**Web Concept**

Diagram

Description automatically generated

**Hypertext Transfer Protocol**

Request – Reply protocol (RR)

HTTP Resources are identify by URI (or URL)

2 types of message

* Request message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Request  Line | General header | Response header | Blank line | Message body |

* Response message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Status line | General header | Response header | Blank line | Message body |

**Example of HTTP Exchange**

**Request message:**

GET /index.html HTTP/1.1 *Request line*

Host: www.example.com *header line*

[Blank Line]

**Response message:**

HTTP/1.1 200 OK *Status line*

Date: Fri, 31 Dec 1999 23:59:59 GMT *Header lines*

Content-Type: text/html *Header lines*

Content-Length: 1354 *Header lines*

[Blank Line]

<html>

… *Body*

**HTTP Protocol**

2 common methods

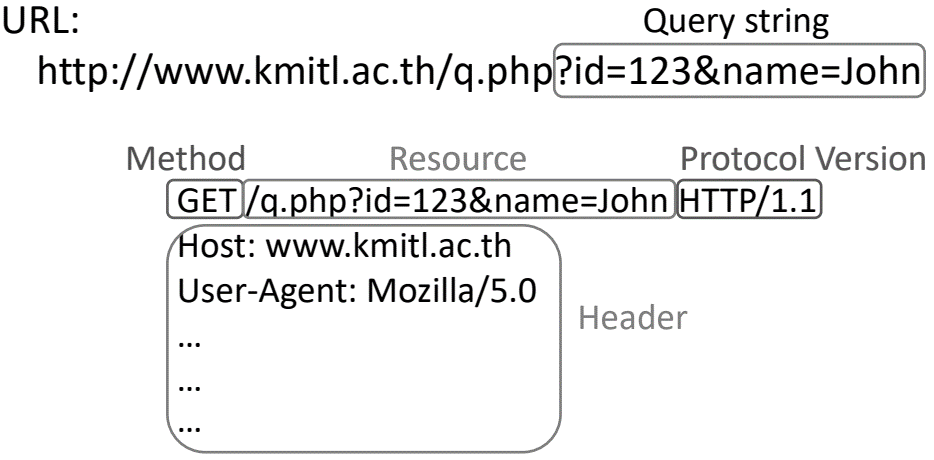
* Get
* Post

**HTTP GET Method**

URL: http://www.kmitl.ac.th/page.html

* Requesting resource



****

**HTTP POST Method**

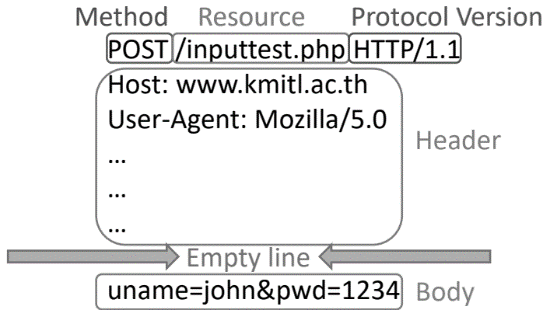
<form action=”kmitl/inputtest.php” **method=”post”**>

Username: <input type=”text” **name=”uname”** value=”” size=”20” maxlength=”20”> <br>

Password: <input type=”password” **name=”pwd”** value=”” size=”20” maxlength=”20>

<input type=”submit” value=”Submit”>

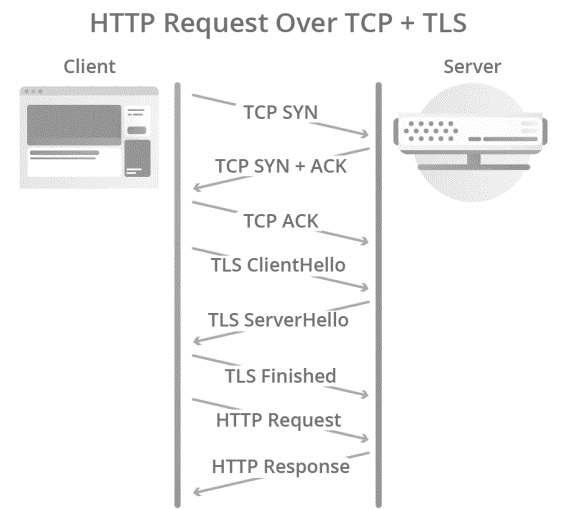
</form>

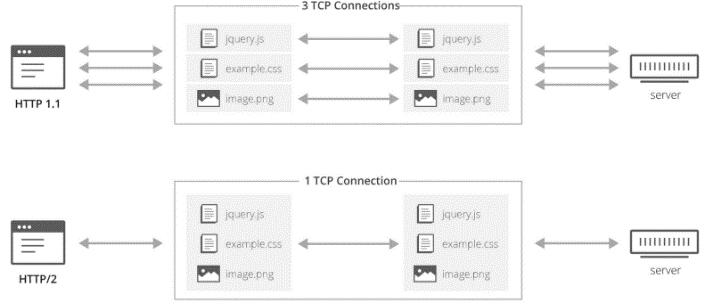


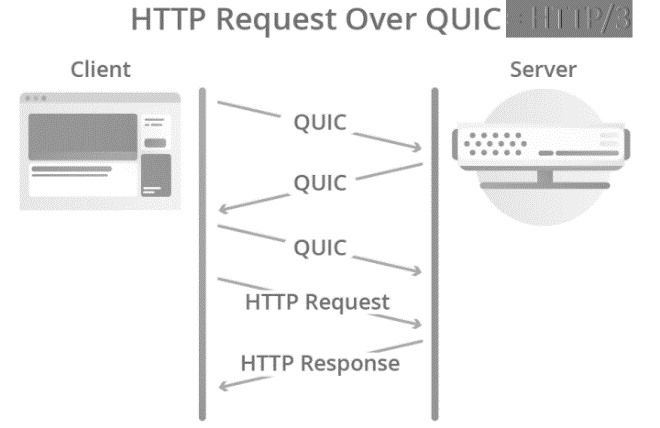
**Get VS Post**

|  |  |  |
| --- | --- | --- |
|  | Get | Post |
| Data length to be sent | ~4000 | unlimited |
| Can send a chunk of data | No | Yes |
| Location of data in message | URL | Body |
| Can access CGI without using Form | Yes | No |
| Can retrieve file or other resource | Yes | No |

**How does it work?**







**WWW คืออะไร**

WWW ย่อมาจาก Wold Wide Web คือ เครือข่ายที่เชื่อมต่อกันทั่วโลก เรามักเรียกย่อๆกันว่า เว็บ คือรูปแบบหนึ่งของระบบการเชื่อมโยงเครือข่ายข่าวสาร ใช้ในการค้นหาข้อมูลข่าวสารบน Internet จากแหล่งข้อมูลหนึ่ง ไปยังแหล่ง ข้อมูลที่อยู่ห่างไกล ให้มีความง่ายต่อการใช้งานมากที่สุด WWW จะแสดงผลอยู่ในรูปแบบของเอกสารที่เรียกว่า ไฮเปอร์เท็กซ์ (Hyper Text) ซึ่งเป็นฐานข้อมูลชนิดหนึ่งที่ทำหน้าที่รวบรวมข่าวสารข้อมูลที่อยู่กระจัดกระจายในที่ต่าง ๆ ทั่วโลกให้สามารถนำมาใช้งานได้เสมือนอยู่ในที่เดียวกัน โดยใช้เว็บเบราเซอร์

การรวมกลุ่มคอมพิวเตอร์อย่างกว้างขวางของ WWW ทำให้เว็บกลายเป็นแหล่งข้อมูลขนาดใหญ่ เป็นบริการข้อมูลแบบมัลติมีเดีย บนอินเทอร์เน็ตที่ได้รับความนิยมสูงสุดใน ปัจจุบันจุดเด่น ได้แก่ความง่ายดายต่อการใช้งาน ที่เชื่อมโยงจากข้อมูลชุดหนึ่งไปยังข้อมูลอีกชุดหนึ่งได้ ซึ่งอาจอยู่ในศูนย์บริการข้อมูลเดียวกันหรือต่างศูนย์กัน จึงเป็นเสมือนเครือข่ายที่โยงใยข้อมูลทั่วโลกเข้าหากัน เมื่อใช้งานศูนย์บริการแห่งหนึ่งแล้วผู้ใช้สามารถเชื่อมต่อ เพื่อค้นข้อมูลที่ศูนย์อื่นๆได้ข้อมูลใน WWW มีทั้งข้อความปกติ หรือแบบมัลติมีเดีย ที่ประกอบด้วยเสียง ภาพนิ่ง และภาพเคลื่อนไหว WWW ยังได้ผนวกบริการอินเทอร์เน็ตอื่นไว้ภายใน

**Webpage**

เอกสารใน WWW มีชื่อเรียกว่า เว็บเพ็จ ถูกสร้างขึ้นจากภาษาคอมพิวเตอร์ ที่เรียกว่า HTML ภาษา HTML จะกำหนดรูปแบบและหน้าตาของเว็บเพจที่ปรากฎบนหน้าจอและส่วนที่เชื่อมต่อกับเว็บเพจอื่น

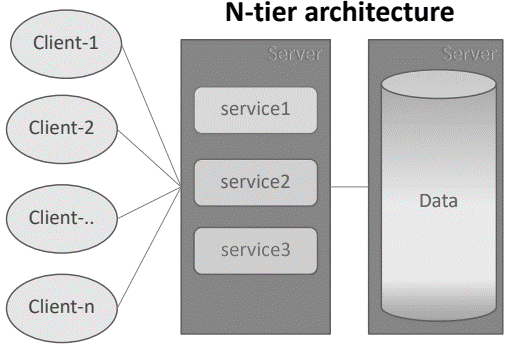
**Link**

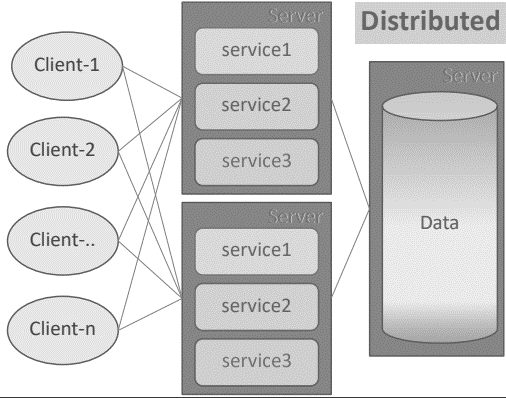
เว็บเพจแต่ละหน้าใน WWW มีการเชื่อมต่อถึงกันทำให้สามารถเรียกดูเว็บเพจหนึ่งจากเว็บเพจอื่นได้โดยในเว็บเพจจะมีจุดเชื่อมโยงที่เรียกว่า ลิงค์ ลิงค์อาจอยู่ในรูปแบบของข้อความ รูปภาพ หรือปุ่ม เมื่อเลื่อนเม้าส์ไปเหนือ ลิงค์ มันจะเปลี่ยนเป็นรูปมือสีขาว ลิงค์เป็นคุณสมบัติที่ทำให้เว็บเพจมีความแตกต่างจากเอกสารทั่วไป เพราะผู้อ่านสามารถโต้ตอบกับข้อมูลได้โดยการคลิกเม้าส์ เพื่อเปิดดูข้อมูลในส่วนที่ต้องการได้

โดย World Wide Web เป็นระบบเปิด ไม่มีผู้ใดเป็นเจ้าของ ทำให้ผู้คนที่สร้างเว็บไซต์สามารถทำงานบนเวิลด์ไวล์เว็บได้อย่างมีประสิทธิภาพ โดยการเข้าอินเตอร์เน็ตแต่ละครั้งต้องพิมพ์ URL ผ่านBrowser หลังจากนั้นบราวเซอร์จะขอใช้บริการของ WWW เพื่อเข้าถึงข้อมูลที่ต้องการผู้ใช้งานอินเตอร์เน็ตก็สามารถที่จะเห็นข้อมูลต่างๆผ่านหน้าจอคอมเป็นตัวอักษรและที่มนุษย์เข้าใจได้ โดยผู้ใช้งานสามารถโต้ตอบสื่อสารกับทาง WWW ได้ด้วยเช่นกัน การโต้ตอบเราจะทำผ่านเว็บบราวเซอร์เช่นเดียวกัน

**Diagram

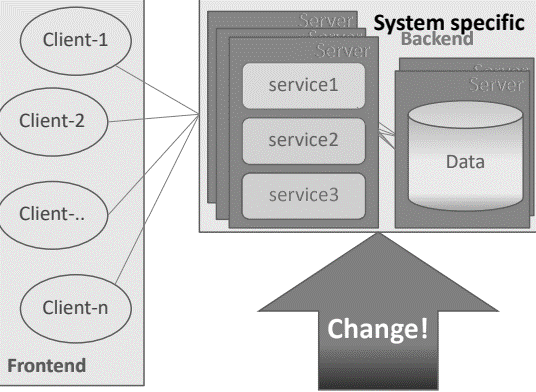
Description automatically generated**

****

****

**Diagram

Description automatically generated**

**Diagram

Description automatically generated**

**HTML (HyperText Markup Language)**

เป็นภาษาที่ใช้สําหรับสร้างเว็บเพจ มีโครงสร้างภาษาโดยใช้ตัวกํากับ (Markup Tag) เพื่อ

ควบคุมการแสดงผลข้อมูล รูปภาพ และวัตถุอื่น ๆ ผ่านทาง Web Browser เช่น GoogleChrome , Firefox , Safari , Microsoft Edge เป็นต้น

ในแต่ละ Tag จะมีส่วนที่เรียกว่า Attribute เพื่อควบคุมการทํางานของ Tag แต่ละตัว

**การสร้างไฟล์ HTML**

จะต้องอาศัย Text Editor เพื่อใช้สําหรับเขียนคําสั่งต่าง ๆ ที่ต้องการแสดงผลทางจอภาพ / เว็บเบราว์เซอร์ และเก็บเป็นไฟล์โดยมีนามสกุล .html

**HTML 5.0**

มาตรฐานของภาษา HTML มีการจัดโครงสร้างและการแสดงผลของเนื้อหา

สําหรับ www มาตรฐานใหม่จะมีคุณลักษณะเด่นที่สําคัญ เช่น

* เล่นวิดีโอ
* แสดงตําแหน่งทางภูมิศาสตร์
* เก็บไฟล์ในลักษณะออฟไลน์
* แสดงกราฟิก
* การป้อนข้อมูลแบบใหม่ เช่น search, number, range, color, tel, url, email, date, month, week, time, datetime, datetime-local

**DOCTYPE**

การประกาศว่าเว็บเพจที่ได้สร้างขึ้นมาอ้างอิงตามมาตรฐานใด

**HTML 5**

<!DOCTYPE html>

**HTML 4.01:**

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" ">

**XHTML 1.1:**

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "">

**กําหนดรูปแบบ Character encoding ในหน้าเว็บ**

การกําหนดรูปแบบการเข้ารหัสอักขระ(Character encoding) โดยใช้แท็ก <meta> กําหนด Attribute charset ลงไป

* <meta charset="utf-8">
* <meta http-equiv="Content-Type" content="text/html; charset=utf-8">

**โครงสร้างภาษา HTML**

**โครงสร้าง HTMLจะแบ่งออกเป็น 2 ส่วน** ได้แก่ ส่วน head และส่วน body โดยเรียงจากแท็ก <head> และ <body> ตามลําดับ โดยทั้ง 2 แท็กจะอยู่ภายใน <html> … </html>

**ส่วน head**

เป็นส่วนที่อยู่ภายใน <head> … </head> ใช้สําหรับอธิบายข้อมูลเกี่ยวกับเว็บ เช่น ชื่อเรื่องของเว็บเพจ (Title) ชื่อผู้จัดทําเว็บ (Author) คีย์เวิร์ด (Keywords) เพื่อใช้สําหรับให้ผู้ใช้ค้นหาข้อมูลเกี่ยวกับเว็บได้

**ส่วน body**

เป็นส่วนที่อยู่ระหว่าง <body> … </body> ใช้อธิบายเนื้อหาหลักของเว็บ เช่น ใส่ข้อความต่างๆ รูปภาพ แบบฟอร์ม วิดีโอและยังสามารถกําหนดคุณสมบัติ พื้นฐานของเว็บได้ เช่น รูปแบบของพื้นหลัง สีของตัวอักษร

**HTML Element**

* ทุกคําสั่งที่อยู่ระหว่างแท็กเปิดและแท็กปิด HTML element บางอย่างไม่มีเนื้อหา (content) ซึ่งจะจบคําสั่งในแท็กเปิดเลย
* โดยส่วนใหญ่ HTML element มักจะมี attributeประกอบอยู่ในแท็กด้วย

|  |  |  |
| --- | --- | --- |
| Tag เปิด | Element Content | Tag ปิด |
| <h1> | หัวข้อเรื่อง | </h1> |
| <a href=“www.google.com”> | เข้าสู่เว็บไซต์ | </a> |
| <br /> |  |  |

**Comment**

ส่วนที่ใช้ในการการอธิบายโค้ด ซึ่งจะช่วยให้สามารถ

เข้าใจและสามารถแก้ไขโค้ดได้ในภายหลังได้

รูปแบบ <!-- ข้อความอธิบายโค้ด -->**การกําหนดหัวเรื่อง (Heading)**

จะใช้ Tag <h1> จนถึง <h6> โดย <h1> จะเป็นการกําหนดหัวเรื่องที่มีขนาดใหญ่ที่สุดส่วน <h6> เป็นกําหนดหัวเรื่องที่มีขนาดเล็กสุด

**แสดงข้อมูลเป็น Paragraphs <p>**

จุดเริ่มต้นของ Paragraphs จะเริ่มที่บรรทัดใหม่ และประโยคที่ไม่ได้อยู่ใน Paragraphs เดียวกัน แต่อยู่ในตําแหน่งที่ต่อจากParagraphs ก็จะถูกจัดให้ขึ้นบรรทัดใหม่ทันที

**แท็กสําหรับขึ้นบรรทัดใหม่**

<br /> เพื่อให้เนื้อหาดูเป็นระเบียบและอ่านได้ง่ายขึ้น

**แท็กสําหรับสร้างเส้นคั่นในแนวนอน**

<hr> สร้างเส้นคั่นให้กับเนื้อหา

**แท็กรูปภาพ (HTML Images)**

<img src=“ตําแหน่งรูปภาพ” alt=“ท่องเที่ยว” width=”ความกว้าง” height=”ความสูง”>**ใส่ link ให้กับรูปภาพ**

<a href=”url”>

<img src=“ตําแหน่งรูปภาพ” alt=“ท่องเที่ยว”> </a>

**การแสดงรายการ (Lists)**

ใช้แสดงข้อมูลในรูปแบบของรายการมี 2 รูปแบบ

* **รายการแบบใช้ตัวเลข (Order List : Ol)**
* **รายการแบบใช้สัญลักษณ์ (Unorder List : Ul)**

<ol type=“รูปแบบการแสดงผล”>

<li>หัวข้อย่อยรายการที่ 1</li>

<li>หัวข้อย่อยรายการที่ 2</li>

</ol>

1. "A" - ตัวอักษรพิมพ์ใหญ่ เช่น A, B, C
2. "a" - อักษรพิมพ์เล็ก เช่น a, b, c
3. "I" - เลขแบบโรมัน เช่น I, II, III

<ul type=“รูปแบบการแสดงผล”>

<li>หัวข้อย่อยรายการที่ 1</li>

<li>หัวข้อย่อยรายการที่ 2</li>

</ul>

* disc – จุดสีดํา
* circle – จุดวงกลมโปร่ง
* square - สี่เหลี่ยมทึบดํา (ตัวเล็กทั้งหมด)

**การสร้างตาราง (Table)**

<table></table> ใช้กําหนดสําหรับสร้างตาราง

<thead></thead> ใช้กําหนดกลุ่มเนื้อหาส่วนหัวตาราง

<tbody></tbody> ใช้กําหนดกลุ่มเนื้อหาตาราง

<tfoot></tfoot> ใช้กําหนดกลุ่มส่วนใต้ตาราง

<tr></tr> ใช้กําหนดแถวในตาราง

<td></td> กําหนดคอลัมน์

<th></th> กําหนดคอลัมน์ที่แสดงผลในส่วนหัวของตาราง

**Attribute ของตาราง**

* border=“ความหนา” กําหนดเส้นขอบและความหนาของเส้นขอบตาราง ค่าเริ่มต้น = 0
* width=“%” กําหนดความกว้างหน่วยเป็น %
* bgcolor=“สี” กําหนดสีพื้นหลังในตาราง
* <table bgcolor=“สี”> กําหนดสีทั้งแถวและคอลัมน์
* <tr bgcolor=“สี”> สีของแถว
* <td bgcolor=“สี”> สีของคอลัมน์

**Attribute ของตาราง**

* colspan=“x” รวมคอลัมน์ ค่า x คือจํานวนคอลัมน์ที่ต้องการรวมเข้าด้วยกัน (ช่อง <td>)
* rowspan=“x” รวมแถว ค่า x คือจํานวนแถวที่ต้องการรวมเข้าด้วยกัน
* align=“left, center, right” จัดตําแหน่งของภาพ หรืออักษรภายในช่องตาราง <td> ค่าปกติคือ left
* Cellpadding แสดงข้อมูลภายในตาราง หากมีค่ามากก็จะมีพื้นที่การแสดงผลเป็นที่ว่างมากขึ้น โดยมีค่าเริ่มต้นเป็น 0 (หน่วย Pixel)
* Cellspacing กําหนดขนาดเส้นตาราง หากมีค่ามากขึ้นเส้นตารางก็ จะมีขนาดมากตามไปด้วย โดยมีค่าเริ่มต้นเป็น 0 (หน่วย Pixel)

**การจัดกลุ่มด้วย div , span**

**<span></span>** ใช้จัดกลุ่มข้อความหรือแท็กต่าง ๆ เข้าเป็นกลุ่มเดียวกันเพื่อกําหนด สี รูปแบบตัวอักษร หรือ style ให้กับข้อความและแท็กภายใต้ <span> ให้เป็นรูปแบบเดียวกัน

**<div></div>** ใช้จัดกลุ่มข้อความหรือแท็กต่าง ๆ เข้าเป็นกลุ่มเดียวกันลักษณะคล้ายๆกับ <span> แต่แตกต่างกันตรงที่แท็ก div จะมีการขึ้นบรรทัดใหม่ก่อนเริ่มแสดงข้อความภายใต้แท็ก div**HTML FORM**

การพัฒนาเว็บแอปพลิเคชั่นจําเป็นต้องมีการสร้างแบบฟอร์มที่ผู้ใช้งานสามารถป้อนข้อมูลต่างๆได้ เพื่อนําข้อมูลที่ป้อนนั้นไปทําการประมวลผลอีกทีโดยการรับค่าข้อมูลจะดําเนินการผ่าน <form>....</form>

|  |  |
| --- | --- |
| Tag | คําอธิบาย |
| <input> | สร้างช่องรับข้อความต่างๆ |
| <select> | แสดงตัวเลือกในรูปแบบ Drop-down |
| <option> | สร้างตัวเลือก |
| <button> | สร้างปุ่ม |
| <label> | กําหนดป้ายชื่อให้ช่องรับข้อมูล |
| <textarea> | สร้างช่องรับข้อความแบบหลายบรรทัด |

**Block vs Inline**

* Block คือ ความยาวเต็มบรรทัด
* Inline คือ ความกว้างเท่ากับข้อความที่แสดง

**Class & ID**

* Class การประกาศค่า Attribute “class” ในแท็กที่ต้องการ
* ID เป็นการกําหนดรหัสเฉพาะของแท็กด้วยการประกาศค่า Attribute “id” ในแท็กที่ต้องการ เพื่อนําไปแสดงผลเหมือนกับ Class แต่ค่า id จะไม่สามารถซ้ำกันได้

**Semantic Tag**

การใช้ Semantic tag ถูกนํามาใช้แทน div หลายๆชั้นในหน้าเว็บจะส่งผลทําให้โครงสร้าง html มีความหมายตรงตัวชัดเจนมากยิ่งขึ้น

Table

Description automatically generated

<header> คือ ส่วนหัวของเว็บ

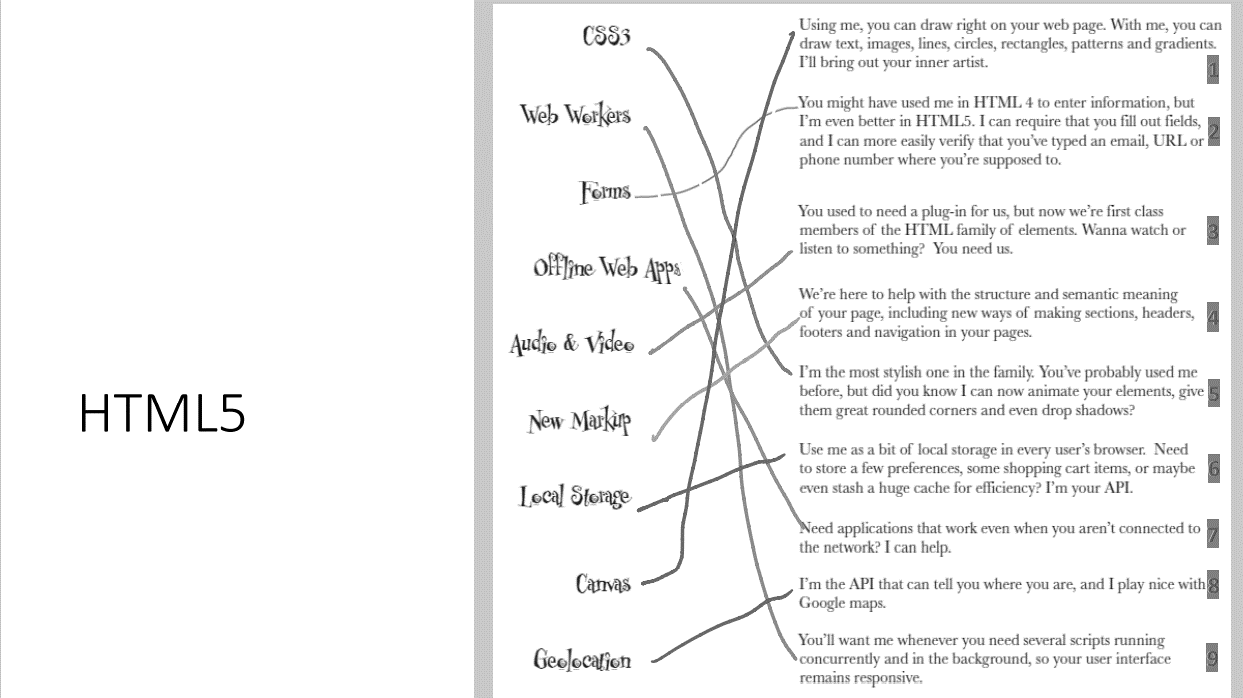
<nav> คือ เมนูของเว็บ หรือ ลิงค์ไปเว็บอื่นๆ

<article> คือ ส่วนที่แสดงเนื้อหาของเว็บ

<section> คือ กลุ่มหัวข้อย่อย

<aside> คือ เนื้อหาอื่นๆที่แยกจากเนื้อหาหลัก

<footer> คือ ส่วนท้ายของหน้าเว็บ



**CSS (Cascading Style Sheet)**

CSS (Cascading Style Sheets) เป็นเทคโนโลยีที่นํามาใช้จัดรูปแบบและควบคุมการแสดงผลของเว็บ เช่น การกําหนดขนาด สี แบบอักษร เส้นขอบ พื้นหลัง และตําแหน่งของข้อความ รูปภาพ เป็นต้น

**โครงสร้างของ CSS**

มีส่วนประกอบอยู่ 2 ส่วน คือ Selector และ Declaration

* Selector คือแท็ก HTML, ID และ Class ที่ต้องการกําหนด
* Declaration ใช้สําหรับกําหนดค่าให้กับ Selector

|  |  |
| --- | --- |
| Selector{  Declaration  } | แท็กอ้างอิง{  คุณสมบัติ  } |

**Selector** คือแท็ก HTML, ID และ Class ที่ต้องการกําหนด

คุณสมบัติ แบ่งออกเป็น 3 รูปแบบ

* Element Selector คือ การกําหนด style ให้แท็ก HTML โดยตรง
* Class Selector คือ การกําหนด style โดยใช้ชื่อ Class เป็น Selector โดยใช้เครื่องหมายจุด (.) สามารถมีค่าซ้ำกันได้
* ID Selector คือ การกําหนด style โดยระบุชื่อ ID เป็นรหัสเฉพาะของแท็กหลังเครื่องหมาย # โดย ID ไม่สามารถกําหนดค่าซ้ำกันได

**Declaration** ใช้กําหนดค่าให้กับ selector ซึ่งมีส่วนประกอบ

อยู่ด้วยกัน 2 ส่วน คือ

* Property หรือ Attribute คือคุณสมบัติที่ต้องการกําหนดให้กับ Selector
* Value คือ ค่าที่กําหนดให้กับ Property หรือ Attribute

การกําหนดส่วนของ property กับ value จะแยกด้วยเครื่องหมายcolon (:) ถ้ามีมากกว่า 1 property จะต้องแยกด้วยเครื่องหมายsemi-colon (;) โดยมีรูปแบบดังนี้

selector { property : value; property : value …}

**รูปแบบการประกาศใช้ CSS**

**แบบ Internal**

* Inline เป็นการแทรก attribute style ลงในแท็ก HTML
* Embeded การกําหนดรูปแบบ style sheet ภายในไฟล์เดียวกันกับ

ไฟล์เว็บโดยการเขียนภายใต้ <style> … </style>

**แบบ External**

จะกําหนดไฟล์ style sheet แยกเป็นอีกไฟล์โดยมีนามสกุล .css แล้วค่อยเชื่อมเข้ามาใช้ภายในเว็บเพจ เหมาสําหรับการกําหนดให้หลายๆเว็บเพจใช้รูปแบบ style เดียวกันผ่านการเขียน style แค่ครั้งเดียว

<link rel=“stylesheet”href=“style.css”>

**แบบ Inline**

ใช้กําหนด style ให้กับอิลิเมนต์ของ HTML เพียงอันเดียวเป็นการเฉพาะ <p style="color:green">TEXT.</p>

**การเขียน Comment**/\* ข้อความอธิบายโค้ด \*/

**หน่วย (Unit) ใน CSS (Absolute) แบบตายตัว**

|  |  |
| --- | --- |
| px | pixel (พิกเซล) เป็นหน่วยที่นิยมใช้มากที่สุด โดยที่ 1px = 0.75pt สัมพันธ์กับรายละเอียดหน้าจอ |
| pt | point โดยที่ 1 pt = 1/72 inchs ใช้ในงานสิ่งพิมพ์ |
| cm | เซนติเมตร |
| mm | มิลลิเมตร |
| in | inches (1 in = 96px = 2.54cm) |
| pc | picas (1pc = 12pt) |

**หน่วย (Unit) ใน CSS (Relative) แบบอัตราส่วน**

|  |  |
| --- | --- |
| % | เป็นการกําหนดขนาดเป็นเปอร์เซ็นต์ มักใช้กับความกว้างหรือสูง |
| em | อ้างอิงขนาดกับ element parent ที่ใกล้ที่สุด ใช้ในการกําหนดขนาดจํานวนเท่า ของขนาดปัจจุบันเช่น หากขนาดที่ใช้ตอนนี้คือ 10px   * ถ้ากําหนดขนาดเป็น 2em จะหมายถึง 2 เท่าของขนาดปัจจุบัน คือ 20px * ถ้ากําหนดขนาดเป็น 1.4em จะหมายถึง 1.4 เท่าของขนาดปัจจุบัน คือ 14px * ถ้ากําหนดเป็นขนาดเดิมก็กําหนดเป็น 1em = 10px |
| rem | กําหนดขนาดโดยอ้างอิงกับ root element ปกติ font-size จะอยู่ที่ 16px |
| vw | 1% หรือ 1/100 ของ viewport width   * width ของ browser viewport เท่ากับ 750px ค่า 1vw = 7.5px |
| vh | 1% หรือ 1/100 ของ viewport height   * height ของ browser viewport เท่ากับ 900px ค่า 1vh = 9px |
| vmin, vmax | กําหนดค่าต่ำาสุด และค่าสูงสุดของ viewport |

**Universal Selector**

กําหนด style sheet มีผลต่อทุกแท็กของเว็บเพจ โดยใช้เครื่องหมาย \*

**การกําหนดชนิดฟอนต์ (Font)**

**font-family** กําหนดชนิดของฟอนต์ในหน้าเว็บกําหนดได้ทั้งแบบ

ฟอนต์เดียวหรือหลายฟอนต์ โดยกําหนดได้ดังนี้ คือ

ฟอนต์แบบที่ 1;

ฟอนต์แบบที่ 1 , ฟอนต์แบบที่ 2, ชนิดฟอนต์;

สามารถกําหนดฟอนต์ได้มากกว่า 1 ชนิดในกรณี Browserไม่รองรับฟอนต์ชนิดแรก ก็จะนําฟอนต์ลําดับถัดไปมาใช้งานแทน

**การกําหนดขนาดข้อความ**

**font-size** กําหนดขนาดข้อความหรือตัวอักษรในหน้าเว็บโดย

กําหนด 2 รูปแบบ คือ

* แบบค่าคงที่ : xx-small, x-small, medium, large, x-large, xx-large
* แบบตัวเลข : 10px 20px

**การกําหนดความหนาข้อความ**

**font-weight** กําหนดได้ 2 รูปแบบ

* แบบค่าคงที่ : lighter , bold (ตัวหนา) bolder (ตัวหนากวา)
* แบบตัวเลข : 100, 200, …., 800, 900

**font-style** กําหนดตัวเอียง โดยค่าที่กําหนดได้คือ italic , oblique (45 องศา)

**การกําหนดสี (Colors)**

**color** กําหนดได้ 4 รูปแบบ

**ชื่อสี** : green , red , yellow , pink

**RGB** : rgb(red, green, blue) , rgb(30%, 20%, 45%)

**Hexadecimal** : #000 #FFF

**HSL** : (hue, saturation, lightness)

**การกําหนดลักษณะข้อความ**

**Text-decoration** กําหนดได้ทั้ง 4 แบบ

none : (ค่าว่าง)

underline : (ขีดเส้นใต้)

overline : (ขีดเส้นเหนือข้อความ)

line-through : (ขีดเส้นทับข้อความ)

**การจัดแนวข้อความ**

**Text-align** กําหนดได้ 4 รูปแบบ

* left, right, center (ชิดด้านซ้าย ขวา และกึ่งกลางตามลําดับ)
* justify (การกระจายข้อความให้ทุกบรรทัดมีความกว้างเท่ากัน)

**กําหนดความกว้างและความสูง**

**width** ใช้กําหนดความกว้างของวัตถุที่ต้องการ

* auto คือ browser จะกําหนดให้เอง
* length คือ ความกว้างแบบหน่วยวัด เช่น px cm เป็นต้น
* % คือ ความกว้างแบบ % (แบบยืดหยุ่น)

**height** ใช้กําหนดความสูงของวัตถุที่ต้องการ

* auto คือ browser จะกําหนดให้เอง
* length คือ ความสูงแบบหน่วยวัด เช่น px cm เป็นต้น
* % คือ ความสูงแบบ % (แบบยืดหยุ่น)

**ความกว้างและความสูงด้วย max , min**

* width-min ใช้กําหนดความกว้างต่ำสุดของวัตถุ
* width-max ใช้กําหนดความกว้างสูงสุดของวัตถุ
* height-min ใช้กําหนดความสูงต่ำสุดของวัตถุ
* height-max ใช้กําหนดความสูงสูงสุดของวัตถุ

**หน่วยที่รองรับ**

* none ไม่กําหนด (default)
* length คือ กําหนดแบบหน่วยวัด เช่น px cm เป็นต้น
* % คือ กําหนดแบบ % (แบบยืดหยุ่น)

**การกําหนดเส้นขอบ**

Border คือการกําหนดเส้นขอบ โดยมีรูปแบบดังนี้

**border : <รูปแบบเส้น> <ขนาด> <สี>**

รูปแบบเส้น ประกอบด้วย solid (เส้นทึบ) dotted (แบบจุด) dashed (เส้นปะ)

ขนาด คือ ความกว้างของเส้นกําหนดเป็นตัวเลขตามหน่วย

สี กําหนดรูปแบบเหมือน font เลย เช่น ชื่อสี , rgb , hex

**การกําหนดรูปแบบเส้นขอบ (style)**

Border คือการกําหนดเส้นขอบ โดยมีรูปแบบดังนี้

**border-style : <รูปแบบเส้น>**

รูปแบบเส้น ประกอบด้วย solid (เส้นทึบ) dotted (แบบจุด) dashed (เส้นปะ)

**border-xxx-style : <รูปแบบเส้น>**

xxx คือ ทิศทางของรูปแบบเส้นขอบ

**การกําหนดสีเส้นขอบ (color)**

Border คือการกําหนดเส้นขอบ โดยมีรูปแบบดังนี้

**border-color : <สี>**

สี ประกอบด้วย RGB , ชื่อสี , HEX

**border-xxx-color: <สี>**

xxx คือ ทิศทางของรูปแบบเส้นขอบ

**การกําหนดขนาดเส้นขอบ (width)**

Border คือการกําหนดเส้นขอบ โดยมีรูปแบบดังนี้

border-width : <ขนาด>

border-xxx-width: <ขนาด>

xxx คือ ทิศทางของรูปแบบเส้นขอบ

**การกําหนดขนาดเส้นขอบ (width)**

Border คือการกําหนดเส้นขอบ โดยมีรูปแบบดังนี้

**border-width : <ขนาด>**

แบบระบุค่าคงที่ :

* medium – ขอบปานกลาง
* thin – ขอบบาง
* thick – ขอบหนา
* initial – ค่าเริ่มต้น
* inherit - อ้างอิงตาม parent element

**การกําหนดความโค้งเส้นขอบ (radius)**

Border คือการกําหนดเส้นขอบ โดยมีรูปแบบดังนี้

**border-radius : <ค่าความโค้ง>**

**border-xxx-radius: <ค่าความโค้ง>**

xxx คือ ทิศทางของรูปแบบเส้นขอบ

**การกําหนดความโค้งเส้นขอบ**

**เขียน 4 รูปแบบ**

border-radius: 10px 5px 15px 20px;

- top-left = 10px

- top-right = 5px

- bottom-right = 15px

- bottom-left = 20px

**เขียน 3 รูปแบบ**

border-radius: 10px 5px 15px;

- top-left = 10px

- top-right , bottom-left = 5px

- bottom-right = 15px**เขียน 2 รูปแบบ**

border-radius : 10px 5px;

- top-left , bottom-right = 10px

- top-right , bottom-left = 5px

**เขียน 1 รูปแบบ**

border-radius : 10px;

ทุกทิศทางมีค่า = 10px

**Box Model**

**Diagram

Description automatically generated**

เส้นขอบ

**การกําหนดพื้นที่รอบ Element**

**Margin** กําหนดระยะห่างจากเส้นขอบหรือพื้นที่ภายนอกของ Element เทียบกับวัตถุอื่นๆ (ช่องว่างข้างนอก)

**Padding** กําหนดพื้นที่ระยะห่างภายในของ Element (ช่องว่างข้างใน)

**Border** กําหนดเส้นขอบ

กําหนดได้ทั้งรูปแบบด้านบน (top) ด้านล่าง (bottom) ด้านขวา (right) และด้านซ้าย (left) ทั้ง margin และ padding เช่น margin-left , margin-right , padding-left, padding-bottom

หน่วยที่กําหนดได้ เช่น px, pt, cm อื่นๆ หรือแบบเปอร์เซ็นต์ (%) โดยมีค่า default เป็น 0

**การกําหนดพื้นที่รอบ ElementMargin** กําหนดระยะห่างจากเส้นขอบหรือพื้นที่ภายนอกของ Element

* margin: ค่าที่กําหนด ;
* margin-left:ค่าที่กําหนด;
* margin-right:ค่าที่กําหนด;
* margin-top:ค่าที่กําหนด;
* margin-bottom:ค่าที่กําหนด;
* margin:auto;(ให้ browser จัดให้เอง ปกติจะกําหนดกึ่งกลาง)

**เขียน 4 รูปแบบ**

margin: 10px 5px 15px 20px;

- top margin = 10px

- right margin = 5px

- bottom margin = 15px

- left margin = 20px

**เขียน 3 รูปแบบ**

margin: 10px 5px 15px;

- top margin = 10px

- right left = 5px

- bottom margin = 15px

**เขียน 2 รูปแบบ**

margin: 10px 5px;

- top bottom margin = 10px

- right left = 5px

**เขียน 1 รูปแบบ**

margin: 10px;

ทุกทิศทางมีค่า = 10px

\*\* ลําดับ top, right, bottom, left

**Padding** กําหนดพื้นที่ระยะห่างภายในของ Element

* padding: ค่าที่กําหนด ;
* padding-left:ค่าที่กําหนด;
* padding-right:ค่าที่กําหนด;
* padding-top:ค่าที่กําหนด;
* padding-bottom:ค่าที่กําหนด;
* padding:auto;(ให้ browser จัดให้เอง ปกติจะกําหนดกึ่งกลาง)

**การกําหนดสีพื้นหลัง (background)**

**background-color** ใช้เปลี่ยนสีพื้นหลังหรือกําหนดความโปร่งใสให้กับพื้นหลังและ Element ต่าง ๆ

**background-color : <สี> | transparent**

สี ประกอบด้วย RGB , ชื่อสี , HEX

**การกําหนดภาพพื้นหลัง (background)**

**background-image** ใช้เปลี่ยนภาพพื้นหลัง

**background-image : url(รูปภาพ)**

**url(‘image.png’)**

**url(‘http://www.unsplash…..png’)**

**รูปแบบภาพพื้นหลัง (background-repeat)**

**background-image:url(xxx);**

**background-repeat : xxx;**

repeat (ค่า default ทําให้ภาพเต็มพื้นที่ของอิลิเมนต์)

repeat-x (จัดเรียงเฉพาะในแนวแกน X)

repeat-y (จัดเรียงเฉพาะในแนวแกน Y)

no-repeat (ไม่จัดเรียงแนวแกน x,y แสดงภาพพื้นหลังภาพเดียว)**กําหนดคุณสมบัติพื้นหลัง**

โดยทั่วไปภาพพื้นหลังจะถูกเลื่อนตามการ scroll ของเว็บเบราเซอร์ ถ้าต้องการตรึงภาพพื้นหลังให้อยู่กับที่สามารถทําได้โดยการใช้ **background-attachment**

* scroll (คือการเลื่อนพื้นหลังตาม scrollbar)
* fixed (คือการตรึงพื้นหลังไว้กับที่)

**กําหนดตําแหน่งพื้นหลัง**

สามารถกําหนดได้ว่าจะเริ่มต้นวางพื้นหลังที่ตําแหน่งใด โดยใช้

**background-position** เช่น

background-position : center right ; (กลาง ขวา)

* left top หรือ left center หรือ left bottom
* right top หรือ right center หรือ right bottom
* center top หรือ center center หรือ center bottom

**กําหนดขนาดภาพ (Background-size)**

**background-size** : กําหนดค่า

* auto คาปกติภาพเทาเดิม
* px หรือ %
* cover ขยายภาพทั้งความกวางความÿูงเต็มจอในอัตราÿวนเทากัน
* contain ขยายภาพทั้งความกวางความÿูงชนขอบ Browser

**กําหนดพื้นหลังแบบหลายคุณสมบัติพร้อมกัน**

สามารถกําหนดโดยใช้ property ชื่อว่า **background**background : white url(images/logo.jpg) no-repeat fixed center center;

**กําหนดตําแหน่งด้วย float**

**float** กําหนดให้อิลิเมนต์สามารถลอยอยู่ด้านใดด้านหนึ่ง

โดยค่าที่กําหนดได้ คือ

* left, right (ให้ลอยอยู่ทางด้านซ้ายและขวาตามลําดับ)
* inherit ลอยตาม parent element
* none ไม่ให้มีการลอย (default)
* clear ไม่อนุญาตให้มีการลอยของอิลิเมนต์โดยค่าที่กําหนดได้คือ
* left, right (ไม่อนุญาตให้มีลอยทางด้านซ้ายและขวาตามลําดับ)
* both (ไม่อนุญาตให้มีการลอยทั้งสองด้าน)

**กําหนด style ให้กับ Link**

* a : link กําหนด style ให้กับ link ที่ยังไม่ถูกคลิก
* a : hover กําหนด style ให้กับ link เมื่อนําเมาส์ไปวาง
* a : visited กําหนด style ให้กับ link ที่ถูกคลิกแล้ว
* a : active กําหนด style ให้กับ link ขณะที่ถูกคลิก

**Display Inline , Block , Inline-Block**

**display** คือการจัดรูปแบบการแสดงผลข้อมูล layout

* none ไม่มีการแสดงผล
* block แสดงผลแบบ block โดยการขึ้นบรรทัดใหม่ก่อน(เรียงในแนวตั้ง)
* inline แสดงผลแบบ inline โดยไม่มีการขึ้นบรรทัดใหม่(เรียงในแนวนอน)
* inline-block แสดงผลแบบแนวนอนและขยายพื้นที่ด้านใน

**Visibility**

**visibility** คือ การแสดงหรือซ่อนวัตถุโดยไม่กระทบ layout

* hidden ซ่อนวัตถุ
* visible แสดงวัตถุ

**การจัดตําแหน่งด้วย Position**

**position** ใช้ในการจัดตําแหน่งของวัตถุ

* static จัดวางแบบปกติเป็นค่า default
* relative ใช้ในการจัดวางและกําหนดตําแหน่งวัตถุโดยการนับจากจุดที่วัตถุนั้นๆอยู่
* absolute ใช้ในการจัดวางวัตถุ ให้ไปอยู่ตําแหน่งใดก็ได้แต่ต้องกําหนดตําแหน่งจากขอบของ elememt ที่บรรจุวัตถุนั้นๆอีกที (ถ้าไม่มีอะไรครอบก็ถือว่า body เป็นelememt ที่ครอบวัตถุที่ระบุ absolute)
* fixed ใช้ในการจัดวางวัตถุให้อยู่ตําแหน่งเดิม
* sticky ให้วัถตุติดขอบเมื่อเลื่อนไปถึง

**ลําดับความสําคัญของ Style**

* เรียงลําดับความสําคัญจากล่างขึ้นบน
* !important กําหนดให้มีความสําคัญที่สุด

**จํากัดการแสดงผลข้อมูลด้วย Overflow**

**overflow**

* visible แสดงข้อมูลทั้งหมด
* hidden ซ่อนข้อมูลที่เกิน
* scroll ให้มี scroll bar แสดงผลเมื่อมีข้อมูลเกิน
* overflow:scrolll
* overflow-x : scroll
* overflow-y : scroll
* auto ให้มี scroll bar แสดงผลออกมาอัตโนมัติ

**กําหนดเงาให้วัตถุด้วย Box-shadow**

**box-shadow : x y blur spread color**

* x คือ เงาแกน x (+ ขวา, - ซ้าย)
* y คือ เงาแกน y (+ ล่าง, - บน)
* blur คือ ขนาดความมัวของเงา
* spread คือ ขนาดของเงา
* color คือ สีของเงา (color name,rgb,...)

**กําหนดค่าความทึบ (Opacity)**

**opacity : value**

* ค่าอยู่ระหว่าง 0.0 - 1.0
* ยิ่งค่าน้อย ยิ่งจาง
* ยิ่งค่ามาก ยิ่งทึบ

**Responsive Web Design**

การออกแบบเว็บที่ตอบสนองและแสดงผลได้ดีบนอุปกรณ์ต่างๆหรือขนาดหน้าจอที่หลากหลาย เช่น จอคอมพิวเตอร์ หรือ สมาร์ทโฟน เป็นต้น

**Responsive Web Design (Media Query)**

**Media Query** คือ รูปแบบการเขียน Style ให้แสดงผลตามขนาดหน้าจอที่แตกต่างกัน

@media ขนาดอุปกรณ์ {

…..style…..

}

@media screen , printer {

…..style…..

}

**ความกว้างของขนาดอุปกรณ์**

● 320px—480px: Mobile devices

● 481px—768px: iPads, Tablets

● 769px—1024px: Laptops

● 1025px—1200px: Desktops

● 1201px เป็นต้นไป — TV , Widescreen

**Viewport Units**

ในอดีตเว็บถูกพัฒนาให้ทํางานบนจอคอมพิวเตอร์อย่างเดียว ถ้านํามาทํางานบน Smart Phone , Tablet ในปัจจุบันอาจจะไม่สามารถแสดงผลได้ตามที่ต้องการเช่น ถ้าต้องการดูเนื้อหาในหน้าเว็บต้องทําการ zoom และ scroll หน้าจอเท่านั้น

ในปัจจุบันจึงได้มีการพัฒนาและเพิ่ม meta tag ชื่อว่า viewport เข้ามาในหน้าเว็บเพื่อบอกว่าให้เว็บแสดงผลบน Smart Phone , Tablet ได้ผ่านการอ้างอิงพื้นที่หรือ

สัดส่วนของ Web Browser ที่อยู่ในอุปกรณ์นั้นๆ ในการแสดงผลหน้าเว็บให้เหมาะสมกับอุปกรณ์ดังกล่าว

**องค์ประกอบของ Viewport**

* Vw แนวนอน(ความกว้าง)
* Vh แนวตั้ง(ความสูง)

**Viewport Units**

* vw บอกสัดส่วนความกว้างของ viewport (%) เช่น เรามีจอกว้าง 100px ต้องการแสดงผล 10% ของความกว้างทั้งหมด จะมีค่า 10vw หรือพื้นที่ 10pxหรือ 10/100 เป็นต้น ถ้าอยากกําหนดความกว้างของ viewport มีค่าเท่ากับความกว้างของ browser จะมีค่าเท่ากับ 100vw
* vh บอกสัดส่วนความสูงของ viewport (%)
* vmin ค่า % ต่ำสุดของ viewport
* vmax ค่า % สูงสุดของ viewport

**รู้จักกับ Flexbox**

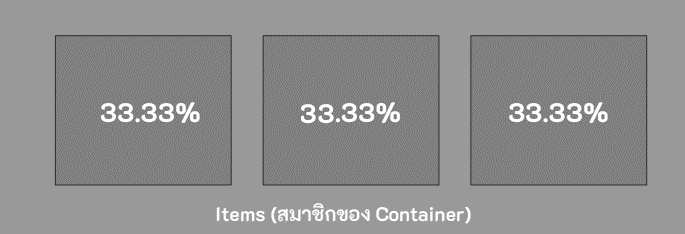
**Flexbox** คือเครื่องมือใน CSS ที่ช่วยให้การจัดการ element ต่างๆในหน้าเว็บ มีความง่ายและยืดหยุ่นมากยิ่งขึ้น โดยทั่วไปการจัดตําแหน่ง element ต่างๆต้องใช้ layout mode คือ block , inline , position และอื่นๆ

การพัฒนาเว็บในปัจจุบันมีความซับซ้อนมากยิ่งขึ้นทําให้แบบเดิมไม่ตอบโจทย์เท่าที่ควร จึงได้มีการพัฒนา flexbox ขึ้นมาเพื่อจัดใช้ในการจัดการ element ให้มีความยืดหยุ่นสูง โดยมีคุณสมบัติดังนี้

* จัดเรียงตําแหน่งของ element ได้ง่ายขึ้น เรียงจากบนลงล่าง ซ้ายไปขวา อื่นๆ
* กําหนดขนาดให้พอดีกับพื้นที่ว่างแบบอัตโนมัติ (Sizing)

**องค์ประกอบของ Flexbox**

**Container (กล่องที่ครอบ Items)**



**Flexbox เบื้องต้น**

.container{

display:flex;

box-sizing: border-box;

}

* flex: ให้จัดวางในรูปแบบ flex
* border-box – กําหนดขนาดให้พอดีกับพื้นที่ว่างโดยคํานวณจากค่าจริงที่ผู้ใช้กําหนด (border + padding) เพื่อไม่ให้ item แสดงผลเพี้ยน (ระบุหรือไม่ระบุก็ได้)

**กําหนดทิศทางด้วย flex-direction**

* row (ค่าเริ่มต้น) จัดวาง items ในแนวนอนทิศทางเดียวกับแกนหลัก
* column จัดวาง items ในแนวตั้งทิศทางเดียวกับแกนหลัก
* row-reverse จัดวาง items ในแนวนอนทิศทางตรงข้ามกับแกนหลัก
* column-reverse จัดวาง items ในแนวตั้งทิศทางตรงข้ามกับแกนหลัก

**กําหนดขนาดด้วย flex-wrap**

**กรณีที่ขนาด items ใหญ่กว่าพื้นที่ container**

* nowrap จัดวาง items ที่เกินพื้นที่ container ไปด้านขวามือ
* wrap จัดวาง items ที่เกินพื้นที่ container เรียงจากบนลงล่าง
* wrap-reverse จัดวาง items ที่เกินพื้นที่ container เรียงจากล่างขึ้นบน

**Flex - properties (Items)**

* flex-shrink : ให้ item หดตัวจํานวนเท่าใดเมื่อเทียบกับ item อื่นๆ ค่าเริ่มต้นเป็น 1
* flex-grow : ให้ item ขยายจํานวนเท่าใดเมื่อเทียบกับ item อื่นๆ ค่าเริ่มต้นเป็น 0
* flex-basis : กําหนดค่าความยืดหยุ่นเริ่มต้น
* flex:1 ทําให้ item ที่อยู่ในแถวเดียวกันมีขนาดเท่ากัน

**Flex Justify (จัดวางตําแหน่ง item)**

\*\*เทียบกับแกนหลัก เช่น กําหนดแกนหลักเป็นแนวนอน

**justify-content**

● flex-start ชิดซ้าย container ทิศทางตามแนวนอน

● center กึ่งกลาง container ทิศทางตามแนวนอน

● flex-end ชิดขวา container ทิศทางตามแนวนอน

● space-around ระยะห่างซ้ายขวาและขนาด item เท่ากัน

● space-between ระยะห่างซ้ายขวาและขนาด item เท่ากัน (ติดมุม)

**Flex Alignment (จัดวางตําแหน่ง item)**

\*\*เทียบกับแกนตรงข้าม เช่น กําหนดแกนหลักเป็นแนวนอน

**align-items** (item ทุกตัว) และ **align-self** (Item ที่ต้องการ)

● stretch - กําหนดขนาด item เท่ากับขนาด container

● flex-start ด้านบน container ทิศทางตามแนวนอน

● center กึ่งกลาง container ทิศทางตามแนวนอน

● flex-end ด้านล่าง container ทิศทางตามแนวนอน

**CSS Grid Layout**

**Flexbox** ถูกออกแบบให้จัดการ layout แบบทิศทางเดียว คือ 1 มิติเช่น เรียงลําดับ

จากบนลงล้าง ซ้ายไปขวา เป็นตอน

|  |  |  |
| --- | --- | --- |
|  |  |  |

แต่ Grid Layout ถูกออกแบบมาเพื่อจัดการ layout แบบ 2 มิติคือ มีทั้งแบบแนวนอน และแนวตั้งในเวลาเดียวกับ หรือมองง่ายๆก็เป็นแบบตาราง

|  |  |
| --- | --- |
|  |  |
|  |  |

**CSS Grid Layout**

**การใช้งาน**

display : grid;

**กําหนดขนาดแถว (ความสูง) :**

grid-template-rows : ความสูงของแถวที่ 1 , 2 , 3;

**กําหนดขนาดคอลัมน์ (ความกว้าง) :**

grid-template-columns : ความกว้างของคอลัมน์ที่ 1 , 2 , 3;

**Gird Properties อื่นๆ**

* กําหนดสัดส่วนพื้นที่ด้วย span
* กําหนดสัดส่วนพื้นที่ด้วยหน่วย fr
* กําหนดระยะห่างพื้นที่ด้วย gap
* กําหนดชื่อพื้นที่ด้วย grid-template-area

**กําหนดเงาให้ข้อความด้วย text-shadow**

text-shadow : x y blur color

* x คือ เงาแกน x (+ ขวา, - ซ้าย)
* y คือ เงาแกน y (+ ล่าง, - บน)
* blur คือ ขนาดความมัวของเงา
* color คือ สีของเงา (color name,rgb,...)

**CSS Variable**

การสร้างตัวแปรใน css เพื่อช่วยอํานวยความสะดวกในการกําหนด style ให้แต่ละ element โดยลดขั้นตอนการทํางานที่ซ้ำซ้อนใน css ให้โค้ดมีความเป็นระเบียบและง่ายต่อการจัดการมากยิ่งขึ้น

การกําหนดตัวแปรสามารถกําหนดชื่อได้เอง (custom variable) โดยขึ้นต้นด้วยเครื่องหมาย - ตามด้วยชื่อตัวแปร ซึ่งตัวแปรส่วนใหญ่ใน css จะนํามาเก็บค่าที่ใช้เรียกทํางานซ้ำๆ เช่น สี เงา แอนิเมชั่น เป็นต้น

**จัดการ Element ด้วย Transform**

* translate (x,y) กําหนดตําแหน่ง element
* scale(x,y) ขยาย element
* rotate (มุม deg) กําหนดการหมุนของ element
* skewX(มุม deg) กําหนดการเอียงของ element แกน x
* skewY (มุม deg) กําหนดการเอียงของ element แกน y

**เปลี่ยนแปลง Element ด้วย Transition**

**transition** คือการเปลี่ยนค่าใน element จากค่าหนึ่งไปสู่อีกค่าหนึ่งในช่วงเวลาที่กําหนดประกอบด้วย

* transition-properties คือ รูปแบบคุณสมบัติที่การเปลี่ยนแปลงค่า
* transition-duration คือ ระยะเวลาในการเปลี่ยนแปลงค่า
* transition-timing-fuction คือ รูปแบบฟังก์ชั่นของการเปลี่ยนแปลงค่า (ease-in)
* transition-delay คือ ระยะเวลาที่จะเริ่มต้นเปลี่ยนแปลงค่า

**transition-timing-function**

Chart, histogram

Description automatically generated with medium confidence

**CSS Animation**

* animation-name คือ ชื่อ animation
* animation-duration คือ ระยะเวลาของ animation จากเริ่มต้นไปสิ้นสุด
* animation-timing-function คือ รูปแบบการเล่นanimation
* animation-iteration-count คือ จํานวนการเล่น animation (infinite คือ ไม่สิ้นสุด)
* animation-direction คือ ทิศทางการเล่น (เล่นจาก frame 1 ไป frame 10 หรือ แบบย้อนกลับก็ได้)
* animation-delay คือ ระยะเวลาที่จะเริ่มต้น

**JavaScript เบื้องต้น ใช้ร่วมกับ HTML5 CSS3**

**JavaScript คืออะไร**

เป็นภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนาเว็บร่วมกับ HTML เพื่อให้เว็บมีลักษณะแบบ ไดนามิก คือ เว็บสามารถตอบสนองกับผู้ใช้งานหรือแสดงเนื้อหาที่แตกต่างกันไปโดยจะ อ้างอิงตามเว็บบราวเซอร์ที่ผู้เข้าชมเว็บใช้งานอยู่

เป็นภาษาที่ทํางานฝั่ งผู้ใช้ (Client Side Script)โดยเว็บเบราว์เซอร์จะทําหน้าที่ ประมวลผลคําสั่งที่ถูกเขียนขึ้นมาและตอบสนองต่อผู้ใช้ได้ทันที เช่น การแสดงข้อความแจ้ง เตือน (Alert) การตรวจสอบข้อมูลที่ผู้ใช้ป้อน (Validation) เป็นต้น

**ความสามารถของ JavaScript**

* สามารถเปลี่ยนแปลงรูปแบบการแสดงผลของ HTML,CSS ได้
* ตรวจสอบความถูกต้องของข้อมูลได้
* ตรวจสอบ Browser ของผู้ใช้ได้
* เก็บข้อมูลผู้ใช้ได้ เช่น การใช้ Cookie , Local Storage เป็นต้น

**รูปแบบการเขียน JavaScript**

**1.แบบ Internal** คือ กําหนด JavaScript ไว้ในส่วนของ <head></head> หรือ <body></body><script type=“text/javascript”>

document.write(“Hello World”); </script>

**2. แบบ External** คือ กําหนด JavaScript ไว้เป็นไฟล์ด้านนอกที่มีนามสกุล .js จากนั้นก็นําเข้ามาทํางานในหน้าเว็บ หรือ HTML ไฟล์<script src=“ชื่อไฟล์.js”></script>

**การแสดงข้อมูล**

* **document.write**(“ข้อความที่ต้องการแสดง”) แสดงเป็นข้อความ ตัวเลข ตัวแปร หรือแท็ก HTML ก็ได้ในหน้าเว็บ
* **alert**(“ข้อความแจ้งเตือน”) สําหรับแจ้งเตือนผู้ใช้ในหน้าเว็บ
* **Console.log**(“ข้อความ หรือ ตัวแปร”) สําหรับ debug ค่าต่างๆ แต่จะไม่แสดงผลในหน้าเว็บ

**การเขียนคําอธิบาย (Comment)**

วิธีที่ 1 โดยใช้เครื่องหมาย Slash ( / ) ใช้ในการอธิบายคําสั่งสั้นๆในรูปแบบบรรทัดเดียว

วิธีที่ 2 เขียนคําอธิบายไว้ในเครื่องหมาย /\* … \*/ ใช้ในการอธิบายคําสั่งยาวๆหรือแบบหลายบรรทัด

**ตัวแปรและชนิดข้อมูล**

ตัวแปร คือ ชื่อที่ถูกนิยามขึ้นมาเพื่อใช้เก็บค่าข้อมูลสําหรับนําไปใช้งานในโปรแกรม โดยข้อมูลอาจจะประกอบด้วย

ข้อความ ตัวเลข ตัวอักษรหรือผลลัพธ์จากการประมวลผล

ข้อมูล

**รูปแบบการตั้งชื่อ**

1. ขึ้นต้นด้วยตัวอักษรในภาษาอังกฤษตามด้วยตัวอักษรหรือตัวเลข

2. ห้ามขึ้นต้นด้วยตัวเลขหรือสัญลักษณ์พิเศษ

3. ขึ้นต้นด้วย $ (dollar sign) และ \_ (underscore) ได้

4. มีลักษณะเป็น case sensitive คือ ตัวพิมพ์เล็กพิมพ์ใหญ่จะมีค

วามหมายที่แตกต่างกัน

5. ไม่ซ้ำกับคําสงวน (Keyword)

**ตัวแปรใน JavaScript เป็นรูปแบบ Dynamic Typing**

* ตัวแปรแบบ Dynamic Typing คือชนิดตัวแปรจะเป็นอะไรก็ได้ตามค่าที่ตัวมันเก็บโดยไม่ต้องประกาศชนิดข้อมูล
* ตัวแปรแบบ Static Typing ต้องประกาศชนิดข้อมูลในตอนเริ่มต้น เช่น int, double, char เพื่อบอกว่าตัวแปรนี้จะเก็บข้อมูลชนิดไหน

**การนิยามตัวแปร**

var money;

var money=100;

money=200;

var a, b, c, d;

var x = 10, y = 20, z = 30;

\*\*\*ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กําหนดค่า จะมีค่าเป็น undefined โดยอัตโนมัติ

**หัวข้อที่เกี่ยวข้องกับตัวแปร**

**typeof** คือ เช็คชนิดข้อมูล

**null** คือ ไม่มีการกําหนดค่าถูกกําหนดค่าโดยผู้เขียน

**undefined** ไม่มีการกําหนดค่า (เป็นค่าเริ่มต้นของโปรแกรม)

**การแปลงชนิดข้อมูล (Type Conversion)**

**แปลงจาก String เป็น Number**

* x = parseInt('1.2');
* x = parseFloat('1.2');
* ใช้เครื่องหมาย (+...) เพิ่มไปข้างหน้า

**แปลงจาก Number เป็น String**

* ใช้เครื่องหมาย " " + ตัวแปร หรือ ค่าที่เป็นตัวเลข
* ใช้ toString() เช่น x.toString()

**โครงสร้างควบคุม (Control Structure)**

if(เงื่อนไขที่ 1){

คําสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง ;

}else if(เงื่อนไขที่ 2){

คําสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;

}else{

คําสั่งเมื่อทุกเงื่อนไขเป็นเท็จ ;

}

while(เงื่อนไข){

คําสั่งที่จะทําซ้ำเมื่อเงื่อนไขเป็นจริง ;

}for(let i = 1;i<=10;i++) {

คําสั่งเมื่อเงื่อนไขเป็นจริง;

}

**รูปแบบของฟังก์ชั่น**

**1.ฟังก์ชั่นที่ไม่มีการรับและส่งค่า**

function ชื่อฟังก์ชั่น(){

// คําสั่งต่างๆ

}

**การเรียกใช้งานฟังก์ชั่น**

ชื่อฟังก์ชั่น ();

**2.ฟังก์ชั่นที่มีการรับค่าเข้ามาทํางาน**

function ชื่อฟังก์ชั่น(parameter1,parameter2,.....){

// กลุ่มคําสั่งต่างๆ

}

**อาร์กิวเมนต์ คือ** ตัวแปรหรือค่าที่ต้องการส่งมาให้กับฟังก์ชัน (ตัวแปรส่ง)

**พารามิเตอร์ คือ** ตัวแปรที่ฟังก์ชันสร้างไว้สําหรับรับค่าที่จะส่งเข้ามาให้กับฟังก์ชัน (ตัวแปรรับ)

**การเรียกใช้งานฟังก์ชั่น**

ชื่อฟังก์ชั่น (argument1,argument2,.....);

**3.ฟังก์ชั่นที่มีส่งค่าออกมา**

function ชื่อฟังก์ชั่น(){

return ค่าที่จะส่งออกไป

}**4.ฟังก์ชั่นที่มีการรับค่าเข้ามาและส่งค่าออกไป**

function ชื่อฟังก์ชั่น(parameter1,parameter2,.....){

retrun ค่าที่จะส่งออกไป

}

**ฟังก์ชั่นแบบกําหนดค่าเริ่มต้น**

function ชื่อฟังก์ชั่น(name=”Mon”,parameter2,......){

// คําสั่งต่างๆ

}

**ขอบเขตตัวแปร**

* **local variable** ตัวแปรที่ทํางานอยู่ในฟังก์ชั่นมีขอบเขตการทํางานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของฟังก์ชั่น
* **global variable** ตัวแปรที่ทํางานอยู่นอกฟังก์ชั่นมีขอบเขตการทํางานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของไฟล์ที่ประกาศใช

**Array Properties & Function**

**หาจํานวนสมาชิกและเรียงลําดับ**

let color = [“แดง”, “น˵าเงิน”, “เหลือง”];

let x = color.length;

let y = color.sort();

**สมาชิกตัวแรกและตัวสุดท้าย**

let first = color[0];

let last = color[color.length-1];

**การเพิ่มสมาชิก**

color.push("สีเทา");

**แปลง Array เป็น String**

* .toString() //แปลงเป็น String
* .join(" \* "); // นําค่าแต่ละค่าในตัวแปร array มารวมกันเป็นข้อความและส่งค่ากลับเป็นข้อความที่มีตัวคั่นค่าตัวแปรแต่ละค่าตามที่กําหนด
* color.pop(); // เอาตัวสุดท้ายออก
* let x = color.pop(); //เอาตัวท้ายออกแล้วเก็บในตัวแปร x

**เรียงลําดับใน Array** let fruits = ["ส้ม", "องุ่น"];

fruits.sort();

fruits.reverse();**JavaScript Object**

let ชื่อวัตถุ = {propertyName:value}

ยกตัวอย่าง เช่น

let user = {

name:"mon", age:20 email:"mon@gmail.com"

};

let product ={name:”มะม่วง”,price:150,category:”ผลไม้”}

**การเข้าถึงข้อมูล**

objectName.propertyName

objectName["propertyName"]

ยกตัวอย่าง เช่น

user.name

user[“name”]

**JavaScript Object (Method)**

let user = {

name:"mon",

age:20,

email:"mon@gmail.com",

**getUser:function(){**

**return this.name + " " + this.email;**

**}**

};

การเรียกใช้งาน

objectName.methodName();

let data = user.getUser();

**ความแตกต่างของ Array และ Object**

* Array มี Index เป็นตัวเลข , Object กําหนดเป็นชื่อ
* Array ใช้ [] , ส่วน Object ใช้ {}

**HTML DOM (Document Object Model)**

เมื่อหน้าเว็บโหลดเสร็จเรียบร้อย Web Browser มันจะสร้าง DOM ของหน้านั้นขึ้นมา โดยมอง HTML เป็นโครงสร้างต้นไม้ (ก้อน Object) หรือเรียกว่า DOM

Diagram

Description automatically generated

Tag ต่าง ๆ ใน HTML จะเรียกว่า Element

**คุณสมบัติของ HTML DOM**

* เข้าถึงและเปลี่ยนคุณสมบัติทั้งหมดของ Element ในหน้าเว็บได้
* ควบคุมและเปลี่ยนรูปแบบ CSS ได้
* สามารถตอบสนองกับทุกเหตุการณ์ที่เกิดขึ้นหน้าเว็บได้

**เข้าถึง Element ผ่าน Id , Tag , Class**

* document.getElementById ( "ชื่อไอดี");
* document.getElementsByTagName ( "ชื่อแท็ก");
* document.getElementsByClassName("ชื่อคลาส");

**DOM Document**

* เปลี่ยนเนื้อหา HTML : element.innerHTML
* เปลี่ยนเนื้อหา Text : element.innerText
* เปลี่ยน style Element : element.style.properties = value

**ดําเนินการผ่าน Method**

* element.setAttribute(attribute, value)

**DOM Nodes**

* document.createElement(element) // สร้าง element ใหม่
* document.removeChild(element) // ลบ node ลูก
* document.appendChild(element) // นํา element ไปต่อใน node แม่
* document.replaceChild(new, old) แทนที่ element

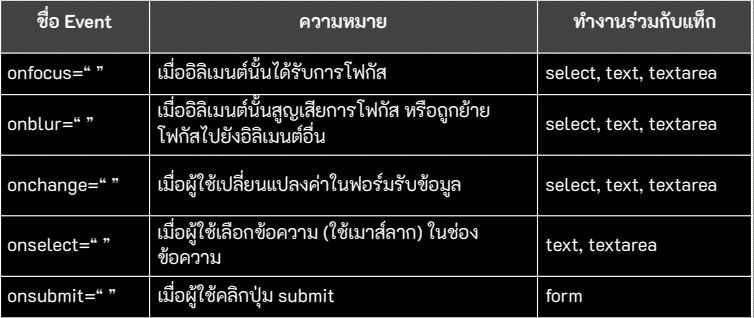
**DOM CSS Add & Remove Class**

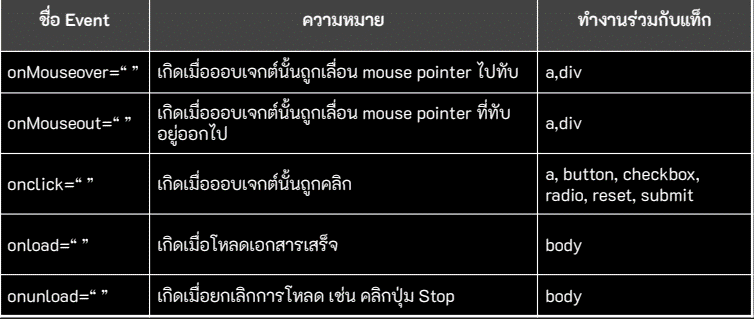
* element.classList.add(“class”); // เพิ่ม class style
* element.classList.remove(“class”); // ลบ class style
* element.classList.toggle(“class”); // สลับ class style
* element.classList.contains(“class”);// เปรียบเทียบ class style

**DOM Event**

คือ เหตุการณ์หรือการกระทําบางอย่างที่เกิดขึ้นกับอิลิเมนต์ เช่น การคลิกเมาส์ การเคลื่อนย้ายเมาส์ การกดปุ่มคีย์บอร์ด เป็นต้น

โดยผู้พัฒนาสามารถใช้อีเวนต์ที่เกิดขึ้นเป็นตัวกําหนดให้ตอบสนอง หรือกระทําบางอย่างได้ เช่น การคลิกแล้วแจ้งเตือน เป็นต้น



****

**EventListener**

คือ เหตุการณ์หรือการกระทําบางอย่างที่เกิดขึ้นกับอิลิเมนต์แต่รูปแบบการเขียนจะเขียนในฝั่ ง javascript ทั้งหมด

โครงสร้าง : element.addEventListener(event,callback)**JavaScript ES6 (For React)**

**Block Scope (let/constant)**

การประกาศใช้งาน let แทน var เนื่องจาก var เป็นตัวแแปรที่ถูกมอง global variable สามารถทํางานทะลุขอบเขต (block scope) ได้

* การใช้งาน const ในการประกาศตัวแปรที่เก็บค่าคงที่

**Arrow Function**

เป็นรูปแบบการเขียน Function ให้มีความกระชับมากยิ่งขึ้น

**แบบเดิม**

function fullname(fname,lname){

return fname+lname

}Arrow Function เป็นรูปแบบการเขียน Function ให้มีความกระชับมากยิ่งขึ้น

**แบบใหม่**

fullname=(fname,lname)=>fname+lname

**Object**

const user {

name:name,

email:email,

address:address

}

**String**

* MultiLine String สามารถทํางานกับข้อความยาวๆได้ โดยการขึ้นบรรทัดใหม่แล้วไม่มีข้อผิดพลาดเกินขึ้นโดยใช้ `
* Interpolation สามารถแทรกตัวแปรลงในพื้นที่ String ได้โดยใช้${ชื่อตัวแปร} ร่วมกับ `

**Spread Operator**

ใช้ในการกระจายสมาชิกใน Array ออกมาใช้งานโดยเติมเครื่องหมาย … ด้านหน้าตัวแปร Array

**Rest Parameter**

ใช้ในการส่งค่า Parameter เข้าไปทํางานใน Function โดยไม่จํากัดจํานวนโดยใช้เครื่องหมาย ...

**Destructuring (การสลายโครงสร้าง)**

คือ การกําหนดค่าที่อยู่ภายใน Array หรือ Object ให้กับตัวแปรโดยใช้วิธีการจับคู่ตัวแปรกับค่าใน array หรือ Object เช่น

const colors = [“ขาว”,”แดง”]

const [a,b]=colors

**Default Parameter**

การกําหนดค่าเริ่มต้นให้กับ Parameter ภายในฟังก์ชั่น

**Array เบื้องต้น**

* **join ()** แปลง Array ให้กลายเป็น string และคั่นด้วยเครื่องหมาย , (comma) หรือ เครื่องหมายอื่นๆที่กําหนดขึ้นมา
* **concat ()** รวม Array และ return Array ก้อนใหม่กลับมา โดยที่ไม่เปลี่ยนแปลงโครงสร้าง Array เดิม (ส่ง Array หรือ Element เข้าไปได้)
* **slice()** คัดลอก Array แล้ว return Array ออกไปใช้งาน

**React**

**React** คือ JavaScript Library ที่ใช้สําหรับสร้าง User Interface (UI) หรือหน้าจอของแอปพลิเคชั่น ถูกพัฒนาขึ้นโดยFacebook**Framework vs Library**

**Framework ข้อดี**

* มีรูปแบบที่ชัดเจนเป็นแบบแผนเหมาะกับงานที่ทําเป็นทีมต้องเขียนตามรูปแบบที่กําหนดไว้
* มีเครื่องมือทุกอย่างพร้อมให้เราใช้งานได้เลย

**Framework ข้อเสีย**

* ไม่มีความยืดหยุ่น
* ปรับปรุงแก้ไขการทํางานได้ยาก

**Library**

**Library ข้อดี**

* เลือกเครื่องมือหรือสิ่งที่ต้องการมาใช้งานในระบบของเราได้เลย
* มีความยืดหยุ่นสูง

**Library** **ข้อเสีย**

* ต้องทําทุกอย่างด้วยตนเอง

**แนวคิดของ React**

React จะใช้นํามาสร้างหน้าเว็บ โดยที่สามารถแบ่งการแสดงผลหน้าเว็บออกเป็นส่วนย่อยหลายๆส่วนได้ โดยที่ไม่ต้องเขียนหน้าเว็บเก็บไว้ในไฟล์เดียว เพื่อให้ง่ายต่อการจัดการ แล้วค่อยนําส่วนย่อยดังกล่าวมาประกอบกันในภายหลัง เราจะเรียกองค์ประกอบที่แบ่งออกเป็นส่วนย่อย ๆ นี้ว่า **“Component”**

ข้อดีของการสร้าง Component คือ สามารถที่จะออกแบบแล้วนํากลับมาใช้งานในภายหลังได้ โดยที่ไม่ต้องเสียเวลาเขียนใหม่

Graphical user interface, treemap chart

Description automatically generated

สําหรับการสร้าง Component นั้นก็จะมีรูปแบบการเขียนคล้ายๆกับ HTML เพียงแต่ว่า React จะใช้ส่วนที่เรียกว่า **JSX** ในการเขียนซึ่งมีความคล้ายคลึงกับ HTML มาก โดยการเขียน **JSX** นั้น คือการเขียนในไฟล์ JavaScript ไม่ใช่ ในไฟล์ HTML ซึ่งจะลงรายละเอียดในหัวข้อของ Component

**Imperative Programming**

เป็นรูปแบบการเขียนโปรแกรมแบบเดิมที่ใช้ใน JavaScript ในการบอกว่าเว็บของเราต้องการอยากจะสร้างอะไร แล้วให้กระทําอะไร เช่น

const btnEl = document.createElement(‘button’)

btnEl.innerHTML = ‘Send Data’

document.body.appendChild(btn)

**Declarative Programming**

เป็นรูปแบบการเขียนโปรแกรมที่ได้รับความนิยมในปัจจุบันเนื่องจากทําให้โค้ดมีความกระชับอ่านแล้วเข้าใจง่าย มีความหมายที่ตรงตัว

<Button>Send Data</Button>

**แนวคิดของ React**

* **Component** คือ ส่วนประกอบต่างๆที่ถูกนํามาประกอบรวมกันเป็นหน้า เว็บ (คล้ายๆ สร้าง tag ขึ้นมาใช้เอง เช่น <mon/> เป็นต้น)
* **State** คือ ข้อมูลที่อยู่ภายใน Component แต่ละตัว
* **Props** คือ ข้อมูลที่ถูกส่งจาก Component (แนวคิดมาจากการกําหนด Attribute หรือ Properties ใน HTML)

**โครงสร้างโปรเจค**

* **node\_modules** เก็บโมดูลหรือไลบรารี่ที่ทํางานภายในโปรเจคของเรา
* **public** เก็บไฟล์ public ต่างๆ เช่น รูปภาพ และ index.html เป็นต้น
* **src** สําหรับเก็บ Component หรือโครงสร้างของแอพพลิเคชั่น
* **package.json** เก็บข้อมูลต่างๆรวมถึง pakcage ที่จะใช้ทํางานภายในโปรเจค

**/Public/Index.html**

ไฟล์ที่ใช้สําหรับการแสดงผลข้อมูลบนเบราเซอร์ จะรันผ่านไฟล์ index.html สิ่งที่เราสนใจใน index.html ก็คือ คําสั่งส่วนที่อยู่ใน <body> ตรงส่วนของ <div id=”root”></div>

**/Src/App.js**

เป็นไฟล์หลักในการรันแอพพลิเคชั่นขึ้นมา โดยนําส่วนประกอบต่างๆมาประกอบกันแล้วนําไปแสดงผลในบราวเซอร์

**/Src/index.js**

เป็นไฟล์สําหรับเชื่อมการทํางานระหว่าง App.js และ Index.html

**ReactDOM Render HTML**

ReactDOM เป็นไลบราลี่เหมือนกับ React ทําหน้าที่เฉพาะในการจัดการกับ DOM และใช้เฉพาะกับ React เท่านั้น

**คําสั่ง ReactDOM.render()** จะทําการสร้าง DOM (Virtual DOM) ที่มีลักษณะของ โครงสร้างต้นไม้ แล้วนําโครงสร้างดังกล่าวใส่ลงไปยัง DOM จริงๆ (Real DOM) ซึ่งเป็น

วิธีการในการ Render JSX ออกมาแสดงผลทางหน้าจอ ยกตัวอย่าง เช่น

ReactDOM.render(<p>Hello World</p>, document.getElementById('root'));

**Virtual DOM**

**การทํางานของ Virtual DOM**

Virtual DOM มีลักษะคล้ายๆกับ DOM ใน HTML โดย Virtual DOM จะทําการคัดลอก DOM จริง (Real DOM) มาเก็บลง Memory ถ้ามีการเปลี่ยนแปลงเกิดขึ้นที่ Component ก็จะอัพเดตเฉพาะ Component ที่เปลี่ยนแปลงเท่านั้น โดยไม่จําเป็นต้องอัพเดต DOM ใหม่หมดทั้งหน้า ทําให้เกิดการทํางานได้อย่างรวดเร็ว

\*\*\*ถ้าใช้ (DOM แบบปกติจะ Refresh ทั้งหน้าเพื่ออัพเดตหน้าเว็บที่เปลี่ยนแปลงไป)

**รูปแบบการสร้าง Component**

สามารถสร้างได้ 2 รูปแบบ

● **Class Component**

● **Functional Component**

โดยทั้งคู่จะ Return HTML ออกมาและเขียน JSX ด้านในส่วนของการ Return

**Functional Component**

สร้างแบบ Component ที่ง่ายที่สุด คือ เป็นรูปแบบฟังก์ชั่นโดยสร้างฟังก์ชันธรรมดาๆ และ Return HTML ออกมา โดย กําหนดให้ตัวอักษรตัวแรกของชื่อฟังก์ชั่นเป็นตัวพิมพ์ใหญ่เสมอ เช่น

function HelloComponent() {

return <h1>สวัสดี React </h1>;

}

ReactDOM.render(<HelloComponent />, document.getElementById('root'));

**Class Component**

สร้าง Class ที่ extends มาจาก React.Component และ Return HTML ออกมา แต่ถ้าสร้างเป็นแบบ Class จะมีความสามารถในการใช้งานมากกว่าแบบ Functional

class HelloComponent extends React.Component {

render() {

return <h1>สวัสดี React</h1>;

}

}

ReactDOM.render(<HelloComponent />, document.getElementById('root'));**External Component**

การสร้าง Component ไว้เป็นไฟล์ด้านนอกที่มีนามสกุล .js จากนั้นก็นําเข้ามาทํางาน

ในหน้าเว็บ

function HelloComponent(){

return <h1>สวัสดี Component แบบ External</h1>

}

export default HelloComponent

**React JSX**

**รูปแบบการเขียน JSX**

* สามารถเขียนใน <div> / section / article / Fragment / <> ก็ได้ และต้องมีการกําหนด Tag เปิด - ปิด ทุกครั้ง
* การใส่ Class Style ที่เป็น Attribute ใน JSX จะมีการกําหนด className แทน class เนื่องจากคําว่า class เป็น keyword

**รูปแบบการเขียนแบบ div**

function HelloComponent(){

return (

<div>

<div><h3>สวัสดี</h3></div>

</div>

);

}**รูปแบบการเขียนแบบ section / article**

function HelloComponent(){

return (

<section>

<article><h3>สวัสดี </h3></article>

</section>

);

}

**รูปแบบการเขียนแบบ React.Fragment**

function HelloComponent(){

return (

<React.Fragment>

<div><h3>สวัสดี React </h3></div>

</React.Fragment>

);

}**รูปแบบการเขียนแบบ Fragment แบบ <>**

function HelloComponent(){

return (

<>

<div><h3>สวัสดี React </h3></div>

</>

);

}

**React Props**

**Props (Properties)** คือ ตัวแปรที่สามารถส่งเข้าไปใน Componentsได้ ผ่านการกําหนด Attribute ส่งผลให้ Component แต่ละตัวสามารถรับค่าจากด้านนอกเข้าไปทํางานได้ <ชื่อคอมโพเนน ชื่อพร็อพเพอร์ตี้ =ค่าที่กําหนดให้พร็อพเพอร์ตี้/>

**Keys**

**Keys** เป็นพร็อพเพอร์ตี้ที่อยู่ใน JSX โดย Keys จะมีค่าไม่ซ้ำกัน กําหนดขึ้นเพื่อให้ React นําไปเช็คว่ามี Component ใดบ้างที่เปลียนแปลงค่าไปในการ Render หน้าเว็บ

**React PropsType (Validation)**

เป็นการประกาศรูปแบบหรือชนิดของ Props ที่ส่งเข้าไปทํางานใน Component เช่น กําหนดชนิดข้อมูลของ Props หรือ บังคับให้ต้องกําหนดค่า Props ทุกครั้งที่มีการเรียกใช้งาน Component เป็นต้น

**การใช้งาน**

ชื่อคอมโพเนน.propsTypes = {

ชื่อพร็อพเพอร์ตี้ : รูปแบบของพร็อพเพอร์ตี้

}**State**

**State** คือ ตัวแปรที่เก็บข้อมูลภายใน Component คล้ายๆกับ Props แต่การใช้งาน Props นั้น ข้อมูลจะไม่สามารถเปลี่ยนแปลงค่าได้ แต่ State สามารถทําได้

ฉะนั้น ถ้าต้องการให้ข้อมูลภายในแอพสามารถเปลี่ยนแปลงค่าได้ในระหว่างรันแอพก็จะใช้ State ซึ่งรูปแบบเดิมจะเขียนภายใน

**Stateless vs Stateful**

* Stateless คือ State ที่ไม่มีการเปลี่ยนแปลงค่า
* Stateful คือ State ที่มีการเปลี่ยนแปลงค่า

**REACT HOOK**

**Hook** เป็นฟีเจอร์ที่มีอยู่ใน React เวอร์ชั่น 16.8 เป็นต้นไปใช้สําหรับจัดการเกี่ยวกับ State หรือฟีเจอร์ต่างๆที่อยู่ภายใน React โดยที่ไม่ต้องเขียนใน Class Component แต่จะใน Functional Component แทน**การใช้งาน React Hook**1. เขียนใน Functional Component

2. เขียนในส่วนของ Top-Level ของ Function (อย่าเขียนใน Condition , Loop , Nested Function นอกจากจะเขียน Custom Hook)**ตัวอย่าง React Hook**

import {useState} from ‘react’

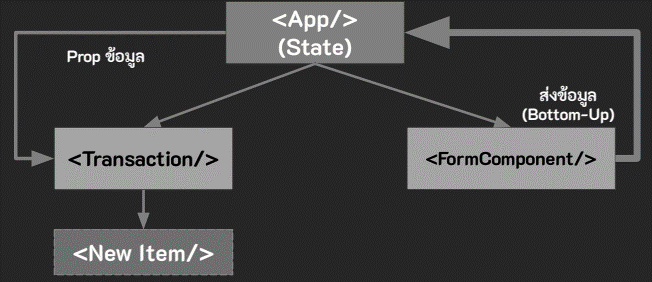
[ชื่อ State , ฟังก์ชั่นที่ใช้เปลี่ยนแปลงข้อมูลใน State ] = useState(ค่าเริ่มต้นของ State)

**ตัวอย่าง**

const [name,setName] = useState (‘kongruksiam’)

const [age,setAge] = useState(30)

จะได้ Array ที่ Destructuring จาก useState



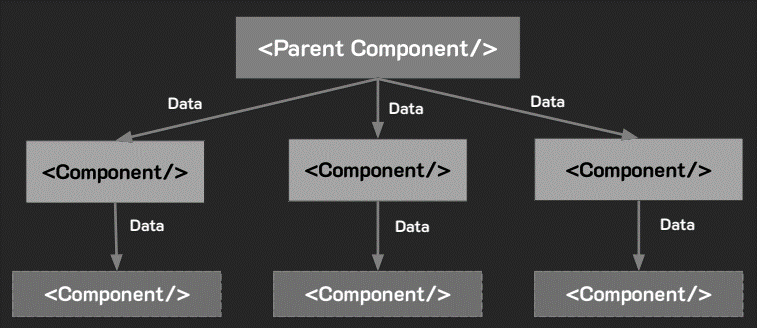
**useEffect**

**\*\* Effect** คือ ผลกระทบ

**\*\* useEffect** คือ ผลกระทบที่เกิดขึ้นภายใน Component ใช้เพื่อต้องการทราบว่าเกิดการอัพเดตหรือเปลี่ยนแปลงอะไรขึ้นบ้างภายใน Component จนส่งผลให้ Component เกิดการ Render ใหม่

โดยสาเหตุหลักๆที่ Component Render ใหม่จะมาจากการเปลี่ยนแปลงค่าที่อยู่ภายใน Props และ State นั่นเอง **การใช้งาน useEffect** จึงนํามาใช้งานเพื่อตรวจสอบการเปลี่ยนแปลงที่เกิดขึ้นภายใน Application ของเราว่ามีข้อมูลส่วนใดบ้างที่เปลี่ยนแปลงไปจากค่าหนึ่งไปสู่อีกค่าหนึ่ง จนส่งผลให้ Render Component ใหม่อีกครั้ง**Context API (Global State)**

ในกรณีที่ต้องการอยากให้แอพของเรามีข้อมูลกลางและอยากให้มีการแชร์ข้อมูลเกิดขึ้นใน Component จะทําอย่างไร เช่น ต้องการให้ทุกๆ Component สามารถเข้าถึงข้อมูลชุดเดียวกันได้ โดยที่ไม่ต้องใช้รูปแบบส่งข้อมูลจาก Component แม่ไปยัง Component ลูก (Props)



**Context API มีองค์ประกอบ 2 ส่วน** **คือ**

* Provider (Parent) ดูแลและจัดการข้อมูลแล้วนําไปส่งให้ Consumer
* Consumer (Children) นําข้อมูลที่ได้จาก Provider ไปสร้างหรือแสดงผลใน Component

**useReducer**

**\*\*\*\* เป็นการจัดการ State ในรูปแบบของ Redux**

**โดยทั่วไป State สามารถอ่านค่าได้อย่างเดียว ถ้าต้องการอยากจะเปลี่ยนแปลงค่า State จะใช้ useReducer โดยการกําหนดรูปแบบการกระทําที่เกิดขึ้นกับ State ของเราผ่านส่วนที่เรียกว่า ActionReact Router** การพัฒนาแอพด้วย React ประกอบไปด้วยการแสดงผลมากกว่า 1 หน้าจอ การที่จะทําให้ผู้ใช้งานไปยังส่วนการแสดงผลต่างๆที่เกิดขึ้น จากการใช้งาน Component ในแอพ เราจะใช้ส่วนที่เรียกว่า Route (การสร้างเส้นทาง) !!!

1.สร้าง Router/ Route เพื่อกําหนดเส้นทางการเข้าถึง Component ของการกําหนด Path

2.สร้าง Switch ในการเปลี่ยนเส้นทางการเข้าถึง Component

3.กําหนด Link เพื่อเชื่อมโยง Path กับ Component

**AJAX**

AJAX ย่อมาจากคำว่า Asynchronous JavaScript and XML ซึ่งหมายถึงการพัฒนาเว็บแอพพลิเคชั่นที่ประมวลผลในเบื้องหลังเป็นเทคนิคในการพัฒนาเว็บแอปพลิเคชันเพื่อให้สามารถโต้ตอบกับผู้ใช้ได้ดีขึ้น ทำให้ความรู้สึกการใช้งานโปรแกรมเหมือนกับเดสก์ท็อปแอปพลิเคชัน

**ตัวอย่างโค้คการทำงาน AJAX ของโดยใช้ jQuery**

$.ajax({

type: 'GET',

url: 'send-ajax-data.php',

dataType: "JSON", // data type expected from server

success: function (data) {

console.log(data);

},

error: function(error) {

console.log('Error: ' + error);

}

});

**วิธีที่ 1 ใช้ XMLHttpRequest**

var request = new XMLHttpRequest();

request.open('GET','https://tutorialzine.com/misc/files/my\_url.html', true);

request.onload = function (e) {

if (request.readyState === 4) {

// Check if the get was successful

if (request.status === 200) {

console.log(request.responseText);

} else {

console.error(request.statusText);

}

}

};

// Catch errors:

request.onerror = function (e) {

console.error(request.statusText);

};

request.send(null);

**วิธีที่ 2 ใช้ Fetch API**

fetch('./api/some.json')

.then(

function(response) {

if (response.status !== 200) {

console.log('Looks like there was a problem. Status Code:' +

response.status);

return;

}

// Examine the text in the response

response.json().then(function(data) {

console.log(data);

});

}

)

.catch(function(err) {

console.log('Fetch Error :-S', err);

});