วิชาการออกแบบและเขียนโปรแกรมเว็บเพจ

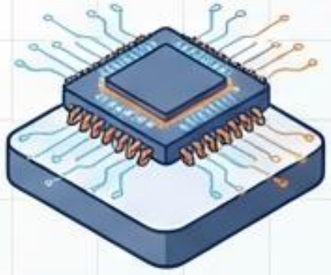
หลักการทำงานของเว็บเซิร์ฟเวอร์

สถาปัตยกรรม โปรโตคอล และการประยุกต์ใช้

ผู้ช่วยศาสตราจารย์สมเกียรติ ช่อเหมือน

เว็บเซิร์ฟเวอร์คืออะไร?

ซอฟต์แวร์ที่ให้บริการข้อมูลผ่าน HTTP
และจัดเก็บไฟล์เว็บไซต์



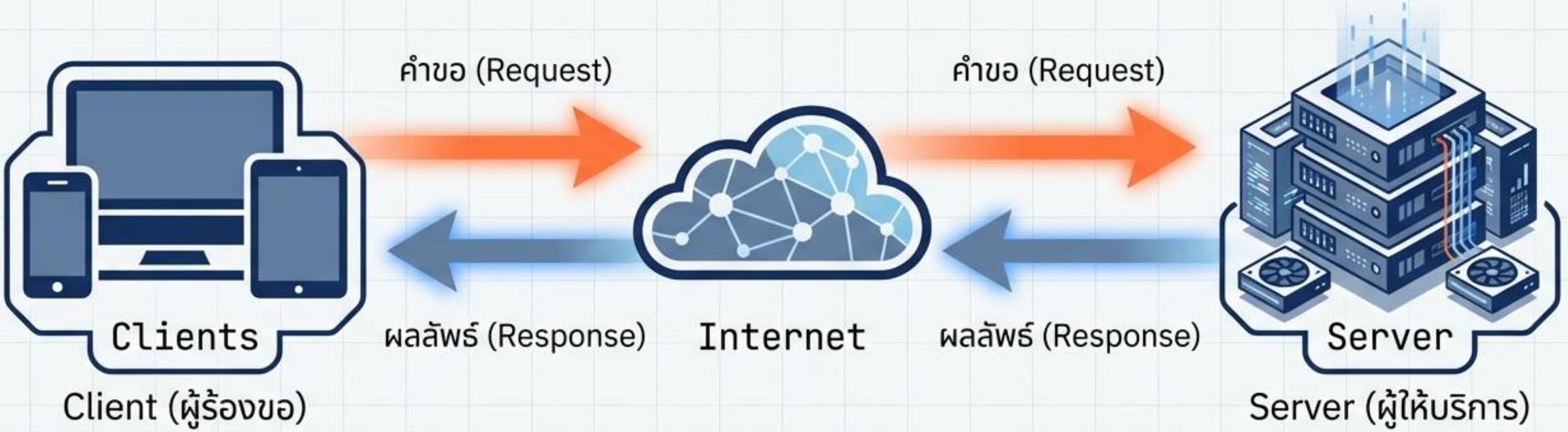
ทำงานบนเครื่องคอมพิวเตอร์
ประสิทธิภาพสูง

เชื่อมต่ออินเทอร์เน็ต
ความเร็วสูงเสมอ

เปิดให้บริการตลอด
24 ชั่วโมง

รอรับคำขอจากผู้ใช้งาน
(Clients)

โครงสร้าง Client-Server



Client (ผู้ร้องขอ)

เบราว์เซอร์ทำหน้าที่ส่งคำขอข้อมูล

Server (ผู้ให้บริการ)

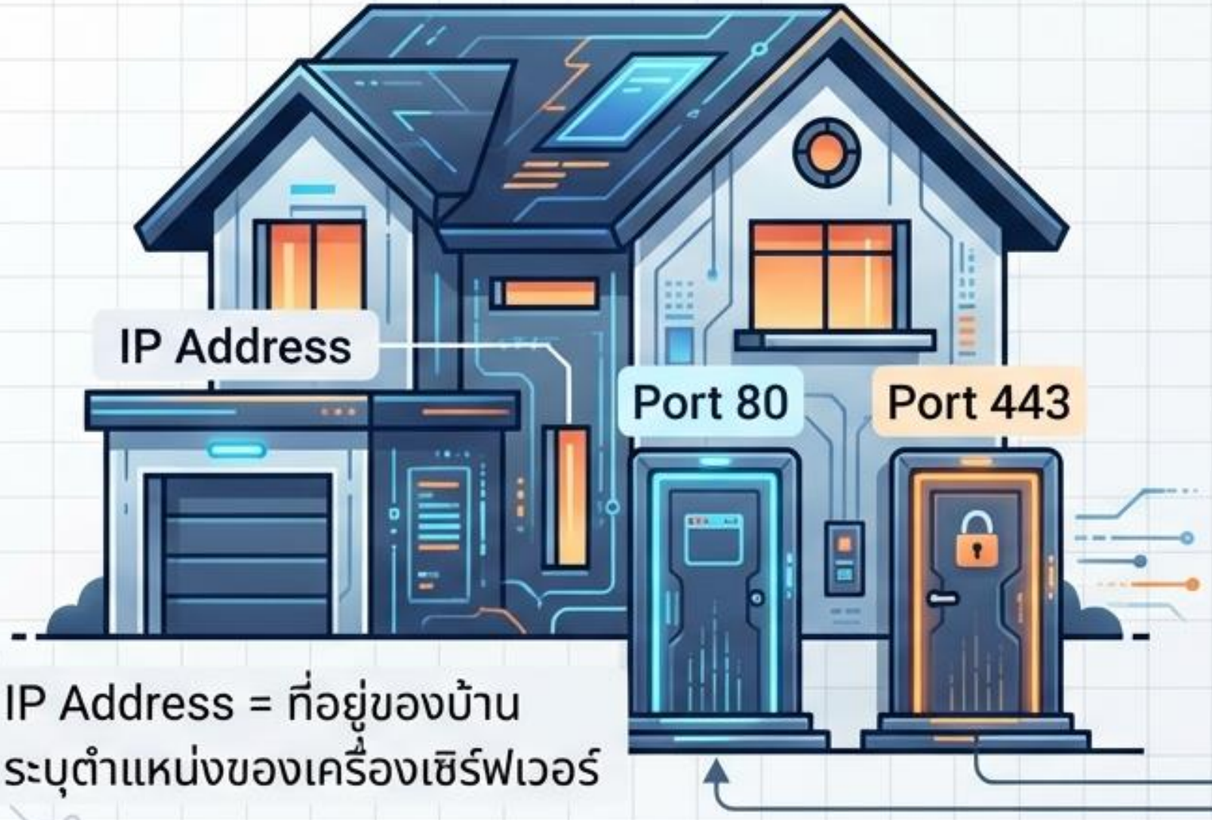
เครื่องแม่ข่ายทำหน้าที่รับคำขอและส่งผลลัพธ์

ใช้มาตรฐานการสื่อสารเดียวกันและรองรับผู้ใช้งานพร้อมกันจำนวนมาก

การระบุตัวตน: หมายเลข IP และ Port

JetBrains Mono

House Analogy



IP Address = ที่อยู่ของบ้าน
ระบุตำแหน่งของเครื่องเซิร์ฟเวอร์

Technical Reality

Port = ประตูเข้าบ้าน (ช่องทางสื่อสารเฉพาะบริการ)

- 80: สำหรับเว็บทั่วไป
- 443: สำหรับเว็บแบบปลอดภัย

ระบบ DNS (Domain Name System)

User
จำชื่อเว็บ
(google.com)

DNS
แปลงชื่อ
เป็น IP

Browser
เชื่อมต่อ
IP ปลายทาง

เบราว์เซอร์จะถาม DNS ก่อนเสมอ

DNS Mechanism

ขั้นตอนการทำงาน5 ระยะ

1. Input
ผู้ใช้พิมพ์ URL ในเบราว์เซอร์

2. Lookup
เบราว์เซอร์หา IP จาก DNS

3. Request
ส่ง HTTP Request ไปยังเซิร์ฟเวอร์

4. Process
เซิร์ฟเวอร์ประมวลผลคำขอ

5. Response
ส่ง HTTP Response กลับมา

ส่วนประกอบของ HTTP Request

Method

คำสั่งการกระทำ

`GET, POST, PUT, DELETE`

URL

เส้นทางเข้าถึงทรัพยากร

`/index.html`

Headers

ข้อมูลเสริม (Browser type, file type)

วงจร Request-Response

การสื่อสารแบบรอบต่อรอบ: Server ทำงานจบในหนึ่งรอบการร้องขอ ไม่มีการจำค่าเดิม

Communication Loop

รหัสสถานะ HTTP (HTTP Status Codes)

Kanit และ JetBrains Mono

SUCCESS

200 OK

ทำงานสำเร็จ ข้อมูลส่งถึงครบ

NOT FOUND

404 Not Found

ไม่พบไฟล์ที่ต้องการบนระบบ

ERROR

500 Server Error

เซิร์ฟเวอร์ทำงานผิดพลาดภายใน

Static vs Dynamic Content

Static Content (ไฟล์คงที่)

- ไฟล์คงที่ ไม่เปลี่ยนแปลง
- HTML, CSS, รูปภาพ
- เซิร์ฟเวอร์ส่งไฟล์ให้โดยตรง
- ทำงานเร็วมาก

Dynamic Content (เปลี่ยนแปลงได้)

- สร้างข้อมูลใหม่ตามคำขอ
- ใช้ภาษา PHP, Node.js, Python
- ดึงข้อมูลจาก Database
- มีความยืดหยุ่นสูง

ซอฟต์แวร์เว็บเซิร์ฟเวอร์ที่นิยม

Apache: เสถียรและใช้งานแพร่หลาย

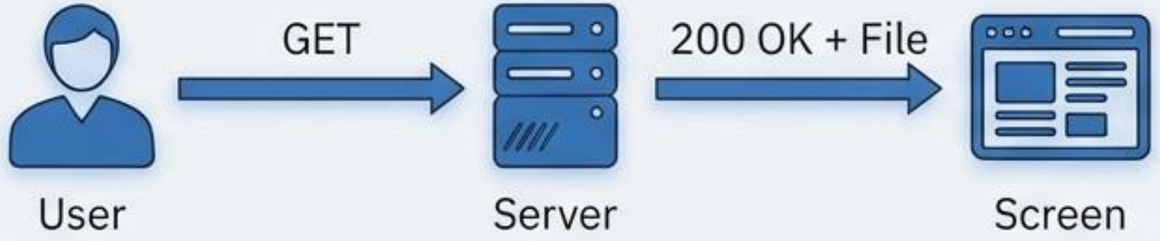
Nginx: ประสิทธิภาพสูง รองรับงานหนัก

IIS: ของ Microsoft สำหรับ Windows

Node.js: สำหรับเว็บแอปพลิเคชันยุคใหม่

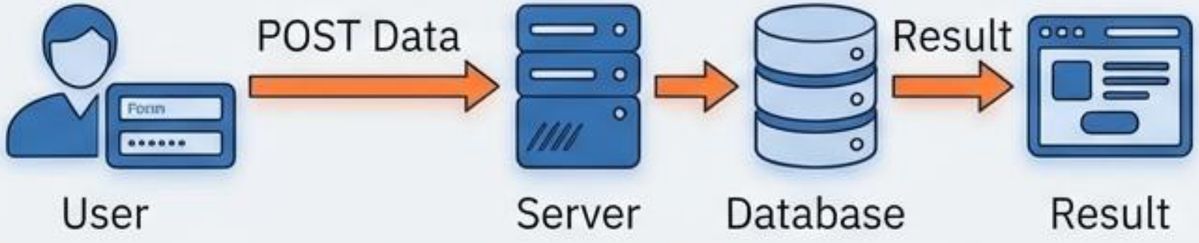
ตัวอย่างการทำงานจริง: การเรียกดู vs การเข้าสู่ระบบ

Browsing (GET)



- เข้าชมหน้า Home Page
- อ่านไฟล์ index.html
- แสดงผลหน้าจอ

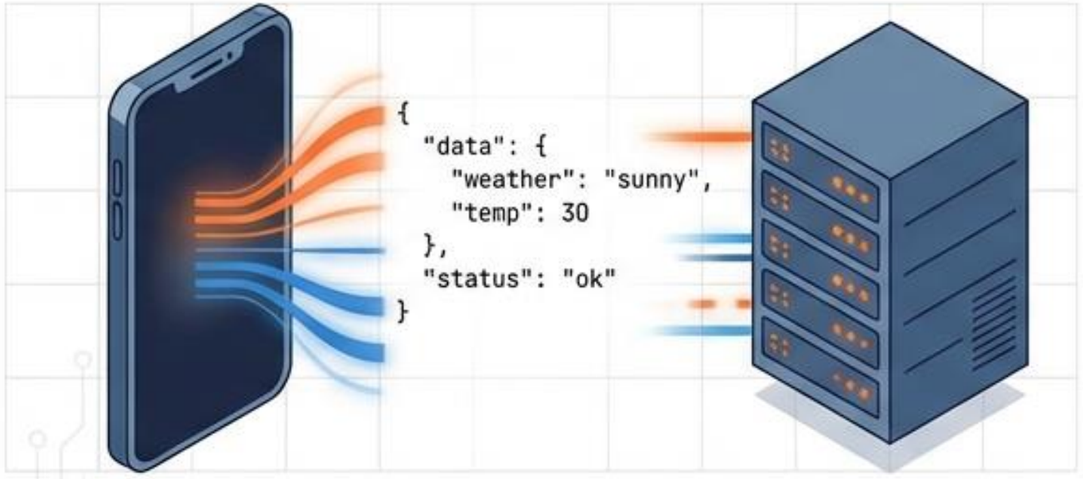
Login (POST)



- กรอกรหัสผ่าน
- ตรวจสอบกับ Database
- อนุญาต/ไม่อนุญาต

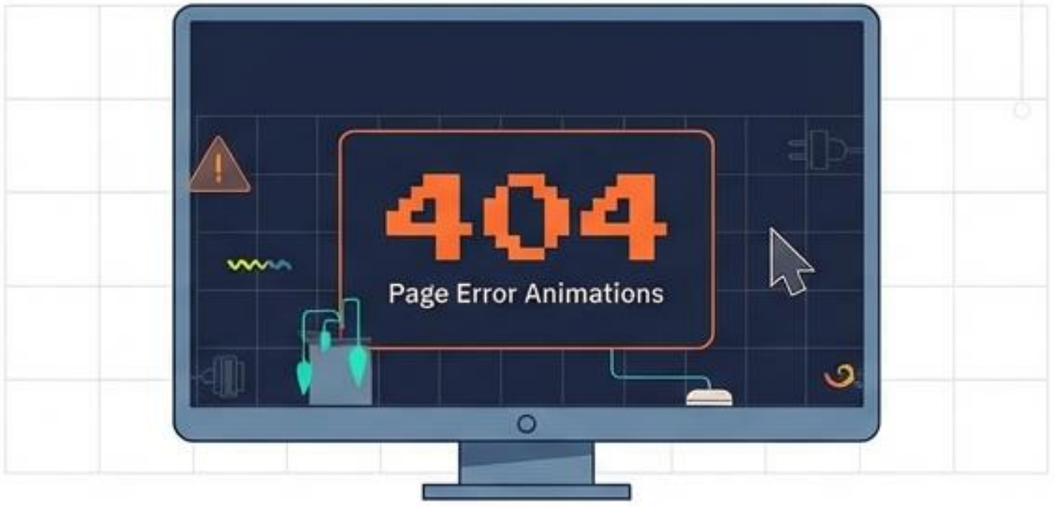
Advanced Examples.
การรับส่งข้อมูลและการจัดการข้อผิดพลาด

JSON Data (API)



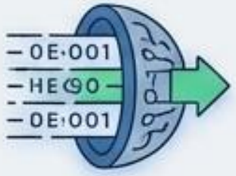
- สำหรับแอปพลิเคชันมือถือ
- รับส่งข้อมูลรวดเร็ว ไม่เน้นหน้าจอ HTML

Error 404



- ผู้ใช้สะกดชื่อไฟล์ผิด
- Server หาไฟล์ไม่เจอ
- แจ้งเตือนให้ผู้ใช้ตรวจสอบ URL

ความปลอดภัย (HTTPS)



HTTPS คือ HTTP ที่มีการเข้ารหัส

เบราว์เซอร์แสดงรูปกุญแจสีเขียว

ใช้ SSL/TLS ป้องกันการดักข้อมูล

จำเป็นมากสำหรับระบบธุรกรรม

บทสรุปสำคัญ (Key Takeaways)

1		เซิร์ฟเวอร์เป็นหัวใจขององการส่งข้อมูล	2		Request-Response คือกลไกพื้นฐาน
3		ต้องเข้าใจรหัสสถานะ (Status Codes)	4		แยกแยะไฟล์คงที่และพลวัตได้
5		ความปลอดภัยเป็นเรื่องที่ไม่ควรมองข้าม	6		ฝึกฝนการอ่าน Log เพื่อแก้ไขปัญหา