รายวิชา 88726065 Front-end Web Development

สัปดาห์ที่ 1: ภาพรวมวิชา เครื่องมือพัฒนา และพื้นฐานเว็บ & HTTP/HTTPS

วัตถุประสงค์การเรียนรู้ (หลังจบบทเรียนนี้ นักศึกษาจะสามารถ)

- อธิบายสถาปัตยกรรมเว็บโดยรวม (Client–Server–Internet) และมุมมอง Front-end/Back-end ได้
- อธิบายองค์ประกอบของ URL, แนวคิดทรัพยากร (Resource), และเส้นทาง (Path/Query) ได้
- อธิบายหลักการของ HTTP request/response, methods พื้นฐาน (GET/POST), status codes ชั้นหลัก (2xx/3xx/4xx/5xx) ได้
- อธิบายบทบาทของ HTML/CSS/JavaScript ในการสร้างเว็บเพจเชิงโต้ตอบได้
- ระบุเครื่องมือพัฒนา (VS Code, เบราว์เซอร์ + DevTools, Node.js, Git) และหน้าที่ของแต่ละเครื่องมือได้

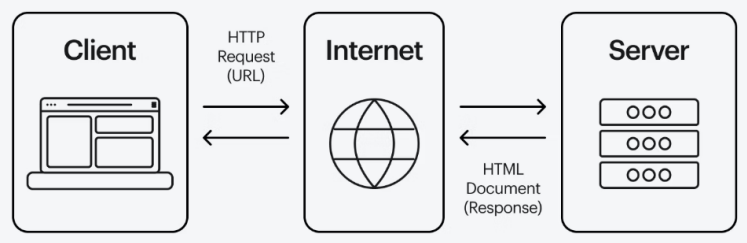
หัวข้อบรรยาย

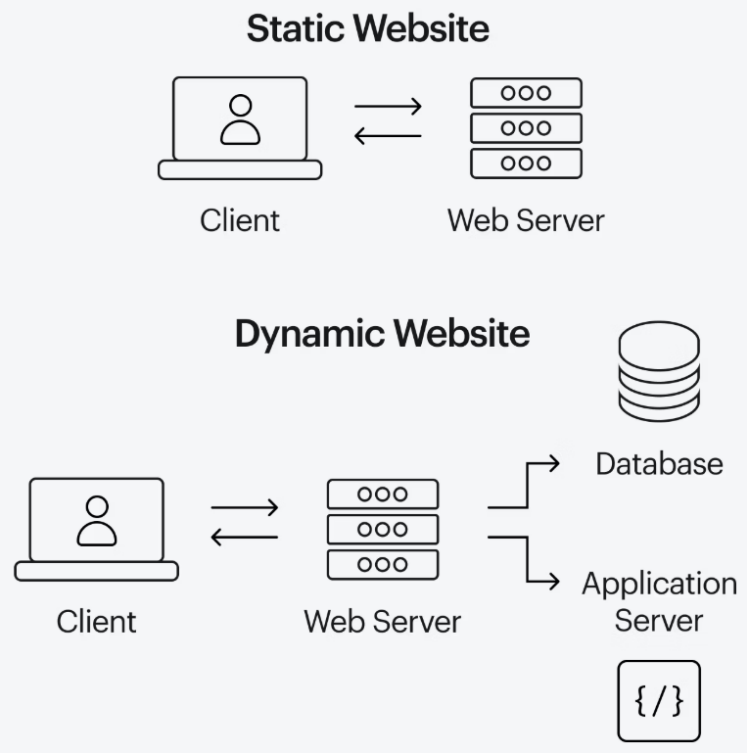
1. แนะนำรายวิชาและคาดหวังการเรียนรู้ (CLOs, รูปแบบประเมินผล, แนวทางการส่งงาน)
2. สถาปัตยกรรมเว็บและวงจรการร้องขอ–ตอบกลับ (HTTP)
3. URL, DNS, Hosting/CDN, HTTPS/TLS (ภาพรวม)
4. โครงสร้างเว็บเพจและบทบาท HTML/CSS/JS
5. เครื่องมือพัฒนา: VS Code, Extensions, Browser DevTools, Node.js, npm, Git
6. ตัวอย่างการวิเคราะห์ Request/Response ด้วย Network Panel
7. Q&A และสรุปเชื่อมสู่ Lab

1) ภาพรวมรายวิชาและหัวข้อ

- แนวคิดหลักของรายวิชา: พัฒนาเว็บฝั่งลูกข่าย (Front-end) เป็นหลัก, เชื่อมต่อ API, คำนึงถึง Performance, Accessibility, Security
- โครงสร้างรายสัปดาห์โดยย่อ + จุดประสงค์ของ Lab
- เกณฑ์ให้คะแนน (Quiz/Assignments/Labs/Project/Midterm/Final) และนโยบายการส่งงาน/ความซื่อสัตย์ทางวิชาการ

2) สถาปัตยกรรมเว็บ (ภาพใหญ่)





https://www.ramotion.com/blog/what-is-a-web-server/

ความเข้าใจทั่วไป

- Client-Server Model: ผู้ใช้ส่งคำร้องขอ ⟶ เซิร์ฟเวอร์ตอบกลับ
- Static vs Dynamic content
    - Static: ไฟล์เดียวกันไปให้ทุกคน (HTML/CSS/JS)
    - Dynamic: เนื้อหาแตกต่างตามผู้ใช้ (Database-driven)
- CDN (Content Delivery Network): เซิร์ฟเวอร์กระจายเพื่อความเร็ว
- Caching: เก็บข้อมูลเพื่อไม่ต้องดึงหลายครั้ง
- DNS (Domain Name System): แปลงชื่อโดเมน (example.com) → IP Address (93.184.216.34)
- HTTPS/TLS Encryption: เข้ารหัสการสื่อสาร เพื่อป้องกันการแอบดูข้อมูล

ตัวอย่าง DNS

User พิมพ์: https://example.com

↓

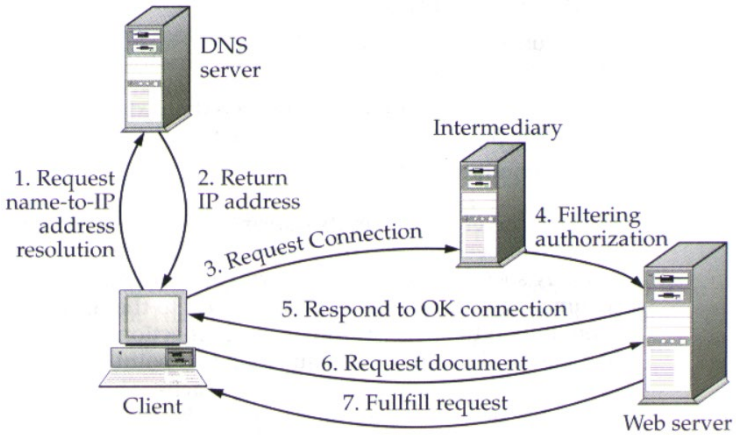
DNS Query: What is IP of example.com?

↓

DNS Response: 93.184.216.34

↓

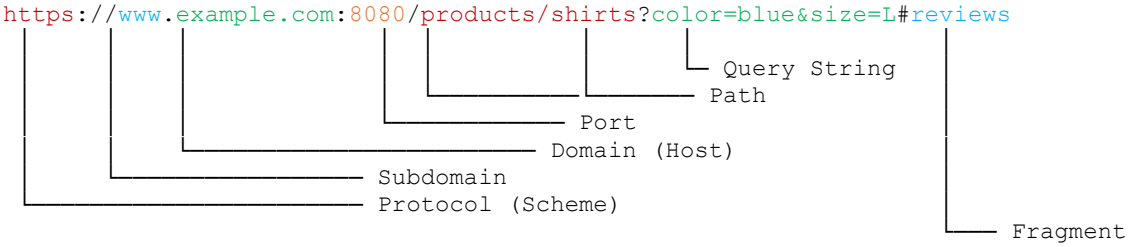
Browser connects to 93.184.216.34:443 (HTTPS)



https://technology.niagaracollege.ca/courses/elnc1221/notes/tcp-ip-10.html

3) URL และ HTTP พื้นฐาน

โครงสร้าง URL (Uniform Resource Locator)



ส่วนประกอบ:

ส่วน	ตัวอย่าง	หมายความ
Scheme	https	โปรโตคอล (http, https, ftp)
Host	www.example.com	ชื่อเซิร์ฟเวอร์
Port	8080	พอร์ต (ค่าเริ่มต้น: 80 HTTP, 443 HTTPS)
Path	/products/shirts	เส้นทางทรัพยากร
Query	?color=blue&size=L	พารามิเตอร์ค้นหา
Fragment	#reviews	ตำแหน่งในหน้า (ไม่ส่งไปเซิร์ฟเวอร์)

ตัวอย่าง Query String

?color=blue&size=L&page=2

พารามิเตอร์ตัวที่ 1: color = blue

พารามิเตอร์ตัวที่ 2: size = L

พารามิเตอร์ตัวที่ 3: page = 2

HTTP Methods (วิธีการร้องขอ)

GET - ขอข้อมูล (Safe, Idempotent)

ตัวอย่าง: GET /users/123 → ดึงข้อมูลผู้ใช้ ID 123

ข้อมูลใน URL ได้เห็น (ไม่ปลอดภัย)

POST - สร้าง/ส่งข้อมูลใหม่

ตัวอย่าง: POST /users + body {"name":"John","age":25}

ข้อมูลในเนื้อหา (ปลอดภัยกว่า)

PUT - แก้ไขทั้งหมด (Replace)

ตัวอย่าง: PUT /users/123 + body {"name":"Jane","age":26}

PATCH - แก้ไขบางส่วน (Partial update)

ตัวอย่าง: PATCH /users/123 + body {"age":27}

DELETE - ลบข้อมูล

ตัวอย่าง: DELETE /users/123 → ลบผู้ใช้ ID 123

REST Principles

GET /users → ดึงรายชื่อผู้ใช้ทั้งหมด

GET /users/123 → ดึงผู้ใช้ ID 123

POST /users → สร้างผู้ใช้ใหม่

PUT /users/123 → แก้ไขผู้ใช้ ID 123 (ทั้งหมด)

PATCH /users/123 → แก้ไขผู้ใช้ ID 123 (บางส่วน)

DELETE /users/123 → ลบผู้ใช้ ID 123

HTTP Status Codes (รหัสสถานะการตอบสนอง)

2xx - Success (สำเร็จ)

200 OK → ขอสำเร็จและมีเนื้อหา

201 Created → สร้างทรัพยากรใหม่สำเร็จ

204 No Content → สำเร็จแต่ไม่มีเนื้อหา

3xx - Redirection (เปลี่ยนหน้า)

301 Moved Permanently → โยกย้ายถาวร

302 Found (Temporary) → โยกย้ายชั่วคราว

304 Not Modified → ใช้แคชไฟล์เดิม

4xx - Client Error (ผิดพลาดฝั่ง Client)

400 Bad Request → ร้องขอไม่ถูกต้อง

401 Unauthorized → ยังไม่ login

403 Forbidden → ห้าม (ไม่มีสิทธิ์)

404 Not Found → ไม่พบไฟล์/หน้า

5xx - Server Error (ผิดพลาดฝั่ง Server)

500 Internal Server Error → เซิร์ฟเวอร์มีข้อผิดพลาด

502 Bad Gateway → Gateway ไม่ดี

503 Service Unavailable → เซิร์ฟเวอร์ busy/maintenance

ตัวอย่างการใช้

User ร้องขอ: https://example.com/users/999

Server ตอบ: 404 Not Found

(เพราะไม่มีผู้ใช้ ID 999)

User ร้องขอ: https://old-site.com

Server ตอบ: 301 Moved Permanently

Header: Location: https://new-site.com

(Browser อัตโนมัติไปที่ new-site.com)

HTTP Headers (ข้อมูลเพิ่มเติม)

Request Headers (Client → Server)

Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml
Accept-Language: th,en-US;q=0.9
Content-Type: application/json
Authorization: Bearer token123456

Response Headers (Server → Client)

Content-Type: text/html; charset=utf-8
Content-Length: 1234
Set-Cookie: sessionId=abc123
Cache-Control: max-age=3600

ตัวอย่าง **HTTP Request/Response**

========== REQUEST ==========
GET /products?category=books HTTP/1.1
Host: example.com
User-Agent: Chrome/120.0
Accept: application/json

========== RESPONSE ==========
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 456

{ "products": [...] }

4) เบราว์เซอร์และวงจรการเรนเดอร์

1. HTML Parsing

   อ่านโค้ด HTML

   สร้าง DOM Tree

2. CSS Parsing

   อ่านโค้ด CSS

   สร้าง CSSOM Tree

3. Render Tree

   รวม DOM + CSSOM

   เก็บเฉพาะสิ่งที่แสดง (ไม่รวม display:none)

4. Layout (Reflow)

   คำนวณตำแหน่ง/ขนาด

   สร้าง layout tree

5. Paint (Rasterize)

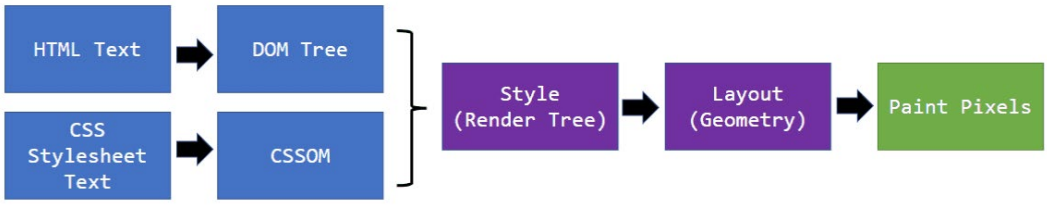
   เพิ่มสี, เงา, border

   แบ่งเป็น layers

6. Composite

   รวม layers

   แสดงผลบนหน้าจอ



https://webperf.tips/tip/browser-rendering-pipeline/

หมายเหตุ: JavaScript สามารถแก้ไข DOM ได้ ⟶ ทำให้ Layout/Paint ทำงานซ้ำ

ตัวอย่าง:

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .box {
        width: 100px;
        background: blue;
      }
    </style>
  </head>
  <body>
    <div class="box">Hello</div>
    <script>
      document.querySelector(".box").style.width = "200px";
      // ↑ ทำให้ Reflow ทำงานอีกครั้ง
    </script>
  </body>
</html>
```

5) เครื่องมือพัฒนาที่ใช้ในรายวิชา

5.1 VS Code (Code Editor)

ติดตั้ง

1. ไปที่ https://code.visualstudio.com/
2. ดาวน์โหลด ⟶ Install → Launch

Extensions แนะนำ

- Live Server → Preview HTML แบบ real-time
- Prettier → Format Code
- ESLint → Detect Code Issues
- HTML Snippets → Auto-complete HTML tags
- CSS Intellisense → CSS suggestions
- Thunder Client → Test API (ถ้าทำ backend)

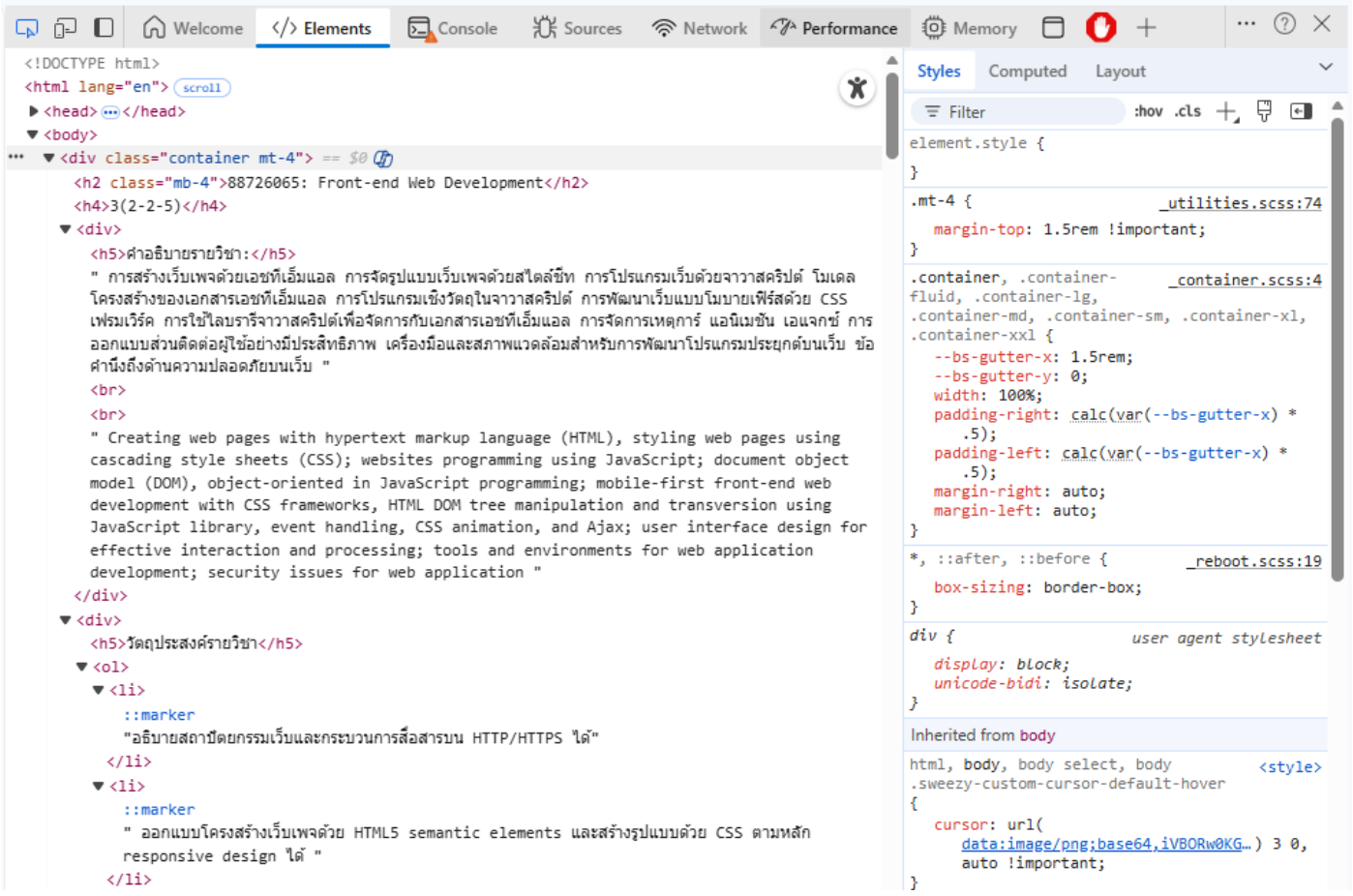
การติดตั้ง Extension

1. คลิก Extensions (ด้านซ้าย)
2. ค้นหาชื่อ
3. Install

5.2 Browser DevTools

เปิด DevTools:

- Windows/Linux: F12 หรือ Ctrl+Shift+I
- Mac: Cmd+Option+I
- Right-click → "Inspect"



Tab สำคัญ:

📄 Elements (Inspector)

ดูและแก้ไข HTML/CSS แบบ live

- Click element selector
- Hover บน element
- ดู CSS rules ที่ใช้
- ดู Computed styles
- Edit HTML (เฉพาะใน memory)

🌐 Network

ดูการ request/response ทั้งหมด

- ชี้ HTTP method (GET/POST)
- ดู Status codes
- ดู Response headers/body
- วัด Load time
- Filter by type (Document/XHR/Image)

💻 Console

รันคำสั่ง JavaScript และดู logs

- console.log('debug message')
- console.error('error')
- console.warn('warning')
- document.querySelector('.box')
- fetch('https://dog.ceo/api/breeds/image/random').then(res => res.json())

ตัวอย่างใช้ Console:

// ดูชื่อหน้า

document.title
→ "Example Page"

// เปลี่ยนชื่อหน้า

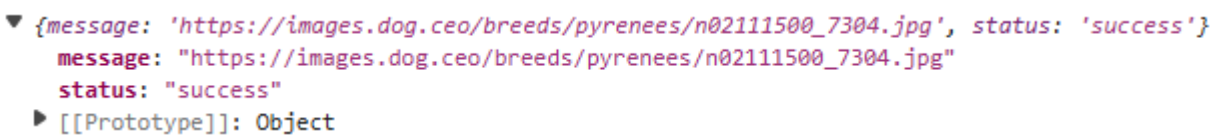
document.title = "New Title"

// หาองค์ประกอบ

document.querySelector('h1')
→ <h1>Hello</h1>

// ดึงข้อมูล JSON จาก API

```
fetch('https://dog.ceo/api/breeds/image/random')
        .then(res => res.json())
        .then(data => console.log(data))
```



5.3 Node.js & npm

Node.js: JavaScript runtime (รัน JS นอก browser)

npm: Package manager (ติดตั้ง libraries)

การติดตั้ง

1. ไปที่ https://nodejs.org/
2. ดาวน์โหลด LTS (Long Term Support)
3. Install → Finish

ตรวจสอบ

```
$ node --version
v20.10.0

$ npm --version
10.2.3
```

ใช้งาน

$ node hello.js	→ รัน script
$ npm install express	→ ติดตั้ง package express
$ npm start	→ รัน script ที่กำหนดใน package.json

5.4 Git & GitHub

Git: Version control (จัดการการเปลี่ยนแปลง)

GitHub: Web platform (เก็บ repositories)

ติดตั้ง Git

1. ไปที่ https://git-scm.com/
2. ดาวน์โหลด ⟶ Install

ตั้งค่า:

```
$ git config --global user.name "Your Name"
$ git config --global user.email "your@email.com"
```

Git Commands พื้นฐาน

$ git init	→ สร้าง repo ใหม่
$ git clone <url>	→ Clone จาก GitHub
$ git status	→ ดูสถานะ
$ git add .	→ เตรียม commit
$ git commit -m "message"	→ บันทึก
$ git push origin main	→ ส่ง GitHub
$ git pull origin main	→ ดึง GitHub
$ git log	→ ดูประวัติ

ตัวอย่าง **Workflow** การทำงาน

1. Clone: git clone https://github.com/user/repo.git
2. Edit: สร้าง/แก้ไขไฟล์
3. Add: git add .
4. Commit: git commit -m "Add feature X"
5. Push: git push origin main

6) DevTools Demo

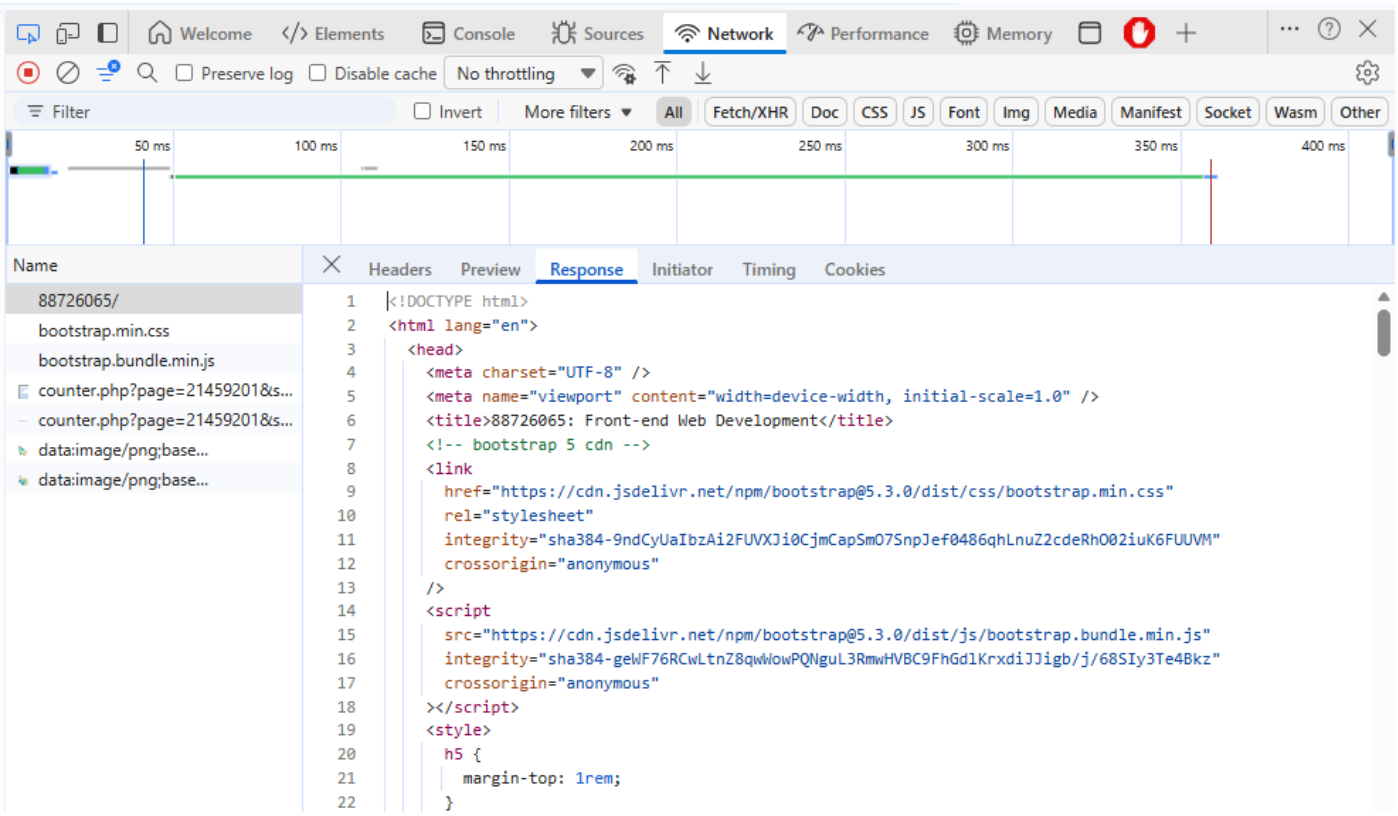
Demo 1: ตรวจสอบ HTTP Request/Response ด้วย Network Tab

ขั้นตอน:

1. เปิด DevTools (F12) → Tab "Network"
2. ไปที่ https://angsila.cs.buu.ac.th/~wittawas/682/88726065/
3. รีเฟรชหน้า (Ctrl+R)
4. ดูรายการ request:
    - Click ไฟล์ HTML แรก
    - ดู Headers: URL, Method, Status Code, Headers
    - ดู Response: HTML content

สิ่งที่เห็น:

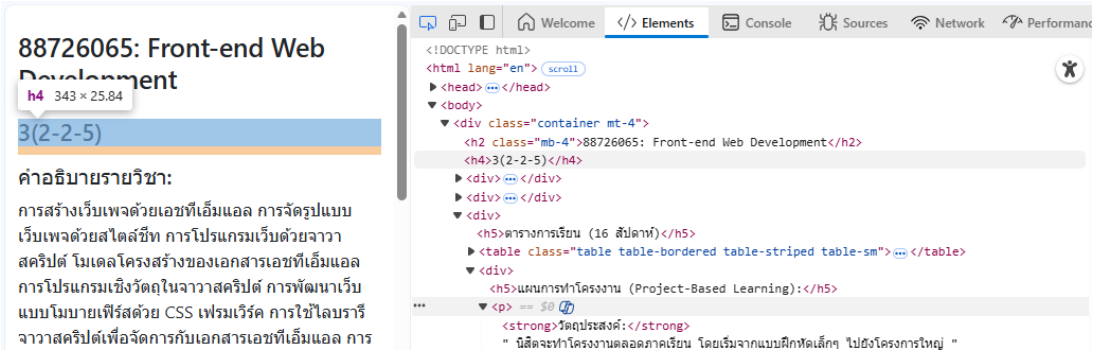
Request: GET / HTTP/1.1
Response: 200 OK
Content-Type: text/html
Content-Length: 1256



Demo 2: ตรวจสอบ HTML Structure ด้วย Elements Tab
ขั้นตอน:

1. Tab "Elements" (หรือ "Inspector")
2. Click "Select element" (มุมบนซ้าย)
3. Hover ที่ heading
4. ดูโครงสร้าง HTML ด้านซ้าย

สิ่งที่เห็น:
<h1 class="title">Example Domain</h1>



Demo 3: รันคำสั่งใน Console
ขั้นตอน:

1. Tab "Console"
2. พิมพ์:

document.title
→ "Example Domain"

document.querySelector('h1').textContent
→ "Example Domain"

document.querySelectorAll('p').length
→ 3

Demo 4: ตรวจสอบ Performance
ขั้นตอน:

1. Tab "Network"
2. ดู "Waterfall" (แสดงลำดับการโหลด)
3. ดู "Load time" (เวลารวม)
4. ดู "Size" (ขนาดไฟล์)

สิ่งที่เห็น:
HTML: 50 KB, 200ms
CSS: 20 KB, 180ms
JS: 100 KB, 300ms
Images: 500 KB, 1000ms

Total: ~1.5s

7) ตัวอย่างโค้ด: บทบาท HTML/CSS/JavaScript

HTML (โครงสร้าง) - index.html

```
<!DOCTYPE html>
<html lang="th">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta name="description" content="สิ่งที่ปรากฏใน Google Search" />
    <title>ตัวอย่างหน้าแรก</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <!-- Semantic HTML -->
    <header>
      <h1>📄 ยินดีต้อนรับเข้า Web Development</h1>
    </header>

    <main>
      <section>
        <h2>บทนำ</h2>
        <p>นี่คือตัวอย่างหน้า HTML เบื้องต้น</p>
      </section>

      <section>
        <h2>ปุ่มโต้ตอบ</h2>
        <button id="myBtn" class="btn btn-primary">คลิกฉัน!</button>
        <p id="result"></p>
      </section>
    </main>

    <footer>
      <p>&copy; 2025</p>
    </footer>

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

หมายเหตุ HTML:

- <!DOCTYPE html> - บอก browser ว่าเป็น HTML5
- <meta charset="UTF-8"> - รองรับภาษาไทย
- <meta name="description"> - ปรากฏใน Google Search
- <link rel="stylesheet"> - เชื่อม CSS
- <script> - เชื่อม JavaScript
- Semantic tags: <header>, <main>, <section>, <footer>

CSS (การนำเสนอ) - style.css

```
/* reset margin and padding */
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

body {
  font-family: "Segoe UI", system-ui, sans-serif;
```

```
  line-height: 1.6;
  color: #333;
  background: #f5f5f5;
}

/* Header */
header {
  background: linear-gradient(135deg, #369 0%, #abc 369%);
  color: white;
  padding: 2rem;
  text-align: center;
}

header h1 {
  font-size: 2rem;
}

/* Main content */
main {
  max-width: 800px;
  margin: 2rem auto;
  background: white;
  padding: 2rem;
  border-radius: 8px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

section {
  margin-bottom: 2rem;
}

section h2 {
  color: #369;
  margin-bottom: 1rem;
}

/* Button styles */
.btn {
  padding: 0.75rem 1.5rem;
  border: none;
  border-radius: 4px;
  font-size: 1rem;
  cursor: pointer;
  transition: background-color 0.3s ease;
}

.btn-primary {
  background: #963;
  color: white;
}

.btn-primary:hover {
  background: #5568d3;
  transform: translateY(-2px);
  box-shadow: 0 4px 8px rgba(102, 126, 234, 0.3);
}

.btn-primary:active {
  transform: translateY(0);
}

/* Result text */
#result {
  margin-top: 1rem;
  padding: 1rem;
```

```
  background: #e3f2fd;
  border-left: 4px solid #667eea;
  border-radius: 4px;
  display: none;
}

#result.show {
  display: block;
}

/* Footer */
footer {
  background: #333;
  color: white;
  text-align: center;
  padding: 1rem;
  margin-top: 2rem;
}
```

หมายเหตุ CSS:

- Flexbox/Grid สำหรับ layout
- Transition สำหรับ smooth animation
- :hover, :active states สำหรับ interactivity
- CSS Variables (optional): --primary-color: #667eea

JavaScript (พฤติกรรม) - script.js

```
// รอให้โหลดเสร็จก่อน
document.addEventListener("DOMContentLoaded", function () {
  // ดึง elements
  const btn = document.getElementById("myBtn");
  const resultDiv = document.getElementById("result");
  let clickCount = 0;

  // เพิ่ม event listener
  btn.addEventListener("click", function () {
    clickCount++;
    resultDiv.textContent = `คุณคลิก ${clickCount} ครั้ง`;
    resultDiv.classList.add("show");

    // เก็บใน localStorage
    localStorage.setItem("clickCount", clickCount);

    // Animate button
    btn.style.transform = "scale(0.95)";
    setTimeout(() => {
      btn.style.transform = "scale(1)";
    }, 100);
  });

  // ดึงจาก localStorage เมื่อโหลดหน้า
  const savedCount = localStorage.getItem("clickCount");
  if (savedCount) {
    clickCount = parseInt(savedCount);
    resultDiv.textContent = `ก่อนหน้านี้คลิก ${clickCount} ครั้ง`;
    resultDiv.classList.add("show");
  }
});
```

หมายเหตุ JavaScript:

- DOMContentLoaded - รอให้ HTML โหลดเสร็จ
- getElementById() - ดึง element
- addEventListener() - รับ event
- classList.add() - เพิ่ม CSS class
- localStorage - เก็บข้อมูล local

ดู HTTP Request/Response Example

========== REQUEST ==========
GET / HTTP/1.1
Host: example.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
Accept: text/html,application/xhtml+xml
Accept-Language: th-TH,th;q=0.9
Connection: keep-alive

========== RESPONSE ==========
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 2456
Cache-Control: max-age=3600
Set-Cookie: sessionId=abc123; Path=/

<!DOCTYPE html>
<html>
<!-- ... HTML content ... -->
</html>

========== SUMMARY ==========
✓ Status: 200 OK
✓ Response Time: 142ms
✓ Total Size: 2.45 KB
✓ Cached for: 1 hour

8) Quiz/Self-Assessment (ประเมินความเข้าใจเบื้องต้น)

ข้อ 1: Concept ทั่วไป

HTTP/HTTPS คืออะไร และแตกต่างกันอย่างไร?

คำตอบ:

HTTP (HyperText Transfer Protocol):
- โปรโตคอลสำหรับสื่อสารบนเว็บ
- Port 80
- ไม่เข้ารหัส (ข้อมูล plain text)
- ล้าสมัย ไม่ปลอดภัย

HTTPS (HTTP Secure):
- HTTP + TLS/SSL encryption
- Port 443
- มีการเข้ารหัสข้อมูล
- ปลอดภัย

ข้อ 2: URL Components

ระบุส่วนของ URL นี้:

https://shop.example.com:8080/products/shirts?color=blue&size=L#reviews

คำตอบ:

https://	→ Scheme
shop.example.com	→ Host/Domain
8080	→ Port
/products/shirts	→ Path
?color=blue&size=L	→ Query String
#reviews	→ Fragment

ข้อ 3: HTTP Methods

ความแตกต่างระหว่าง GET กับ POST:

คำตอบ:

GET:
- ขอข้อมูลจากเซิร์ฟเวอร์
- ข้อมูลใน URL (มองเห็น)
- ไม่ปลอดภัยสำหรับข้อมูลลับ
- ตัวอย่าง: GET /users/123

POST:

- ส่งข้อมูลไปยังเซิร์ฟเวอร์
- ข้อมูลในเนื้อหา request (ซ่อน)
- ปลอดภัยกว่า
- ตัวอย่าง: POST /users + {name, email}

ข้อ 4: Status Codes

Status code เหล่านี้หมายความว่า อะไร?

คำตอบ:

200 → OK (สำเร็จ)

201 → Created (สร้างใหม่สำเร็จ)

301 → Moved Permanently (เปลี่ยนหน้าถาวร)

404 → Not Found (ไม่พบ)

500 → Internal Server Error (เซิร์ฟเวอร์มีข้อผิดพลาด)

ข้อ 5: DevTools

บอกว่าคุณสามารถทำอะไรได้ด้วย Network Tab?

คำตอบ:

- ดูทั้งหมด HTTP requests ที่เกิดขึ้น
- ดู status codes ของแต่ละ request
- ดู request/response headers
- ดู response body
- วัด load time
- ดู file size
- ดู waterfall (ลำดับการโหลด)

ข้อ 6: localStorage

localStorage ใช้ทำอะไร?

คำตอบ:

- เก็บข้อมูลใน browser ของผู้ใช้
- ข้อมูลคงอยู่ถึงแม้ปิด browser
- ไม่ต้องขึ้น server (เร็ว)
- เก็บได้ 5-10 MB
- ตัวอย่างใช้งาน
    * เก็บค่า form
    * เก็บ user preferences
    * เก็บ session tokens

ข้อ 7: Meta Tags

ทำไม meta tags จึงสำคัญสำหรับ SEO?

คำตอบ

- <title> ปรากฏในแถบ browser และ Google Search
- <meta name="description"> แสดง snippet ใน Google
- <meta name="viewport"> ทำให้ responsive
- Open Graph tags: ใช้เมื่อแชร์ใน social media
- ถูกใช้โดย Search Engine Crawlers

9) Glossary (ศัพท์สำคัญ)

ศัพท์	ความหมาย
API	Application Programming Interface - ทางเข้าโปรแกรม
Browser	โปรแกรมเว็บเบราวเซอร์ (Chrome, Firefox, Safari)
Client	อุปกรณ์ผู้ใช้ที่ขอข้อมูล
Cache	หน่วยเก็บข้อมูลชั่วคราวเพื่อความเร็ว
CDN	Content Delivery Network - เซิร์ฟเวอร์กระจาย
DOM	Document Object Model - โครงสร้างที่ JS เข้าถึงได้
DNS	Domain Name System - แปลงชื่อโดเมนเป็น IP
HTML	HyperText Markup Language - ภาษามาร์กอัป
HTTP	HyperText Transfer Protocol - โปรโตคอลเว็บ
HTTPS	HTTP Secure - HTTP ที่เข้ารหัส
IP	Internet Protocol - ที่อยู่เน็ต
JSON	JavaScript Object Notation - รูปแบบข้อมูล
META	ข้อมูลเกี่ยวกับข้อมูล
Port	ประตูการสื่อสารในคอมพิวเตอร์
Query	พารามิเตอร์ค้นหา (ในหาง URL)
Request	การร้องขอข้อมูลจาก client
Response	การตอบกลับข้อมูลจาก server
Server	คอมพิวเตอร์ที่เก็บข้อมูลและตอบสนอง
TLS	Transport Layer Security - เข้ารหัสสื่อสาร
URL	Uniform Resource Locator - ที่อยู่เว็บ

---

10) เตรียมตัวสำหรับ Lab

- อ่านไฟล์ wk01-lab.md เพื่อเข้าใจสิ่งที่จะทำ
- สร้างโฟลเดอร์สำหรับ project week 1
- เตรียม GitHub username/password

ก่อนการปฏิบัติการสัปดาห์นี้:

- ติดตั้ง VS Code → ลองเปิด
- ติดตั้ง Node.js LTS → ตรวจ node --version
- ติดตั้ง Git → ตรวจ git --version
- สร้าง GitHub Account (ถ้ายังไม่มี)
- ติดตั้ง Live Server extension

อ่านเพิ่มเติม:

- MDN: What is HTML?
- MDN: What is HTTP?
- web.dev: How the Web Works
- Git Documentation: Getting Started

---

11) Key Takeaways

✅ **จำสิ่งนี้ให้ดี**

1. **HTTP/HTTPS** = โปรโตคอลสื่อสารบนเว็บ (HTTPS ปลอดภัยกว่า)
2. **URL** = ที่อยู่เว็บ (Scheme + Host + Path + Query)
3. **HTTP Methods** = GET (ขอ), POST (ส่ง), PUT (แก้), DELETE (ลบ)
4. **Status Codes** = 2xx ✓, 3xx ⤴️, 4xx ❌ client, 5xx ❌ server
5. **DevTools** = สำหรับ debug และตรวจสอบ HTTP traffic
6. **HTML/CSS/JS** = โครงสร้าง/นำเสนอ/พฤติกรรม
7. **localStorage** = เก็บข้อมูลใน browser

⚠️ **ยังไม่ต้องกังวล**

- ยังไม่ต้องเขียน JavaScript มากมาย
- ยังไม่ต้องเข้าใจ backend/server
- ยังไม่ต้องเข้าใจ async/await (สัปดาห์ที่ 9)
- ยังไม่ต้องเข้าใจ Framework (Vue/React)

---

12) เอกสารอ้างอิง

Official Documentation

- MDN Web Docs - หลักสูตรเรียนรู้อย่างเป็นทางการ
- W3C HTML Specification - มาตรฐาน HTML
- HTTP/2 Specification - มาตรฐาน HTTP

Tools

- VS Code Documentation
- Chrome DevTools Documentation
- Git Documentation
- GitHub Docs

Learning Resources

- freeCodeCamp - Web Development
- Khan Academy - Internet
- web.dev - Learn

---

13) FAQ & Troubleshooting

Q: ทำไมใช้ HTTPS แทน HTTP?

A: เพราะ HTTPS เข้ารหัสข้อมูล ป้องกันการแอบดูรหัสผ่าน/ข้อมูลส่วนตัว Google ยังไม่แนะนำให้ใช้ HTTP อีกต่อไป

Q: ความแตกต่าง GET vs POST?

A: GET ขอข้อมูล (ข้อมูลเห็นใน URL) VS POST ส่งข้อมูล (ข้อมูลซ่อนในเนื้อหา)

Q: ทำไม DevTools ถึงสำคัญ?

A: DevTools ช่วยให้มองเห็น Request/Response, Error, Console logs ทำให้ debug ได้ง่าย

Q: localStorage คือ cache หรือเปล่า?

A: ไม่เหมือน localStorage ถูกควบคุมโดย JavaScript, cache ถูกควบคุมโดย HTTP headers

Q: ต้องเรียนรู้ SEO ไปทำไม?

A: เพื่อให้เว็บเราปรากฏใน Google Search ซึ่งใช้ meta tags เพื่ออ่านเนื้อหา
