i = 1; $i <= 5; $i++) {
    echo "รอบที่ $i <br>";
}
```

☐ **ฟังก์ชัน**

```
function sayHello($name) {
    return "สวัสดีคุณ $name";
}

echo sayHello("สมปอง");
```

☐ **สรุป**

บทนี้ช่วยให้คุณ:

- เข้าใจว่า PHP คือภาษา server-side ที่ทำงานบนเว็บเซิร์ฟเวอร์
- ติดตั้ง PHP ผ่าน XAMPP/MAMP ได้ง่าย
- รู้จักโครงสร้างของโค้ด PHP
- เขียนไฟล์ PHP แรกและรันบนเบราว์เซอร์
- เข้าใจพื้นฐาน Syntax เช่น ตัวแปร คำสั่ง echo ลูป และฟังก์ชัน

รู้จักกับ PHP และการตั้งค่าสิ่งแวดล้อม (เชิงลึก)

---

1. PHP คืออะไร และทำงานอย่างไร (Server-side Scripting)

☐ **1.1 ความเป็นมาของ PHP**

- PHP เริ่มต้นพัฒนาในปี 1994 โดย **Rasmus Lerdorf**
- เดิมเรียกว่า “Personal Home Page Tools” ⟶ ปัจจุบันเป็น “PHP: Hypertext Preprocessor”
- พัฒนาโดยชุมชนภายใต้ **PHP Group** มีการออกเวอร์ชันอย่างต่อเนื่อง (PHP 7, PHP 8)

☐ **1.2 ประเภทของ PHP**

- **Server-side scripting**: PHP ทำงานบนเซิร์ฟเวอร์ ⟶ ประมวลผลและส่งผลลัพธ์ไปยังเบราว์เซอร์
- **Command-line scripting (CLI)**: ใช้ PHP รันใน shell/terminal เช่นสคริปต์ cron job
- **Desktop application**: ใช้งานน้อย (เช่น ร่วมกับ PHP-GTK)

☐ **1.3 ขั้นตอนการทำงานของ PHP แบบละเอียด**

1. ผู้ใช้เรียก URL เช่น http://domain.com/index.php
2. เว็บเซิร์ฟเวอร์ (Apache/Nginx) ตรวจสอบว่าไฟล์นั้นต้องส่งไปให้ **PHP interpreter** ประมวลผล
3. PHP ทำงานกับ **Zend Engine** (compiler/interpreter) ⟶ แปลง PHP เป็น opcode
4. ถ้ามีฐานข้อมูล เช่น MySQL ⟶ PHP สื่อสารผ่าน Extension เช่น mysqli, PDO
5. ผลลัพธ์ถูกแปลงเป็น **HTML บริสุทธิ์** และส่งกลับไปยังเบราว์เซอร์ของผู้ใช้

☐ **1.4 ข้อดี/ข้อเสียของ PHP**

ข้อดี	ข้อเสีย
ติดตั้งง่าย, ฟรี	syntax ไม่ strict
เรียนรู้ง่าย	ไม่มี type safety
เอกสารครบ, community ใหญ่	legacy code เยอะ
รองรับ OOP, MVC, API	อาจทำงานช้ากว่า compiled language

---

2. การติดตั้ง PHP (XAMPP, MAMP, LAMP)

☐ **2.1 โครงสร้างการพัฒนา PHP ต้องมี 3 ส่วน:**

- **Interpreter**: โปรแกรมแปล PHP (php.exe หรือ php binary)
- **Web Server**: โปรแกรมจัดการ HTTP (Apache/Nginx)
- **Database (Optional)**: เช่น MySQL/PostgreSQL

☐ **2.2 วิธีติดตั้งโดยใช้ Bundle**

☐ **XAMPP (Windows / macOS / Linux)**

- ติดตั้งง่ายมาก แค่ดาวน์โหลดแล้วคลิก
- ติดตั้งพร้อม Apache, MySQL, PHP, phpMyAdmin
- GUI จัดการบริการ (Start/Stop)
- โฟลเดอร์ root อยู่ที่ htdocs/

☐ **MAMP (Mac)**

- เหมาะกับ macOS โดยเฉพาะ
- สามารถรัน Apache, PHP หลายเวอร์ชันได้
- GUI ใช้งานง่าย

☐ **LAMP (Linux)**

- ติดตั้งผ่าน terminal:

sudo apt update

sudo apt install apache2 php libapache2-mod-php mysql-server

- Web Root: /var/www/html
- แก้ไฟล์คอนฟิกใน /etc/apache2/sites-available/000-default.conf

☐ **2.3 วิธีติดตั้งแบบแยก (ขั้นสูง)**

- ติดตั้ง Apache, PHP, และฐานข้อมูลแยก
- ต้องแก้ไฟล์ httpd.conf หรือ php.ini เอง
- ควบคุมเวอร์ชัน PHP ได้ละเอียด

---

3. การใช้งาน PHP กับเว็บเซิร์ฟเวอร์ (Apache, Nginx)

☐ 3.1 การทำงานของ Apache + PHP

- Apache ใช้ **mod_php** เพื่อประมวลผล PHP
- เมื่อไฟล์ .php ถูกเรียก ⟶ mod_php ส่งให้ PHP interpreter ⟶ คืน HTML กลับมา

Apache Config (ตัวอย่าง):

```
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>
```

☐ 3.2 Nginx + PHP-FPM

- Nginx ไม่มี mod_php ⟶ ต้องใช้ FastCGI ผ่าน **PHP-FPM**
- ทำงานเร็วและประหยัดทรัพยากรมากกว่า Apache
- เหมาะกับ production

Nginx Config (ตัวอย่าง):

```
location ~ \.php$ {
    fastcgi_pass unix:/run/php/php8.1-fpm.sock;
    include fastcgi_params;
    fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
}
```

☐ 3.3 PHP CLI Server (สำหรับพัฒนา)

- เริ่มตั้งแต่ PHP 5.4 ขึ้นไป

```
php -S localhost:8080
```

- ใช้สำหรับรัน local dev แบบง่าย (ไม่มี .htaccess, ไม่มี Virtual Host)

---

4. สร้างไฟล์ PHP แรกและรันบนเบราว์เซอร์

☐ 4.1 ตัวอย่าง hello.php

```
<?php
echo "Hello, world!";
?>
```

☐ 4.2 อธิบาย:

- <?php ... ?> คือ **PHP opening tag**
- echo ใช้พิมพ์ข้อความ
- ต้องเก็บไฟล์นี้ใน web root เช่น htdocs/ หรือ /var/www/html/
- เรียกผ่าน URL:
- http://localhost/hello.php

4.3 แนะนำฟังก์ชันแสดงผลอื่น

- print() ⟶ เหมือน echo แต่คืนค่าด้วย
- var_dump() ⟶ แสดงชนิดข้อมูลและค่า (debug)
- print_r() ⟶ แสดง array อย่างสวยงาม

---

5. Syntax พื้นฐานของ PHP

5.1 ตัวแปร

```
$name = "สมชาย";  // String
$age = 30;          // Integer
$pi = 3.14;         // Float
$isAdmin = true;    // Boolean
```

5.2 คำสั่งเงื่อนไข

```
if ($age >= 18) {
    echo "บรรลุนิติภาวะแล้ว";
} elseif ($age >= 13) {
    echo "วัยรุ่น";
} else {
    echo "เด็ก";
}
```

5.3 ลูป

```
for ($i = 1; $i <= 5; $i++) {
    echo "รอบที่ $i<br>";
}
```

5.4 ฟังก์ชัน

```
function greet($name) {
    return "สวัสดีคุณ $name";
}

echo greet("สมปอง");
```

5.5 Array

```
$fruits = ["apple", "banana", "mango"];
echo $fruits[1]; // banana

$person = ["name" => "A", "age" => 20];
```

```
echo $person["name"];
```

☐ **สรุปแนวคิดสำคัญ**

หัวข้อ	สิ่งที่ควรเข้าใจ
PHP คือ	ภาษาที่รันบนเซิร์ฟเวอร์ สร้างหน้าเว็บแบบไดนามิก
ติดตั้งอย่างไร	ใช้ XAMPP, MAMP หรือ LAMP ตามระบบปฏิบัติการ
ทำงานกับเซิร์ฟเวอร์อย่างไร	ผ่าน Apache หรือ Nginx โดยใช้ mod_php หรือ PHP-FPM
สร้างไฟล์แรกอย่างไร	เขียนใน .php แล้วรันผ่าน localhost
โครงสร้างพื้นฐาน	Syntax ของตัวแปร, คำสั่ง echo, if, loop, function

---

PHP คืออะไร และทำงานอย่างไร (Server-side Scripting)

☐ **1. PHP คืออะไร?**

PHP (ย่อจาก *PHP: Hypertext Preprocessor*) คือ:

ภาษาโปรแกรมประเภท **Server-side Scripting** ที่ออกแบบมาสำหรับการพัฒนาเว็บแอปพลิเคชันแบบไดนามิก โดยรันโค้ดบนเซิร์ฟเวอร์ และส่งผลลัพธ์ที่เป็น HTML หรือข้อมูลอื่นๆ (เช่น JSON) กลับไปยังผู้ใช้ผ่านเว็บเบราว์เซอร์

☐ **จุดเด่น:**

- เป็น **Open Source** ใช้งานฟรี
- เรียนรู้ง่าย เหมาะกับทั้งผู้เริ่มต้นและผู้เชี่ยวชาญ
- เขียน **ฝังอยู่ใน HTML ได้** (Embedded)
- ใช้งานได้กับเว็บเซิร์ฟเวอร์หลากหลาย (Apache, Nginx, IIS)
- รองรับฐานข้อมูล เช่น MySQL, PostgreSQL, SQLite, Oracle
- มี framework ระดับโลก เช่น Laravel, Symfony, CodeIgniter

☐ **2. Server-side Scripting คืออะไร?**

☐ **ความหมาย:**

Server-side scripting หมายถึง:

การเขียนโปรแกรมที่ **ทำงานบนเซิร์ฟเวอร์** ก่อนจะส่งผลลัพธ์ให้ผู้ใช้ (ผ่านเบราว์เซอร์)

PHP จึง **ทำงานในฝั่งเซิร์ฟเวอร์** ไม่ได้ทำงานในเบราว์เซอร์ของผู้ใช้เหมือน JavaScript

☐ **กระบวนการทำงาน:**

1. ผู้ใช้พิมพ์ URL เช่น https://example.com/index.php

2. เบราว์เซอร์ส่ง HTTP request ไปยัง **Web Server** (เช่น Apache)
3. Web Server รู้ว่าไฟล์ .php ต้องถูกส่งให้ **PHP interpreter**
4. PHP ประมวลผลคำสั่ง, เชื่อมต่อฐานข้อมูล, สร้าง HTML หรือ JSON
5. **เฉพาะผลลัพธ์ (output)** จะถูกส่งกลับไปที่เบราว์เซอร์
6. ผู้ใช้เห็นหน้าเว็บที่ถูก render แล้ว — ไม่สามารถดูโค้ด PHP ได้

✳ □ ผู้ใช้เห็นเฉพาะสิ่งที่ echo, print, หรือ HTML ธรรมดาส่งออกเท่านั้น

□ 3. ตัวอย่างภาพจำลอง

□ ตัวอย่างการเรียกหน้าเว็บ

ผู้ใช้เปิด:

https://myapp.com/welcome.php?name=John

□ **welcome.php**

```
<?php
$name = $_GET['name'] ?? 'Guest';
echo "<h1>Welcome, $name!</h1>";
?>
```

□ **ผลลัพธ์ที่ส่งไปยังเบราว์เซอร์:**

```
<h1>Welcome, John!</h1>
```

โค้ด PHP จะไม่ถูกส่งออก! เพราะมัน **ประมวลผลก่อนแล้วที่ฝั่งเซิร์ฟเวอร์**

□ 4. PHP ใช้ทำอะไรได้บ้าง?

- แสดงข้อมูลแบบไดนามิก (Dynamic Web Page)
- รับข้อมูลจากแบบฟอร์ม HTML
- เชื่อมต่อและจัดการฐานข้อมูล (MySQL, PostgreSQL ฯลฯ)
- สร้างไฟล์ PDF, ZIP, Excel, ภาพ
- ทำ API (RESTful, GraphQL)
- เขียน Command-line script
- ทำ Backend สำหรับเว็บหรือ Mobile App

□ 5. PHP ทำงานร่วมกับอะไรบ้าง?

ส่วนประกอบ	ตัวอย่าง	หน้าที่
Web Server	Apache, Nginx	รับ HTTP request, ส่งให้ PHP ประมวลผล
PHP Engine	Zend Engine	ประมวลผลโค้ด PHP

ส่วนประกอบ	ตัวอย่าง	หน้าที่
Database	MySQL, SQLite	จัดเก็บข้อมูล
File System	Local, Cloud	เก็บไฟล์, อัปโหลด
Client	Web Browser	แสดงผลลัพธ์ (HTML/JSON)

□ 6. เปรียบเทียบ Server-side vs Client-side

ด้าน	Server-side (PHP)	Client-side (JavaScript)
ทำงานที่	เซิร์ฟเวอร์	เบราว์เซอร์
มองเห็นโค้ดได้ไหม	□ (ซ่อน)	□
ใช้ประมวลผล	ข้อมูล, DB, Logic	UI, Animation, Validation
ความปลอดภัย	สูงกว่า	ต่ำกว่า

□ สรุปสั้น ๆ

- PHP คือภาษา **ฝั่งเซิร์ฟเวอร์** ที่สร้างเว็บแบบไดนามิก
- โค้ด PHP **ทำงานก่อน** แล้วจึงส่งผลลัพธ์ (HTML/JSON) กลับให้ผู้ใช้
- ทำงานร่วมกับ Apache/Nginx และฐานข้อมูล
- เป็นเทคโนโลยีหลักที่ใช้พัฒนาเว็บมานาน เช่น WordPress, Facebook รุ่นแรก

ประวัติความเป็นมา และแนวโน้มของภาษา PHP (จากอดีตถึงปัจจุบัน และอนาคต)

□ 1. กำเนิดของ PHP (1994–1997): ยุคก่อนมี Framework

■ ปี 1994 – จุดเริ่มต้นจาก CGI Script ส่วนตัว

- **Rasmus Lerdorf** โปรแกรมเมอร์ชาวเดนมาร์ก-แคนาดา ต้องการสร้างหน้าเว็บที่ติดตามผู้เข้าเยี่ยมชมเว็บส่วนตัวของเขา
- เขาเขียนชุด **Common Gateway Interface (CGI)** ด้วยภาษา C ⟶ เรียกว่า "**Personal Home Page Tools**"
- ทำให้เว็บแสดงข้อมูลแบบไดนามิก และสามารถอ่านข้อมูลฟอร์ม HTML ได้

■ ปี 1995 – เปิดตัว PHP 1.0

- Rasmus เผยแพร่โค้ดแบบโอเพ่นซอร์สภายใต้ชื่อ **PHP Tools 1.0**
- เริ่มมีการรับข้อมูลผ่าน HTML forms และรองรับฐานข้อมูล

☐ 2. ยุคเติบโต (1997–2004): กลายเป็นภาษายอดนิยมบนเว็บ

- **ปี 1997 – PHP 2.0 (PHP/FI)**
  - เปลี่ยนชื่อเป็น **PHP/FI 2.0** (Form Interpreter)
  - มี Syntax คล้าย C และเริ่มสนับสนุนฐานข้อมูล
- **ปี 1998 – PHP 3.0 (เปลี่ยนสู่ภาษาสคริปต์จริงจัง)**
  - พัฒนาใหม่โดย **Andi Gutmans** และ **Zeev Suraski**
  - เริ่มสนับสนุนฟีเจอร์ที่เหมือนภาษาโปรแกรมจริง ๆ เช่น ฟังก์ชัน, ตัวแปร, Extension
- **ปี 2000 – PHP 4.0: Zend Engine รุ่นแรก**
  - เปิดตัว **Zend Engine 1.0** ซึ่งเป็น Core ของ PHP ในเวลานั้น
  - รองรับ OOP ระดับเบื้องต้น
  - การใช้งานแพร่หลายมากขึ้นในระบบ CMS เช่น phpNuke, phpBB

☐ 3. ยุคเฟื่องฟู (2004–2014): การเข้าสู่ยุคของ PHP Framework

- **ปี 2004 – PHP 5.0: รองรับ OOP เต็มรูปแบบ**
  - มาพร้อม **Zend Engine 2**
  - สนับสนุน **Class, Inheritance, Visibility, Interfaces, Exceptions**
  - ระบบ Framework เริ่มถือกำเนิด เช่น CodeIgniter, CakePHP, Symfony
- **ปี 2008–2011 – Laravel เริ่มปรากฏ**
  - Laravel 1.0 เปิดตัวในปี 2011 โดย **Taylor Otwell**
  - เป็น Framework สมัยใหม่ที่ช่วยให้ PHP มีโครงสร้างแบบ MVC อย่างชัดเจน

☐ 4. ยุคปรับตัวสู่ประสิทธิภาพ (2015–ปัจจุบัน): PHP ไม่ได้ช้าอย่างที่เคยเป็น

- **ปี 2015 – PHP 7.0: ก้าวกระโดดด้านประสิทธิภาพ**
  - เปลี่ยนมาใช้ **Zend Engine 3.0**
  - เร็วขึ้นกว่า PHP 5 ถึง **2 เท่า**
  - เพิ่ม Scalar Type, Return Type, Anonymous Classes, Null Coalescing (??)
  - ปรับปรุง Memory Usage และ Exception Handling ใหม่
- **ปี 2020 – PHP 8.0: ประสิทธิภาพเทียบชั้นภาษาใหม่**
  - เปิดตัว **Just-In-Time Compiler (JIT)**
  - รองรับ **Attributes (Annotation)**, Match Expression, Union Types
  - มีฟีเจอร์ OOP แบบ Modern มากขึ้น
  - ประสิทธิภาพเทียบได้กับ Java, Go ในบางเคส

■ **ปี 2022–2023 – PHP 8.1, 8.2**

- เพิ่ม readonly, enum, fibers และ never return type
- ปรับโครงสร้างให้รองรับ **Asynchronous Programming**

☐ **5. สถานะปัจจุบันของ PHP (2024–2025)**

☐ **ยังเป็น ภาษายอดนิยมด้าน Web Backend**

- **WordPress** (42% ของเว็บไซต์ทั่วโลก) เขียนด้วย PHP
- **Facebook** (ในอดีต) พัฒนาด้วย PHP ⟶ ปรับเป็น Hack (ภาษาที่แปลงจาก PHP)
- Laravel กลายเป็น Framework ยอดนิยมมากที่สุด
- CMS ดัง: Drupal, Joomla, Magento ⟶ PHP ทั้งหมด
- ยังเป็นภาษาที่ใช้ในโครงการระดับ Production อย่างแพร่หลาย

☐ **Ecosystem ที่แข็งแรง:**

- Composer (dependency manager)
- PHPStan / Psalm (static analysis)
- PHPUnit (testing framework)
- Laravel ecosystem (Forge, Vapor, Nova, Envoyer)

☐ **6. แนวโน้มของ PHP ในอนาคต**

☐ **แนวโน้มเชิงบวก:**

- **Modernization ต่อเนื่อง** ⟶ PHP 9.x จะเน้นความปลอดภัย, async/await, lightweight coroutine
- มีความพยายามให้ PHP รองรับ **High Concurrency แบบ Node.js** (เช่น Swoole, ReactPHP)
- ยังคงได้รับการสนับสนุนในวงการ Web + CMS + API + E-Commerce
- ใช้ใน **Microservices**, **REST APIs**, **Serverless (ผ่าน Laravel Vapor)**

☐ **ความท้าทาย:**

- ความนิยมลดลงบ้างในสาย startup ที่หันไปใช้ Node.js, Python, Go
- ภาพลักษณ์เดิมว่า “ช้า-โบราณ” ยังติดอยู่ แม้เทคโนโลยีปัจจุบันจะแก้แล้ว
- การแข่งขันจาก Framework ใหม่ เช่น Fastify, Next.js, Spring Boot

---

☐ **สรุปภาพรวมประวัติและอนาคต PHP**

ยุค	เหตุการณ์สำคัญ
1994	เริ่มโดย Rasmus Lerdorf

ยุค	เหตุการณ์สำคัญ
1998	PHP 3 กลายเป็นภาษา scripting เต็มตัว
2004	PHP 5 รองรับ OOP จริงจัง
2015	PHP 7 พลิกภาพลักษณ์ "เร็ว–ปลอดภัย"
2020+	PHP 8/8.1/8.2 เพิ่ม modern features, JIT, enums
2025+	เดินหน้า async, testing-first, typed programming

สถิติการใช้งาน PHP

- ปัจจุบันมีมากกว่า **10.7 ล้านบริษัท** ใช้ PHP ในระบบของตน (ข้อมูลล่าสุดปี 2024) (webtastic.ai, Seven Square Technologies)
- ประมาณ **78% ของเว็บไซต์ที่มีการใช้งานฝั่งเซิร์ฟเวอร์** ใช้ PHP เท่านั้น (Reddit)

☐ รายชื่อตัวอย่างองค์กรระดับโลกที่ใช้ PHP

1. Facebook

- เริ่มต้นเขียนด้วย PHP รุ่นแรก
- พัฒนา HipHop ⟶ HHVM เพื่อเพิ่มประสิทธิภาพในการประมวลผล PHP (Guru)

2. Wikipedia / MediaWiki

- รันผ่าน PHP + MySQL
- ใช้ HHVM เพื่อการประมวลผลที่เร็วขึ้นกว่าเดิมเกือบเท่าตัว (felipe.lavin.blog, WIRED)

3. WordPress

- CMS ยอดนิยมติดอันดับใช้ PHP
- รองรับเว็บไซต์ถึง ~43% ของโลก (BairesDev)

4. Tumblr

- ใช้ PHP และอัปเกรดเป็น PHP 7 ในปี 2016
- ลด latency และโหลด CPU ถึง 50% (Guru, BairesDev)

5. Slack

- ใช้ PHP ในระบบ backend เพื่อความเร็วและประสิทธิภาพ
- มีผู้ใช้หลายล้านคนต่อวัน (webwishcreation.com)

6. MailChimp

- ใช้ PHP ในการส่งอีเมลหลายร้อยล้านฉบับ/วัน
- PHP เป็นภาษาหลักสำหรับ production

7. Etsy