

# Fundamental Web Programming

## Midterm by Arm IT20



### Unit1 - Introduction to Internet & Web system

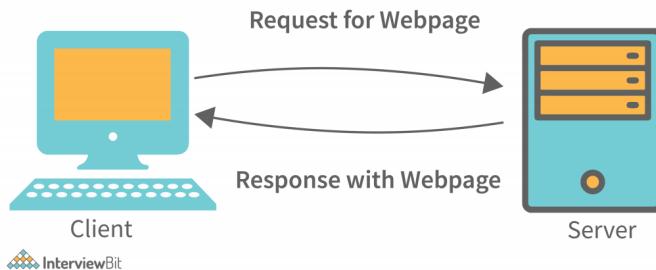
#### Internet

- เครือข่ายที่เชื่อมต่อกันกั่วโลก
- ไม่มีใครเป็นเจ้าของ Internet
- ใช้กำอะไรบ้าง? - Email, Social network, etc.

#### The World Wide Web

- Tim Berners-Lee เป็นผู้คิดค้น world wide web
- ทำงานโดยการส่ง HTML documents ด้วย Internet ผ่าน web server ไปที่ web browser
- **web pages** คือ ข้อมูลที่แสดงบน Internet 1 หน้า
- **web site** คือ web pages จำนวนมากมาประกอบเข้าด้วยกัน โดยแต่ละ web pages อาจเชื่อมโยงถึงกันได้

# **Client-Server Architechute**



## Web Server & Web Browser

- **Web Server (Server-side)** คือ ส่วนที่ใช้สำหรับรับคำขอ(request) จาก Web Client และ ทำงานตามคำขอที่ได้รับ หลังจากนั้นจะทำการส่งผลลัพธ์(response) กลับไปยัง web browser
  - **Web Browser (Client-side)** คือ ส่วนที่ส่งคำร้องขอ(request) ไปยัง web-server และเมื่อได้รับ response จาก web-server จะทำการประมวลผลและแสดงให้ผู้ใช้เห็น

## OSI model

- **Application layer** จัดการประมวลผลในรับ-ส่งข้อมูล เช่น การเข้าหน้า website จะใช้ HTTP (เรียกข้อมูลใน layer นี้ว่า message)
  - **Presentation layer** ควบคุมการบีบอัดและถอดรหัสข้อมูล
  - **Session layer** ควบคุมการเริ่ม-หยุดของการรับและส่งข้อมูล
  - **Transport layer** ควบคุมวิธีการรับส่งข้อมูล กำหนด protocol ว่าจะใช้อะไร (เรียกข้อมูลใน layer นี้ว่า Segment)
  - **Network Layer** หากเส้นทางในการรับ-ส่งข้อมูล(ระบุ IP Address), แบ่งข้อมูลออกเป็น Package (เรียกข้อมูลใน layer นี้ว่า Datagram)
  - **Link layer** แบ่ง Package ให้เล็กลง เช็คและซ่อน error ข้อมูล, ระบุ mac address ที่ใกล้เคียง (เรียกข้อมูลใน layer นี้ว่า frame)
  - **Physical layer** ส่งข้อมูลผ่าน physical(Ex. สายไฟ) เป็น Bit (10100100)

# Essential Protocols

- **TCP (Transmission Control Protocol)** คือ Protocol ที่ใช้ในการส่งข้อมูล โดยจะมีการตรวจสอบความถูกต้องว่าข้อมูลที่ส่งไปถูกต้องหรือไม่ (ส่งข้อมูลแบบเน้นถูกต้อง)
- **IP (Internet Protocol)** จะใช้ในการกำหนด IP Address ให้แต่ละเครื่อง ในการส่งข้อมูล หรือคือกำหนดตัวบานาและปลายทาง
- **UDP (User Datagram Protocol)** ทำหน้าที่คล้ายๆ TCP แต่จะไม่มีการตรวจสอบความถูกต้องของข้อมูลให้ (ส่งข้อมูลเน้นเร็ว)
- **Dynamic Host Configuration Protocol (DHCP)** ช่วยให้การ Configuration ก่อนจะเข้าถึง Internet เป็นไปแบบอัตโนมัติ
- **FTP(File Transfer Protocol)** ใช้เพื่อ copy file จาก host หนึ่งไปยังอีกหนึ่ง
- **HTTP (Hypertext Transfer Protocol)** คือ protocol ที่ใช้สื่อสารกันระหว่าง web server และ web client รู้จักกันในชื่อ “**request-response model**”
- **HTTP Secure (HTTPS)** คือล้วงเสธของ HTTP ที่ช่วยให้ปลอดภัยมากขึ้น โดยจะเพิ่บใน Encrypt โดยอาศัยเทคโนโลยี SSL (Secure Sockets Layer) หรือ TLS (Transport Layer Security)
- **DNS (Domain Name System)** คือส่วนที่ทำการแปลงชื่อ website หรือเรียกว่า domain name (ex. www.google.com) เป็น IP Address (ex. 172.217.26.68) โดย DNS จะทำการจับคู่ระหว่าง Domain name กับ IP Address

## Web Development

1. **Front-end Development** คือส่วนของการพัฒนา website ที่เน้นไปที่การพัฒนาส่วน user interacts หรือเรียกรวบๆว่าพัฒนาในส่วน client-side
2. **Back-end Development** คือส่วนของการพัฒนา website ที่เน้นไปที่การพัฒนาในการที่ user ไม่สามารถมองเห็นและตอบโต้ได้ หรือเรียกรวบๆว่าพัฒนาในส่วน server-side
3. **Full-stack Development** ทำทั้ง front-end และ back-end

## Front-end Technologies

1. **HTML (Hypertext Markup Language)** ใช้เพื่อสร้างโครงหน้าและใส่ข้อมูลใน website
2. **CSS (Cascading Style Sheets)** ใช้ในการตกแต่งหน้า website หรือเป็นส่วนที่กำหนดว่าหน้า website จะแสดงออกไปยังไง
3. **JavaScript** ใช้ในการทำให้ user สามารถ interact กับ website เช่น เมื่อกดปุ่มจะให้เกิดอะไรขึ้น

4. **AJAX (Asynchronous Javascript and XML)** ใช้เพื่อติดต่อกับฝั่ง web server

## Back-end Technologies

1. **PHP, Python, Ruby, Java, Golang, C#, Javascript** ภาษาที่คนส่วนมากใช้ในการพัฒนาฝั่ง back-end
2. **Node.js** เป็นตัวที่ทำให้นำ Javascript มารันนอก web browser ได้ โดยสามารถใช้ตัวนี้ในการพัฒนา back-end service หรือที่เรียกว่า API ได้
3. **API (Application Programming Interface)** เป็นตัวเชื่อมต่อระหว่าง front-end และ back-end โดย API จะกำหนดว่าโปรแกรมกังส่องสามารถส่งข้อมูลระหว่างกันได้อย่างไร ข้อมูลจะໄດ້สามารถแลกเปลี่ยนกันได้
4. **Format of Data** รูปแบบข้อมูลที่ใช้ในส่วน backend จะเป็น
  - **XML (Extensible Markup Language)** จะเป็นรูปแบบข้อมูลที่กำหนดด้วย tag คล้ายๆ HTML ผุดจ่ายๆ คือ เมื่อน PDF ก็คอมพิวเตอร์อ่านได้
  - **JSON (JavaScipy Object Notation)** จะเก็บข้อมูลในรูปแบบโครงสร้าง

## **Unit2 - HTML and CSS**

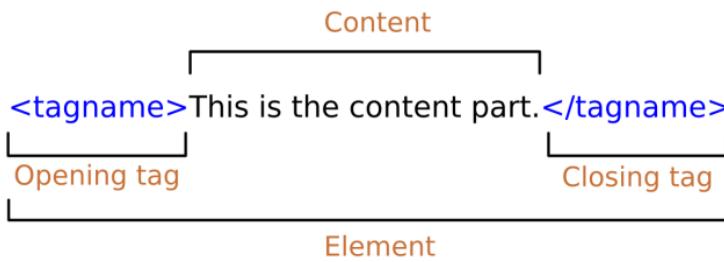
---

### Introduction to HTML

- **HTML (Hypertext Markup Language)** คือ ภาษาพื้นฐานในการสร้าง web pages โดยจะแสดงในรูปแบบ ข้อความ, รูปภาพ, input, etc.
- **Hypertext** ข้อความที่สามารถเชื่อโยงไปยังหน้าอื่นๆ ได้
- **Markup Language** คือภาษาที่ใช้ tag ในการแสดง content

### Element Tags

คือ ส่วนหลักในการสร้าง content ในภาษา HTML โดย structure ประกอบไปด้วย



## Element Attribute

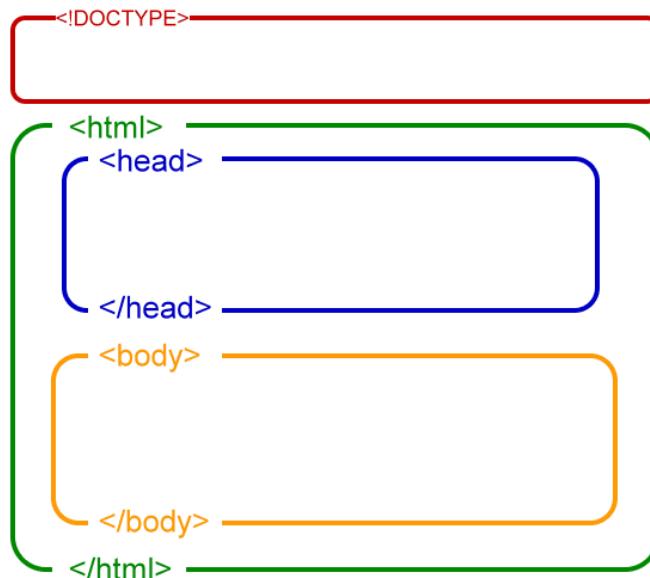
คือข้อมูลเพิ่มเติมที่ใส่进去ใน tag เพื่อวัตถุประสงค์ต่างๆ

```
<tagname attr1="value1" attr2="value2" >
  My text content
</tagname >
```

**For example**

```
<p id="intro">This is my first paragraph.</p>
```

## HTML Document Basic Structure



- **ส่วนสีแดง** เป็นการประกาศว่าเป็นเอกสาร HTML
- **ส่วนสีเขียว** คือ html element จะเป็นการกำหนดจุดเริ่มและจุดสิ้นสุดของเอกสาร เช่น

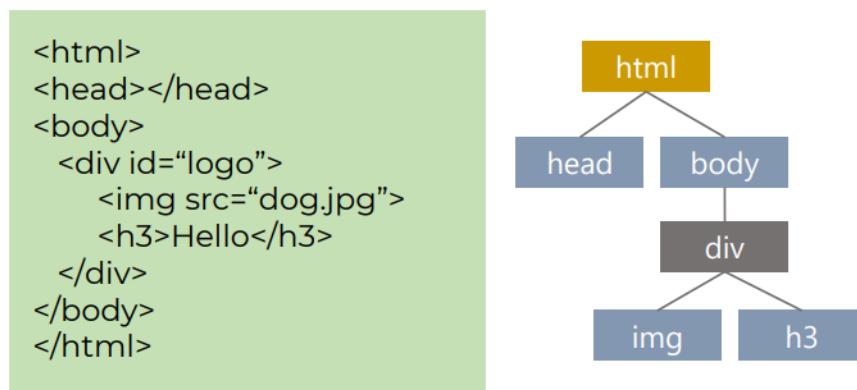
- เก็บวิธีการแสดงผลหรือ CSS ด้วย `<style>`
- เชื่อมต่อ resource ข้างนอกด้วย `<link>`
- เก็บหัวข้อด้วย `<title>`
- เก็บวิธีการ encode หรือ keyword ที่ใช้ในการ search ด้วย `<meta>`



- ส่วนสีน้ำเงิน คือ head element จะทำหน้าที่เก็บรายละเอียดต่างๆของเอกสาร
- ส่วนสีเหลือง คือ body element จะทำหน้าที่เก็บข้อมูลที่จะแสดงใน browser

## Document Hierarchy

คือการที่มี tag ซ้อน tag กัน ซึ่งสามารถอธิบายได้ด้วย tree



จากภาพ `<div id="logo">` จะมี `<img>` และ `<h3>` ซ้อนอยู่ข้างใน ซึ่งเราจะสรุปได้ว่า

- `<div id="logo">` เป็น parents ของ `<img>` และ `<h3>`
- `<img>` และ `<h3>` เป็น children ของ `<div id="logo">`

## Introduction to CSS

- **CSS (Cascading Style Sheets)** ใช้เพื่อควบคุมการแสดงผลของ HTML elements ซึ่ง css style สามารถใส่ใน HTML โดยตรงได้
- **A style sheet** คือการกำหนด “rules” ให้กับ HTML element โดย rules จะกำหนดว่า HTML element แต่ละอันจะแสดงออกไปยังไงเมื่ออยู่ใน browser
- **Cascade** เป็นกฎที่ใช้กำหนดว่า css ตัวไหนจะถูกนำไปใช้กับ HTML element เมื่อมันซ้อนกับกัน



```

● ● ●

<style>
  p {
    color: red;
  }

  .text {
    color: blue;
  }
</style>

<p class="text">hello world</p>

```

จะเห็นได้ว่า มีการกำหนดสีซ้อนกัน โดยในกรณีนี้ จะใช้กฎ cascade ในการเลือกสีได้หนึ่ง ชั้นในกรณีนี้จะเป็นสีน้ำเงิน

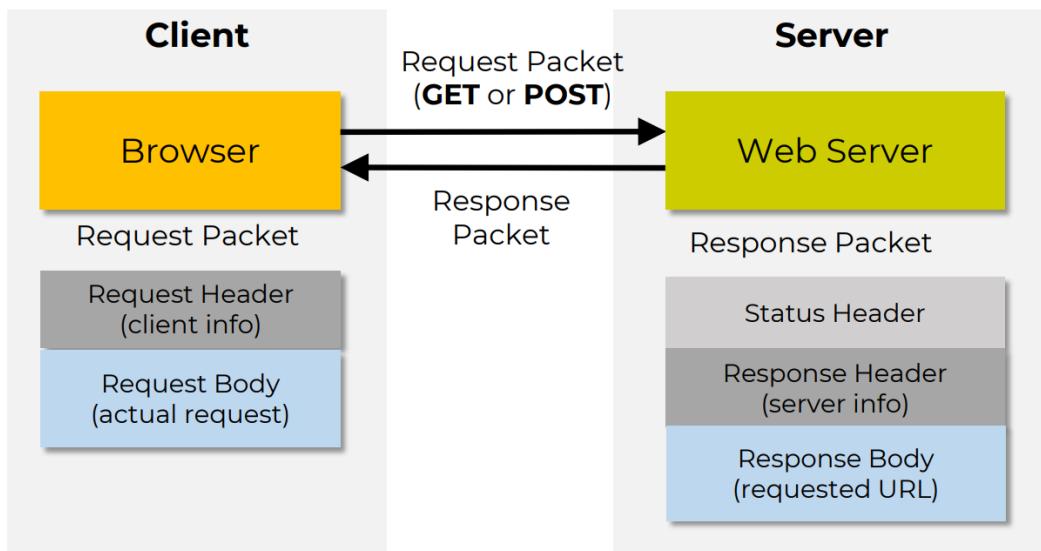
## Type of Style Sheet

- **Browser Style Sheet** คือ css ก็ตัว browser (ex. chrome edge opera) ใส่ให้เอง
- **External Style Sheet** คือ css ภายนอกที่เราใส่เข้าไปใน HTML document เช่น การเขียน css แยกมาเป็นอีกไฟล์ เช่น style.css และนำมา import ใน HTML document โดยใช้ `<link href="style.css" rel="stylesheet" />`
- **Internal / Embedded Style Sheet** คือการกำหนด css ภายใน tag `<style>` ใน HTML Document
- **Inline Styles** คือการกำหนด css โดยตรงใน tag เช่น `<h1 style="color:blue;">this is text</h1>`

## Introduction

**JavaScript** เป็นภาษาการเขียนโปรแกรมแบบสคริปต์ (interpreted) ที่ใช้เพื่อสร้างปฏิสัมพันธ์และไดนามิกให้กับหน้าเว็บ โดยจะสามารถเปลี่ยนเนื้อหาบนเว็บไซต์ได้โดยที่ไม่จำเป็นต้องกด refresh หน้าใหม่

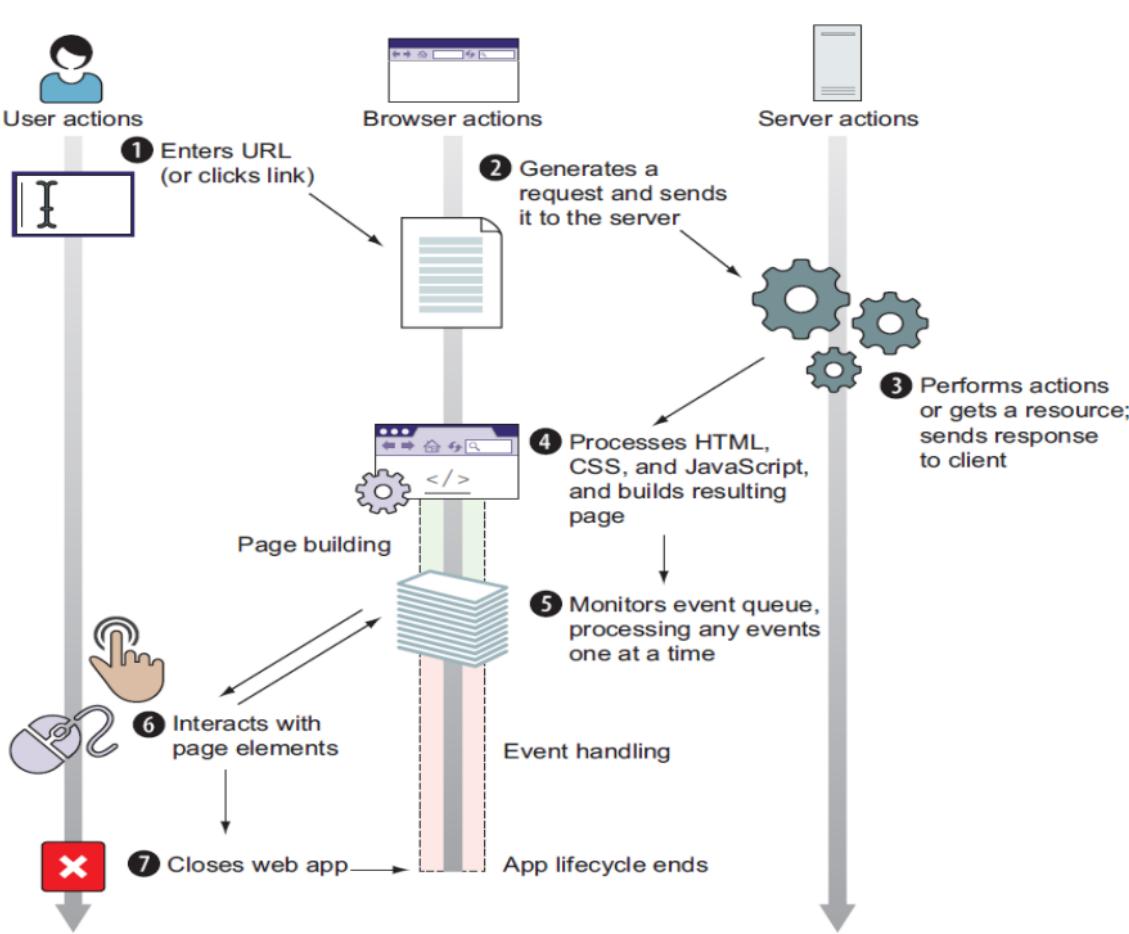
## Request-Response Model



- **GET Method** ข้อมูลจะปรากฏอยู่บน Adress bar และ body
- **POST Method** ข้อมูลจะอยู่ในส่วนของ body โดยข้อมูลจะมีความ private มาากกว่า

## Client-side scripting benefits

- ทำให้ผู้ใช้สามารถ interact กับ HTML และ CSS ได้
- ลดภาระให้กับ server เนื่องจากรับแบบฟัง client (ใช้ hardware ของ user ในการประมวลผล)
  - ลดการ request-response กับฟัง server เนื่องจากโค้ดบางส่วนรับแบบ client ได้เลย
  - เนื่องจากโค้ดรับแบบ client การประมวลผลจะเร็ว ไม่ต้องรอ response จาก server
- สามารถโต้ตอบกับ user ได้ (ทำ event handling ได้) เช่น คลิกเม้าส์, กดปุ่ม



## JavaScript Framework

คือเครื่องมือที่ทำให้เราสามารถทำงานกับภาษา JavaScript ได้ง่ายขึ้น ช่วยลดขั้นตอนในการทำงาน และการเขียนโค้ด ซึ่งมีหลากหลายมาก เช่น

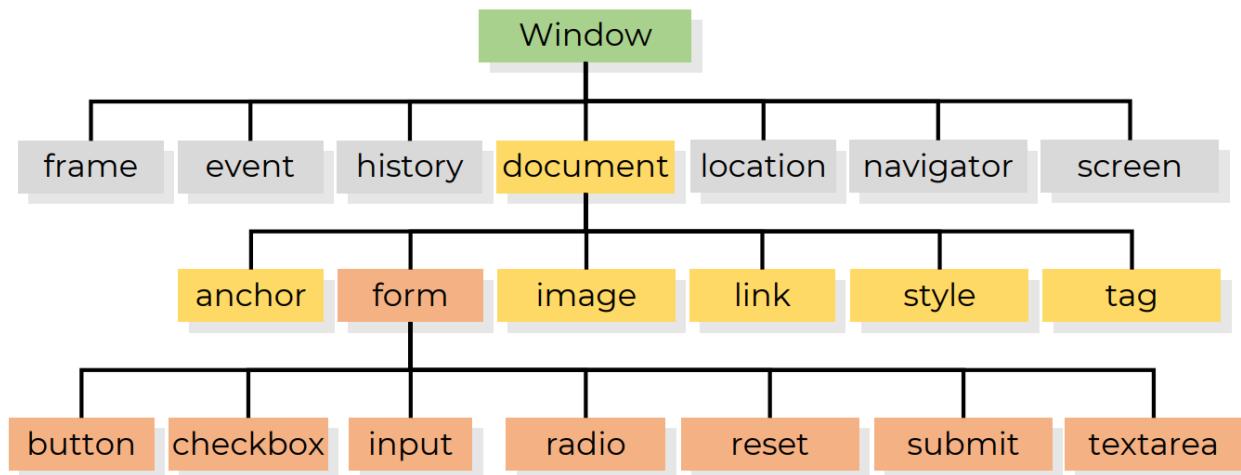
- React.js
- jQuery
- Node.js
- etc.

## Unit4 - Document Object Model

**Document Object Model (DOM)** เป็นโครงสร้างข้อมูลแบบลำดับชั้นที่แสดงเนื้อหาของหน้าเว็บ ประกอบด้วยองค์ประกอบ HTML กั้งหนด ซึ่งเราสามารถใช้ JavaScript ในการจัดการ DOM เพื่อ

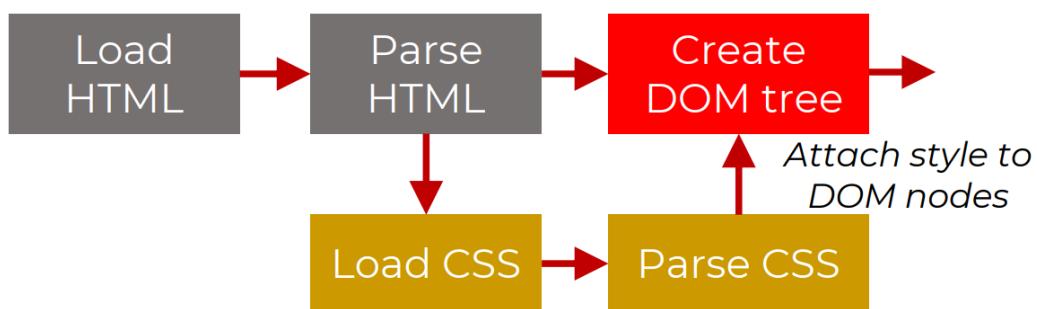
เปลี่ยนองค์ประกอบของ HTML Document ได้ เช่น สร้าง ลบ แก้ไข element ต่างๆใน HTML

- ทุกๆ element ใน HTML จะถูกเปลี่ยนเป็น Object (concent OOP เลย)
- เมื่อเป็น object แล้ว เราสามารถเปลี่ยนแปลง properties หรือเรียกใช้ method ได้
- จึงมองโครงสร้าง HTML ต่างๆเป็น tree



## **Browser and DOM**

เมื่อ browser ได้รับ HTML Document มา ก่อนที่จะแสดงผล browser จะทำการแปลงให้เป็น DOM tree ก่อนถึงจะแสดงผลออกไป



## **ตัวอย่าง**

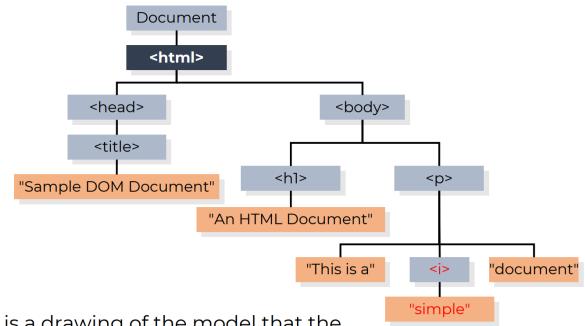
```

● ● ●

<html>
  <head>
    <title>Sample DOM Document</title>
  </head>
  <body>
    <h1>An HTML Document</h1>
    <p>This is a <i>simple</i> document.</p>
  </body>
</html>

```

<i> ทำให้ตัวหนังสืออ่านง่าย



This is a drawing of the model that the

## Type of DOM Nodes

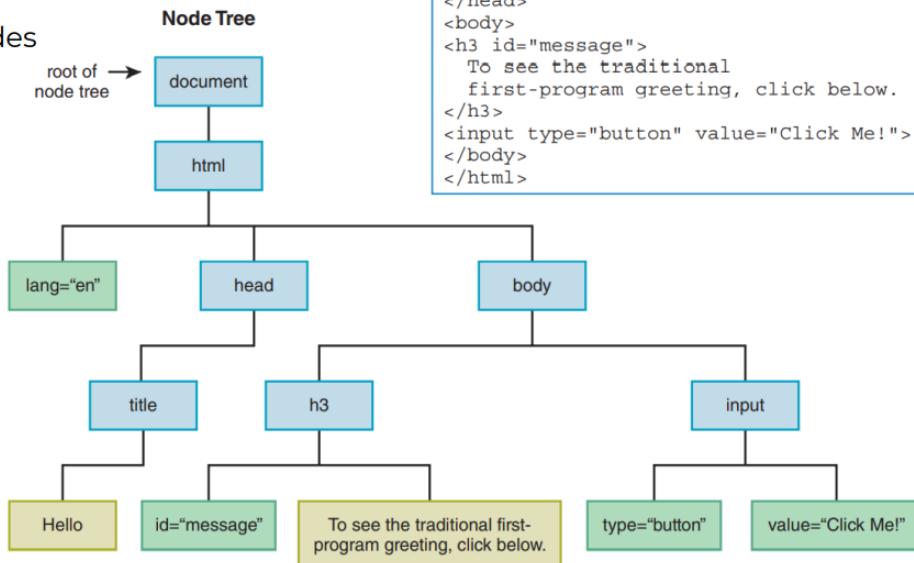
1. **Document node** จุดเริ่มต้นของ tree
2. **Element Node** HTML tag ต่างๆ (สีฟ้า)
3. **Attribute Node** บ่งบอกถึง attribute ของ element node (สีเขียว)
4. **Text Node** หมายถึงข้อความที่อยู่ใน element ต่างๆ (สีเหลือง)
5. **Comment Node** ส่วนที่ comment ไว้
6. **DocumentType** <!DOCTYPE html>

**FIGURE:** Node tree for web page

blue: element nodes

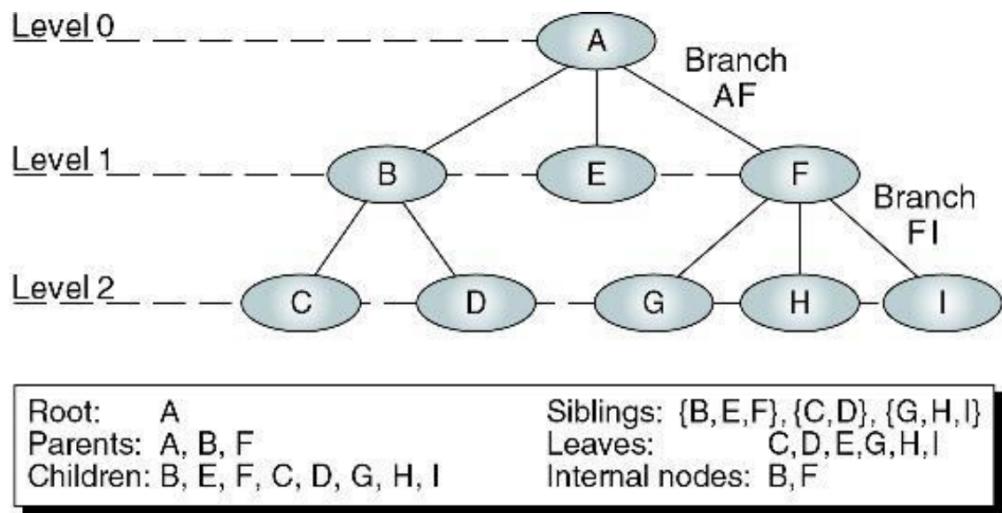
yellow: text nodes

green: attribute nodes



## Relationship among Nodes

- **Root node** គឺជា Node កំពុងស្តុត
  - **Parent node** គឺជា Node ដែលមាន child node
  - **Child node** គឺជា Node ដែលត្រូវការយកចិត្តពី Parent node
  - **Leaf** គឺជា Node ដែលមិនមែនជាកុងការយកចិត្តទេ
  - **Siblings** គឺជាលោកស្រីប្រភេទនៃ node ដែលមាន parent តួនាទីគ្មាន
  - ក្នុងការបង្កើតប្រព័ន្ធដែលមិនមែនជាកុងការយកចិត្តទេ



# Unit5 - Web Data and Local Storage

# JSON

JSON (JavaScript Object Notation) เป็นรูปแบบข้อมูลแบบข้อความที่ใช้เพื่อแลกเปลี่ยนข้อมูลระหว่างแอปพลิเคชันต่างๆ โดยใช้ข้อความธรรมด้า รูปแบบ JSON นั้ง่ายต่อการอ่านและเขียนทั้งโดยมนุษย์และโปรแกรมคอมพิวเตอร์

- ຈັດເກີບເປັນຮູບແບບ key/value(object) ຂີ່ອ ສequence(array)
  - ສ່ວນໃຫຍ່ການສ້ອສາຣະກ່າວງ web server ແລະ web client ຈະໃຊ້ຂໍ້ມູນຮູບແບບນີ້

## ตัวอย่าง 1

```
{  
    "firstname" : "Puwit",  
    "lastname" : "Nunpan",  
    "nickname" : "Arm",  
    "studentID" : "65070185"  
}
```

ในตัวอย่าง JSON จะเก็บเป็น Object เช่น

- key คือ “firstname”
- value คือ “Puwit”

## ตัวอย่าง 2

```
["KMITL", "KMUTT", "KMUTNB"]
```

ในตัวอย่าง JSON จะเก็บเป็น Array ที่มีสมาชิก 3 ตัว

## Convert table to JSON format (ให้ใน lab5 ข้อ 1)

|   | Company             | Contact         | Country |
|---|---------------------|-----------------|---------|
| 1 | Alfreds Futterkiste | Maria Anders    | Germany |
| 2 | Centro commercial   | Francisco Chang | Mexico  |
| 3 | Ernst Handel        | Roland Mendel   | Austria |

1. ดูว่ามี column อะไรบ้างตามตัวอย่างคือ company, contact, country
2. แบ่งข้อมูลว่ามีกี่ชุดตามตัวอย่างคือ 3 ชุด
3. ໄล่ทำกีลະชุด โดยให้ชื่อ column เป็น key และ ข้อมูลเป็น value โดยข้อมูลแต่ละชุดจะเก็บเป็น object

| Company   | Contact      | Country |
|---|--------------|---------|
| Alfreds Futterkiste   | Maria Anders | Germany |
| {   |              |         |
| 1 "company": "Alfreds Futterkiste",<br>"contact": "Maria Anders",<br>"country": "Germany" |              |         |
| }   |              |         |

4. เมื่อทำเสร็จทุกชุดแล้ว ให้นำข้อมูลทุกชุดไปเก็บไว้ใน array

```
[  
  {  
    "company": "Alfreds Futterkiste",  
    "contact": "Maria Anders",  
    "country": "Germany"  
  },  
  {  
    "company": "Centro Commercial",  
    "contact": "Francisco Chang",  
    "country": "Mexico"  
  },  
  {  
    "company": "Ernst Handel",  
    "contact": "Roland Mendel",  
    "country": "Austria"  
  }]
```

## Local Storage

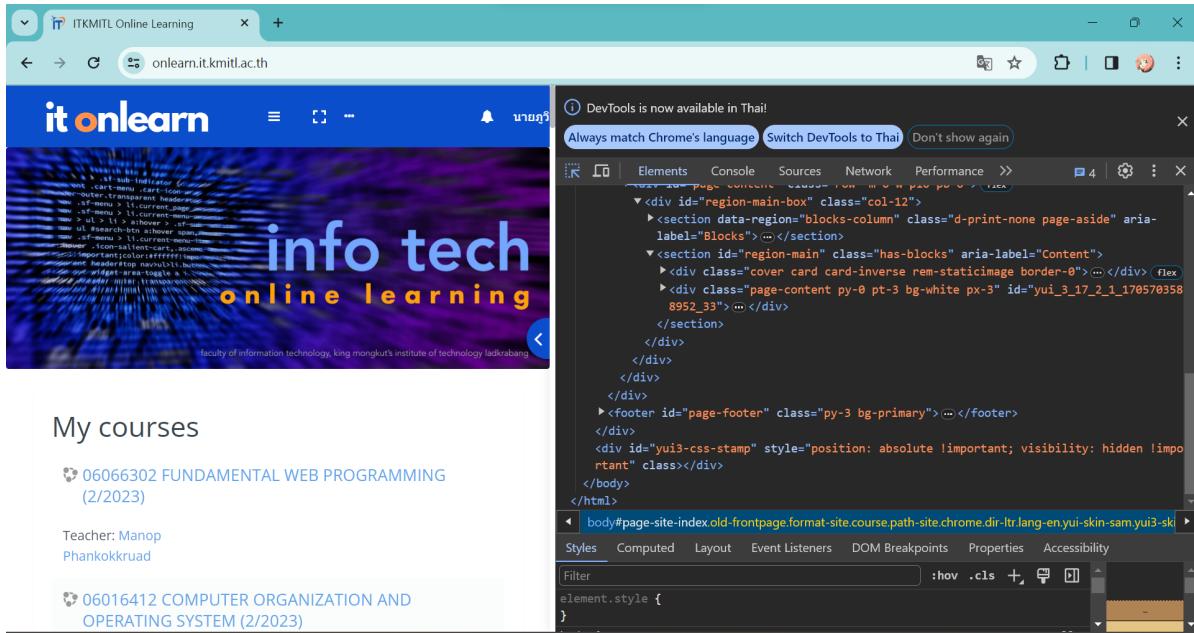
จะเป็นการจัดเก็บข้อมูลของ browser ในรูปแบบ key/value ในฝั่งของ client โดยข้อมูลประเภท localStorage นั้น จะถูกจัดเก็บบนเครื่องของ client หรือคือข้อมูลจะถูกเก็บใน storage ของคอมพิวเตอร์ของผู้ใช้บันทึก

- ข้อมูลที่เก็บจะแยกเว็บไซต์กัน เช่น localStorage ของ onlearn เว็บไซต์ที่จะเรียกใช้ข้อมูลนี้นั้น จะมีแค่ onlearn เว็บอื่นจะไม่สามารถเรียกใช้ข้อมูลตรงนี้ไม่ได้
- ถ้าข้อมูลใน localStorage จะถูกเก็บไว้ตลอดแม้เราจะเปิดเครื่องหรือปิดหน้าเว็บนั้นไป ถ้าไม่ถูกลบออก

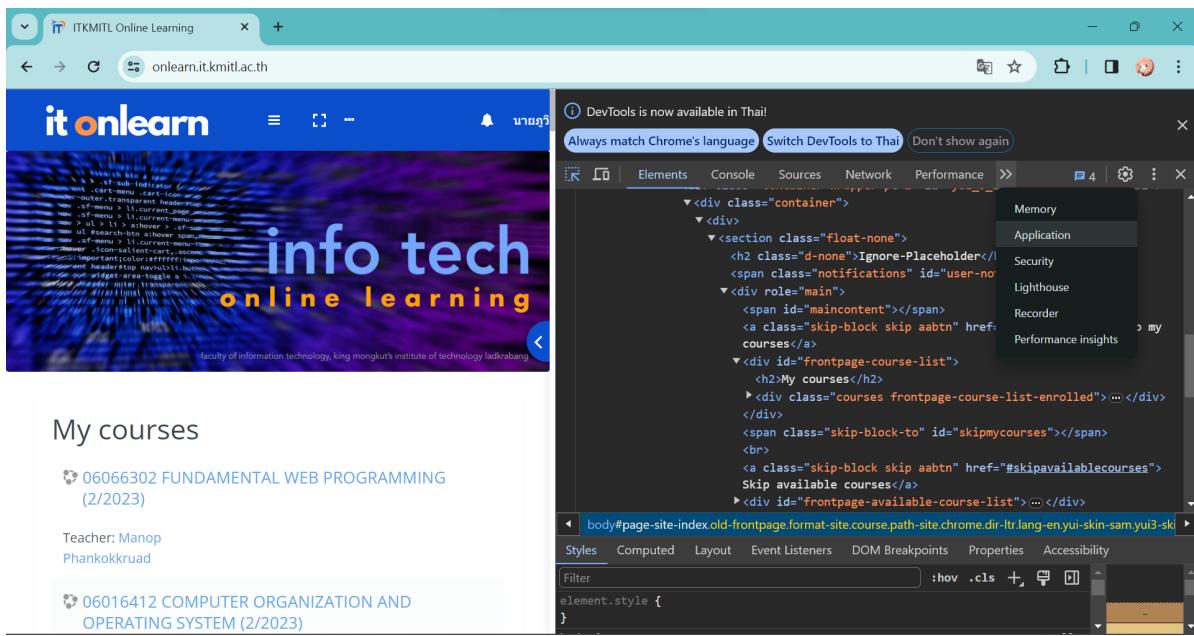
## วิธีดู localStorage ของแต่ละ Website

- อยากรู้ localStorage ของ onlearn

## 2. กด **ctrl+shift+c** เพื่อเปิดหน้า developer tools (ใน safari เนื่องจากทำไม่ได้)



## 3. กด **>>** ข้างๆ tab performance และเลือก Application



## 4. ในเมนู Local Storage กดให้มันแสดงชื่อเว็บ onlearn หลังจากนั้นกดเข้าไปดู ก็จะเห็น localStorage ทุกอันของ Website นี้

The screenshot shows a web browser window with the URL [onlearn.it.kmitl.ac.th](https://onlearn.it.kmitl.ac.th). The main content area displays the 'info tech online learning' logo. Below it, there's a section titled 'My courses' listing two courses:

- 06066302 FUNDAMENTAL WEB PROGRAMMING (2/2023)**
- Teacher: Manop Phankokkrud**
- 06016412 COMPUTER ORGANIZATION AND OPERATING SYSTEM (2/2023)**

To the right of the main content is the Chrome DevTools interface, specifically the Application tab. It shows the storage logs for the current origin. The 'Storage' section is expanded, showing 'Local storage' entries for the domain <https://onlearn.it.kmitl.ac.th>. The log includes items such as 'core\_template/167574...', 'core\_str/savechanges...', 'core\_str/showless...', and 'core\_str/loading/core/th'. The 'Background services' section also lists 'Back/forward cache' and 'Background fetch'.

## Unit 6 - Responsive Web Design

คือการทำให้เว็บไซต์แสดงผลลัพธ์ได้หลายขนาดหน้าจอ เช่น

เว็บที่ทำ responsive

The screenshot shows two views of the 'ToBeIT67 The Second' website. The left view is in landscape mode, and the right view is in portrait mode. Both views show the same content but adapted to their respective screen sizes. The header features the text 'ToBeIT67 THE SECOND' and 'THE SECOND'. Below the header, there is a dark background image of a cathedral or church interior. The main text on the page reads:

สืบสานภารกิจ ก้าวต่อไป สำหรับนักศึกษาและอาจารย์ ที่มีความสนใจในเทคโนโลยีและอาชญากรรม ในการพัฒนาการเดินทางสู่โลกไซเบอร์ ที่มีความปลอดภัย และการเชื่อมต่อ Hackathon ถูกกำหนดให้กับกลุ่มประเทศที่มีความเชี่ยวชาญใน

ปัจจุบันและอนาคต

เรียนรู้เพื่อเติบโต

ดูรายละเอียดเพิ่มเติม

ปัจจุบันและอนาคต

เรียนรู้เพื่อเติบโต

ดูรายละเอียดเพิ่มเติม

## เว็บที่ไม่ทำ responsive

# Coding Session !

## HTML

- **Comment** `<!-- Comment -->` ใช้ในการอธิบาย code โดย code กี่ถูก comment จะไม่ถูกแสดงผล
  - **Heading** `<h1> สิ่ง <h6>` ใช้แสดงข้อความที่เป็นหัวข้อ โดยจะมี 6 ขนาด
  - **Paragraph** `<p>` ใช้แสดงข้อความทั่วไป
  - **List**
    - **List item** `<li>` ใช้แสดงสมาชิกใน list
    - **Unorder list** `<ul>` ใช้แสดง list เป็นแบบไม่มีลำดับ (เช่น แสดงเป็นจุด หรือขีด)



```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

- Coffee
- Tea
- Milk

- **Order list** `<ol>` ใช้แสดง list เป็นแบบ มีลำดับ (1 2 3)



```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

1. Coffee
2. Tea
3. Milk

- **Anchor** `<a>` ใช้เพื่อสร้างลิงค์ไปยังหน้าเว็บ หรือไฟล์อื่นๆได้ โดยจะกำหนดเป้าหมายใน attribute href



```
<a href="www.google.com">Go to Google</a>

<a href="about.html">Go to About Page</a>
```

- **Image** `<img>` ใช้แสดงผลรูปภาพใน website โดยจะกำหนดรูปที่ต้องการใน attribute src



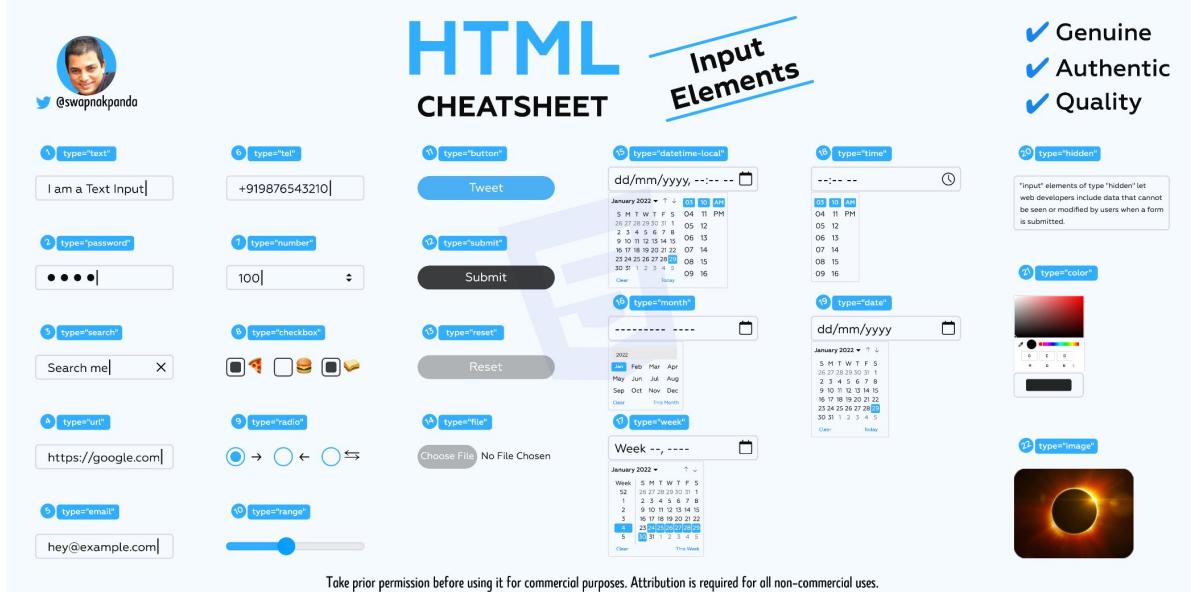
```



```

- **Division** `<div>` เป็น tag ที่ใช้จัดกลุ่มข้อมูลต่างๆ ออกจากกันเพื่อให้ง่ายต่อการใช้งาน เช่น จัด Layout website

- **Input** `<input>` ใช้ในการรับข้อมูลจาก User โดยจะมีหลายรูปแบบ เช่น รับเป็นข้อความ, ตัวเลข, รหัสผ่าน โดยสามารถกำหนดได้ใน attribute ที่ชื่อว่า type



## CSS

- **CSS Selector**

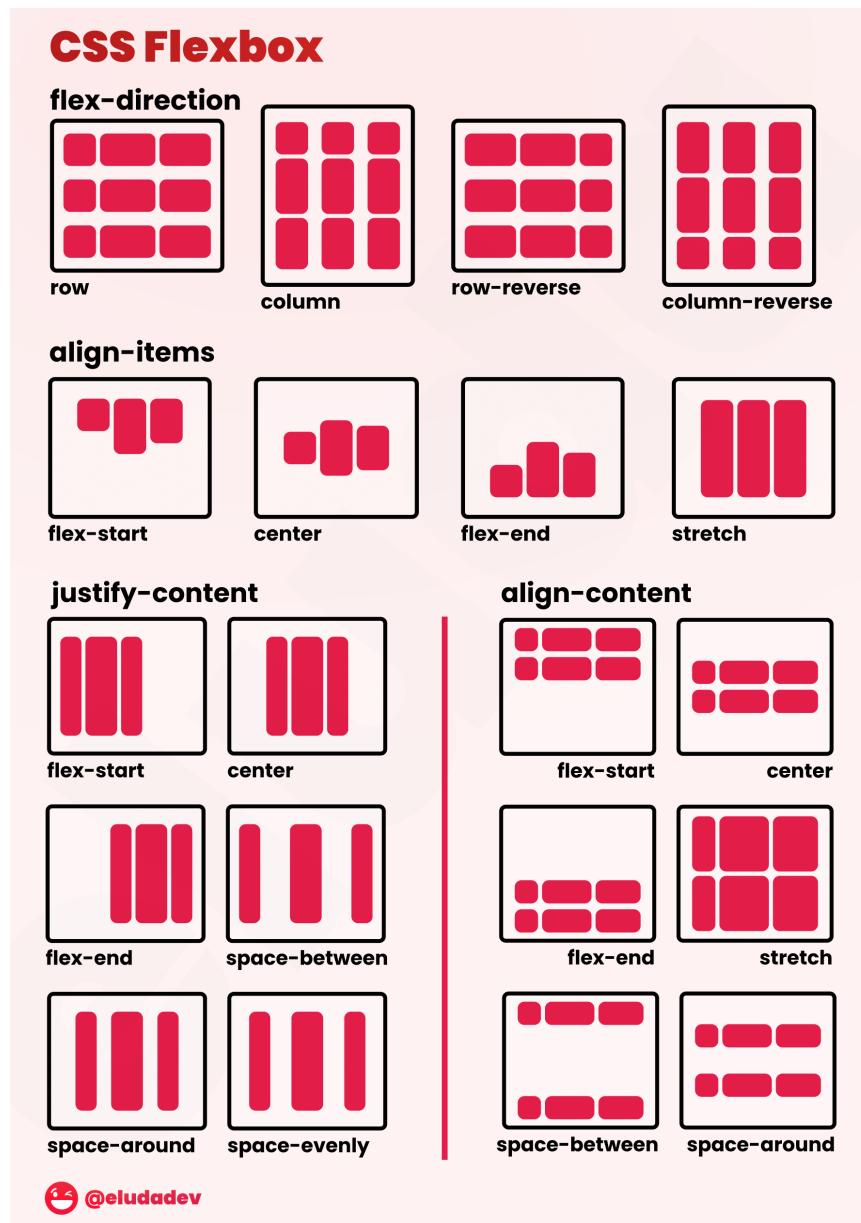
- **Type Selector** เลือก tag html ใช้ชื่อ tag `h1 { color: red; }`
- **Class Selector** เลือก class ใช้จุด `.className { color: green; }`
- **ID Selector** เลือก id ใช้ # `# idName { color: blue; }`

- **CSS Position**

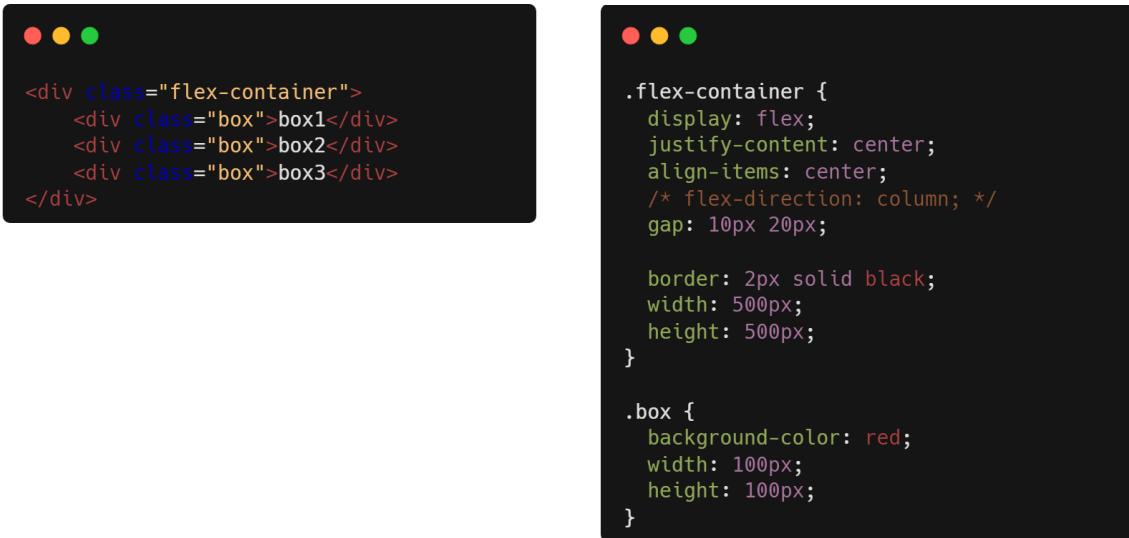
- `position: static;` ตำแหน่งตามค่าเริ่มต้น
- `position: relative;` ตำแหน่งตามค่าเริ่มต้น แต่สามารถย้ายได้โดยใช้ top, right, bottom, left
- `position: absolute;` ตำแหน่งจะขึ้นอยู่กับค่า top, left, right, bottom ไม่สนใจตำแหน่งเริ่มต้น ไม่สนใจว่าจะไปกับ element อื่น
- `position: fixed;` ตำแหน่งจะขึ้นอยู่กับค่า top, left, right, bottom ไม่สนใจตำแหน่งเริ่มต้น ไม่สนใจว่าจะไปกับ element อื่น และจะอยู่ตำแหน่งเดียวกันตลอดเวลาไม่ว่าจะเลื่อนหน้าจอไปไหน

- **CSS Flexbox** กำหนดได้โดยใช้ `display: flex;`

- flex-direction กำหนดทิศทางของ element ภายใน flexbox เช่น `flex-direction: column;`
- align-items จัดตำแหน่งในแนวแกน y เช่น `align-items: center;`
- justify-content จัดตำแหน่งในแนวแกน x `justify-content: center;`
- gap ระยะห่างระหว่าง element ที่อยู่ใน flexbox เช่น `gap: 10px 20px;` หมายถึง ระยะห่างในแกน x 20px และระยะห่างในแกน y 10px
- ในตัวอย่าง ครอบสีดำคือ flexbox ส่วนกล่องสีชมพูก็คือ element ที่อยู่ใน flexbox



- example



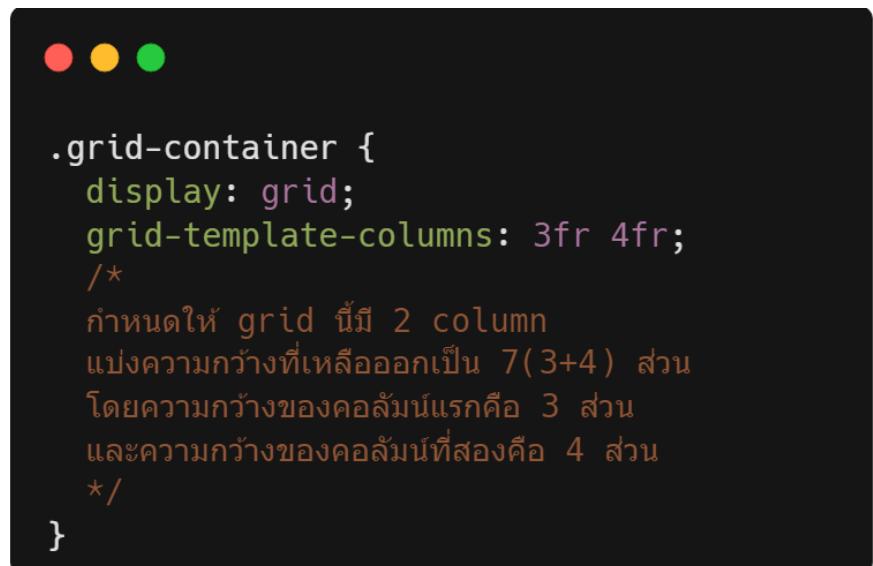
```
<div class="flex-container">
  <div class="box">box1</div>
  <div class="box">box2</div>
  <div class="box">box3</div>
</div>
```

```
.flex-container {
  display: flex;
  justify-content: center;
  align-items: center;
  /* flex-direction: column; */
  gap: 10px 20px;

  border: 2px solid black;
  width: 500px;
  height: 500px;
}

.box {
  background-color: red;
  width: 100px;
  height: 100px;
}
```

- **grid** จัดองค์ประกอบเป็นตาราง กำหนดโดยใช้ `display: grid;`
  - หน่วย fr หรือหน่วยส่วนเศษ ไม่ได้หมายถึงค่าคงที่ใดๆ แต่หมายถึงค่าที่สัมพันธ์กับพื้นที่ว่างที่เหลืออยู่
  - grid-template-columns เป็นการกำหนดความกว้างและจำนวนคอลัมน์ในตาราง



```
.grid-container {
  display: grid;
  grid-template-columns: 3fr 4fr;
  /*
    กำหนดให้ grid นี้มี 2 column
    แบ่งความกว้างที่เหลือออกเป็น 7(3+4) ส่วน
    โดยความกว้างของคอลัมน์แรกคือ 3 ส่วน
    และความกว้างของคอลัมน์ที่สองคือ 4 ส่วน
  */
}
```

- example

```
<div class="grid-container">
  <div class="box">box1</div>
  <div class="box">box2</div>
  <div class="box">box3</div>
  <!-- ลองใส่ให้มากกว่า 3 box ดู -->
  <!-- <div class="box">box4</div> -->
</div>
```

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  gap: 10px 20px;

  border: 2px solid black;
  width: 500px;
  height: 500px;
}

.box {
  background-color: red;
  width: 100px;
  height: 100px
}
```

- Responsive web design คือการทำให้ website รองรับในการแสดงในหลายขนาดหน้าจอ โดยใช้ media queries ตัวอย่างที่ใช้บ่อยคือ การทำให้เป็น grid และเมื่อขนาดจอเล็กลงก็จะปรับให้ column ลดลง

```
.grid-container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  /* เดิม .grid-container จะมี 3 คอลัมน์ */
  gap: 10px 20px;
  margin: 0 auto;

  max-width: 768px;
}

.box {
  border: 1px solid black;
  padding: 20px;
}

/*
  เมื่อขนาดหน้าจอน้อยกว่า 768px
  - จะเปลี่ยน .grid-container ให้มี 2 คอลัมน์
  - จะเปลี่ยนขนาดตัวอักษรใน div ให้เล็กลง
```

```

/*
@media screen and (max-width: 768px) {
    .grid-container {
        grid-template-columns: 1fr 1fr;
    }

    div {
        font-size: 14px;
    }
}

/*
    เมื่อขนาดหน้าจอบigger กว่า 480px
    - จะเปลี่ยน .grid-container ให้มี 1 คอลัมน์
    - จะเปลี่ยนขนาดตัวอักษรใน div ให้เล็กลง
*/
@media screen and (max-width: 480px) {
    .grid-container {
        grid-template-columns: 1fr;
    }

    div {
        font-size: 12px;
    }
}

```

- **trick!** ก่อนเขียน CSS ทุกครั้ง ให้แปลง code นี้ไป ทุกอย่างจะง่ายขึ้น ไม่เหนื่อยการคำนวน

```

* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

```

## JavaScript

- **Array** เป็นการเก็บค่าเป็นลำดับ ซึ่งจะมี method กี่สำหรับคือ
  - map จะเป็นการนำค่าแต่ละทำงานตามฟังก์น์ที่ใส่ไป และ return ออกมาเป็น array ตัวใหม่
  - filter จะเป็นการกรองสมาชิกใน array ให้เหลือตามที่ต้องการ
  - foreach จะเป็นการนำสมาชิกแต่ละตัวมาทำงานตามฟังก์ชันที่ใส่ไป

```
const num = [1, 2, 3, 4, 5]

function filterOdd (num) {
    return num % 2 === 1
}
const odd = num.filter(filterOdd)
console.log(odd) // [1, 3, 5]

function multiple (num) {
    return num * 2
}
const double = num.map(multiple)
console.log(double) // [2, 4, 6, 8, 10]

function print (num) {
    console.log(num)
}
num.forEach(print) // 1 2 3 4 5
```

- **Object** เป็นการเก็บค่าโดยใช้ key/value
  - value เป็นได้ทั้ง string boolean int float object array

```
const student = {
    "name": "Puwit Nunpan",
    "nickName": "Arm",
    "studentID": 65070185,
    "address": {
```

```

        "street": "abcd"
        "city": "maria wall"
    },
    "language": ["th", "en"]
}

console.log(student.name) // Puwit Nunpan
console.log(student.studentID) // 65070185

```

- สามารถแปลงจาก list หรือ object เป็น string โดยใช้ `JSON.stringify(data)`
- สามารถแปลงจาก string เป็น list เป็น object โดยใช้ `JSON.parse(string)`.
- Document.getElementById**
  - หมายถึงการเข้าถึง element ใน HTML

```

<div id="form">
    <input type="text" id="username" />
</div>
<script>
    let form = document.getElementById("form");
    console.log(form);
</script>

```

▼<div id="form">  
 | <input type="text" id="username">  
 |</div>

## Attribute ต่างๆที่ใช้บ่อย

- innerHTML คือค่าที่อยู่ภายใน tag นั้นๆ

```

console.log(form.innerHTML); // <input type="text" id="use
// สามารถเปลี่ยนค่าได้

```

```
form.innerHTML = "Hello World"  
// ภายใน div จะเปลี่ยนจาก กล่อง input เป็นคำว่า hello world เมน
```

- value คือค่าที่อยู่ในกล่อง input

```
let username = document.getElementById("username");  
console.log(username.value); // ค่าที่อยู่ใน input นั้นๆ
```

- style กล่าวง่ายๆคือ css ที่อยู่ในนั้น

```
form.style.width = "300px";  
form.style.backgroundColor = "red";
```

- ถ้าใน HTML tag นั้นสามารถใส่ attribute ไหนได้บ้าง ก็สามารถนำมาเข้าถึงใน javascript ได้ด้วยเช่นกัน เช่น tag `<img>`

```
  
<script>  
    let image = document.getElementById('img');  
    image.src = 'anotherimage.webp';  
    image.alt = 'another example';  
    // เปลี่ยนได้แม้กระทั่ง id กับ class  
    image.id = 'anotherId';  
    image.className = 'anotherClass';  
</script>
```

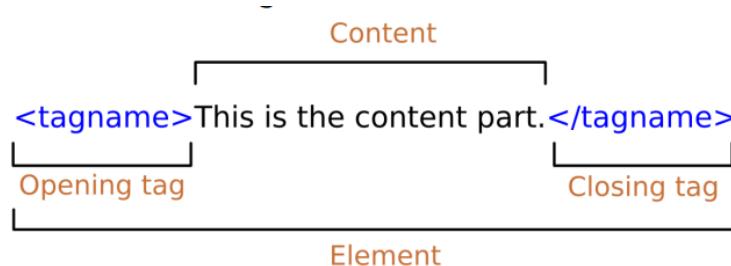
**onclick** คือการกำหนดว่า เมื่อกด element นั้นๆแล้ว ให้ทำอะไร

```
<button onclick="seyhi()">PressME</button>  
<script>  
    function seyhi(){  
        alert("hi midterm");  
    }
```

```
</script>
// เมื่อกดปุ่ม ให้กำฟังก์ชัน sayhi()
```

## CreateElement

- ก่อนจะไปทำ กบกวน structure ของ tag ก่อน



- `document.createElement` คือการสร้าง tag เป็นๆมา โดยที่สร้างมาจะมีแค่ Opening tag และ Closing tag เช่น `document.createElement('p')` จะได้ `<p></p>`
- `document.createTextNode` การสร้างข้อความ
- `appendChild` คือการเพิ่ม content ใน element

```
<!-- เริ่มด้วย tag div เป็นๆ -->
<div id="container"></div>
<script>
    const container = document.getElementById('container');

    // สร้าง element p
    const p = document.createElement('p');
    // สร้าง text node ที่มีค่าเป็น 'Hello World!'
    var text = document.createTextNode('Hello World!');
    // เอา text node ที่สร้างไว้ใส่ใน element p
    p.appendChild(text);
    // เอา element p ที่สร้างไว้ใส่ใน div#container
    container.appendChild(p);

    // สร้าง element ul
```

```

const ul = document.createElement('ul');

// สร้าง element li และ text node ที่มีค่าเป็น 'item 1'
const li1 = document.createElement('li');
const text1 = document.createTextNode('item 1');
li1.appendChild(text1);

// สร้าง element li และ text node ที่มีค่าเป็น 'item 2'
const li2 = document.createElement('li');
const text2 = document.createTextNode('item 2');
li2.appendChild(text2);

// เอา element li ที่สร้างไว้ใส่ใน element ul
ul.appendChild(li1);
ul.appendChild(li2);

// เอา element ul ที่สร้างไว้ใส่ใน div#container
container.appendChild(ul);
</script>

```

## Insert table

- `insertRow()` คือการเพิ่มแถวตาม index ที่ระบุ เช่น `insertRow(-1)` คือการเพิ่มต่อจากแถวสุดท้าย
- `insertCell()` คือการเพิ่มข้อมูลในแถว
- สร้างข้อความโดยใช้ `createTextNode` เมื่อเสร็จ แล้วนำไป `appendChild` ใน cell

```

<table id="show-data">
  <tr>
    <th>ລຳດັບກີ່</th>
    <th>ຮັສນັກສຶກເຫາ</th>
    <th>ຊື່ອ</th>
    <th>ນາມສກູດ</th>
  </tr>

```

```

</table>
<script>
    let table = document.getElementById("show-data");

    // สร้างแควใหม่
    let newRow = table.insertRow(-1);

    // สร้างตัวแปรกึ่งไว้
    let cell, text;

    // เพิ่บข้อมูลลงในแคว เริ่มจาก ลำดับที่ ต้องเพิ่บจากซ้ายไปขวา
    cell = newRow.insertCell(-1);
    text = document.createTextNode("1");
    cell.appendChild(text);

    // เพิ่บข้อมูลลงในแคว รหัสนักศึกษา
    cell = newRow.insertCell(-1);
    text = document.createTextNode("65070185");
    cell.appendChild(text);

    // เพิ่บข้อมูลลงในแคว ชื่อ
    cell = newRow.insertCell(-1);
    text = document.createTextNode("Puwit");
    cell.appendChild(text);

    // เพิ่บข้อมูลลงในแคว นามสกุล
    cell = newRow.insertCell(-1);
    text = document.createTextNode("Nunpan");
    cell.appendChild(text);
</script>

```

### Load ข้อมูล JSON จากภายนอก

```

// function ที่ใช้ load ข้อมูล จดลงเลย 2 บรรทัดเท่านั้น
const response = await fetch(url)

```

```
const data = await response.json()

// ลอง check ดูว่าข้อมูลมาบ้าง
console.log(data)
// จะไปอยู่ใน tab console ใน developer tools เปิดโดยใช้ ctrl+shift+c
```

## Save & Load data Form localStorage

เป็นที่รู้กันว่า localStorage จะเก็บเป็น key/value โดย value จะต้องเป็น string เท่านั้น

- save `localStorage.setItem(key, value)`
- load `localStorage.getItem(key)`

```
// save ข้อมูล
localStorage.setItem("studentID", "65070185")

// load ข้อมูล
let studentId = localStorage.getItem("studentID")
```

- **trick!** สามารถใช้ `JSON.stringify` กับ `JSON.parse` มาประยุกต์ใช้เพื่อเก็บ object หรือ array ลงใน localStorage ได้

```
const data = {
    "firstname": "Puwit",
    "lastname": "Nunpan",
    "nickname": "Arm",
    "studentId": "65070185",
}

// เพราะ localStorage เก็บได้แค่ string เลยเอา object ไปแปลงเป็น string
localStorage.setItem('data', JSON.stringify(data))

// และเราถูกใจมากที่แปลงกลับเป็น object ได้
const dataFromLocalStorage = JSON.parse(localStorage.getItem('data'))
```

## Code lab2 - lab6

ເພື່ອໃຄຣອຍາກດູຕັວອຍ່າງໂຄດຂອງຈາຣນົມຕັບ ອຍ່າກ້ອປເຈາໄປສົ່ງຈາຣນະ

<https://github.com/aaapwn/Fundamental-Web-Programing>