HTML5 เบื้องต้น

ฉบับปรับปรุง 2020

ห้ามจำหน่าย!!!!

Introduction

HTML (HyperText Markup Language)
เป็นภาษาที่ใช้สำหรับสร้างเว็บเพจ มีโครงสร้างภาษาโดยใช้ตัวกำกับ (Markup Tag) เพื่อควบคุมการแสดงผลข้อมูล รูปภาพ และวัตถุอื่น ๆ **ผ่านทาง Web Browser** เช่น Google Chrome , Firefox , Safari , Microsoft Edge เป็นต้น

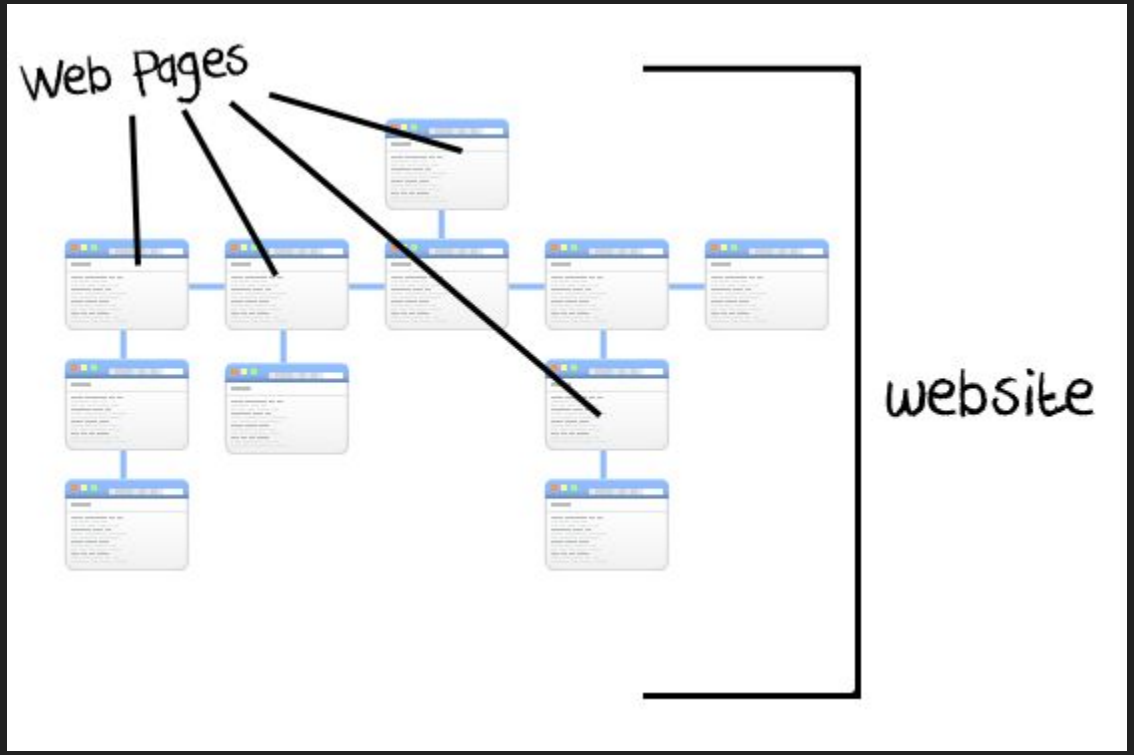
ในแต่ละ Tag จะมีส่วนที่เรียกว่า **Attribute** เพื่อควบคุมการทำงานของ Tag แต่ละตัว

การสร้างไฟล์ HTML
จะต้องอาศัย Text Editor เพื่อใช้สำหรับเขียนคำสั่งต่าง ๆ ที่ต้องการแสดงผลทางจอภาพ / เว็บเบราว์เซอร์ และเก็บเป็นไฟล์โดยมี**นามสกุล .html**

HTML 5.0

มาตรฐานของภาษา HTML มีการจัดโครงสร้างและการแสดงผลของเนื้อหาสำหรับ www มาตรฐานใหม่จะมีคุณลักษณะเด่นที่สำคัญ เช่น

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



https://mysiteauditor.com/blog/wp-content/uploads/2014/05/web-pages-versus-websites.png

DOCTYPE

การประกาศว่าเว็บเพจที่ได้สร้างขึ้นมาอ้างอิงตามมาตรฐานใด

HTML 5

<!DOCTYPE html>

HTML 4.01:

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

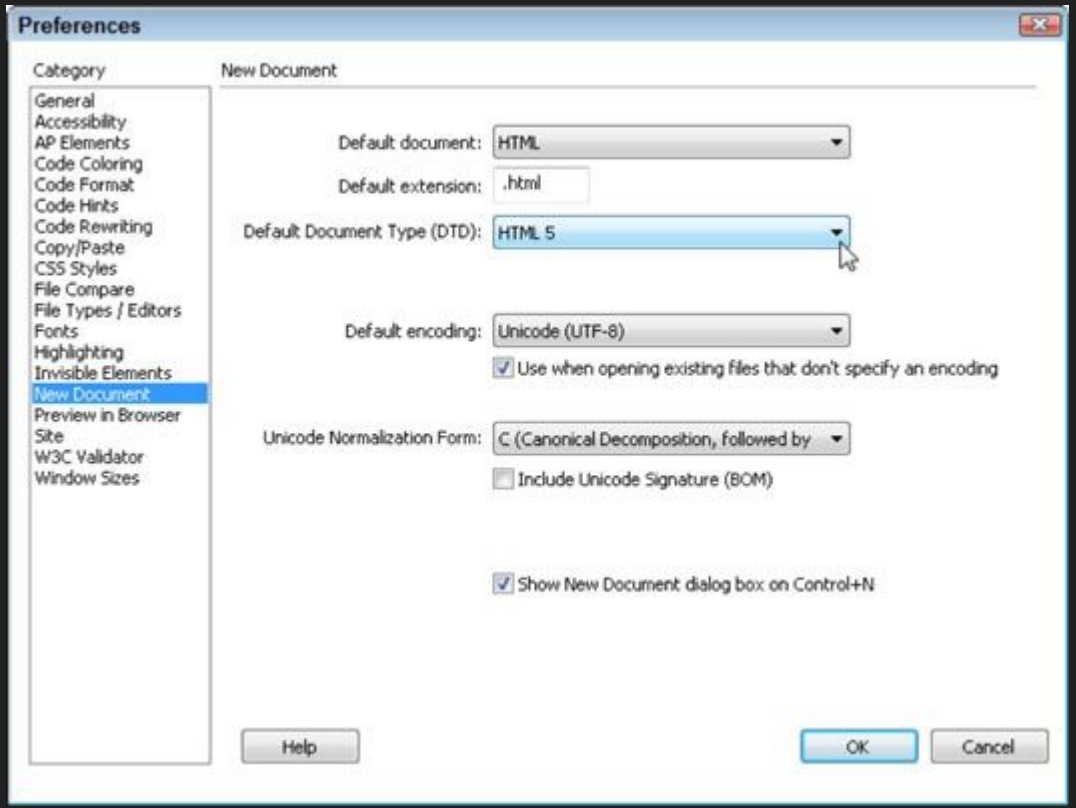
XHTML 1.1:

<!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">


Preferences
Category
General
Accessibility
AP Elements
Code Coloring
Code Format
Code Hints
Code Rewriting
Copy/Paste
CSS Styles
File Compare
File Types / Editors
Fonts
Highlighting
Invisible Elements
New Document
Preview in Browser
Site
W3C Validator
Window Sizes
New Document
Default document:
HTML
Default extension:
.html
Default Document Type (DTD):
HTML 5
Default encoding:
Unicode (UTF-8)
Use when opening existing files that don't specify an encoding
Unicode Normalization Form:
C (Canonical Decomposition, followed by
Include Unicode Signature (BOM)
Show New Document dialog box on Control+N
Help
OK
Cancel

โครงสร้างภาษา HTML

```
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

ส่วน head

ส่วน body

โครงสร้าง HTML
จะแบ่งออกเป็น 2 ส่วน ได้แก่ ส่วน head
และส่วน body โดยเรียงจากแท็ก <head> และ
<body> ตามลำดับ

โดยทั้ง 2 แท็กจะอยู่ภายใน
<html> ... </html>

โครงสร้างภาษา 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 วิดีโอและยังสามารถกำหนดคุณสมบัติพื้นฐานของเว็บได้ เช่น รูปแบบของพื้นหลัง สีของตัวอักษร

โครงสร้างภาษา HTML

<แท็กเปิด>เนื้อหา</แท็กปิด>

HTML Element

- ทุกคำสั่งที่อยู่ระหว่างแท็กเปิดและแท็กปิด HTML element บางอย่างไม่มีเนื้อหา (content) ซึ่งจะจบคำสั่งในแท็กเปิดเลย

- โดยส่วนใหญ่ HTML element มักจะมี attribute ประกอบอยู่ในแท็กด้วย

HTML Element

Tag เปิด	Element Content	Tag ปิด
<h1>	หัวข้อเรื่อง	</h1>
	เข้าสู่เว็บไซต์	

- <h1>หัวข้อเรื่อง</h1>
- เข้าสู่เว็บไซต์

Comment

ส่วนที่ใช้ในการการอธิบายโค้ด ซึ่งจะช่วยให้สามารถเข้าใจและสามารถแก้ไขโค้ดได้ในภายหลังได้

รูปแบบ

```
<!--   ข้อความอธิบายโค้ด -->
```

การกำหนดหัวเรื่อง (Heading)

จะใช้ Tag <h1> จนถึง <h6>

โดย <h1> จะเป็นการกำหนดหัวเรื่องที่มีขนาดใหญ่ที่สุดส่วน <h6> เป็นกำหนดหัวเรื่องที่มีขนาดเล็กสุด

แสดงข้อมูลเป็น Paragraphs <p>

จุดเริ่มต้นของ Paragraphs จะเริ่มที่บรรทัดใหม่ และประโยคที่ไม่ได้อยู่ใน Paragraphs เดียวกัน แต่อยู่ในตำแหน่งที่ต่อจาก Paragraphs ก็จะถูกจัดให้ขึ้นบรรทัดใหม่ทันที เช่น

```
<p>kongruksiam studio</p>
 html เบื้องต้น
```

Tag อื่นๆที่ใช้ร่วมกับ <p> ได้

- <small>
-
- <b>
- <i>
- อื่นๆ

แท็กสำหรับขึ้นบรรทัดใหม่

 เพื่อให้เนื้อหาดูเป็นระเบียบและอ่านได้ง่ายขึ้น

แท็กสำหรับสร้างเส้นคั่นในแนวนอน

<hr> สร้างเส้นคั่นให้กับเนื้อหา

แท็กรูปภาพ (HTML Images)

<img src="ตำแหน่งรูปภาพ" alt="ท่องเที่ยว" width="ความกว้าง" height="ความสูง">

แบบ Internal และแบบ external

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)

การแสดงรายการ (Lists)

```
<ol type="รูปแบบการแสดงผล">
        <li>หัวข้อย่อยรายการที่ 1</li>
        <li>หัวข้อย่อยรายการที่ 2</li>
</ol>
```

```
<ul type="รูปแบบการแสดงผล">
        <li>หัวข้อย่อยรายการที่ 1</li>
        <li>หัวข้อย่อยรายการที่ 2</li>
</ul>
```

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

- 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

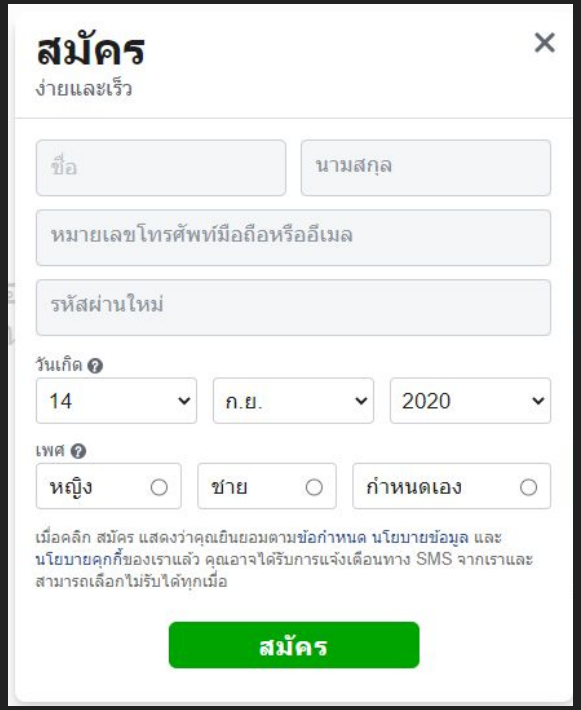
**** ใช้จัดกลุ่มข้อความหรือแท็กต่าง ๆ เข้าเป็นกลุ่มเดียวกัน เพื่อกำหนด สี รูปแบบตัวอักษร หรือ style ให้กับข้อความและแท็กภายใต้ ให้เป็นรูปแบบเดียวกัน

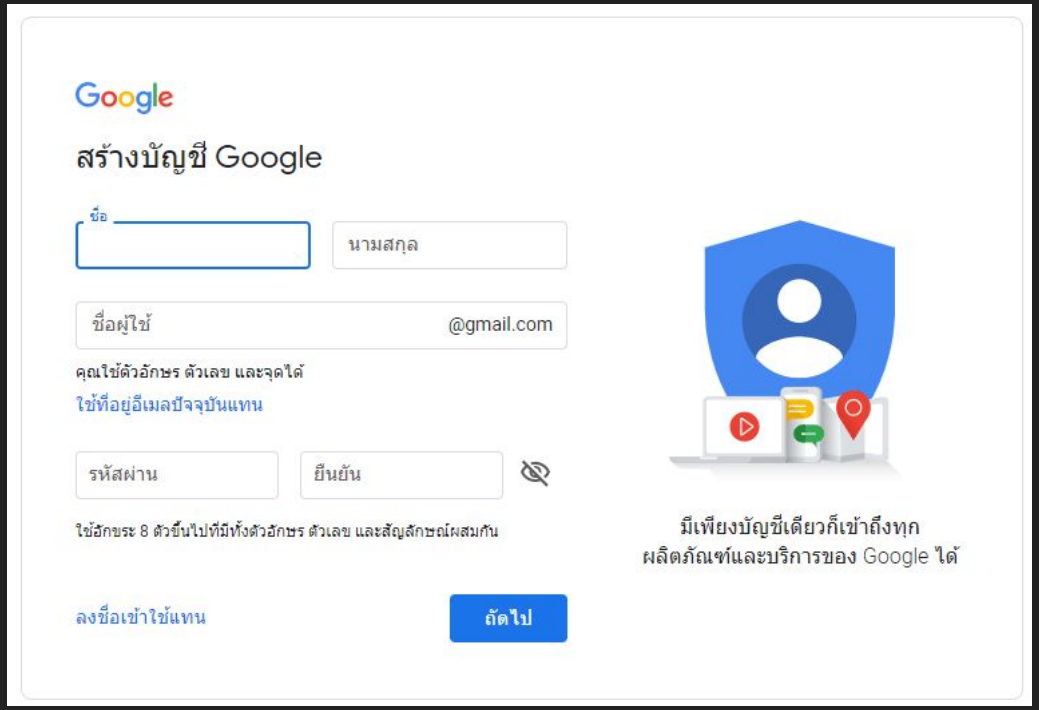
** ** ใช้จัดกลุ่มข้อความหรือแท็กต่าง ๆ เข้าเป็นกลุ่มเดียวกัน ลักษณะคล้ายๆกับ แต่แตกต่างกันตรงที่แท็ก div จะมีการขึ้นบรรทัดใหม่ก่อนเริ่มแสดงข้อความภายใต้แท็ก div

HTML FORM

การพัฒนาเว็บแอปพลิเคชั่นจำเป็นต้องมีการสร้างแบบฟอร์ม ที่ผู้ใช้งานสามารถป้อนข้อมูลต่างๆได้ เพื่อนำข้อมูลที่ป้อนนั้น ไปทำการประมวลผลอีกทีโดยการรับค่าข้อมูลจะดำเนินการ ผ่าน <form>....</form>

HTML FORM


สมัคร
ง่ายและเร็ว
ชื่อ
นามสกุล
หมายเลขโทรศัพท์มือถือหรืออีเมล
รหัสผ่านใหม่
วันเกิด
14
ก.ย.
2020
เพศ
หญิง
ชาย
กำหนดเอง
เมื่อคลิก สมัคร แสดงว่าคุณยินยอมตามข้อกำหนด นโยบายข้อมูล และนโยบายคุกกี้ของเราแล้ว คุณอาจได้รับการแจ้งเตือนทาง SMS จากเราและสามารถเลือกไม่รับได้ทุกเมื่อ
สมัคร


Google
สร้างบัญชี Google
ชื่อ
นามสกุล
ชื่อผู้ใช้
@gmail.com
คุณใช้ตัวอักษร ตัวเลข และจุดได้
ใช้ที่อยู่อีเมลปัจจุบันแทน
รหัสผ่าน
ยืนยัน
ใช้อักขระ 8 ตัวขึ้นไปที่มีทั้งตัวอักษร ตัวเลข และสัญลักษณ์ผสมกัน
ลงชื่อเข้าใช้แทน
ถัดไป
มีเพียงบัญชีเดียวก็เข้าถึงทุกผลิตภัณฑ์และบริการของ Google ได้

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

Block Element

- <address>
- <article>
- <aside>
- <blockquote>
- <canvas>
- <dd>
-
- <dl>
- <table>
- <dt>
- <fieldset>
- <figcaption>
- <figure>
- <footer>
- <form>
- <h1>-<h6>
- <header>
- <tfoot>
- <hr>
-
- <main>
- <nav>
- <noscript>
-
- <p>
- <pre>
- <section>
- <tfoot>
-
- <video>

Inline Element

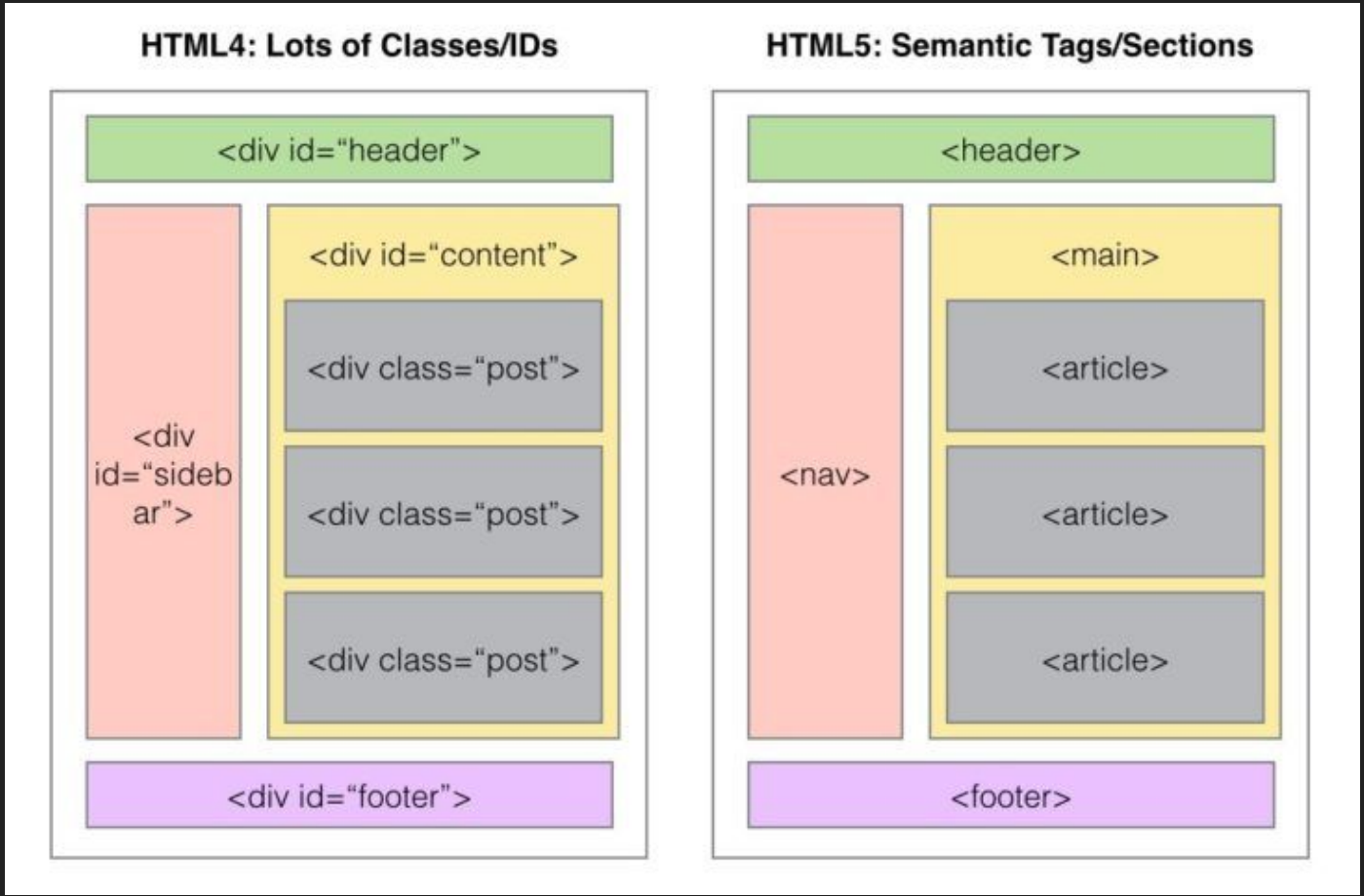
- <a>
- <abbr>
- <acronym>
- <b>
- <bdo>
- <big>
-
- <button>
- <cite>
- <code>
- <dfn>
-
- <i>
- <img>
- <input>
- <kbd>
- <label>
- <map>
- <object>
- <output>
- <q>
- <samp>
- <script>
- <select>
- <small>
-
-
- <sub>
- <sup>
- <textarea>
- <time>
- <tt>
- <var>

Class & ID

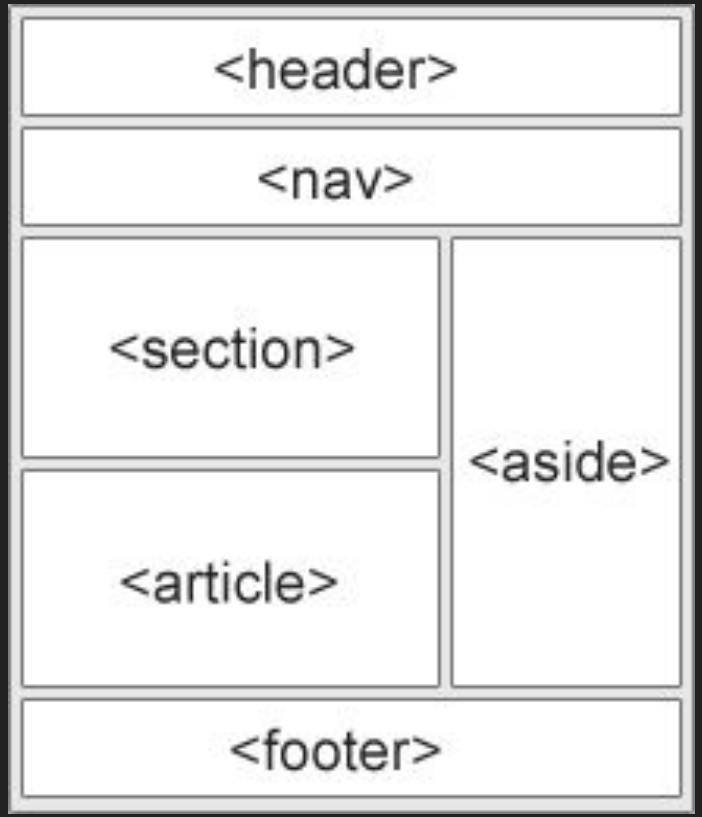
- Class การประกาศค่า Attribute “class” ในแท็กที่ต้องการ
- ID เป็นการกำหนดรหัสเฉพาะของแท็กด้วยการประกาศค่า Attribute “id” ในแท็กที่ต้องการ เพื่อนำไปแสดงผล เหมือนกับ Class แต่ค่า id จะไม่สามารถซ้ำกันได้

Semantic Tag

การใช้ Semantic tag ถูกนำมาใช้แทน div หลายๆชั้น ในหน้าเว็บจะส่งผลทำให้โครงสร้าง html มีความหมายตรงตัวชัดเจนมากยิ่งขึ้น



https://thefrisky.com/wp-content/uploads/2019/05/html-semantic-tags-html5-sectioning-high-level-730x548.jpg



https://www.w3big.com/images/img_sem_elements.gif

- **<header>**

คือ ส่วนหัวของเว็บ

- **<nav>**

คือ เมนูของเว็บ หรือ ลิงค์ไปเว็บอื่นๆ

- **<article>**

คือ ส่วนที่แสดงเนื้อหาของเว็บ

- **<section>**

คือ กลุ่มหัวข้อย่อย

- **<aside>**

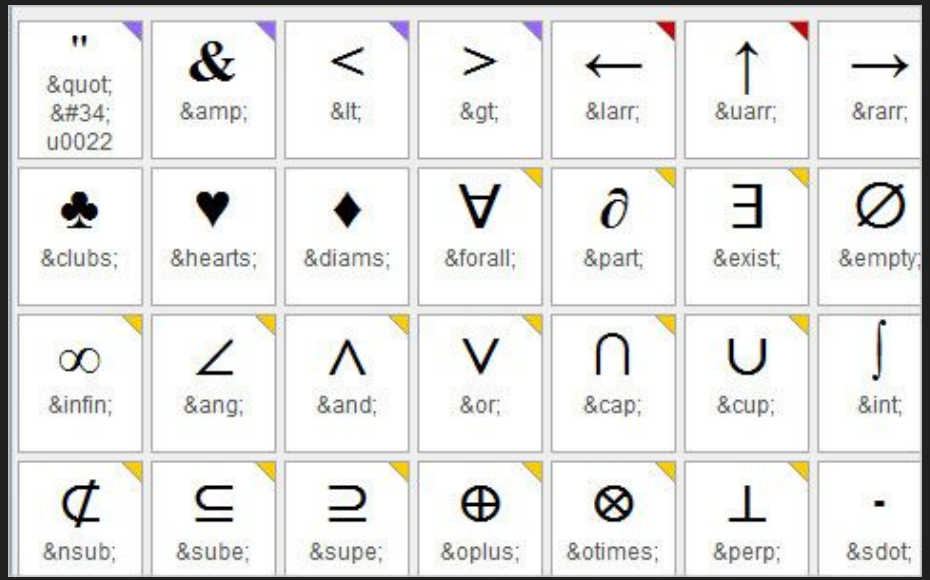
คือ เนื้อหาอื่นๆที่แยกจากเนื้อหาหลัก

- **<footer>**

คือ ส่วนท้ายของหน้าเว็บ

HTML | Character Entity

อักขระพิเศษใน HTML ใช้ในการแสดงผลข้อมูลในหน้าเว็บ



" " " u0022	& &	< <	> >	← ←	↑ ↑	→ →
♣ ♣	♥ ♥	♦ ♦	∀ ∀	∂ ∂	∃ ∃	∅ ∅
∞ ∞	∠ ∠	∧ ∧	∨ ∨	∩ ∩	∪ ∪	∫ ∫
⊄ ⊄	⊆ ⊆	⊇ ⊇	⊕ ⊕	⊗ ⊗	⊥ ⊥	· ⋅

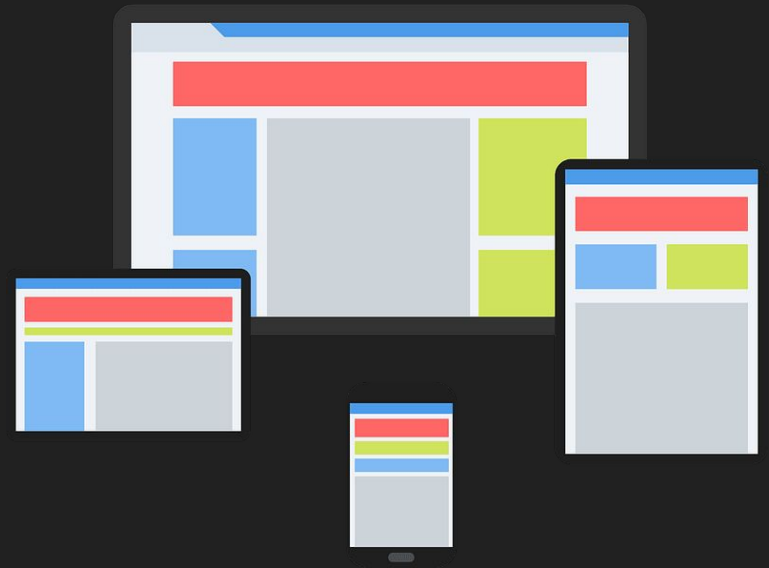
https://rapidpurple.com/v2/wp-content/uploads/2011/05/typography-174.jpeg

CSS เบื้องต้น

Cascading Style Sheet (2020)

Responsive Web Design

การออกแบบเว็บที่ตอบสนองและแสดงผลได้ดีบนอุปกรณ์ต่างๆหรือขนาดหน้าจอที่หลากหลาย เช่น จอคอมพิวเตอร์ หรือ สมาร์ทโฟน เป็นต้น



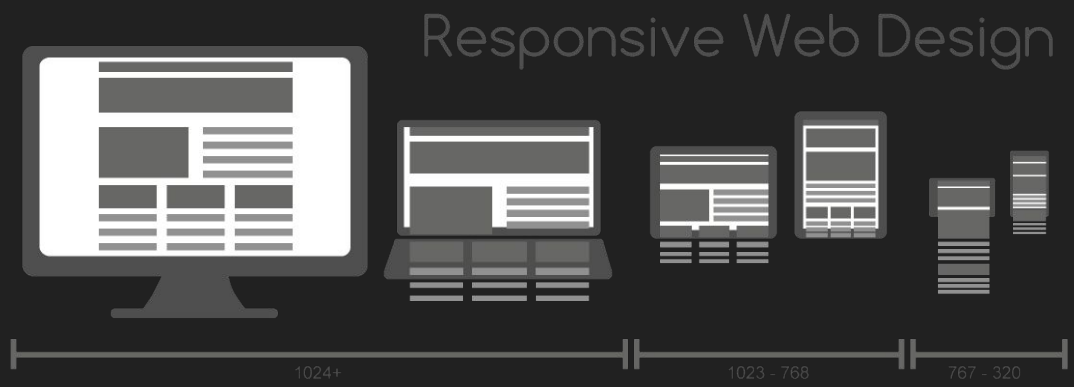
Responsive Web Design (Media Query)

Media Query คือ รูปแบบการเขียน Style ให้แสดงผลตามขนาดหน้าจอที่แตกต่างกัน

```
@media ขนาดอุปกรณ์ {
        .....style.....

}

@media screen , printer {
        .....style.....
}
```



Responsive Web Design (Media Query)

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 ในปัจจุบันอาจจะไม่สามารถแสดงผลได้ตามที่ต้องการ เช่น ถ้าต้องการดูเนื้อหาในหน้าเว็บต้องทำการ zoom และ scroll หน้าจอเท่านั้น

ในปัจจุบันจึงได้มีการพัฒนาและเพิ่ม meta tag ชื่อว่า viewport เข้ามาในหน้าเว็บเพื่อบอกว่าให้เว็บแสดงผลบน Smart Phone , Tablet ได้ผ่านการอ้างอิงพื้นที่หรือสัดส่วนของ Web Browser ที่อยู่ในอุปกรณ์นั้นๆ ในการแสดงผลหน้าเว็บให้เหมาะสมกับอุปกรณ์ดังกล่าว

องค์ประกอบของ Viewport

Viewport Units

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

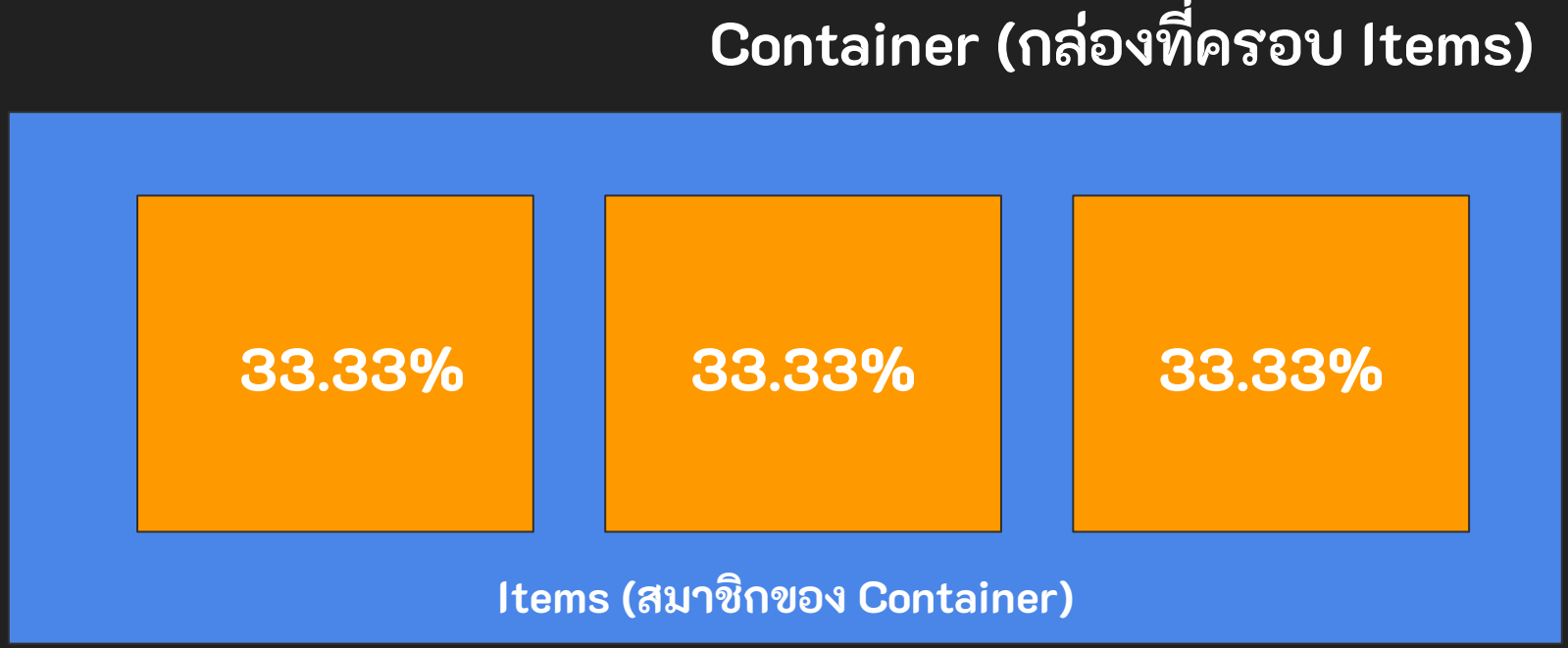
รู้จักกับ Flexbox

Flexbox คือเครื่องมือใน CSS ที่ช่วยให้การจัดการ element ต่างๆในหน้าเว็บมีความง่ายและยืดหยุ่นมากยิ่งขึ้น โดยทั่วไปการจัดตำแหน่ง element ต่างๆต้องใช้ layout mode คือ block , inline , position และอื่นๆ

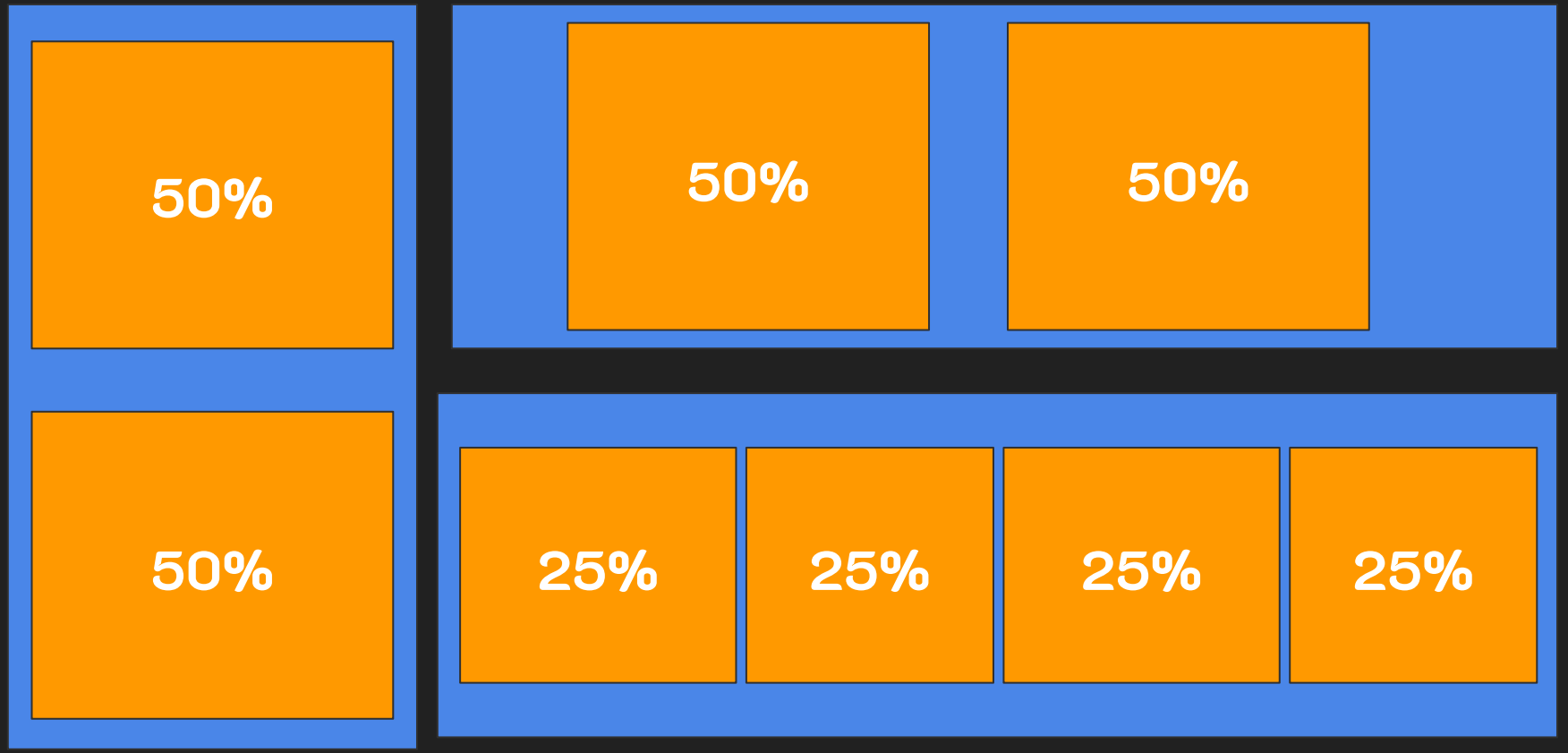
การพัฒนาเว็บในปัจจุบันมีความซับซ้อนมากยิ่งขึ้นทำให้การใช้ layout mode แบบเดิมไม่ตอบโจทย์เท่าที่ควร จึงได้มีการพัฒนา flexbox ขึ้นมาเพื่อจัดใช้ในการจัดการ element ให้มีความยืดหยุ่นสูง โดยมีคุณสมบัติดังนี้

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

องค์ประกอบของ Flexbox



จัดเรียงได้หลากหลายแบบ



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 มิติ เช่น เรียงลำดับจากบนลงล่าง ซ้ายไปขวา เป็นต้น

แต่ Grid Layout ถูกออกแบบมาเพื่อจัดการ layout แบบ 2 มิติ คือ มีทั้งแบบแนวนอนและแนวตั้งในเวลาเดียวกับ หรือมองง่ายๆก็เป็นแบบตาราง

CSS Grid Layout

การใช้งาน
display : grid;

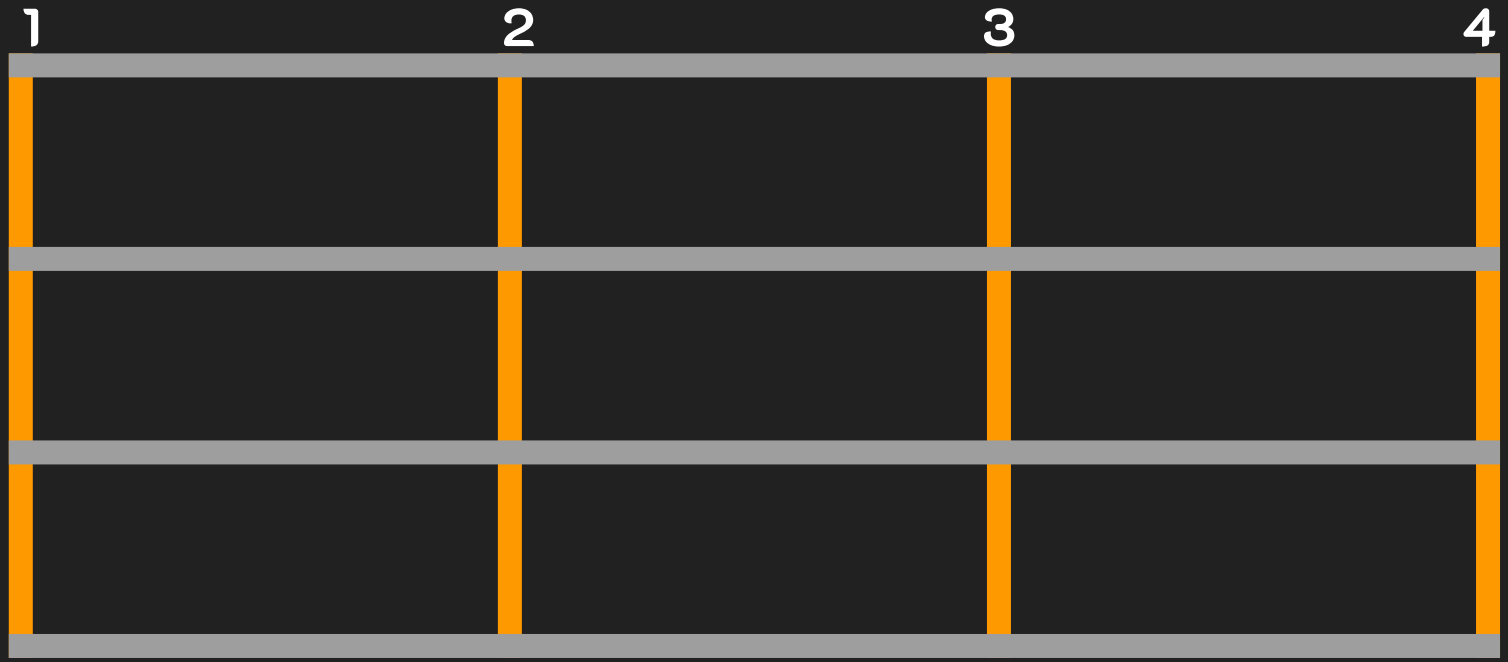
กำหนดขนาดแถว (ความสูง) :

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

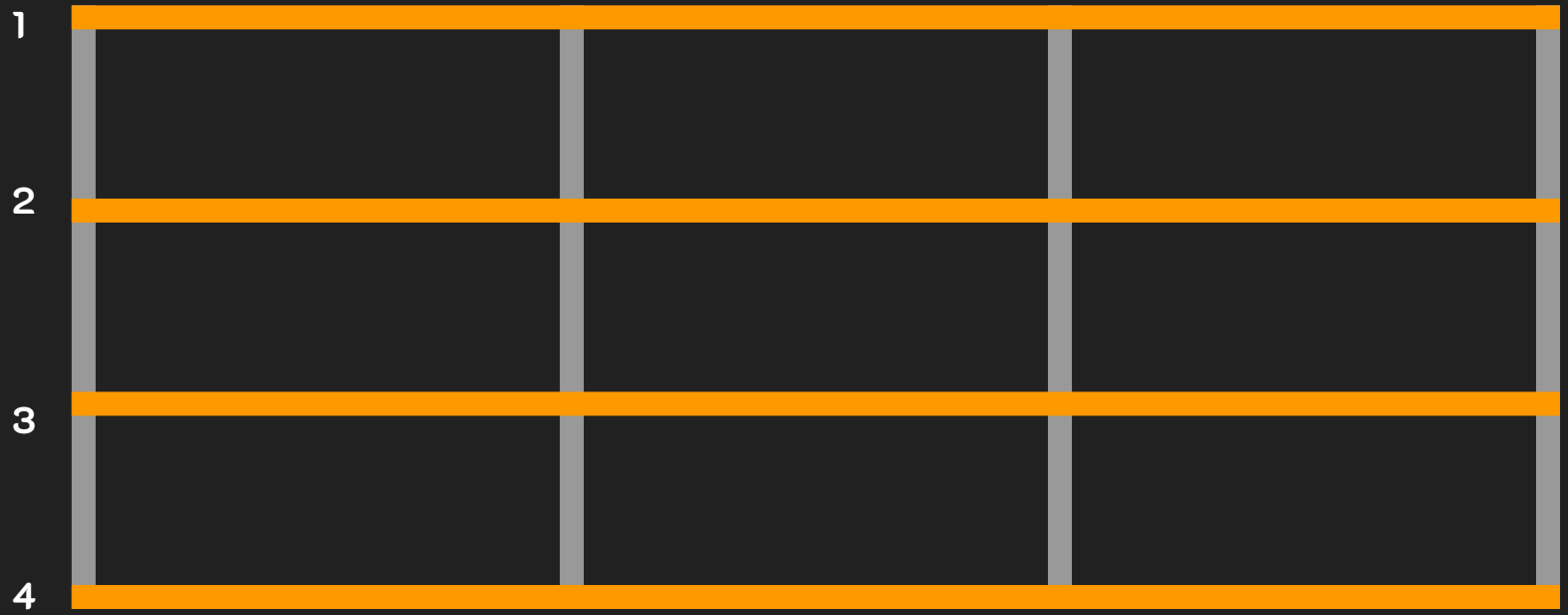
กำหนดขนาดคอลัมน์ (ความกว้าง) :

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

กำหนดคอลัมน์เริ่มต้น -สิ้นสุด



กำหนดแถวเริ่มต้น -สิ้นสุด



Gird Properties อื่นๆ

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

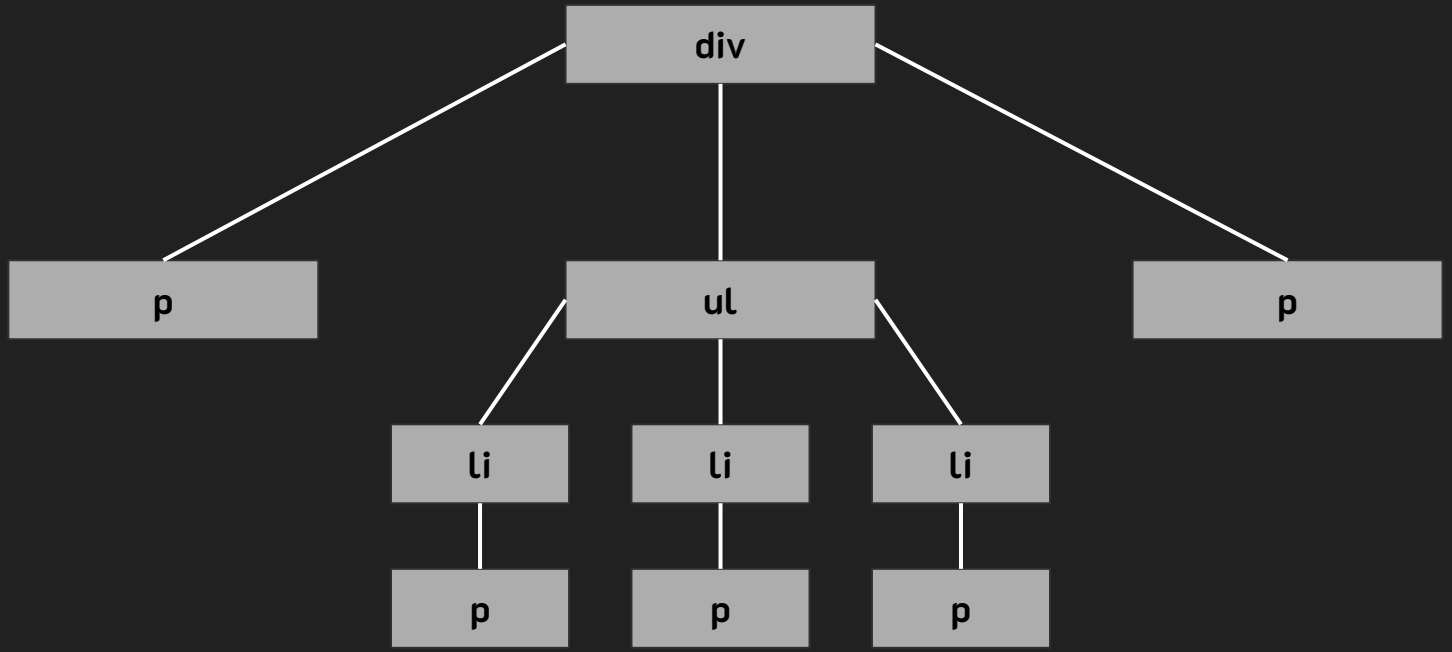
Advance Selectors

Advance Selectors

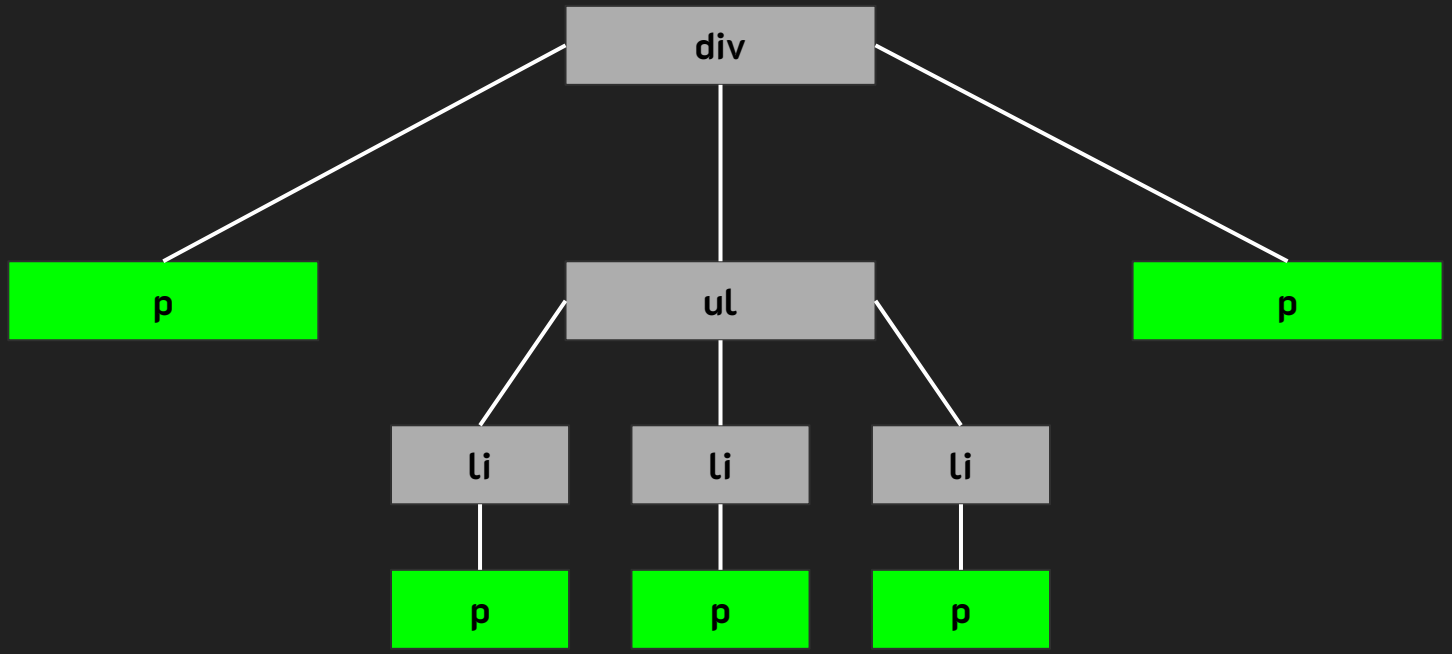
- Targeted Selectors
  - Descendant Selector
  - Child Selector

- Style By Attribute
- Special Attribute
- Pseudo Selectors
  - first-child
  - last-child
  - nth-child
- Before & After Pseudo

Targeted Selectors

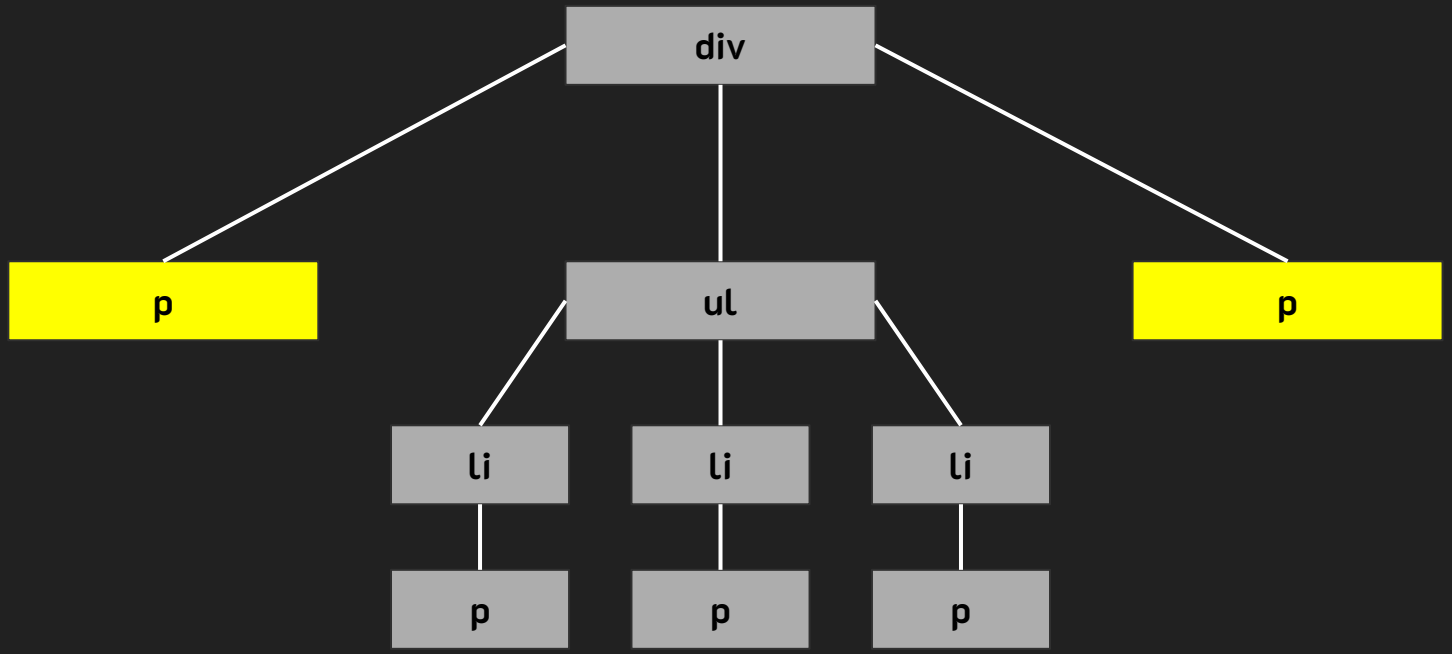


Targeted Selectors



Descendant Selector

Targeted Selectors



Child Selector

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

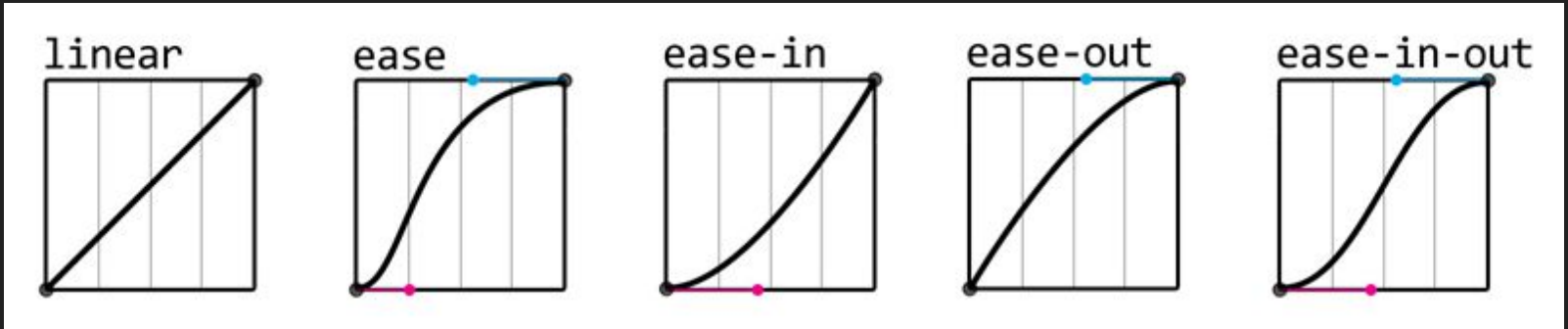
- translate (x,y) กำหนดตำแหน่ง element
- scale(x,y) ขยาย element
- rotate (มุม deg) กำหนดการหมุนของ element
- skewX(มุม deg) กำหนดการเอียงของ element แกน x
- skewY (มุม deg) กำหนดการเอียงของ element แกน y

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



CSS Animation

- **animation-name** คือ ชื่อ animation
- **animation-duration** คือ ระยะเวลาของ animation จากเริ่มต้นไปสิ้นสุด
- **animation-timing-function** คือ รูปแบบการเล่น animation
- **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

ร่วมกับ HTML5 CSS3 [ฉบับปรับปรุง 2020]

สำหรับใช้งานร่วมกับ HTML CSS

JavaScript คืออะไร

เป็นภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนาเว็บร่วมกับ HTML เพื่อให้เว็บมีลักษณะแบบไดนามิก คือ เว็บสามารถตอบสนองกับผู้ใช้งานหรือแสดงเนื้อหาที่แตกต่างกันไปโดยจะอ้างอิงตามเว็บบราวเซอร์ที่ผู้เข้าชมเว็บใช้งานอยู่

เป็นภาษาที่ทำงานฝั่งผู้ใช้ (Client Side Script)โดยเว็บเบราว์เซอร์จะทำหน้าที่ประมวลผลคำสั่งที่ถูกเขียนขึ้นมาและตอบสนองต่อผู้ใช้ได้ทันที เช่น การแสดงข้อความแจ้งเตือน (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">
        ..... Statement.....
</script>
```

```
<script type="text/javascript">
    document.write("Kong Ruksiam");
</script>
```

รูปแบบการเขียน JavaScript

2. แบบ External คือ กำหนด JavaScript ไว้เป็นไฟล์ด้านนอกที่มีนามสกุล .js จากนั้นก็นำเข้ามาทำงานในหน้าเว็บ หรือ HTML ไฟล์

```
<script src="ชื่อไฟล์.js"></script>
```

```
document.write("KongRuksiam");
document.write("<br>");
document.write("JavaScript เบื้องต้น");
```

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 (/) ใช้ในการอธิบาย**คำสั่งสั้นๆในรูปแบบบรรทัดเดียว**

วิธีที่ 2 เขียนคำอธิบายไว้ในเครื่องหมาย /* ... */ ใช้ในการอธิบายคำสั่ง**ยาวๆหรือแบบหลายบรรทัด**

ตัวแปรและชนิดข้อมูล

ตัวแปร คือ **ชื่อที่ถูกนิยามขึ้นมา**เพื่อใช้เก็บค่าข้อมูลสำหรับนำไปใช้งานในโปรแกรม โดยข้อมูลอาจจะประกอบด้วย ข้อความ ตัวเลข ตัวอักษรหรือผลลัพธ์จากการประมวลผลข้อมูล

รูปแบบการตั้งชื่อ

1. ขึ้นต้นด้วยตัวอักษรในภาษาอังกฤษตามด้วยตัวอักษรหรือตัวเลข
2. ห้ามขึ้นต้นด้วยตัวเลขหรือสัญลักษณ์พิเศษ
3. ขึ้นต้นด้วย $ (dollar sign) และ _ (underscore) ได้
4. มีลักษณะเป็น case sensitive คือ ตัวพิมพ์เล็กพิมพ์ใหญ่จะมีความหมายที่แตกต่างกัน
5. ไม่ซ้ำกับคำสงวน (Keyword)

ตัวแปรใน JavaScript เป็นรูปแบบ Dynamic Typing

- **ตัวแปรแบบ Dynamic Typing** คือชนิดตัวแปรจะเป็นอะไรก็ได้ตามค่าที่ตัวมันเก็บโดยไม่ต้องประกาศชนิดข้อมูล

- **ตัวแปรแบบ Static Typing** ต้องประกาศชนิดข้อมูลในตอนเริ่มต้น เช่น int, double, char เพื่อบอกว่าตัวแปรนี้จะเก็บข้อมูลชนิดไหน

การนิยามตัวแปร

var (เปลี่ยนแปลงค่าในตัวแปรได้)

var ชื่อตัวแปร;

var ชื่อตัวแปร = ค่าเริ่มต้น;

var ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

```
var money;

var money=100;

money=200;


var a, b, c, d;
var x = 10, y = 20, z = 30;
```

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น undefined โดยอัตโนมัติ

การนิยามตัวแปร (2015)

let (เปลี่ยนแปลงค่าในตัวแปรได้)

let ชื่อตัวแปร;

let ชื่อตัวแปร = ค่าเริ่มต้น;

let ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

```
let money;

let money=100;

money=200;

let a, b, c, d;
let x = 10, y = 20, z = 30;
```

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น undefined โดยอัตโนมัติ

การนิยามตัวแปร (2015)

const (ค่าคงที่)

const ชื่อตัวแปร = ค่าของตัวแปร;

เช่น

```
const money=100;
money=200;// เปลี่ยนแปลงค่าเดิมไม่ได้
```

Data Type	คำอธิบาย	รูปแบบข้อมูล
boolean	ค่าทางตรรกศาสตร์	True /False
number	ตัวเลขที่ไม่มีจุดทศนิยม	20
	ตัวเลขที่มีจุดทศนิยม	30.15
string	ข้อความ	“kongruksiam”
object	ข้อมูลเชิงวัตถุ	{firstName:”kong”, lastName:”ruksiam”, age:20};
array	ชุดข้อมูล	[”มะม่วง”, ”มะละกอ”, ”ส้ม”]

หัวข้อที่เกี่ยวข้องกับตัวแปร

- **typeof** คือ เช็คชนิดข้อมูล
- **null** คือ ไม่มีการกำหนดค่าถูกกำหนดค่าโดยผู้เขียน
- **undefined** ไม่มีการกำหนดค่า (เป็นค่าเริ่มต้นของโปรแกรม)

จัดการตัวเลข (Number)

```
let x ,y ;
x = 20; // integer
y = 20.15; // float
```

จัดการอักขระและข้อความด้วย string

การประกาศ string ขึ้นมาใช้ ต้องกำหนดเนื้อหาหรือค่าอยู่ในเครื่องหมาย ’ (single quote) หรือ ” (double quote)

```
let a = ’kongruksiam’;
let b = ”สอน javascript เบื้องต้น”;
let c = ’basic to advance’;
```

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()

อาร์เรย์ (Array) คืออะไร

ความหมายที่ 1 ชุดของตัวแปรที่อยู่ในรูปลำดับใช้เก็บค่าข้อมูลให้อยู่ในกลุ่มเดียวกัน ข้อมูลภายในอาร์เรย์จะถูกเก็บบนหน่วยความจำในตำแหน่งที่ต่อเนื่องกัน โดยขนาดของอาร์เรย์จะเล็กหรือใหญ่ขึ้นกับจำนวนมิติที่กำหนดขึ้น

อาร์เรย์ (Array) คืออะไร

ความหมายที่ 2 เป็นตัวแปรที่ใช้ในการเก็บข้อมูลที่มีลำดับที่ต่อเนื่อง ซึ่งข้อมูลมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงได้เพียงชื่อเดียว และใช้หมายเลขกำกับ (index) ให้กับตัวแปรเพื่อจำแนกความแตกต่างของค่าตัวแปรแต่ละตัว

การสร้าง Array

วิธีที่ 1 สร้างโดยใช้คำสั่ง Array()

let ชื่ออาร์เรย์ = new Array( );

let ชื่ออาร์เรย์= Array(สมาชิกตัวที่1, สมาชิกตัวที่2, ...);

เช่น

let myArray = new Array( );

myArray[0] = 2000;

let days = Array(“จันทร์”, “อังคาร”, “พุธ”);

การสร้าง Array

วิธีที่ 2 สร้างโดยใช้เครื่องหมาย []

let ชื่ออาร์เรย์ = [สมาชิกตัวที่1, สมาชิกตัวที่2, ...];

เช่น

let color = [“แดง”, “น้ำเงิน”, “เหลือง”];

การเข้าถึงสมาชิก

ชื่ออาร์เรย์[เลขลำดับ]

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
color[0]
color[1]
```

ตัวดำเนินการ (Operator)

กลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ในการเขียนโปรแกรม

A+B

1. ตัวดำเนินการ (Operator)
2. ตัวถูกดำเนินการ (Operand)

ตัวดำเนินการทางคณิตศาสตร์

Operator	คำอธิบาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ

ตัวดำเนินการเปรียบเทียบ

**** ชนิดข้อมูล boolean

Operator	คำอธิบาย
==	เท่ากับ
!=	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าเท่ากับ
<=	น้อยกว่าเท่ากับ

ตัวดำเนินการทางตรรกศาสตร์

Operator	คำอธิบาย
&&	AND
||	OR
!	NOT

ตัวดำเนินการทางตรรกศาสตร์

a	!a	a	b	a && b	a || b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

ตัวดำเนินการเพิ่มค่า - ลดค่า

Operator	รูปแบบการเขียน	ความหมาย
++ (Prefix)	++a	เพิ่มค่าให้ a ก่อน 1 ค่าแล้วนำไปใช้
++ (Postfix)	a++	นำค่าปัจจุบันใน a ไปใช้ก่อนแล้วค่อยเพิ่มค่า
-- (Prefix)	--b	ลดค่าให้ b ก่อน 1 ค่าแล้วนำไปใช้
-- (Postfix)	b--	นำค่าปัจจุบันใน b ไปใช้ก่อนแล้วค่อยลดค่า

Compound Assignment

Assignment	รูปแบบการเขียน	ความหมาย
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

ลำดับความสำคัญของตัวดำเนินการ

ลำดับที่	เครื่องหมาย	ลำดับการทำงาน
1	( )	
2	++ , --	ซ้ายไปขวา
3	* , / , %	ซ้ายไปขวา
4	+ , -	ซ้ายไปขวา
5	< , <= , > , >=	ซ้ายไปขวา
6	== , !=	ซ้ายไปขวา
7	&&	ซ้ายไปขวา
8	||	ซ้ายไปขวา
9	= , += , -= , *= , /= , %=	ขวาไปซ้าย

กรณีศึกษา

1. 5+8 *9
2. 10 - 4+2
3. 10 - (2+1)
4. 5 * 2 – 40 / 5
5. 7+8/2+25

โครงสร้างควบคุม (Control Structure)

คือ กลุ่มคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรม

- แบบลำดับ (Sequence)
- แบบมีเงื่อนไข (Condition)
- แบบทำซ้ำ (Loop)

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- if
- Switch..Case

รูปแบบคำสั่งแบบเงื่อนไขเดียว

- **if statement**

เป็นคำสั่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม

ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่งต่างๆ ที่กำหนดภายใต้เงื่อนไขนั้นๆ

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบ 2 เงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ ;
}
```

ข้อควรระวังการเขียน if เพื่อตรวจสอบเงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบหลายเงื่อนไข

```
if(เงื่อนไขที่ 1){
        คำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง ;
}else if(เงื่อนไขที่ 2){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
}else if(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        คำสั่งเมื่อทุกเงื่อนไขเป็นเท็จ ;
}
```

if..else แบบลดรูป (Ternary Operator)

ตัวแปร = (เงื่อนไข) ? คำสั่งเมื่อเงื่อนไขเป็นจริง : คำสั่งเมื่อเงื่อนไขเป็นเท็จ;

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ
}
```

การเขียน if ซ้อน if

```
if(เงื่อนไขที่ 1){
    if(เงื่อนไขที่ 2 ){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
    }
}
```

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- **Switch..Case**

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้ายๆกับ if แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดเป็นทางเลือกผ่านคำสั่ง case

รูปแบบคำสั่ง

```
switch(สิ่งที่ต้องการตรวจสอบ) {

      case ค่าที่ 1 : คำสั่งที่ 1;
            break;
       case ค่าที่ 2 : คำสั่งที่ 2;
            break;
           ......
       case ค่าที่ N : คำสั่งที่ N;
            break;

      default : คำสั่งเมื่อไม่มีค่าที่ตรงกับที่ระบุใน case
}
```

***คำสั่ง break จะทำให้โปรแกรมกระโดดออกไปทำงานนอกคำสั่ง switch ถ้าไม่มีคำสั่ง break โปรแกรมจะทำคำสั่งต่อไปเรื่อยๆ จนจบการทำงาน

รูปแบบคำสั่ง

```
switch(month) {

        case 1 : console.log(“มกราคม”);
             break;
        case 2: console.log(“กุมภาพันธ์”);
              break;
                  ……
         case ค่าที่ N : คำสั่งที่ N;
                   break;
        default : console.log(“ไม่พบเดือน”);
}
```

กำหนดให้ตัวแปร month เก็บตัวเลข

Switch..Case VS if Statement

```
switch(month) {
        case 1 : console.log(“มกราคม”);
            break;
        case 2: console.log(“กุมภาพันธ์”);
           break;
                  .....
         case ค่าที่ N : คำสั่งที่ N;
                    break;
        default : console.log(“ไม่พบเดือน”);
}
```

```
if(month==1){
        console.log(“มกราคม”);
}elseif(month==2){
        console.log(“กุมภาพันธ์”);
}elseif(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        System.out.println(“ไม่พบเดือน”);
}
```

แบบทำซ้ำ (Loop)

กลุ่มคำสั่งที่ใช้ในการวนรอบ (loop) โปรแกรมจะทำงานไปเรื่อยๆจนกว่าเงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะหยุดทำงาน

- While
- For
- Do..While

คำสั่งที่เกี่ยวข้องกับ Loop

- **break** ถ้าโปรแกรมพบคำสั่งนี้จะหลุดจากการทำงานในลูปทันที เพื่อไปทำคำสั่งอื่นที่อยู่นอกลูป

- **continue** คำสั่งนี้จะทำให้หยุดการทำงานแล้วย้อนกลับไปเริ่มต้นการทำงานที่ต้นลูปใหม่

คำสั่ง While

- **While Loop**

จะทำงานตามคำสั่งภายใน while ไปเรื่อยๆเมื่อเงื่อนไขที่กำหนดเป็นจริง

```
while(เงื่อนไข){
    คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
}
```

คำสั่ง For

- **For Loop**

เป็นรูปแบบที่ใช้ในการตรวจสอบเงื่อนไข มีการกำหนดค่าเริ่มต้น และเปลี่ยนค่าไปพร้อมๆกัน เมื่อเงื่อนไขในคำสั่ง for เป็นจริงก็จะทำงานตามคำสั่งที่แสดงไว้ภายในคำสั่ง for ไปเรื่อยๆ

โครงสร้างคำสั่ง

```
for(ค่าเริ่มต้นของตัวแปร; เงื่อนไข; เปลี่ยนแปลงค่าตัวแปร) {
        คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

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

คำสั่ง Do..While

- **Do..While**

โปรแกรมจะทำงานตามคำสั่งอย่างน้อย 1 รอบ เมื่อทำงานเสร็จจะมาตรวจสอบเงื่อนไขที่คำสั่ง while ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานที่คำสั่งใหม่อีกรอบ แต่ถ้าเป็นเท็จจะหลุดออกจากลูป

โครงสร้างคำสั่ง

```
do {
    คำสั่งต่างๆ เมื่อเงื่อนไขเป็นจริง;
 } while(เงื่อนไข);
```

ข้อแตกต่างและการใช้งาน Loop

- For ใช้ในกรณีรู้จำนวนรอบที่ชัดเจน
- While ใช้ในกรณีที่ไม่รู้จำนวนรอบ
- Do..while ใช้ในกรณีที่อยากให้ลองทำก่อน 1 รอบ แล้วทำซ้ำไปเรื่อยๆทราบเท่าที่เงื่อนไขเป็นจริง

ค่า null, undefined และ NaN

null คือตัวแปรที่ไม่มีค่าใดๆ เลย ไม่เท่ากับ 0 และไม่เท่ากับสตริงว่าง ไม่สามารถนำไปคำนวณใดๆ ได้ แต่หากนำไปเปรียบเทียบด้วยเงื่อนไขจะมีค่าเท่ากับค่า false

```
let a = null;

if(!a) {
      alert(“a is null”);
} else {
    alert(“a is not null”);
}
```

ค่า null, undefined และ NaN

undefined คือ ตัวแปรที่ประกาศเอาไว้แต่ไม่ได้กำหนดค่าใดๆ ให้กับมัน ยกตัวอย่าง เช่น

```
let a;

alert(a);
```

ค่า null, undefined และ NaN

NaN (มาจาก Not a Number) หมายถึงการนำตัวแปรที่ไม่ใช่ตัวเลขไปคำนวณทางคณิตศาสตร์

```
let a = 10;

let b = “x”;

alert(10-b);
```

ฟังก์ชั่น คืออะไร

ความหมายที่ 1:

ชุดคำสั่งที่นำมาเขียนรวมกันเป็นกลุ่มเพื่อให้เรียกใช้งานตามวัตถุประสงค์ที่ต้องการ และลดความซ้ำซ้อนของคำสั่งที่ใช้งานบ่อยๆ ฟังก์ชั่นสามารถนำไปใช้งานได้ทุกที่และแก้ไขได้ในภายหลัง ทำให้โค้ดในโปรแกรมมีระเบียบและใช้งานได้สะดวกมากยิ่งขึ้น

ความหมายที่ 2 :

โปรแกรมย่อยที่นำเข้ามาเป็นส่วนหนึ่งของโปรแกรมหลัก เพื่อให้สามารถเรียกใช้งานได้โดยไม่จำเป็นต้องเขียนโค้ดคำสั่งใหม่ทั้งหมด

รูปแบบของฟังก์ชั่น

1.ฟังก์ชั่นที่ไม่มีการรับและส่งค่า

```
function ชื่อฟังก์ชั่น(){
    // คำสั่งต่างๆ
}
```

การเรียกใช้งานฟังก์ชั่น

```
ชื่อฟังก์ชั่น ();
```

รูปแบบของฟังก์ชั่น

2.ฟังก์ชั่นที่มีการรับค่าเข้ามาทำงาน

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

// กลุ่มคำสั่งต่างๆ

}
```

อาร์กิวเมนต์ คือ ตัวแปรหรือค่าที่ต้องการส่งมาให้กับฟังก์ชัน (ตัวแปรส่ง)
พารามิเตอร์ คือ ตัวแปรที่ฟังก์ชันสร้างไว้สำหรับรับค่าที่จะส่งเข้ามาให้กับฟังก์ชัน (ตัวแปรรับ)

การเรียกใช้งานฟังก์ชั่น

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

รูปแบบของฟังก์ชั่น

3.ฟังก์ชั่นที่มีส่งค่าออกมา

```
function ชื่อฟังก์ชั่น(){
    return ค่าที่จะส่งออกไป
}
```

รูปแบบของฟังก์ชั่น

4.ฟังก์ชั่นที่มีการรับค่าเข้ามาและส่งค่าออกไป

```
function ชื่อฟังก์ชั่น(parameter1,parameter2,.....){
    retrun ค่าที่จะส่งออกไป
}
```

ฟังก์ชั่นแบบกำหนดค่าเริ่มต้น

```
function ชื่อฟังก์ชั่น (name=”kongruksiam”,parameter2,.....){
    // คำสั่งต่างๆ
}
```

ขอบเขตตัวแปร

- **local variable** ตัวแปรที่ทำงานอยู่ในฟังก์ชั่นมีขอบเขตการทำงานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของฟังก์ชั่น

- **global variable** ตัวแปรที่ทำงานอยู่นอกฟังก์ชั่นมีขอบเขตการทำงานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของไฟล์ที่ประกาศใช้

Array Properties & Function

หาจำนวนสมาชิกและเรียงลำดับ

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
let x = color.length;
let y = color.sort();
```

สมาชิกตัวแรกและตัวสุดท้าย

```
let first = color[0];
let last = color[color.length-1];
```

การเพิ่มสมาชิก

```
color.push(”สีเทา”);
```

เข้าถึงสมาชิกด้วย For Loop

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
let count = color.length;

for (let i = 0; i < count ; i++) {
        console.log(color[i]);
}
```

เข้าถึงสมาชิกด้วย ForEach

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
color.forEach(myData);


function myData(item) {
        console.log(item);
}
```

แปลง Array เป็น String

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

การรวม Array

```
let fruits = ["ส้ม", "องุ่น"];
let vegetables = ["คะน้า", "ผักชี", "ผักกาด"];
let hardware = ["เม้าส์","คีย์บอร์ด"];

let carts = fruits.concat(vegetables,computer);
```

เรียงลำดับใน Array

- let fruits = ["ส้ม", "องุ่น"];
- fruits.sort();
- fruits.reverse();

เรียงลำดับใน Array แบบตัวเลข (น้อยไปมาก)

```
let points = [20, 100, -100, 5, -25, 10];
points.sort(function(a, b){
    return a - b
});
```

a คือ ค่าตัวเลขที่มีค่าลบจะถูกเรียงก่อน

b คือ ค่าตัวเลขที่มีค่าบวกจะถูกเรียงทีหลัง

เรียงลำดับใน Array แบบตัวเลข (มากไปน้อย)

```
let points = [20, 100, -100, 5, -25, 10];
points.sort(function(a, b){
    return b - a
});
```

b คือ ค่าตัวเลขที่มีค่าบวกจะถูกเรียงก่อน

a คือ ค่าตัวเลขที่มีค่าลบจะถูกเรียงทีหลัง

JavaScript Object

let ชื่อวัตถุ = {*propertyName*:value}
ยกตัวอย่าง เช่น

```
let user = {
 name:"kong", age:20, email:"kong@gmail.com"
};
let product ={name:"มะม่วง",price:150,category:"ผลไม้"}
```

JavaScript Object

การเข้าถึงข้อมูล

objectName.propertyName

objectName[“propertyName”]

ยกตัวอย่าง เช่น

user.name

user[“name”]

JavaScript Object (Method)

```
let user = {
        name:"kong",
        age:20,
        email:"kong@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 {}

การยืนยันด้วย confirm( )

เป็นหน้าต่างที่ต้องการสอบถามการยืนยันจากผู้ใช้ ก่อนที่จะทำการใดๆ ต่อไป

confirm(“ข้อความ”);

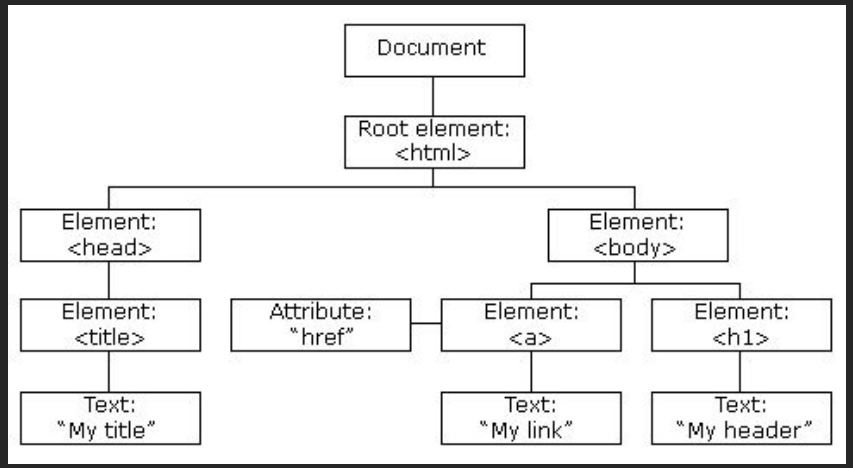
โดยผลลัพธ์จะมีค่าทางตรรกศาสตร์

- มีค่าเป็น true เมื่อผู้ใช้คลิก Ok
- มีค่าเป็น false เมื่อผู้ใช้คลิก cancel

HTML DOM (Document Object Model)

เมื่อหน้าเว็บโหลดเสร็จเรียบร้อย Web Browser มันจะสร้าง DOM ของหน้านั้นขึ้นมา โดยมอง HTML เป็นโครงสร้างต้นไม้ (ก้อน Object) หรือ**เรียกว่า DOM**

```
<html>
<head>
        <title>My title</title>
</head>
 <body>
<a href="#">My link</a>
<h1>My header</h1>
</body>
</html>
```



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

โดยผู้พัฒนาสามารถใช้อีเวนต์ที่เกิดขึ้นเป็นตัวกำหนดให้ตอบสนองหรือกระทำบางอย่างได้ เช่น การคลิกแล้วแจ้งเตือน เป็นต้น

ชื่อ Event	ความหมาย	ทำงานร่วมกับแท็ก
onfocus=“ ”	เมื่ออิลิเมนต์นั้นได้รับการโฟกัส	select, text, textarea
onblur=“ ”	เมื่ออิลิเมนต์นั้นสูญเสียการโฟกัส หรือถูกย้ายโฟกัสไปยังอิลิเมนต์อื่น	select, text, textarea
onchange=“ ”	เมื่อผู้ใช้เปลี่ยนแปลงค่าในฟอร์มรับข้อมูล	select, text, textarea
onselect=“ ”	เมื่อผู้ใช้เลือกข้อความ (ใช้เมาส์ลาก) ในช่องข้อความ	text, textarea
onsubmit=“ ”	เมื่อผู้ใช้คลิกปุ่ม submit	form

ชื่อ Event	ความหมาย	ทำงานร่วมกับแท็ก
onMouseover=“ ”	เกิดเมื่อออบเจกต์นั้นถูกเลื่อน mouse pointer ไปทับ	a,div
onMouseout=“ ”	เกิดเมื่อออบเจกต์นั้นถูกเลื่อน mouse pointer ที่ทับอยู่ออกไป	a,div
onclick=“ ”	เกิดเมื่อออบเจกต์นั้นถูกคลิก	a, button, checkbox, radio, reset, submit
onload=“ ”	เกิดเมื่อโหลดเอกสารเสร็จ	body
onunload=“ ”	เกิดเมื่อยกเลิกการโหลด เช่น คลิกปุ่ม Stop	body

EventListener

คือ เหตุการณ์หรือการกระทำบางอย่างที่เกิดขึ้นกับอิลิเมนต์ แต่รูปแบบการเขียนจะเขียนในฝั่ง javascript ทั้งหมด

โครงสร้าง :

```
element.addEventListener(event,callback)
```