ENGSE611 **การพัฒนาเว็บด้วยเทคโนโลยีสมัยใหม่** 3 (1-4-4) ****วิชาชีพเลือก**

Modern Web Technology Development

Lecture04

ภาษา CSS (Cascading Style Sheets)

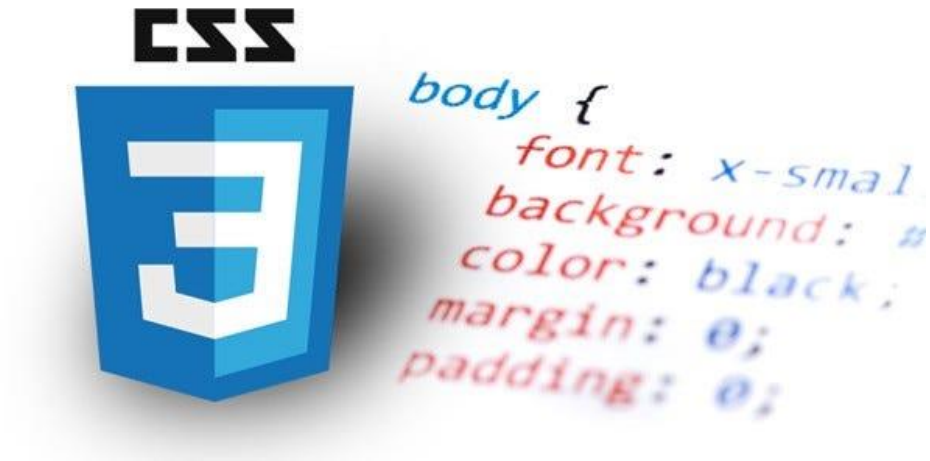
สอนโดย อ.ธนิต เกตุแก้ว

หลักสูตรวิศวกรรมซอฟต์แวร์ คณะวิศวกรรมศาสตร์

มหาวิทยาลัยเทคโนโลยีราชมงคลล้านนา เชียงใหม่

วัตถุประสงค์การเรียนรู้

- เพื่อให้ผู้เรียนมีความรู้ความเข้าใจเกี่ยวกับการจัดรูปแบบเว็บไซต์ด้วย CSS
- เพื่อให้ผู้เรียนมีความรู้ความเข้าใจเกี่ยวกับการพัฒนาเว็บไซต์ด้วย Responsive Web Design
- เพื่อให้ผู้เรียนสามารถประยุกต์ใช้งาน CSS Framework (W3CSS) กับการพัฒนาเว็บไซต์ได้อย่างเหมาะสม



https://www.w3schools.com/whatis/whatis_css.asp

หัวข้อการเรียนรู้

การประยุกต์ใช้ CSS เบื้องต้น

1. พื้นฐานการใช้ CSS ในการจัดรูปแบบเว็บไซต์
2. Selectors และ Specificity ใน CSS
3. การใช้ Flexbox จัดวาง Layout

กิจกรรม : การออกแบบหน้าเว็บด้วย Flexbox

งานที่ต้องทำ : สร้างหน้าเว็บที่ใช้ Flexbox สำหรับจัดวางเนื้อหา

Photo by Ferenc Almasi on Unsplash

สามารถเข้าไปเรียนรู้เพิ่มได้ที่ World Wide Web Consortium (W3C)

What is CSS?

Full CSS Tutorial

CSS คืออะไร ?

Cascading **S**tyle **S**heets (CSS) คือภาษาที่ใช้กำหนดรูปแบบการแสดงผล HTML

CSS ถูกสร้างขึ้นเพื่อใช้ร่วมกับ HTML

กล่าวคือ HTML ใช้สำหรับกำหนดโครงสร้างข้อมูล และ CSS ใช้สำหรับกำหนดรูปแบบการแสดงผล

ตัวอย่างเช่น

HTML:

```
<h1>Welcome to my web site</h1>
```

CSS:

```
h1 {
    color: #0000FF; /* Blue */
 }
```

Welcom to CSS.

Example1

Version ของ CSS

เวอร์ชันของ CSS มีดังต่อไปนี้

CSS 1 หรือ CSS Level 1 คือเวอร์ชันแรกที่ออกสู่สาธารณะและเป็น W3C Recommendation

ซึ่งออกในเดือนธันวาคมปีค.ศ. 1996 CSS 1 นั้นมีฟีเจอร์เพียงไม่มาก

CSS 2 ออกในเดือนพฤษภาคมปีค.ศ. 1998 โดยเพิ่มเติมฟีเจอร์ให้กับ CSS 1

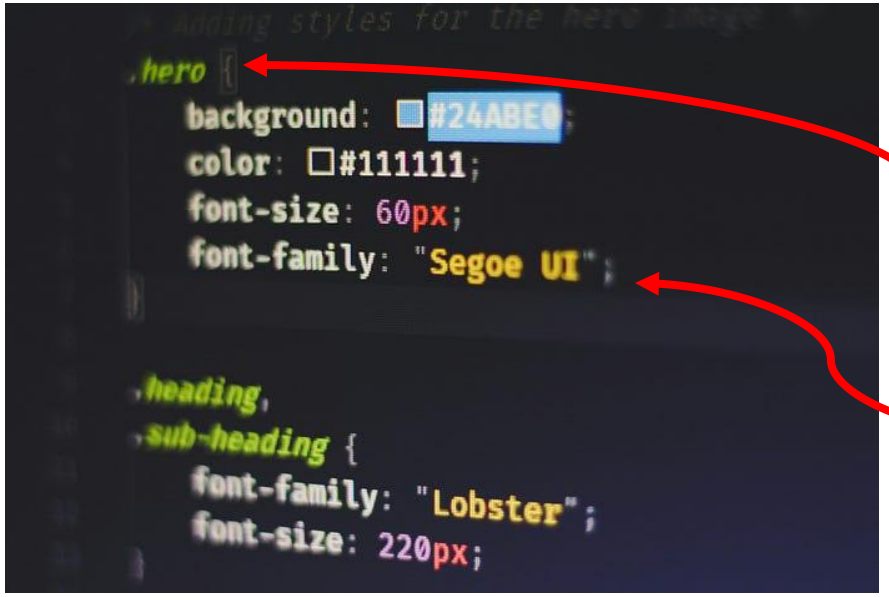
อย่างไรก็ตาม CSS 2 ไม่ประสบความสำเร็จมากนักในแง่ของการยอมรับและการรองรับจากเว็บเบราว์เซอร์ต่างๆ ทำให้ CSS 2.1

ต้องออกมาเพื่อแก้ไขปัญหาต่างๆในเวอร์ชัน 2 CSS 2.1 ได้เป็น Recommendation ในเดือนกรกฎาคมปีค.ศ. 2007

CSS 3 ยังอยู่ระหว่างการพัฒนา และการกำหนดสเปคใน CSS 3 นั้น จะถูกแบ่งออกเป็นส่วน (Module) ต่างๆ

ซึ่งบางส่วนก็เสร็จสมบูรณ์แล้ว บางส่วนก็กำลังพัฒนาอยู่ แต่บางเว็บเบราว์เซอร์ก็เริ่มรองรับสเปคในบางส่วนบ้างแล้ว

โครงสร้างของ CSS



มีส่วนประกอบอยู่ 2 ส่วน คือ

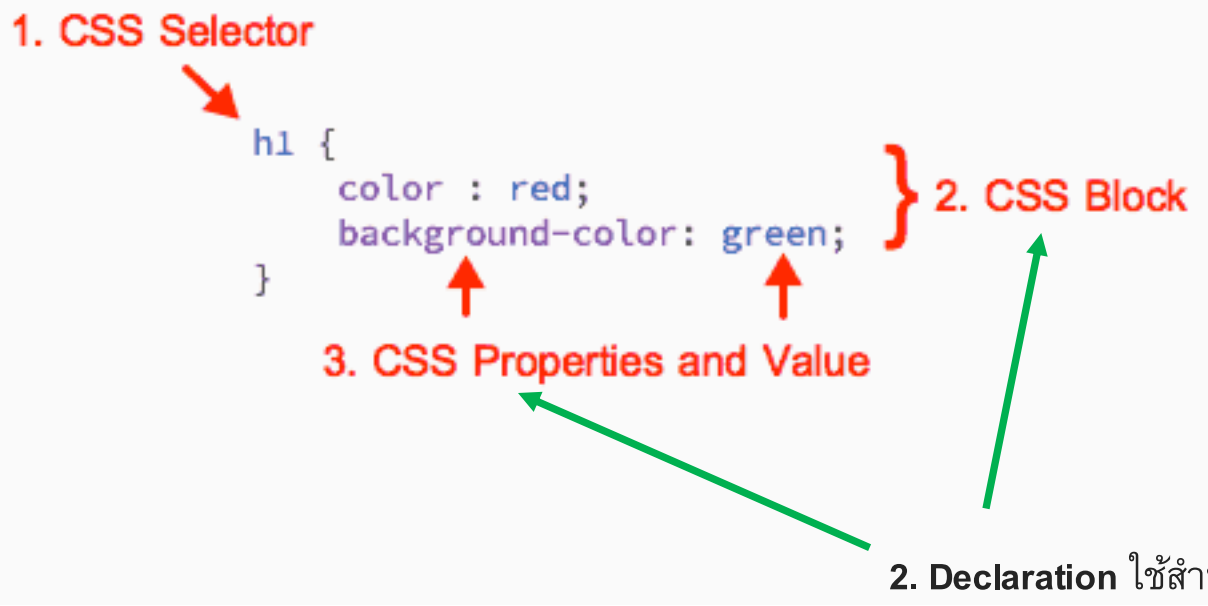
Selector และ Declaration

1. Selector คือแท็ก HTML, ID และ Class ที่ต้องการกำหนด

2. Declaration ใช้สำหรับกำหนดค่าให้กับ Selector

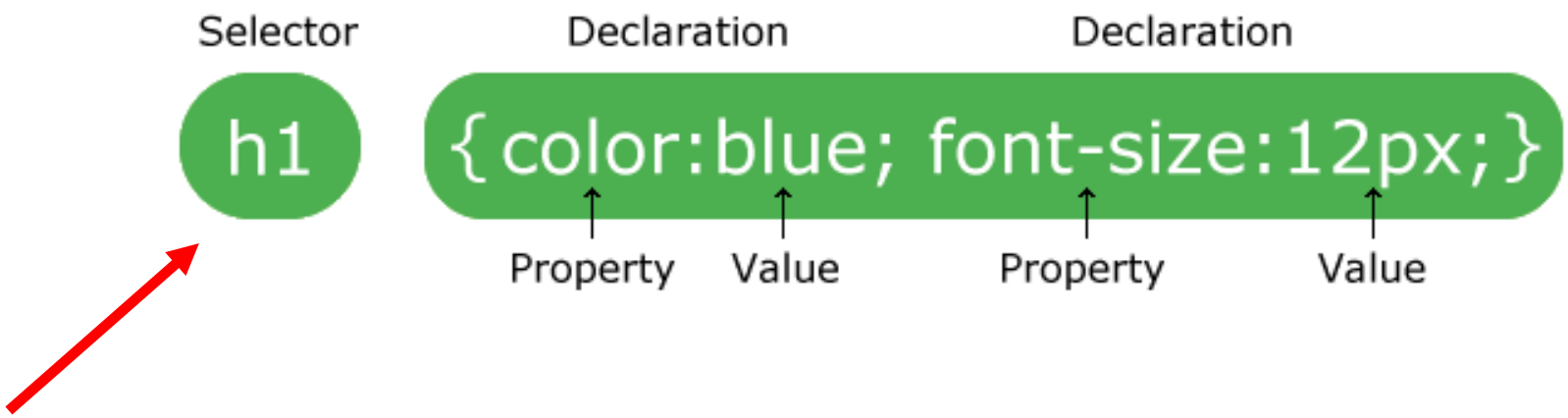
โครงสร้างของ CSS

1. Selector คือแท็ก HTML, ID และ Class ที่ต้องการกำหนด



2. Declaration ใช้สำหรับกำหนดค่าให้กับ Selector

ชนิดของ Selector



Selector คือแท็ก HTML, ID และ Class ที่ต้องการกำหนดคุณสมบัติ 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 *ไม่สามารถกำหนดค่าซ้ำกันได้*

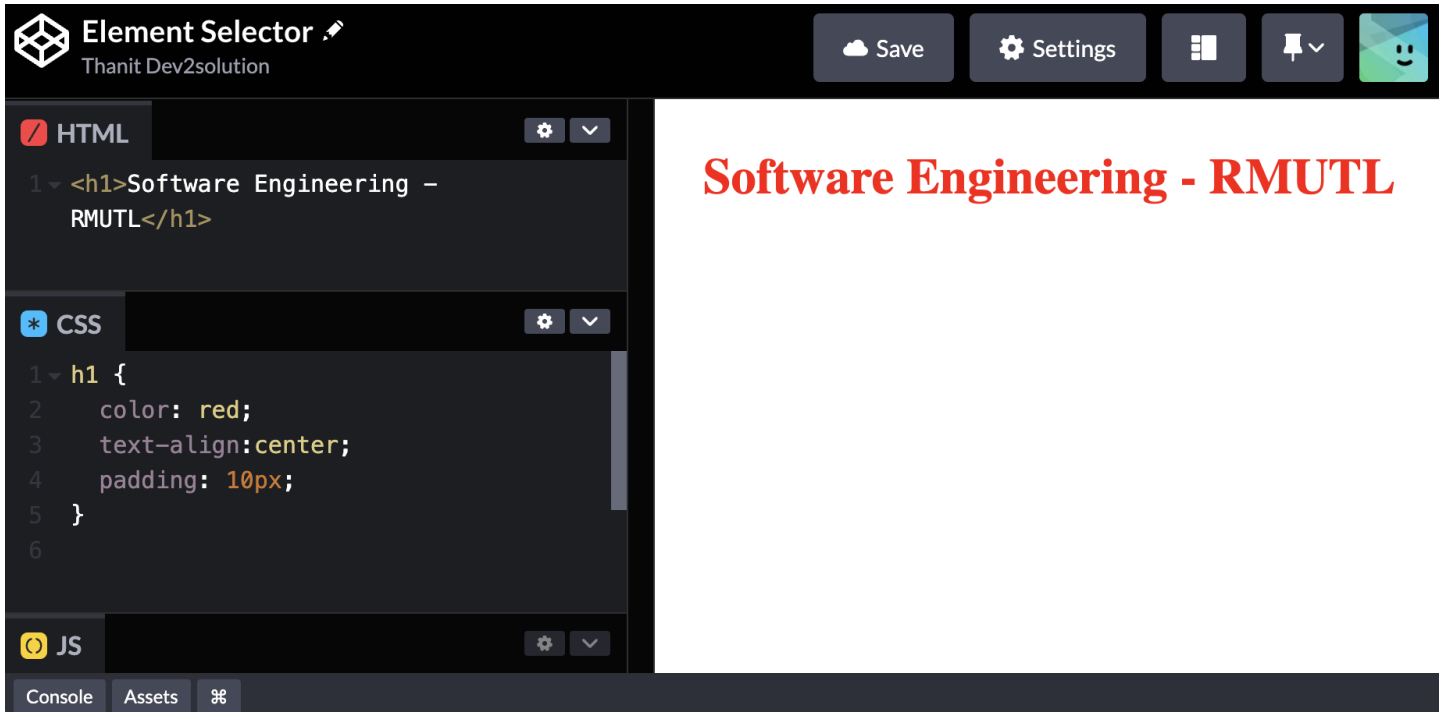
https://www.w3schools.com/whatis/whatis_css.asp

Element Selector

ตัวอย่าง

```
< h1 >Software Engineering - RMUTL</ h1>
```

```
h1 {
    color: red;
    text-align:center;
    padding: 10px;
}
```

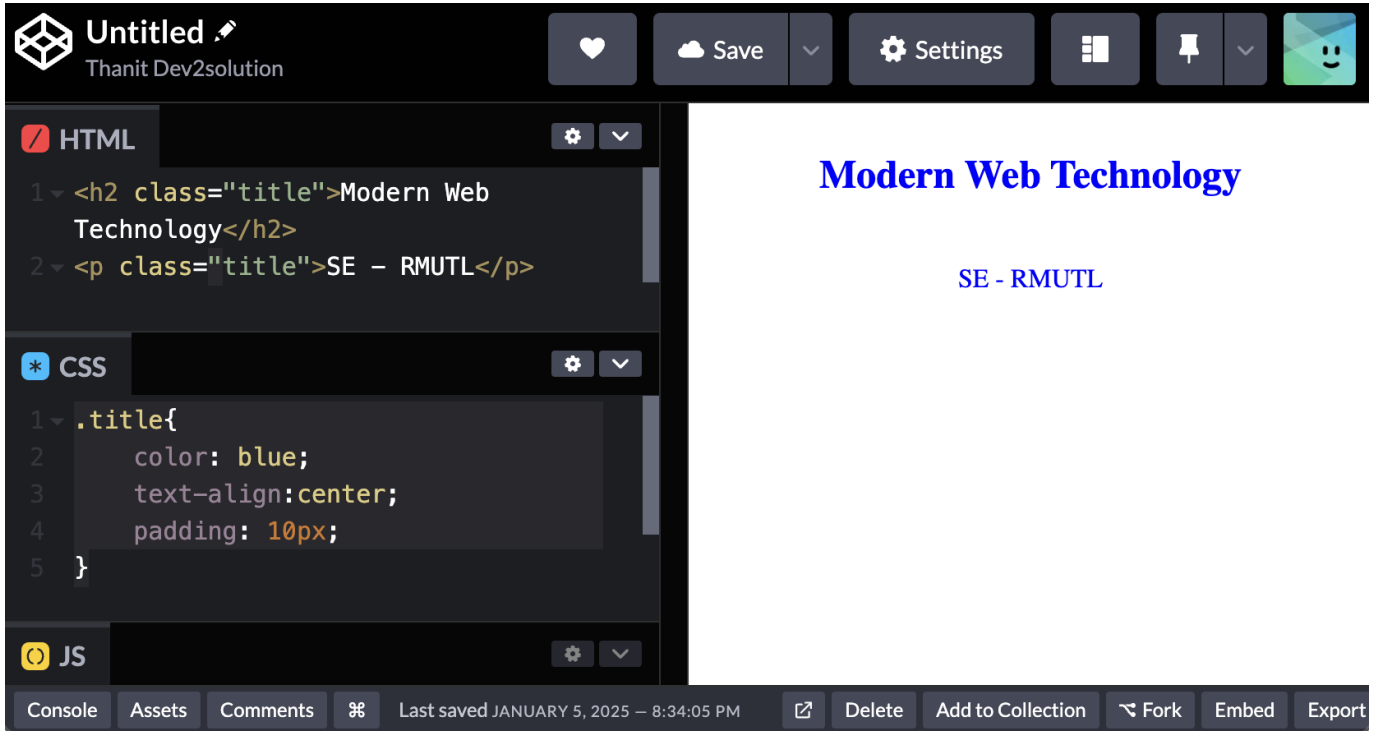


Class Selector

ตัวอย่าง

```
<h2 class="title"> Modern Web Technology</h2>
<p class="title"> SE - RMUTL </p>
```

```
.title{
    color: blue;
    text-align:center;
    padding: 10px;
}
```



ID Selector

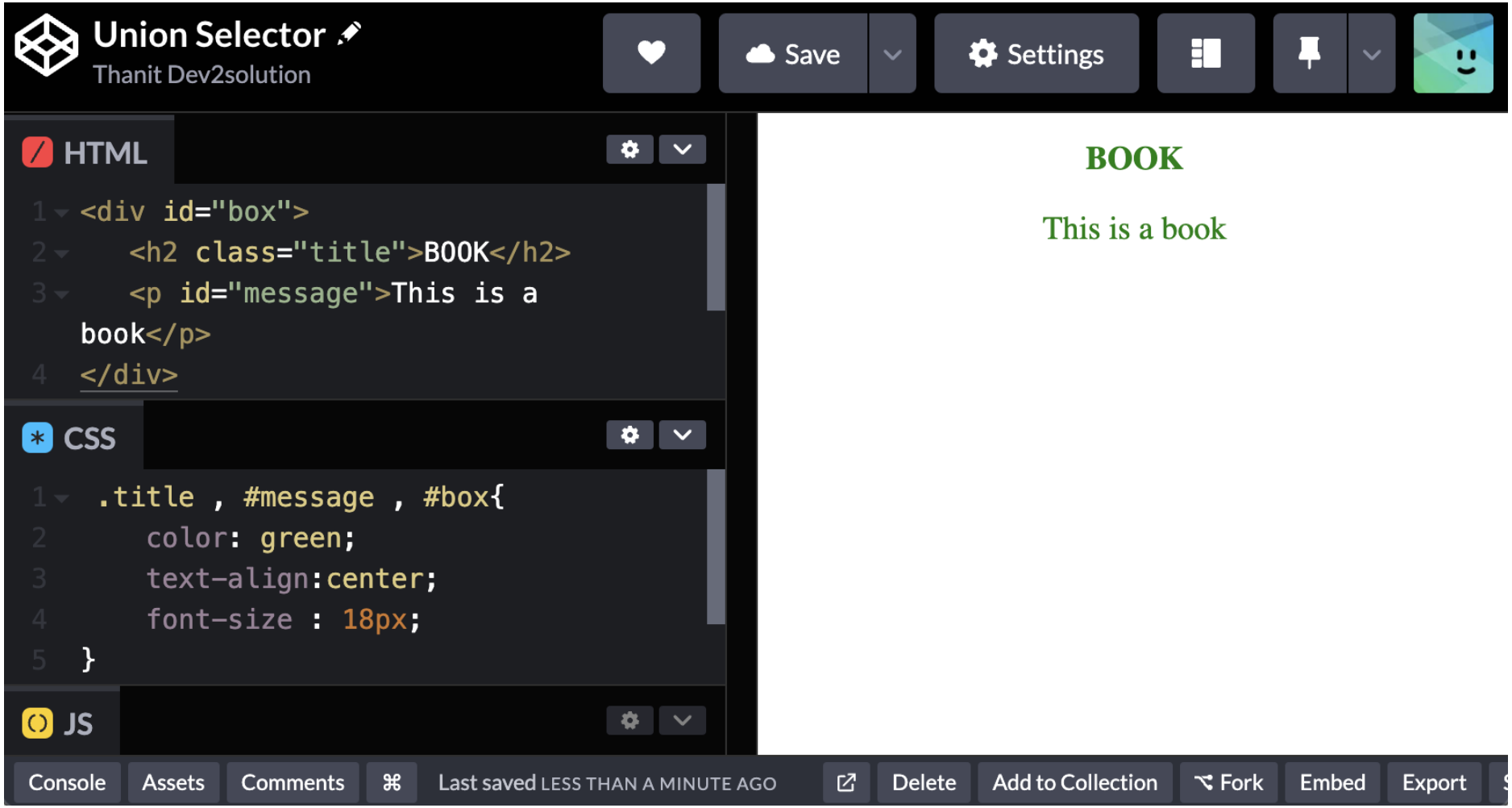
```
<h2 class="title"> SE-RMUTL</h2>
<p class="message"> Welcom to CSS</p>
```

```
#message{
    color: green;
    text-align:center;
    padding: 10px;
}
```

ตัวอย่าง

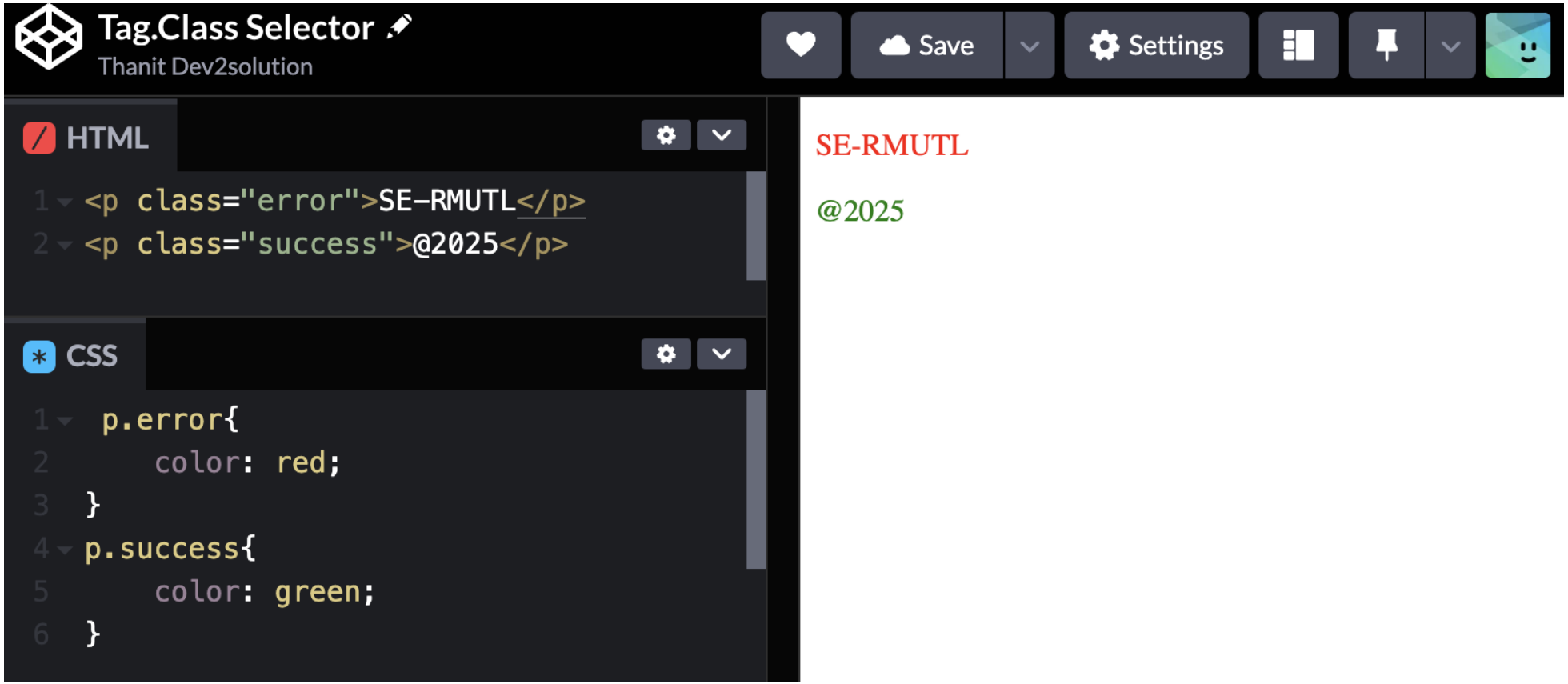
Union Selector

เราอาจนำทั้งสามแบบ มารวมกัน ได้ ดังตัวอย่าง



Tag.Class Selector

กรณี Tag เหมือนกัน แต่เราสามารถแยก class ของ CSS ได้ ดังตัวอย่าง

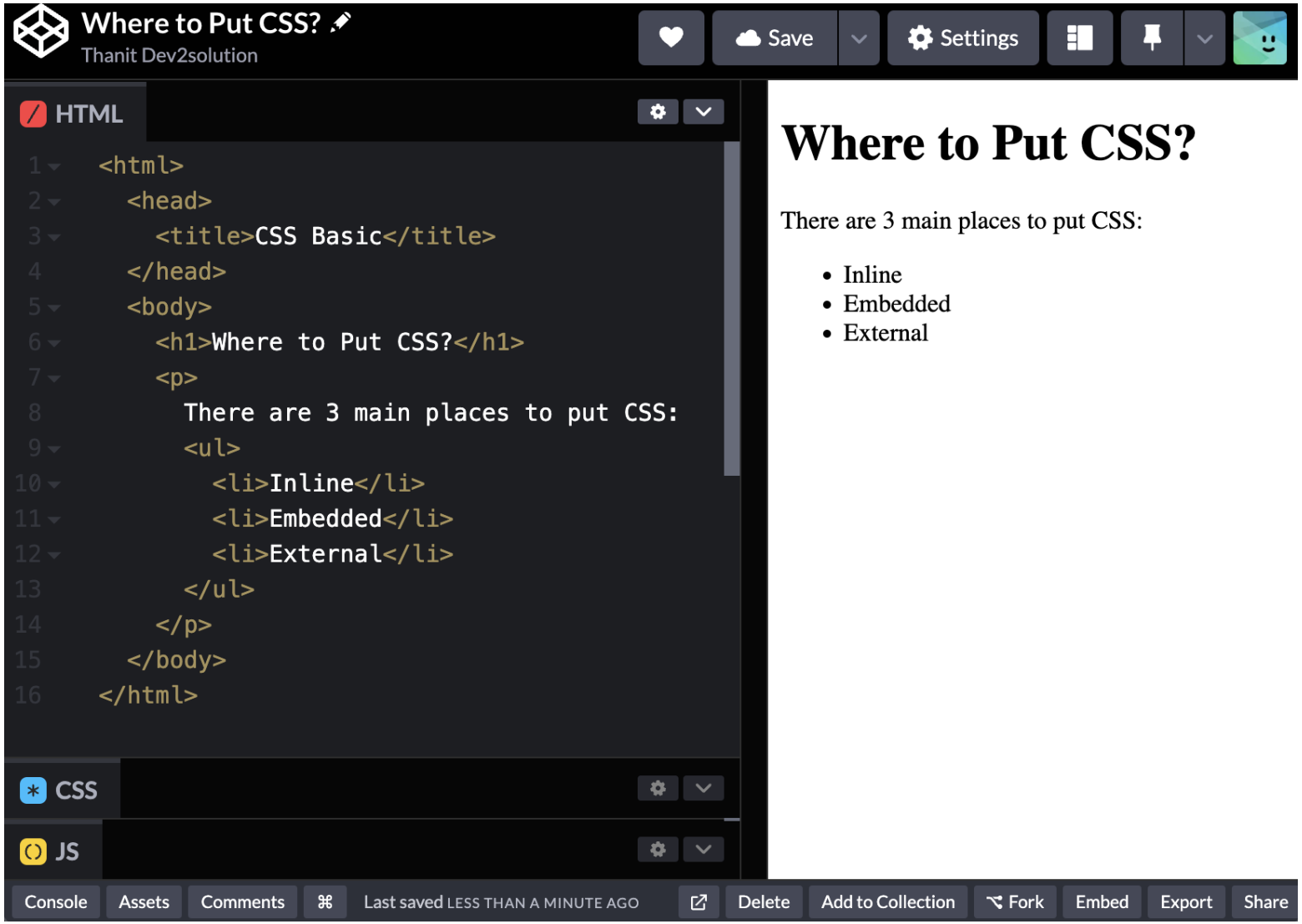


รูปแบบการประกาศโค้ด CSS ที่ใดได้บ้างใน Website

เราสามารถเขียนโค้ด CSS ได้ 3 ที่

หรือ 3 แบบ. หลักๆดังนี้

1. Inline
2. Embedded
3. External



Inline CSS

หรืออาจจะเรียกว่า **Internal**

เป็นการแทรก attribute style ลงในแท็ก HTML

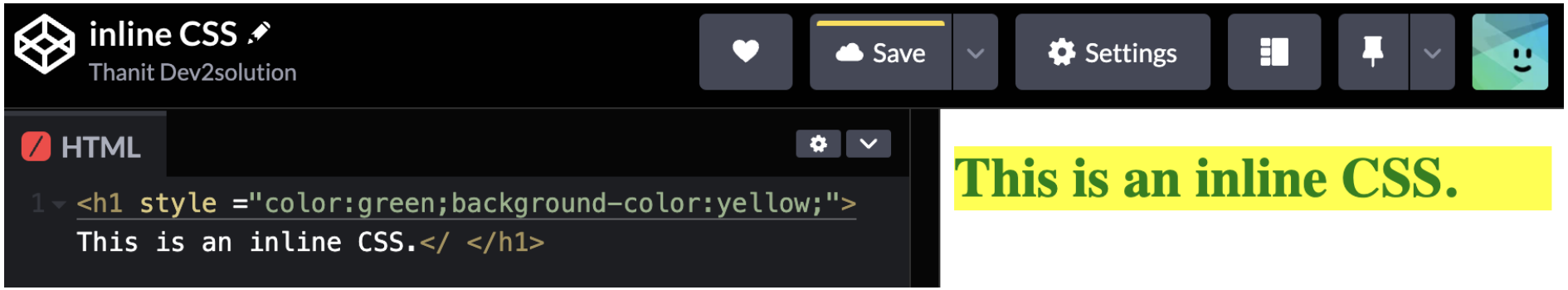
(ใช้กำหนด style ให้กับ Element ของ HTML เพียงอันเดียวเป็นการเฉพาะ)

โดยมี Syntax ดังนี้

<tag **style**="property:value; property:value; . . .">

```
<h1 style ="color:green;background-color:yellow;"> This is an inline CSS.</ </h1>
```

Example



Embedded CSS

แบบที่ 2 แบบ Embedded เป็นการเขียนโค้ด CSS ลงไปในไฟล์ HTML เช่นกัน

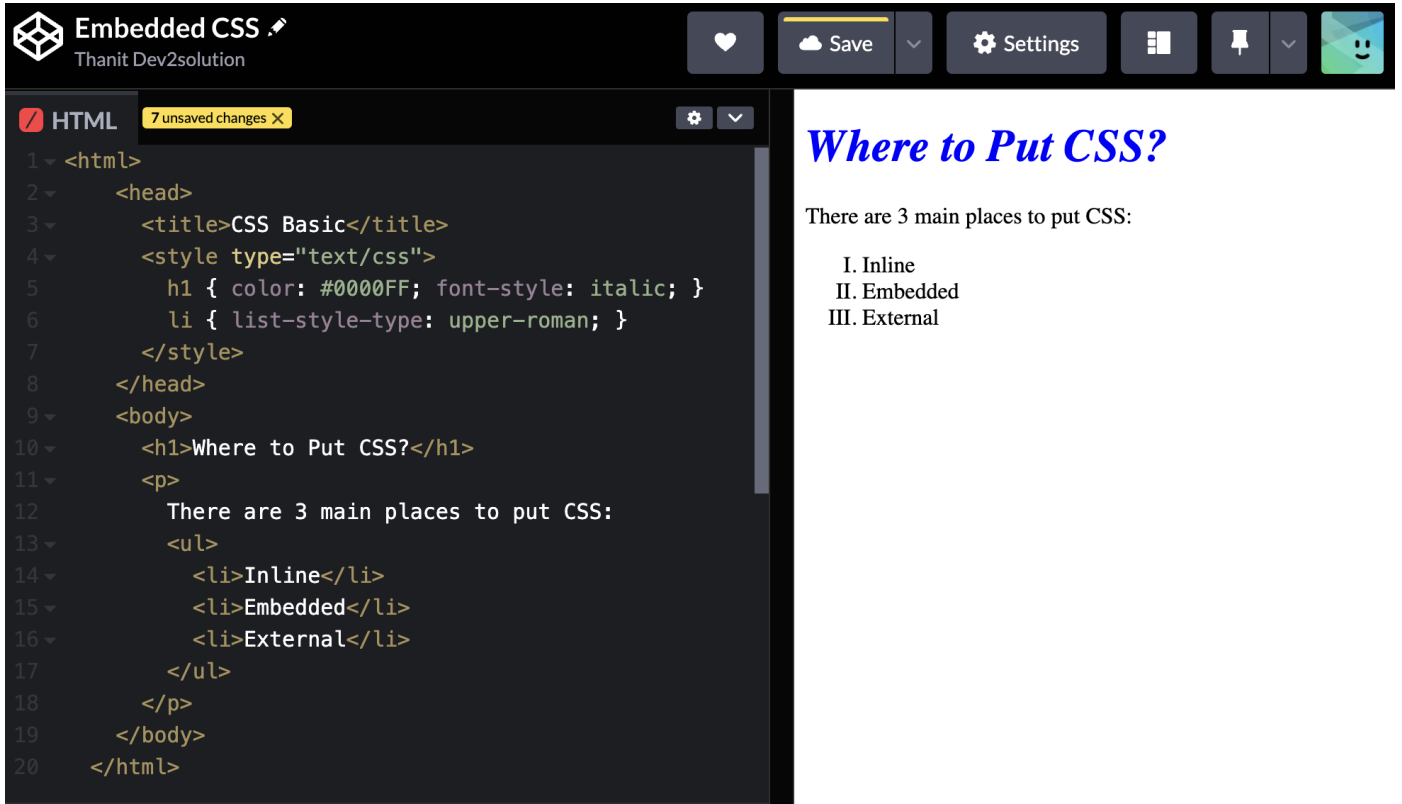
แต่จะรวมโค้ด CSS ทั้งหมดไว้ภายใต้แท็ก <head> โดยมี Syntax ดังนี้

```
<head>
      <title></title>
      <style type="text/css">
            selector { property:value; property:value; . . . }
      </ style >
</head>
```

Embedded CSS

```
<head>
    <title></title>
    <style type="text/css">
        h1 { color: #0000FF; font-style: italic; }  li { list-style-type: upper-roman; }
    </style>
</head>
```

Example



External CSS

แบบสุดท้ายแบบ External เป็นการเขียนแบบแยกโค้ด CSS ออกมายังไฟล์แยกต่างหาก เป็น External File โดยมี Syntax ดังนี้

HTML: index.html

```
<head>
    <title></title>
    <link rel="stylesheet" type="text/css" href="file_name.css" />
</head>
```

CSS: file_name.css

```
selector {
    property: value;
    property: value;
    . . .
}
```

External CSS

Example

External CSS
Thanit Dev2solution

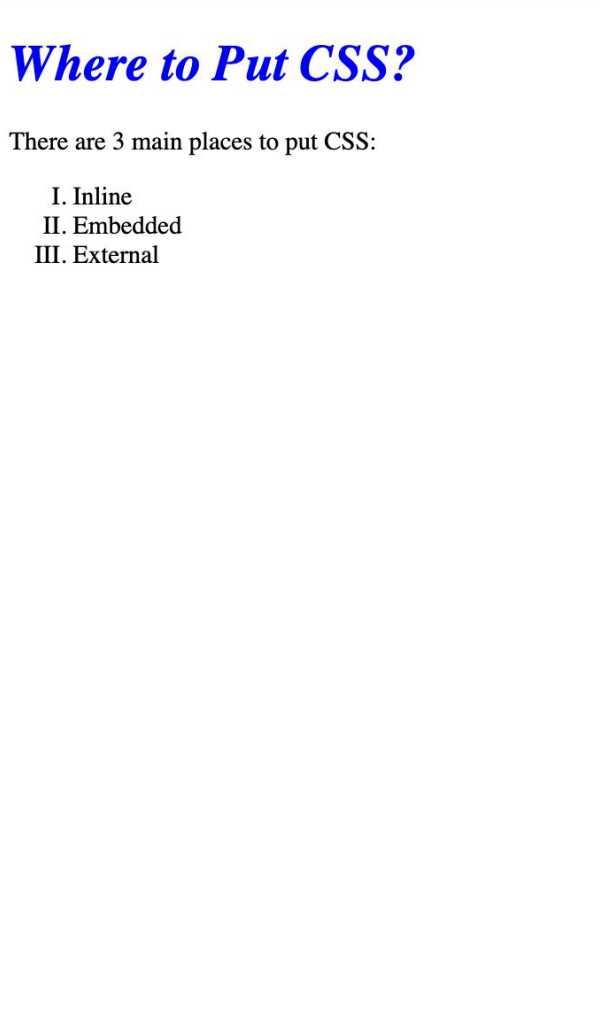
Save | Settings

HTML

```
<html>
    <head>
      <title>CSS Basic</title>
          <link rel="stylesheet" type="text/css" href="External.css" />
    </head>
    <body>
      <h1>Where to Put CSS?</h1>
      <p>
        There are 3 main places to put CSS:
        <ul>
          <li>Inline</li>
          <li>Embedded</li>
          <li>External</li>
        </ul>
      </p>
    </body>
  </html>
```

CSS

```
h1 {
  color: #0000FF;
  font-style: italic;
}
li {
  list-style-type: upper-roman;
}
```



External CSS

- ไฟล์ HTML ยังสามารถลิงค์ไปยังไฟล์ CSS ได้มากกว่าหนึ่งไฟล์ด้วย

- เราสามารถแยกเป็นไฟล์ CSS ตามแต่ละจุดประสงค์ได้ เช่น menu.css สำหรับจัดการกับเมนู, layout.css จัดการกับเลย์เอาท์ เป็นต้น

- การเขียนแบบ External เป็นรูปแบบที่นิยมที่สุดเพราะสามารถนำกลับมาใช้ใหม่ได้ ถ้าทุกหน้ามีสไตล์ที่ตรงกัน ก็ให้ลิงค์ไปที่ไฟล์ CSS เดียวกัน ดังนั้นหากต้องการเปลี่ยนแปลงรูปลักษณ์ใหม่ ก็เปลี่ยนที่ไฟล์ได้โดยตรง ก็จะมีผลในไฟล์ html ทั้งหมดด้วย

สรุปการประกาศโค้ด CSS ทั้ง 3 แบบ

- ✓ CSS ก็คือ มันถูกเรียกว่า **Cascading** Stylesheet โดยมีความสามารถในการ**เรียงลำดับความสำคัญ** ของ Style ได้

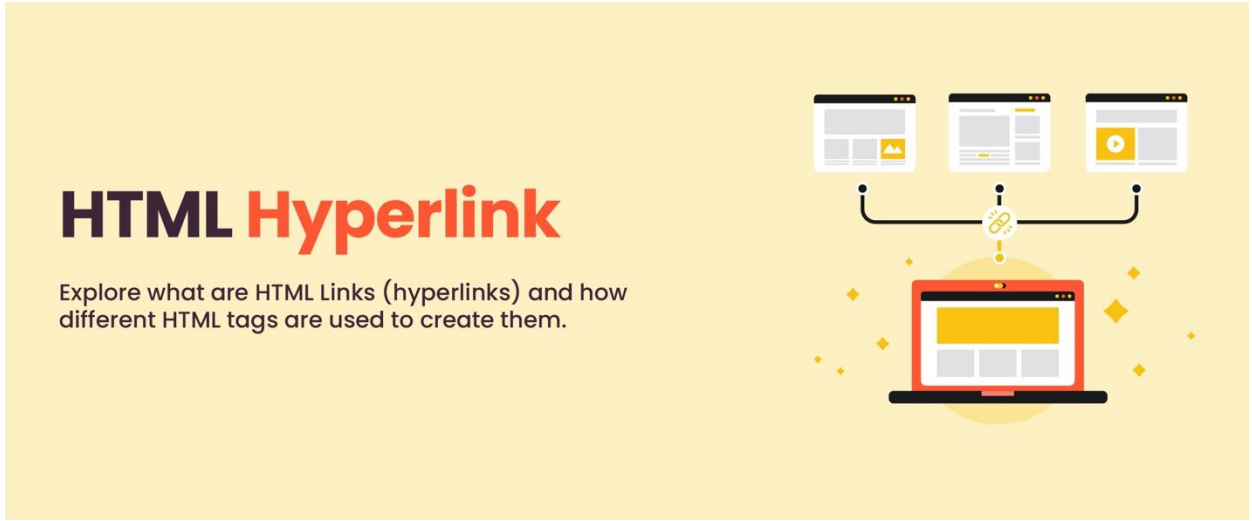
- ✓ Stylesheet แบบ Link เหมือนกัน จะเรียงความสำคัญจากล่างขึ้นบน ตัวมาทีหลังจะทับตัวหน้าหมด

- ✓ ถ้า เป็น Stylesheet ที่ใส่ต่างวิธีกันจะมีความสำคัญดังนี้

    1. Inline (สำคัญสูงสุด)
    2. Embedded
    3. External

- ✓ สามารบังคับให้ สิ่งที่เราเขียนมีลำดับความสำคัญสูงสุดด้วยโดยการใช้คำสั่ง **!important** แต่ไม่นิยมใช้ เพราะการใช้ **!important** หลายๆที่มันควบคุมยาก แต่จะใช้เพื่อลองตอน debug เท่านั้น

การเชื่อมโยงเว็บเพจ (HTML Hyper Link) และ CSS Links

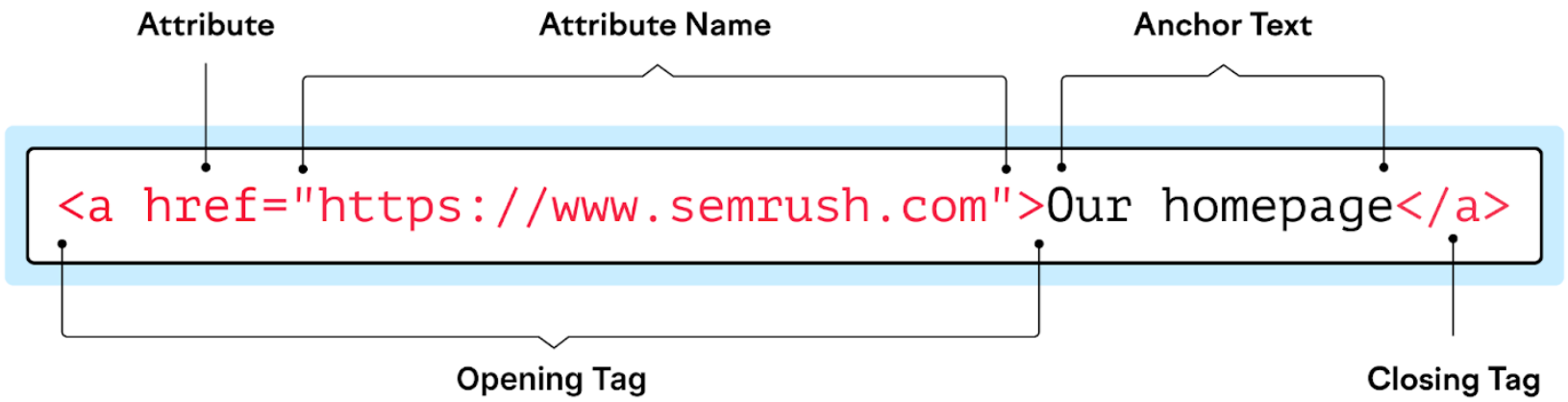


- **Clickable**
- **Text or Image**
- **Destination**
- **URL**
- **Link Types**
- **Functionality**
- **Accessibility**

CSS Links at w3schools
Try me..

https://bignet.in/blog/2631/features-of-hyperlink-in-english

การกำหนด HTML Link



semrush.com

SEMRUSH

ศึกษาเพิ่มเติมได้ที่

- https://www.w3schools.com/html/html_links.asp
- https://code-th.com/html/lesson/links
- https://enfete.co.th/th/html-tags-for-image-and-hyperlink/

การกำหนด **style** ให้กับ **Link**

เกี่ยวการคลิกแล้วลิงค์เชื่อมไปอีก หน้าหนึ่ง คลิกเข้าไปในเว็บไซต์ เช่นการคลิกผ่านทางข้อความ รูปภาพ ไอคอน ปุ่ม แล้วแต่เราจะกำหนด

```
<div class="container">
<div class="m-3">
<a href="https://www.google.com" target="_blank">This LInk</a>
</div>
<div class="m-3">
<a href="https://www.google.com" target="_blank">This LInk</a>
</div>
</div>
```

ศึกษาเพิ่มเติมได้ที่

- https://www.w3schools.com/css/css_link.asp
- CSS Links เชื่อมโยง

Styling Links ออกแบบการลิงค์

```
a {
    color: rgb(243, 255, 80);
}
```

```
a:link {
color: red; //เชื่อมโยง
}
a:visited {
color: yellow; //มาเยือน
}
a:hover {
color: hotpink; //เลื่อนเมาส์ใกล้
}
a:active {
color: blue; //การใช้งาน
}
```

Style Properties ทั่วไป

- หน่วย (Unit)
- ฟอนต์ (Font)
- สี (Colors)
- ภาพพื้นหลัง (background)
- ลักษณะข้อความ (Text)
- เส้นขอบ (Border)

หน่วย (Unit) ใน CSS แบบตายตัว (Absolute)

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

หน่วย (Unit) ใน CSS แบบอัตราส่วน (Relative)

- **%** เป็นการกำหนดขนาดเป็นเปอร์เซ็นต์ มักใช้กับความกว้างหรือสูง
- **em** อ้างอิงขนาดกับ element parent ที่ใกล้ที่สุด ใช้ในการกำหนดขนาดจำนวนเท่าของขนาดปัจจุบัน
- **rem** กำหนดขนาดโดยอ้างอิงกับ root element ปกติ font-size จะอยู่ที่ 16px
- **vw** โดย 1% หรือ 1/100 ของ viewport width (width ของ browser viewport เท่ากับ 750px ค่า 1vw = 7.5px)
- **vh** โดย 1% หรือ 1/100 ของ viewport height (height ของ browser viewport เท่ากับ 900px ค่า 1vh = 9px)
- **vmin , vmax** กำหนดค่าต่ำสุด และค่าสูงสุดของ viewport

การกำหนดชนิดฟอนต์ (Font)

font-family กำหนดชนิดของฟอนต์ในหน้าเว็บกำหนดได้ทั้งแบบฟอนต์เดียวหรือหลายฟอนต์ โดยกำหนดได้ดังนี้ คือ

```
font-family : ฟอนต์แบบที่ 1;
font-family : ฟอนต์แบบที่ 1 , ฟอนต์แบบที่ 2, ชนิดฟอนต์;
```

font-size กำหนดขนาดข้อความหรือตัวอักษรในหน้าเว็บโดยกำหนด 2 รูปแบบ คือ

```
font-size : value
```

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

font-weight กำหนดความหนาของข้อความแบ่งออกเป็น 2 รูปแบบ

```
font-weight : value
```

- แบบค่าคงที่ : lighter , bold (ตัวหนา) , bolder (ตัวหนากว่า)
- แบบตัวเลข : 100, 200, …., 800, 900

การกำหนดสี (Colors)

color : value

- **ชื่อสี** : green , red , yellow , pink
- **RGB** : *rgb(red, green, blue)* ,
  rgb(30%, 20%, 45%)
- **Hexadecimal** : #000 #FFF
- **HSL** : (*hue*, *saturation*, *lightness*)

Try them..

HTML Color Groups

HTML Color Picker

Color names list

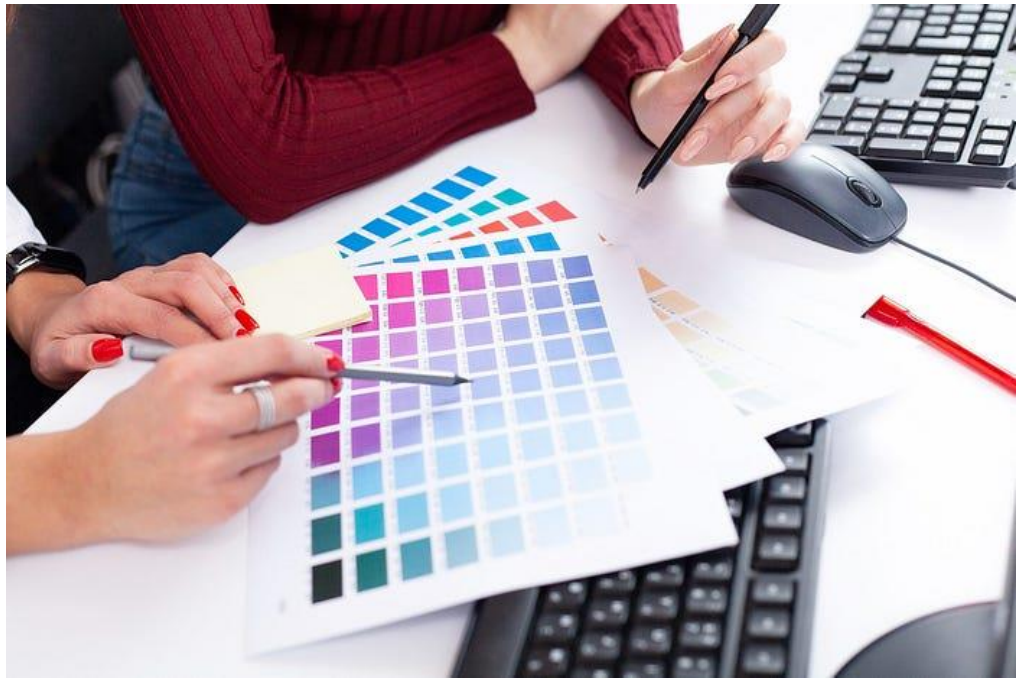
Color Name	Color Code	Color Name	Color Code
Red	#FF0000	White	#FFFFFF
Cyan	#00FFFF	Silver	#C0C0C0
Blue	#0000FF	Gray or Grey	#808080
DarkBlue	#00008B	Black	#000000
LightBlue	#ADD8E6	Orange	#FFA500
Purple	#800080	Brown	#A52A2A
Yellow	#FFFF00	Maroon	#800000
Lime	#00FF00	Green	#008000
Magenta	#FF00FF	Olive	#808000
Pink	#FFC0CB	Aquamarine	#7FFFD4

HTML color codes and names

การกำหนดภาพพื้นหลัง (background)

background-image ใช้เปลี่ยนภาพพื้นหลัง

```
background-image : url(รูปภาพ);
background-repeat : value;
```



ค่าที่กำหนดใน background-repeat

- repeat (ค่า default ทำให้ภาพเต็มพื้นที่ของอิลิเมนต์)
- repeat-x (จัดเรียงเฉพาะในแนวแกน X)
- repeat-y (จัดเรียงเฉพาะในแนวแกน Y)
- no-repeat (ไม่จัดเรียงแนวแกน x,y แสดงภาพพื้นหลังภาพเดียว)

การกำหนดลักษณะข้อความ

การกำหนดลักษณะข้อความ

```
text-decoration : value
```

- none : (ค่าว่าง)
- underline : (ขีดเส้นใต้)
- overline : (ขีดเส้นเหนือข้อความ)
- line-through : (ขีดเส้นทับข้อความ)

การจัดแนวข้อความ

```
text-align : value
```

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

กำหนดความกว้างและความสูง

width ใช้กำหนดความกว้างของวัตถุที่ต้องการ

height ใช้กำหนดความสูงของวัตถุที่ต้องการ

```
width : value
height : value
```

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

ความกว้างและความสูงด้วย max , min

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

การกำหนดลักษณะของเส้นขอบ (Border)

Border คือ การกำหนดเส้นขอบ โดยมีรูปแบบดังนี้

```
border : <รูปแบบเส้น> <ขนาด> <สี>
```

- **รูปแบบเส้น** ประกอบด้วย solid (เส้นทึบ) dotted (แบบจุด) dashed (เส้นปะ)
- **ขนาด คือ** ความกว้างของเส้นกำหนดเป็นตัวเลขตามหน่วย
- **สี** กำหนดรูปแบบเหมือน font เลย เช่น ชื่อสี , rgb , hex

การกำหนดขนาดเส้นขอบ (width)

```
border-width : <ขนาด>
```

ระบุขนาดแบบค่าคงที่ :

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

การกำหนดความโค้งเส้นขอบ (radius)

```
border-radius : <ค่าความโค้ง>
```

CSS สำหรับจัดหน้าเว็บ (Page Layout)

- **Float และ Clear**
- **CSS Align**

การกำหนดตำแหน่งด้วย Float และ Clear

float และ **clear** เป็น property ที่สำคัญอันหนึ่ง ที่มีไว้เพื่อการจัดตำแหน่งของ layout ในแบบ tableless property

float กำหนดให้อิลิเมนต์สามารถลอยอยู่ด้านใดด้านหนึ่ง โดยค่าที่กำหนดได้ คือ

```
float : value
```

- left, right (ให้ลอยอยู่ทางด้านซ้ายและขวาตามลำดับ)
- inherit ลอยตาม parent element
- none ไม่ให้มีการลอย (default)

clear ไม่อนุญาตให้มีการลอยของอิลิเมนต์โดยค่าที่กำหนดได้คือ

```
clear : value
```

- left, right (ไม่อนุญาตให้มีลอยทางด้านซ้ายและขวาตามลำดับ)
- both (ไม่อนุญาตให้มีการลอยทั้งสองด้าน)

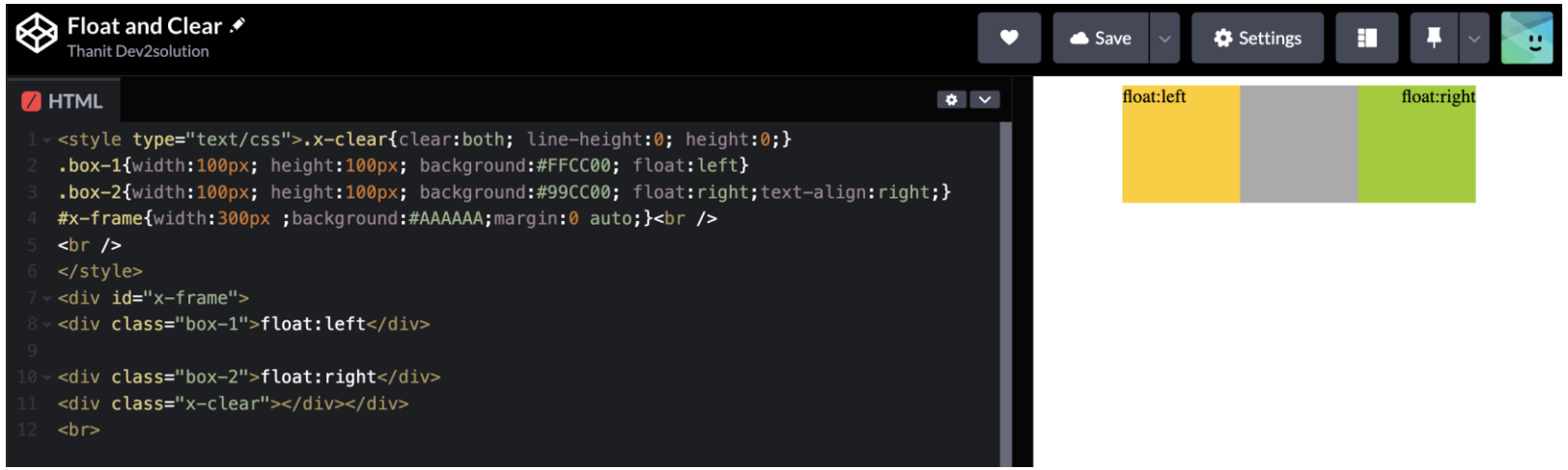
ศึกษาเพิ่มเติมได้ที่

- https://www.w3schools.com/css/css_float.asp
- https://smiletonz.wordpress.com/2013/04/13/css-float-clear/

การกำหนดตำแหน่งโดย CSS ด้วย Float และ Clear

```
<style type="text/css">
      .clear{clear:both; line-height:0; height:0; font-size: 1px}
     .box-1{width:100px; height:100px; background:#FFCC00; float:left}
     .box-2{width:100px; height:100px; background:#99CC00;
                float:right;text-align:right;}
    #frame{width:300px ;background:#AAAAAA}
</style>
<div id="frame">
       <div class="box-1">float:left</div>
       <div class="box-2">float:right</div>
       <br class="clear" />
</div>
```

Example1



การกำหนดตำแหน่งด้วย Float และ Clear

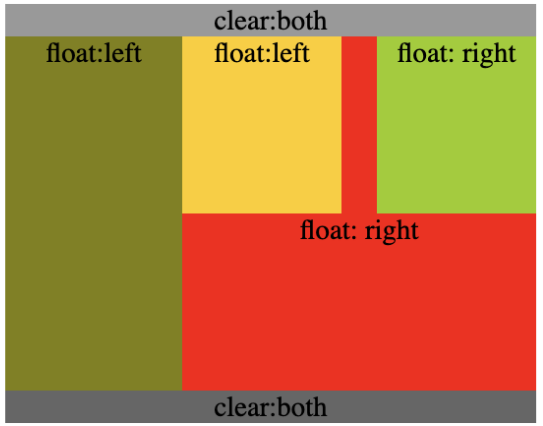
Example2

Float and Clear2
Thanit Dev2solution

Save | Settings

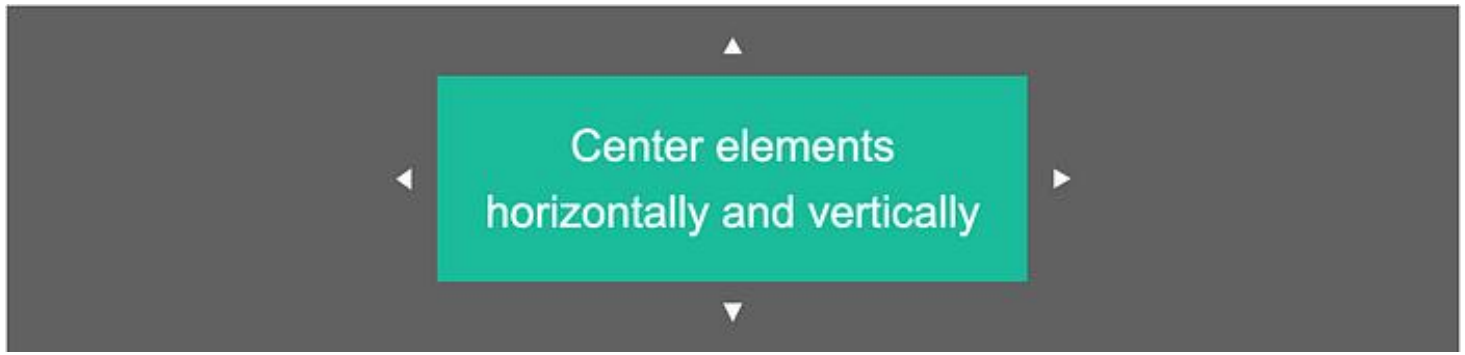
HTML

```
<style type="text/css">
.clear{clear:both; line-height:0; height:0; font-size: 1px}
#menu{float: left; width: 100px; height: 200px; background: olive}
#content{float: right; width: 200px}
.box-1{width:90px; height:100px; background:#FFCC00; float:left}
.box-2{width:90px; height:100px; background:#99CC00; float:right;}
#header{background: #999999}
#footer{background: #666666}
#frame{background: red}
#header,#frame,#footer{clear: both; margin: 0 auto; width: 300px; text-align: center}
</style>
<div id="header">clear:both</div>
<div id="frame">
        <div id="menu">float:left</div>
        <div id="content">
           <div class="box-1">float:left</div>
           <div class="box-2">float: right</div>
           float: right
        </div>
        <div class="clear"></div>
</div>
<div id="footer">clear:both</div>
```



CSS Align

CSS Layout — Horizontal & Vertical Align



ศึกษาเพิ่มเติมได้ที่

- https://tawantawan1997.medium.com/css-align-d502b31d69c2
- https://www.geeksforgeeks.org/css-align/

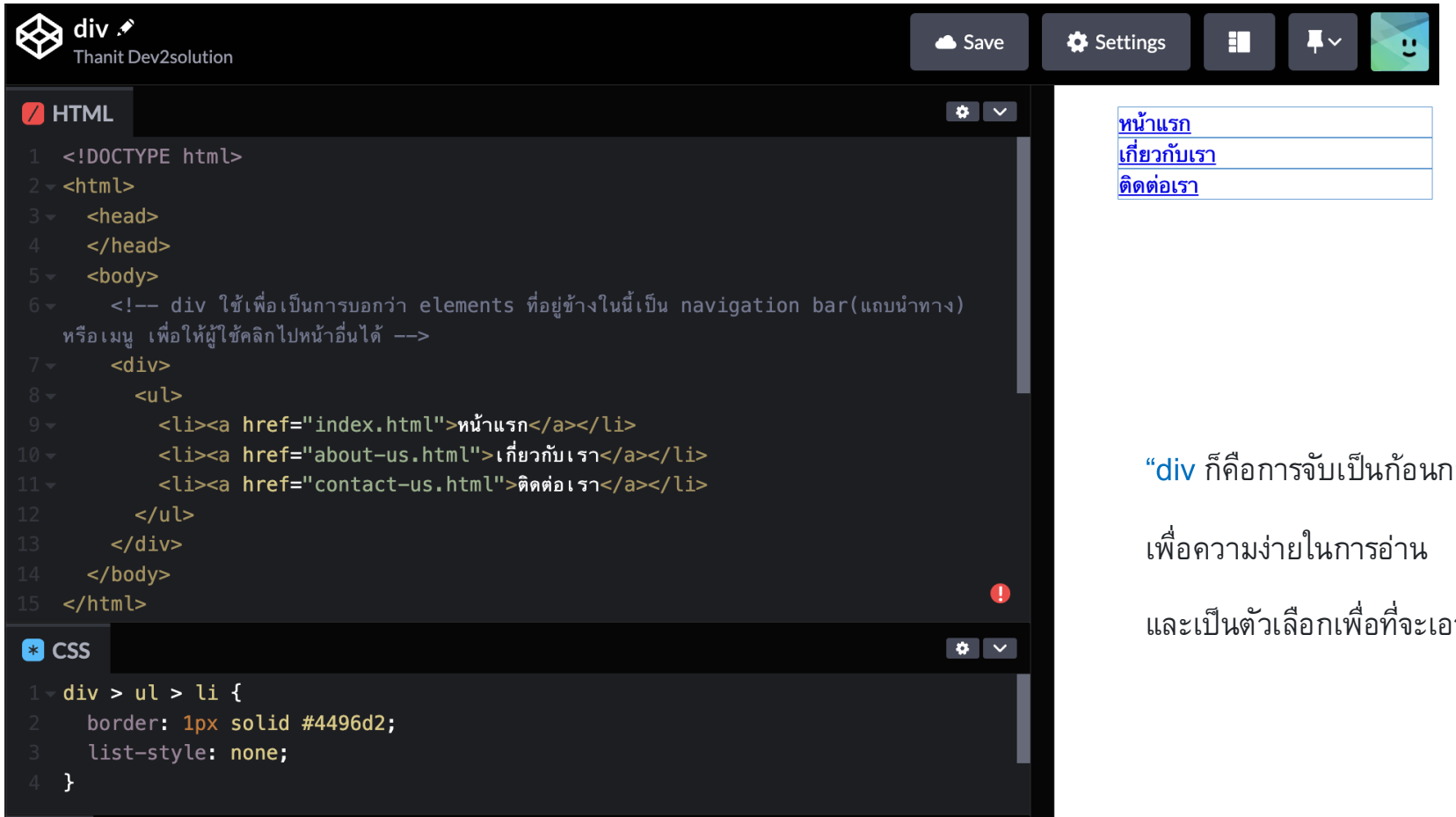
การจัด HTML ให้เป็นระเบียบ

จัดระเบียบด้วย div element

div ย่อมาจากคำว่า division(แผนกหรือกอง) ซึ่งเอาไว้ใช้ในการเป็นภาชนะสำหรับ HTML elements อื่นๆ อีกที

เพื่อความง่ายในการใช้เป็นตัวเลือกเพื่อที่จะตกแต่งด้วย CSS หรือจะทำให้มันมีการเคลื่อนไหวด้วย JavaScript

Example



“div ก็คือการจับเป็นก้อนกลุ่ม

เพื่อความง่ายในการอ่าน

และเป็นตัวเลือกเพื่อที่จะเอาไปทำไรต่อ”

การจัด HTML ให้เป็นระเบียบ

จัดระเบียบ HTML ด้วย ID

ถ้าโครงสร้าง **div** เป็นพ่อแม่และ **ul** เป็นลูก

เหมือนกันจะทำยังไงให้มันแตกต่างกัน

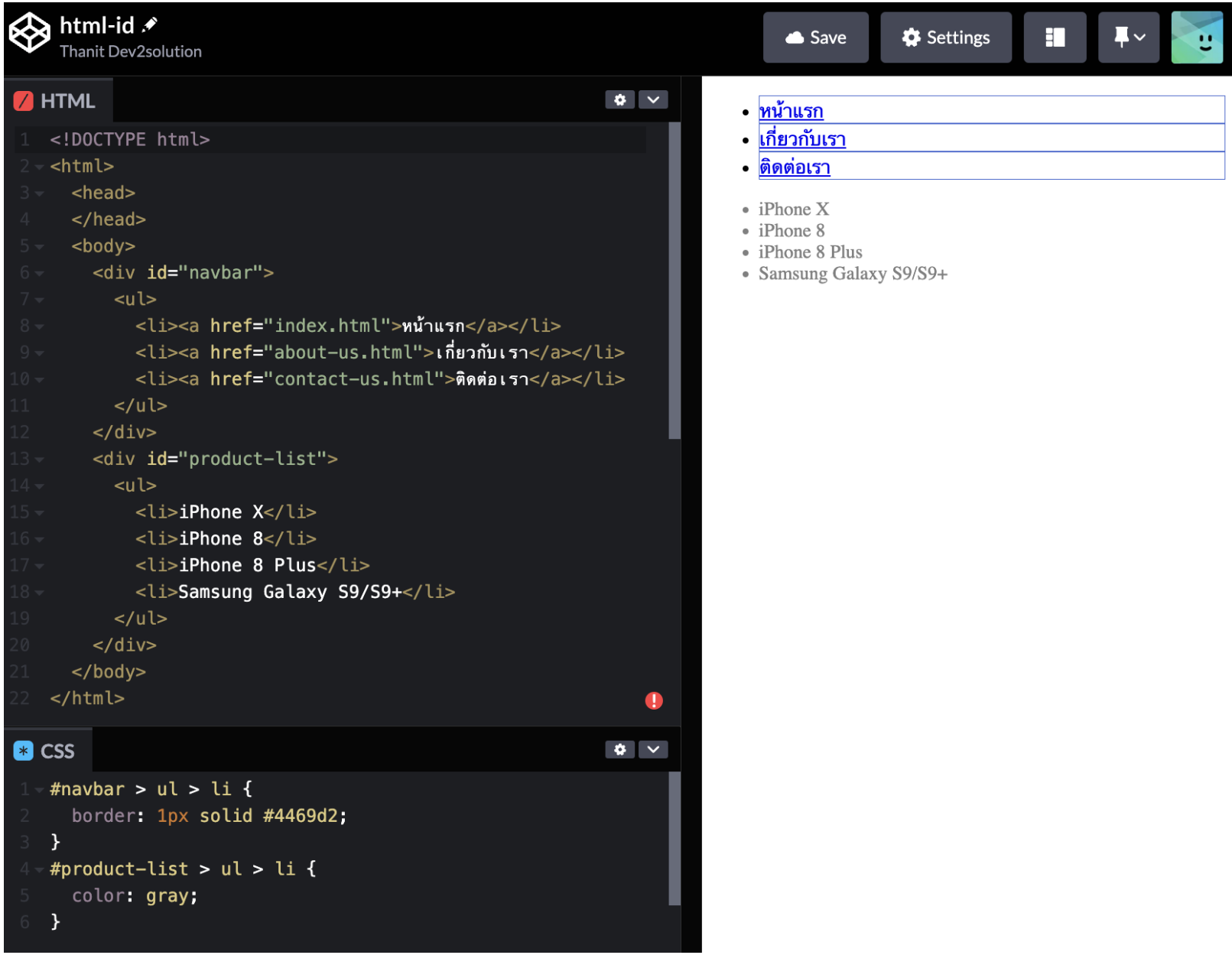
เพื่อที่จะใช้เป็นตัวเลือกง่ายๆ เราสามารถใช้

attribute (คุณสมบัติ) **id** หรือย่อมาจาก

unique identifier (ตัวระบุเฉพาะ)

เพื่อบ่งชี้ว่า element ตัวนี้คืออะไร

Example



การจัด HTML ให้เป็นระเบียบ

จัดระเบียบ HTML ด้วย Class

คุณสมบัติ class (ประเภท) นั้นก็คล้ายๆ id

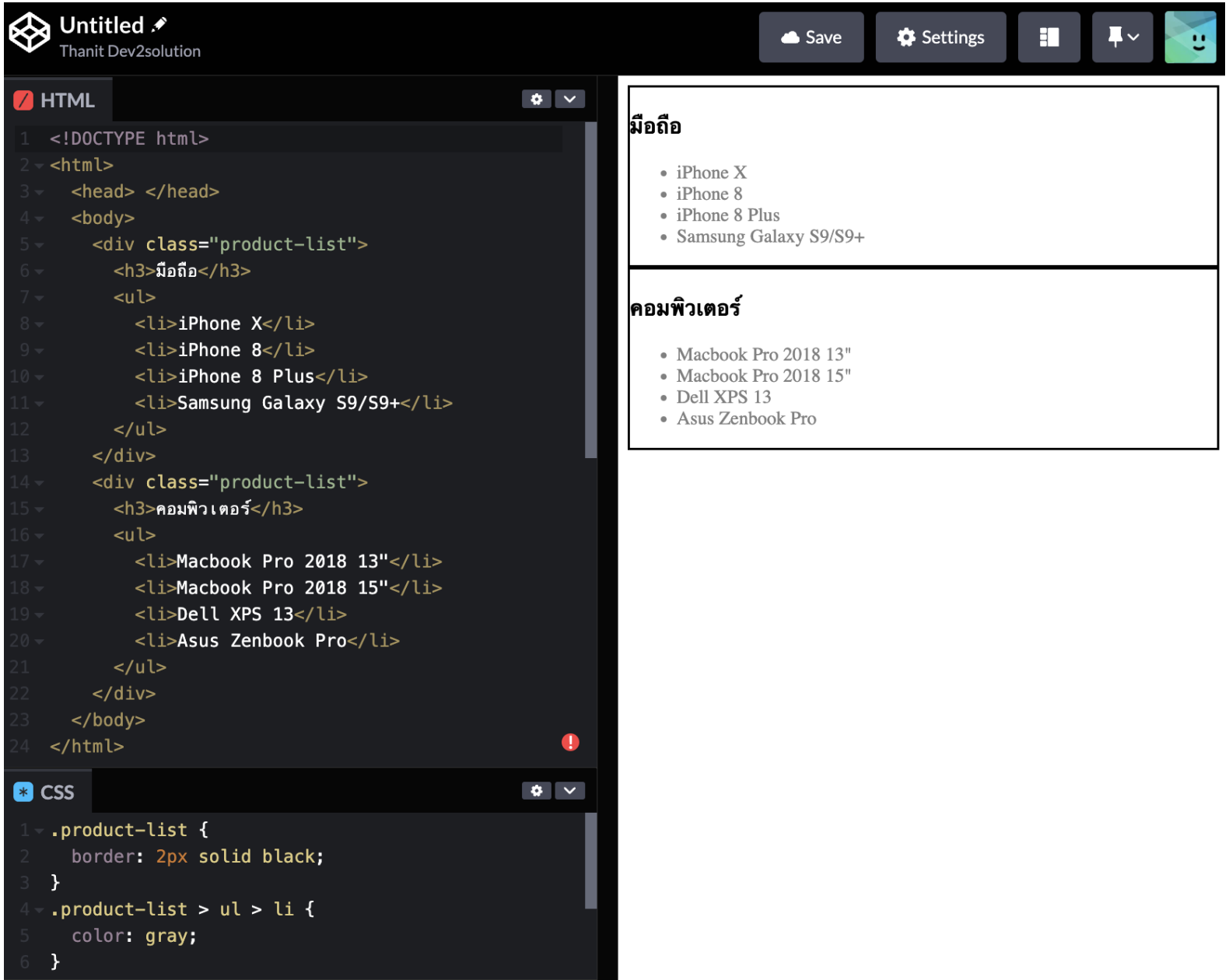
แต่แตกต่าง ตรงที่ว่าสามารถ **ใช้ชื่อซ้ำกันได้**

สังเกตว่า

การที่จะเลือกตัวที่จะตกแต่งโดย class นั้น

เราจะใช้เครื่องหมาย . และตามด้วยชื่อ class

Example



การจัด HTML ให้เป็นระเบียบ

ความแตกต่างระหว่าง **div** และ **span**

span ก็คล้ายกันกับแท็ก **div** แต่มักจะเอาไว้ใช้จัดแต่ง ข้อความสั้นๆ ที่อยู่ใต้การกำหนดรูปแบบของแท็กอื่นอยู่แล้ว เพื่อให้มีรูปแบบที่ต่างไปจากรูปแบบเหล่านั้น

ความแตกต่าง

- div เป็น block
- span เป็น inline
- span ใช้ในการจัดข้อความหรือตกแต่งด้านในของ Element

การจัด HTML ให้เป็นระเบียบ

ความแตกต่างระหว่าง **div** และ **span**

จะเห็นได้ว่า Div จะมีการเรียงลงมา แต่

Span จะเป็นการเรียงข้อความทั้งๆที่โค้ด

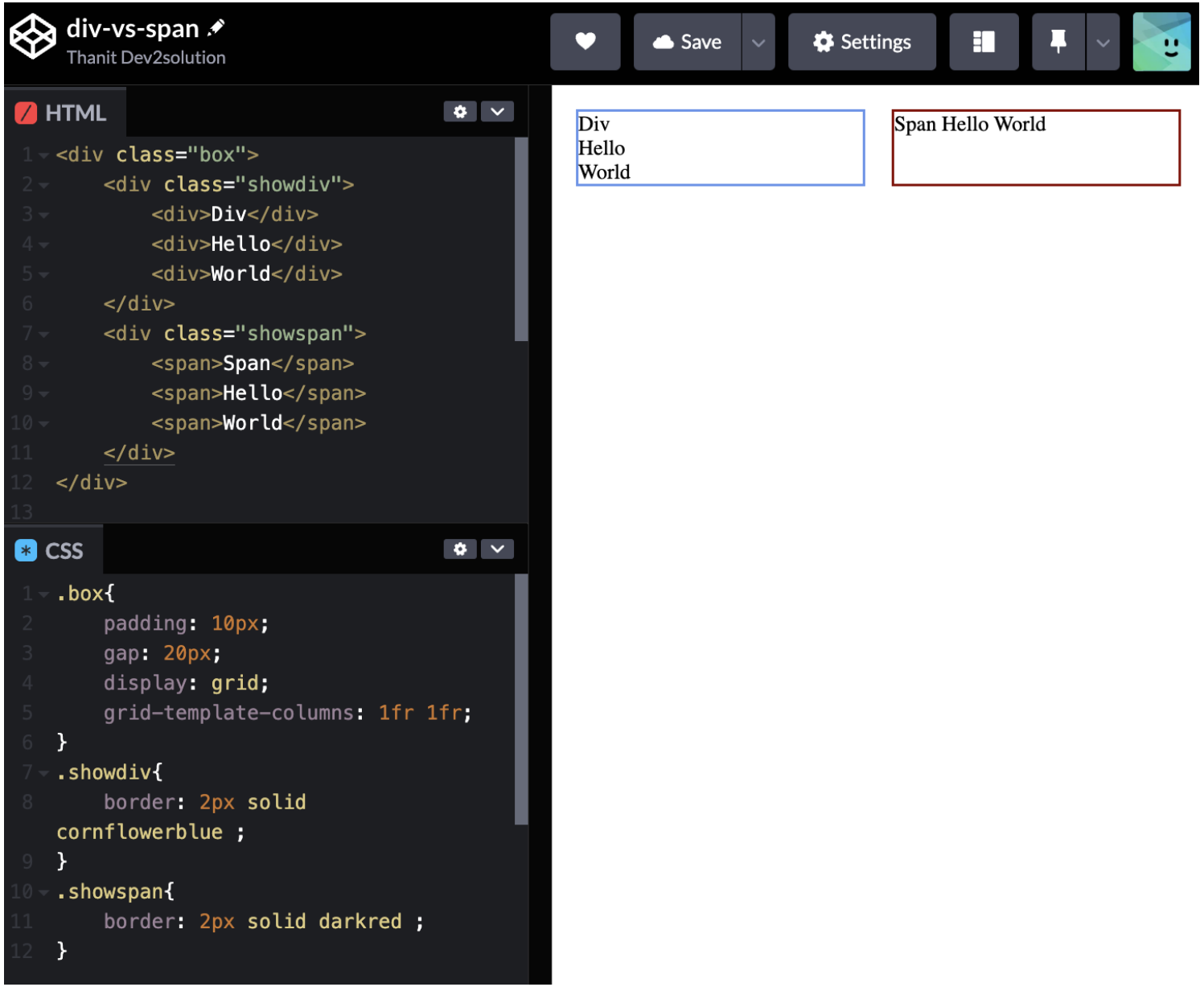
เขียนเหมือนกันแต่เปลี่ยนแค่ Element

“Element **Div** มี Display เป็น block

และ

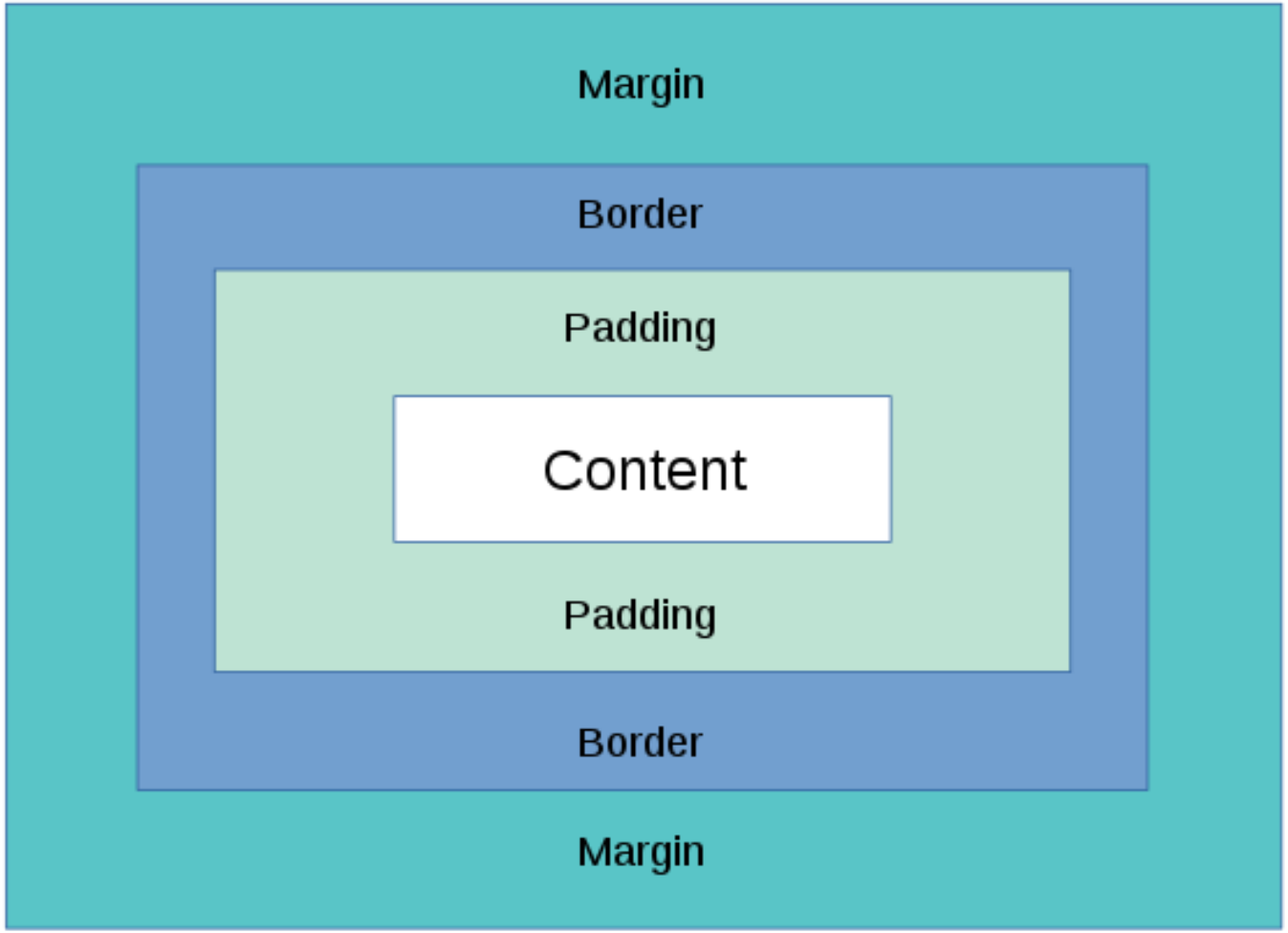
Element **Span** มี Display เป็น inline”

[Example]



CSS Box Model

คือ การกำหนดพื้นที่รอบ Element



- **margin** กำหนดระยะห่างจากเส้นขอบหรือพื้นที่ภายนอก ของ Element เทียบกับวัตถุอื่นๆ **(ช่องว่างข้างนอก)**
- **padding** กำหนดพื้นที่ระยะห่างภายในของ Element **(ช่องว่างข้างใน)**
- **border** กำหนดเส้นขอบ

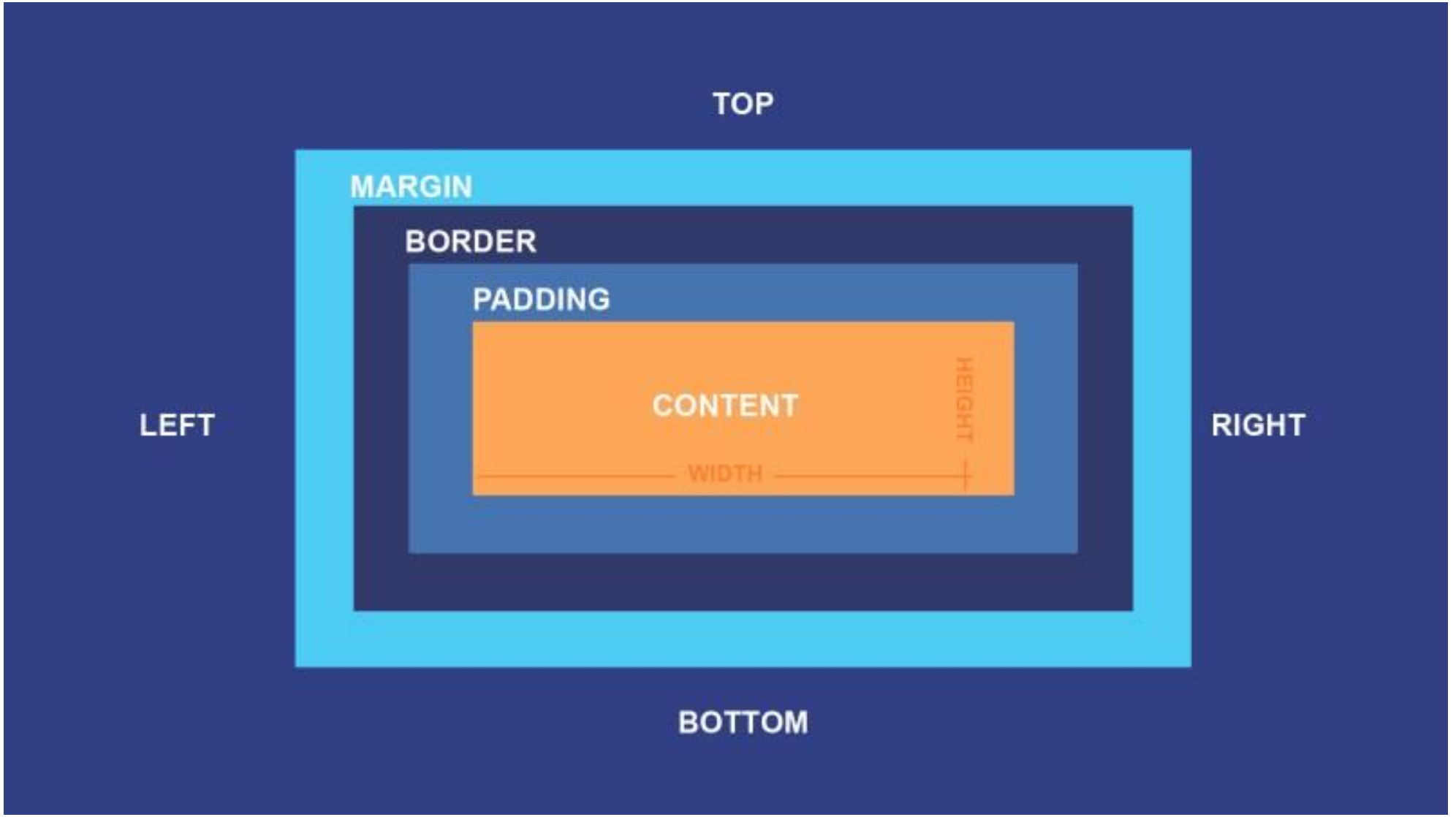
  - กำหนดได้ทั้งรูปแบบ
    ด้านบน (top) ด้านล่าง (bottom)
    ด้านขวา (right) และด้านซ้าย (left)

  - ทั้ง margin และ padding เช่น margin-left , margin-right , padding-left, padding-bottom
    หน่วยที่กำหนดได้ เช่น px, pt, cm อื่นๆ
    หรือแบบเปอร์เซ็นต์ (%) โดยมีค่า default เป็น 0

อ่านเพิ่ม: CSS สำหรับจัดหน้าเว็บ(Page Layout)

https://commons.wikimedia.org/wiki/File:Box-model.svg

CSS Box Model

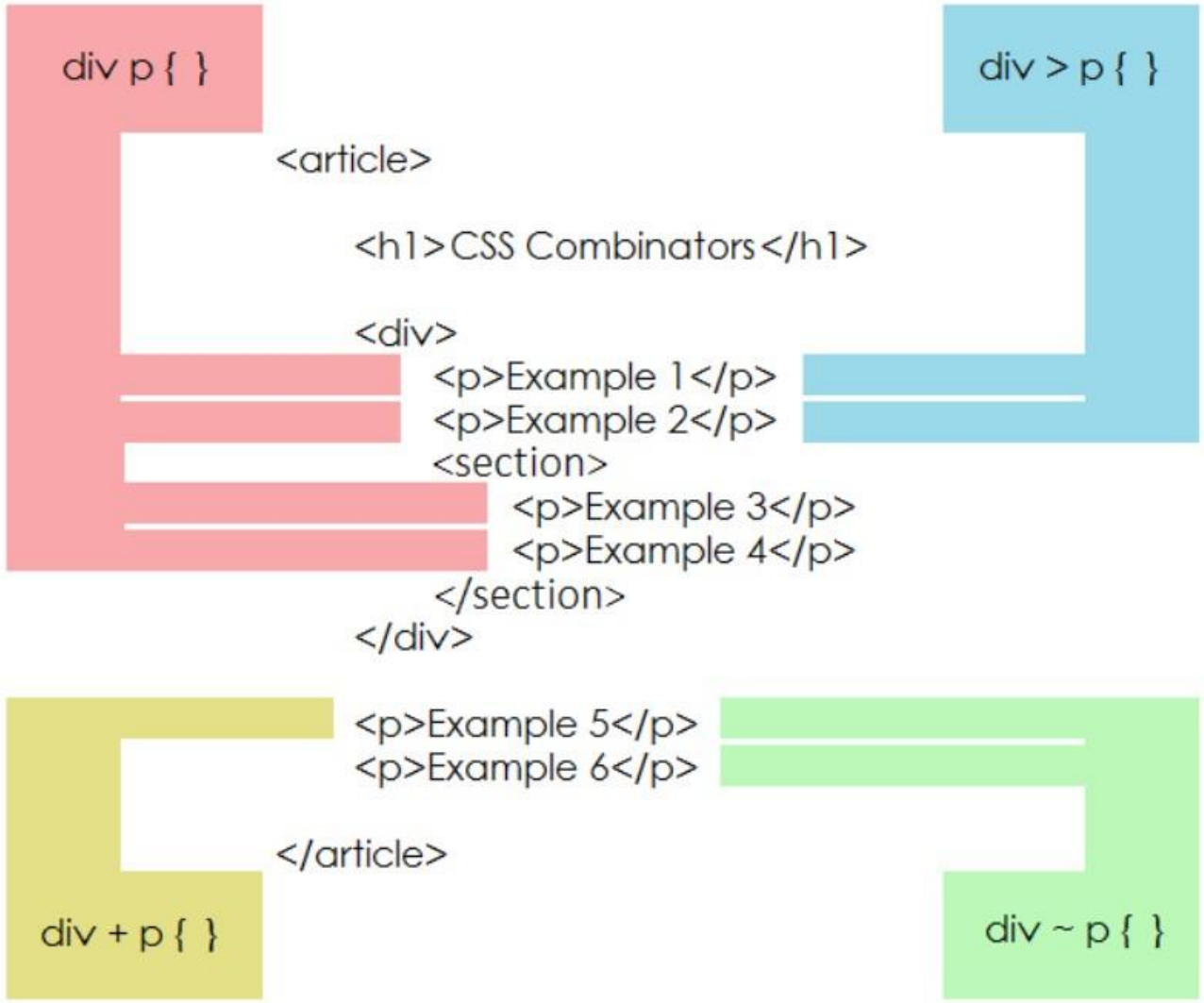


อ่านเพิ่มที่: https://www.geeksforgeeks.org/css-box-model/

CSS Combinators

ตัว CSS Combinators จะเป็นเรื่องของการเลือก selector เลือก adamant ต่างๆที่ต้องการจะเข้าถึง มีด้วยกันทั้งหมด 4 ตัว

1. Descendant Selector
2. Child Selector
3. Adjacent Sibling Selector
4. General Sibling Selector



CSS Combinators

Descendant Selector

```
div p {}
```

จากรูปจะเห็นว่ามีกล่อง div และข้างในกล่อง div ก็จะมี tag p อยู่จะเห็นได้ว่ามีทั้งหมด 4 ตัว 2 ตัวแรกอยู่ใน กล่อง div
อีก 2 ตัวอยู่ใน section ดังนั้นเวลาที่ใช้คำสั่งด้านล่างนี้มันจะเลือก tag p ทุกตัว โดยไม่สนใจว่าจะอยู่ข้างใน adamant ตัวไหนเลย
Descendant Selector จะทำการเลือกทุกตัวในกล่อง div

Child Selector

```
div > p {}
```

จะมีลักษณะการเลือกโดยใช้เครื่องหมาย > ตามภาพตัวอย่างเวลาที่เราใช้ตัว Child Selector มันจะทำการเลือก tag p ที่เป็นตัวลูกของมัน
โดยภาพตัวอย่างจะเห็นว่า จะเลือกใช้ tag p ที่อยู่ด้านใน div ตัว div นั้นเองซึ่งไอตัว tag p ที่อยู่ด้านใน section ก็จะไม่ถูกเลือกนั้นเอง

Adjacent Sibling Selector

```
div + p {}
```

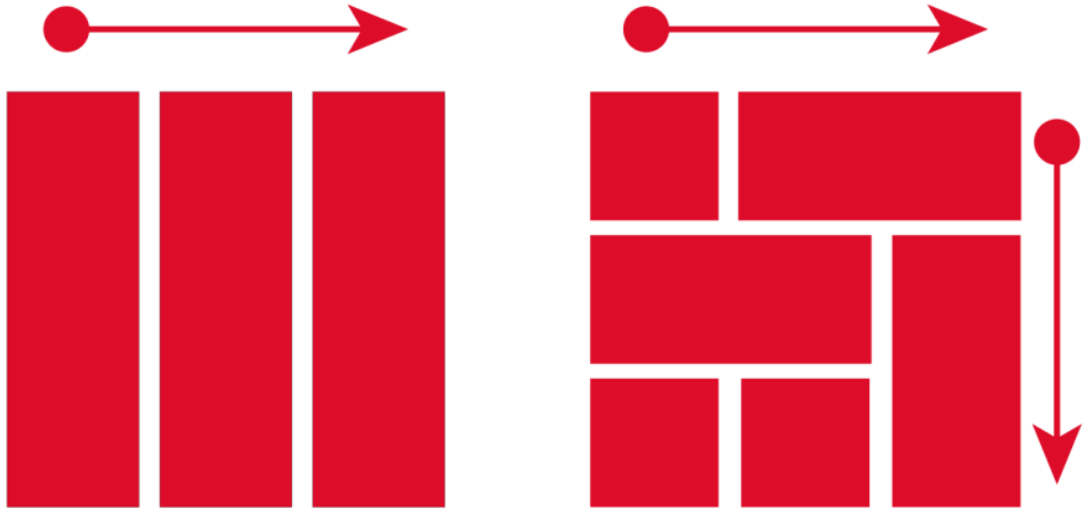
ตัวนี้เวลาที่เราใช้จะเป็นการเลือก adamant ต่อจากที่อยู่ tag div ตัวนี้ดูตามรูปภาพแทบสีเหลือง ตัวนี้เวลาใช้คำสั่ง
จะเห็นตัวตัว tag p ที่อยู่ต่อจาก tag div ด้านบนจะถูกเลือกทันทีเลย

General Sibling Selector

```
div ~ p {}
```

จะคล้ายกับเจ้าตัว Adjacent Sibling Selector แต่ว่าเวลาที่เราใช้เจ้าตัว General Sibling Selector
จะเป็นการเลือก tag p ทั้งหมดเลยที่อยู่ต่อจาก div

CSS Flexbox and Grid



Flexbox

ONE DIMENSION

CSS Grids

TWO DIMENSIONS

CSS Flexbox and Grid

Flex CSS

Flex จัดวางไปในทิศทางใดทิศทางหนึ่ง (One Dimensions) แนวตั้งหรือแนวนอนนั่นเอง
สามารถจัดการตำแหน่งการแสดงผล เฉพาะ Element ด้านในที่เลือก สามารถจัดการพื้นที่ใน Container ได้
และสามารถใช้ร่วมกันกับ Grid ได้ด้วย

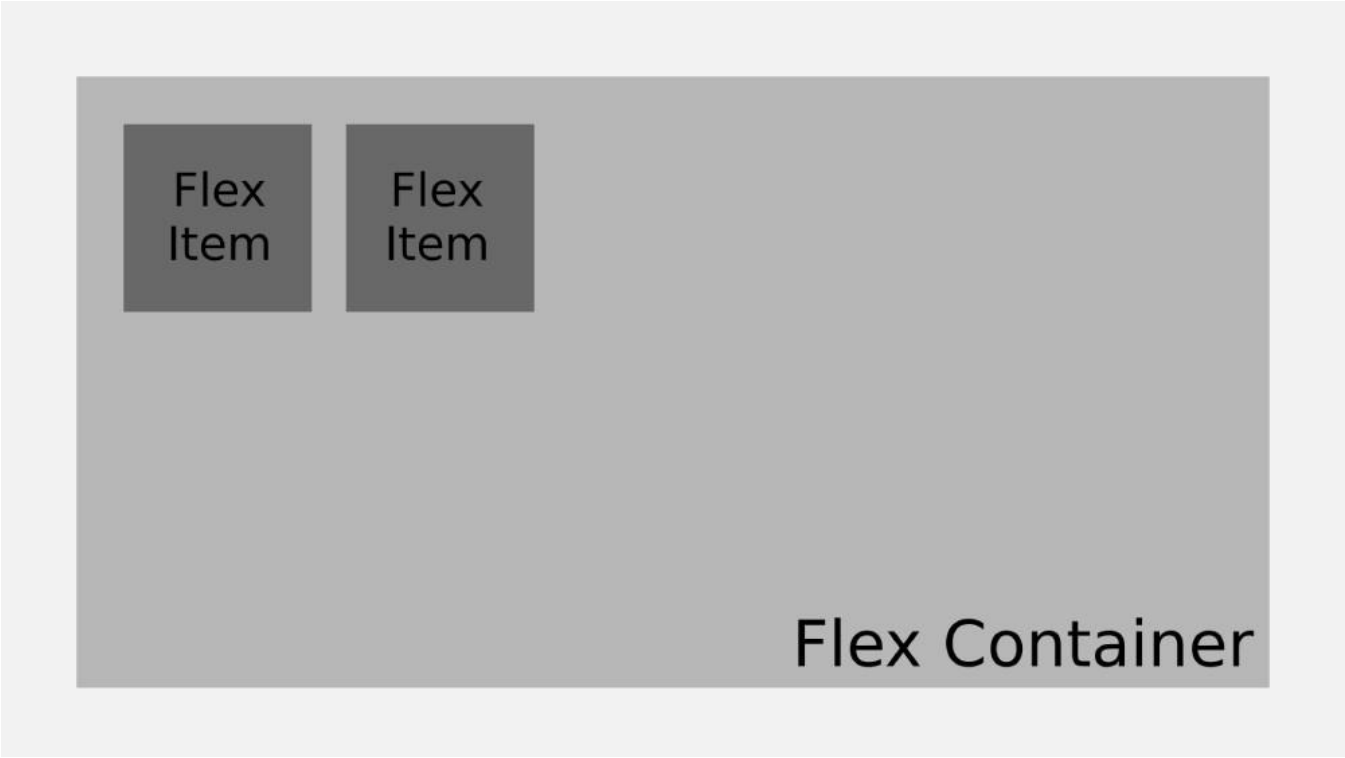
- Grid vs Flexbox คืออะไร?
- มารู้จักและลองใช้งาน CSS Flexbox กันดีกว่า
- CSS Flexbox at w3schools
- CSS Flexbox คืออะไร + สอนวิธีใช้

Flexbox

Flex Container และ Flex Item

สำหรับ Flex Layout จะมีส่วนประกอบด้วย 2 ส่วน ก็คือ

- Flex Container เป็น Layout mode เพื่อกำหนด display: flex เอาไว้
- Flex Item คือ elements ที่อยู่ภายใน Flex Container ดังตัวอย่างรูปภาพประมาณนี้



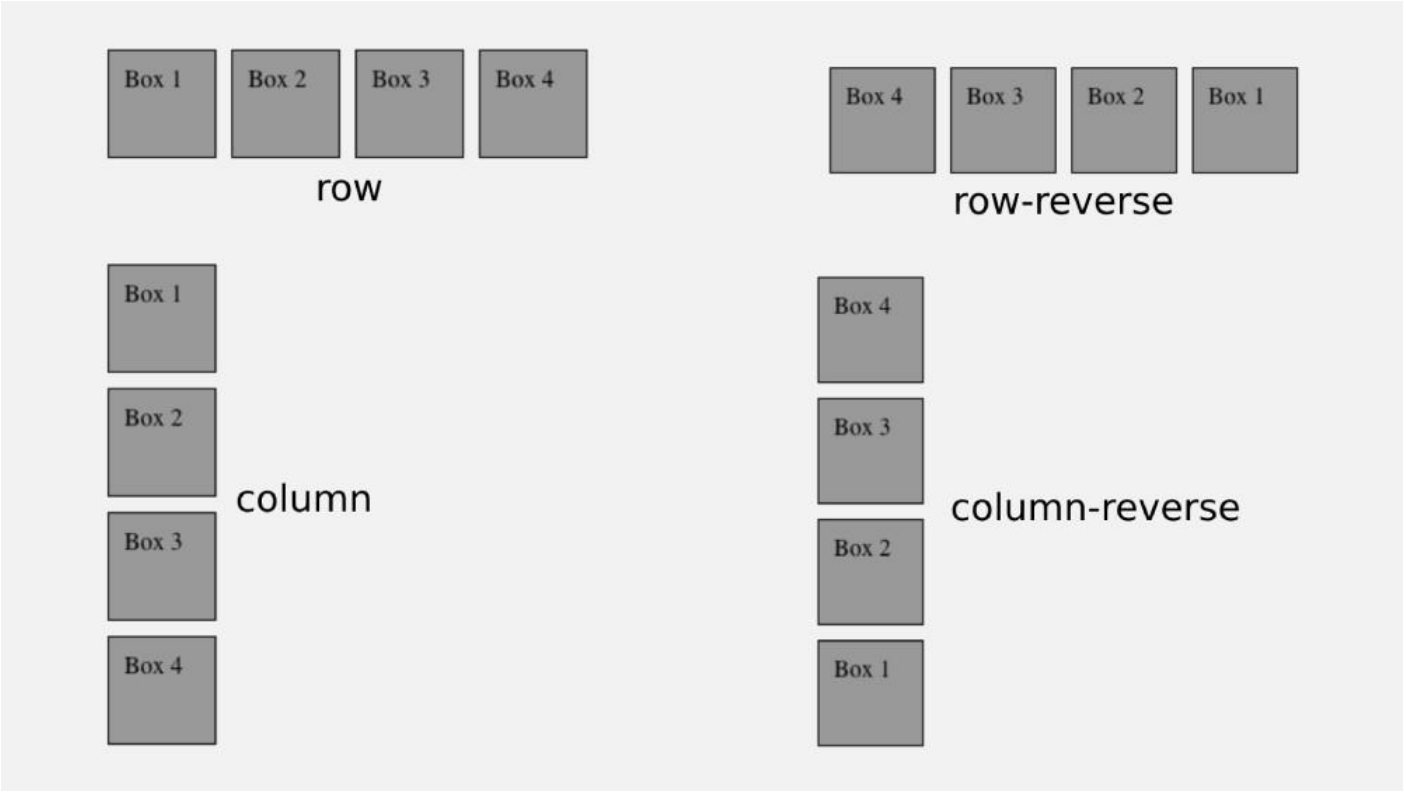
มารู้จักและลองใช้งาน CSS Flexbox

Property ต่าง ๆ ของ Flex Container

flex-direction:

เอาไว้สำหรับกำหนดการจัดเรียง Flex Item ภายใน Flex Container

- **row**: เรียงเป็นแถวแนวนอน จากขวาไปซ้าย
- **row-reverse**: เรียงเป็นแถวแนวนอน จากซ้ายไปขวา
- **column**: เรียงเป็นแถวแนวตั้ง จากบนลงล่าง
- **column-reverse**: เรียงเป็นแถวแนวตั้ง จากล่างขึ้นบน

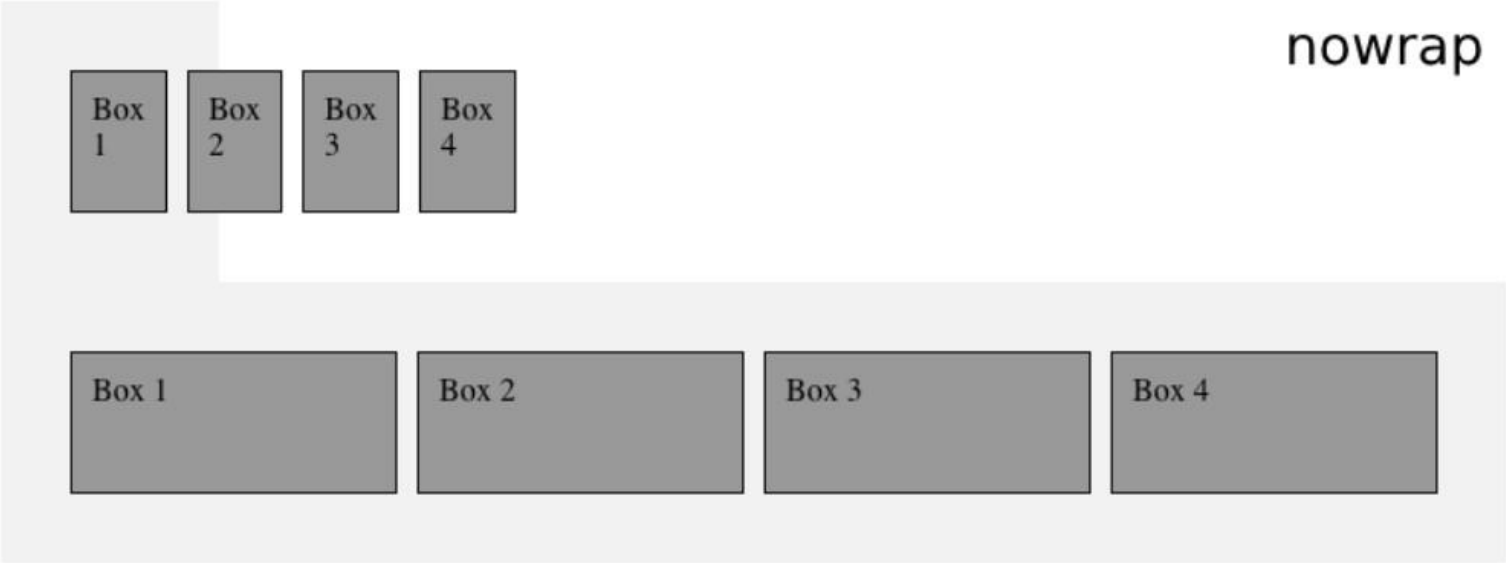


Property ต่าง ๆ ของ Flex Container

flex-wrap:

เป็นการกำหนดว่าจะให้ Flex Item ภายใน Flex Container

- **nowrap**: คือการไม่ให้ Flex Item มีการดันลงมาด้านล่าง และจะมีการลดขนาด Flex Item ให้มีขนาดพอดีกับ Flex Container หาก Flex Container เล็กว่า Flex Item ทั้งหมด ก็จะทำให้ล้นออกไปทางด้านขวา
- **wrap**: การกำหนดให้ Flex Item มีการดันลงมาด้านล่างตามขนาดของ Flex Container โดยมีการจัดเรียง Flex Item จากบนลงล่าง
- **wrap-reverse**: การกำหนดให้ Flex Item มีการดันลงมาด้านล่างตามขนาดของ Flex Container โดยมีการจัดเรียง Flex Item จากล่างขึ้นบน

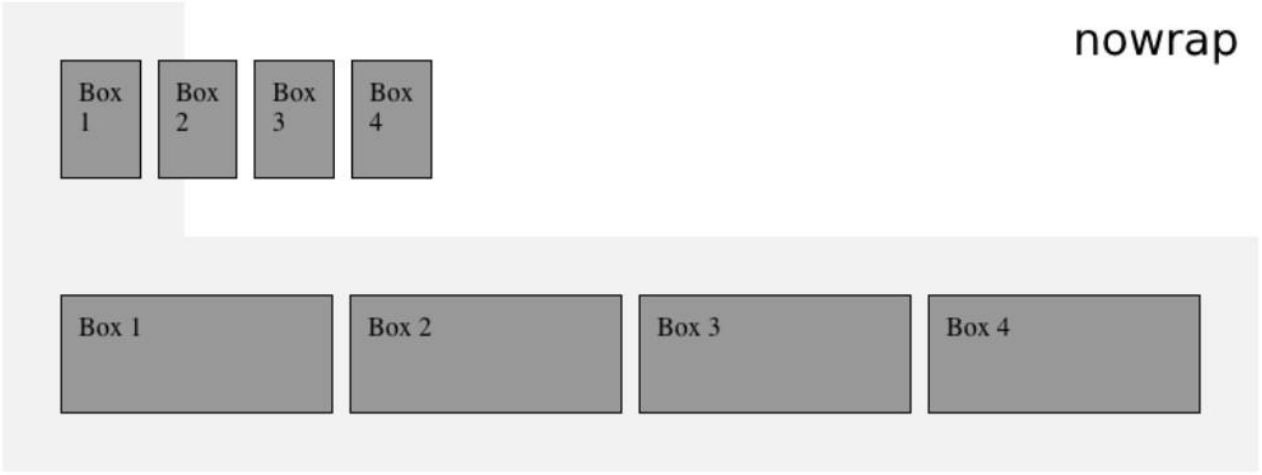


Property ต่าง ๆ ของ Flex Container

flex-wrap:

เป็นการกำหนดว่าจะให้ Flex Item ภายใน Flex Container

- **nowrap**: คือการไม่ให้ Flex Item มีการดันลงมาด้านล่าง และจะมีการลดขนาด Flex Item ให้มีขนาดพอดีกับ Flex Container หาก Flex Container เล็กว่า Flex Item ทั้งหมด ก็จะทำให้ล้นออกไปทางด้านขวา
- **wrap**: การกำหนดให้ Flex Item มีการดันลงมาด้านล่างตามขนาดของ Flex Container โดยมีการจัดเรียง Flex Item จากบนลงล่าง
- **wrap-reverse**: การกำหนดให้ Flex Item มีการดันลงมาด้านล่างตามขนาดของ Flex Container โดยมีการจัดเรียง Flex Item จากล่างขึ้นบน

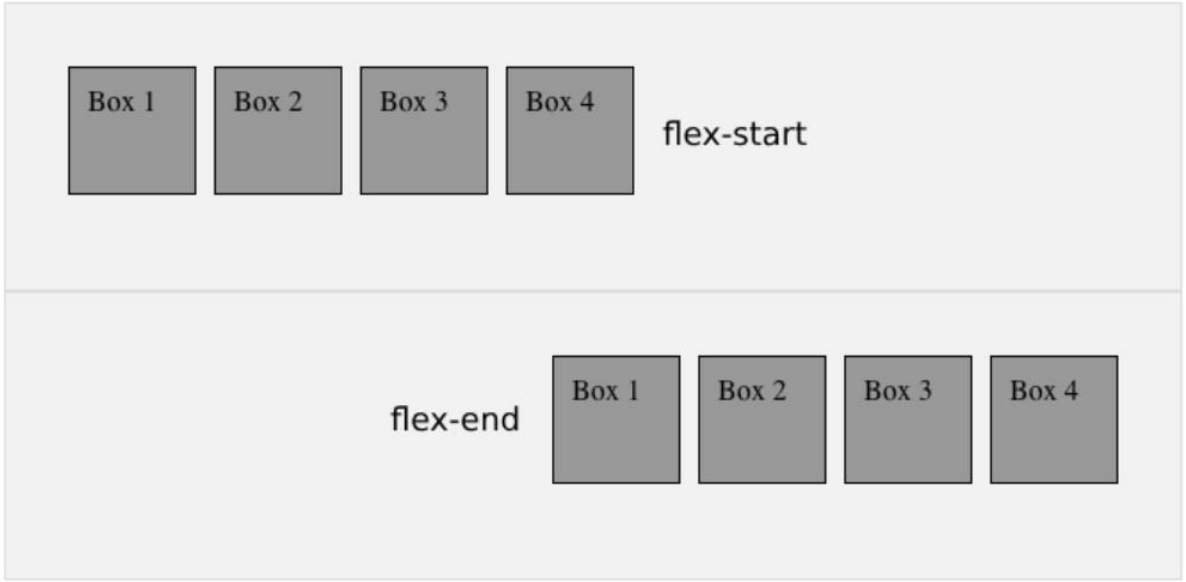


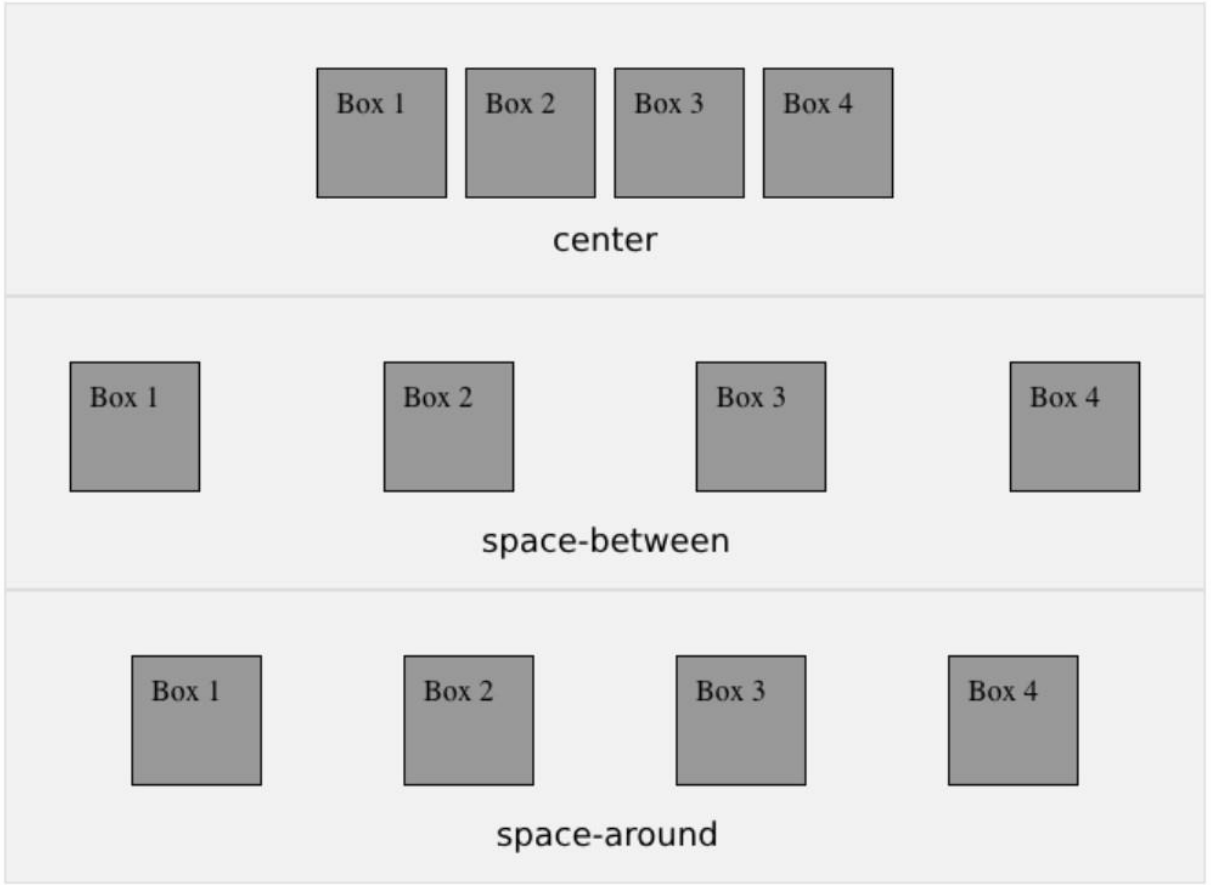
Property ต่าง ๆ ของ Flex Container

justify-content:

เป็นการกำหนดการจัดตำแหน่งของ Flex Item ภายใน Flex Container

- **flex-start**: จัด Flex Item ให้ชิดซ้าย **flex-end**: จัด Flex Item ให้ชิดขวา
- **center**: จัด Flex Item ให้อยู่ตรงกลาง
- **space-between**: จัด Flex Item ให้อยู่ระหว่างกัน
- **space-around**: จัด Flex Item ให้มีช่องว่างระหว่าง ด้านหน้า ตรงกลาง ด้านหลัง ให้เท่ากัน



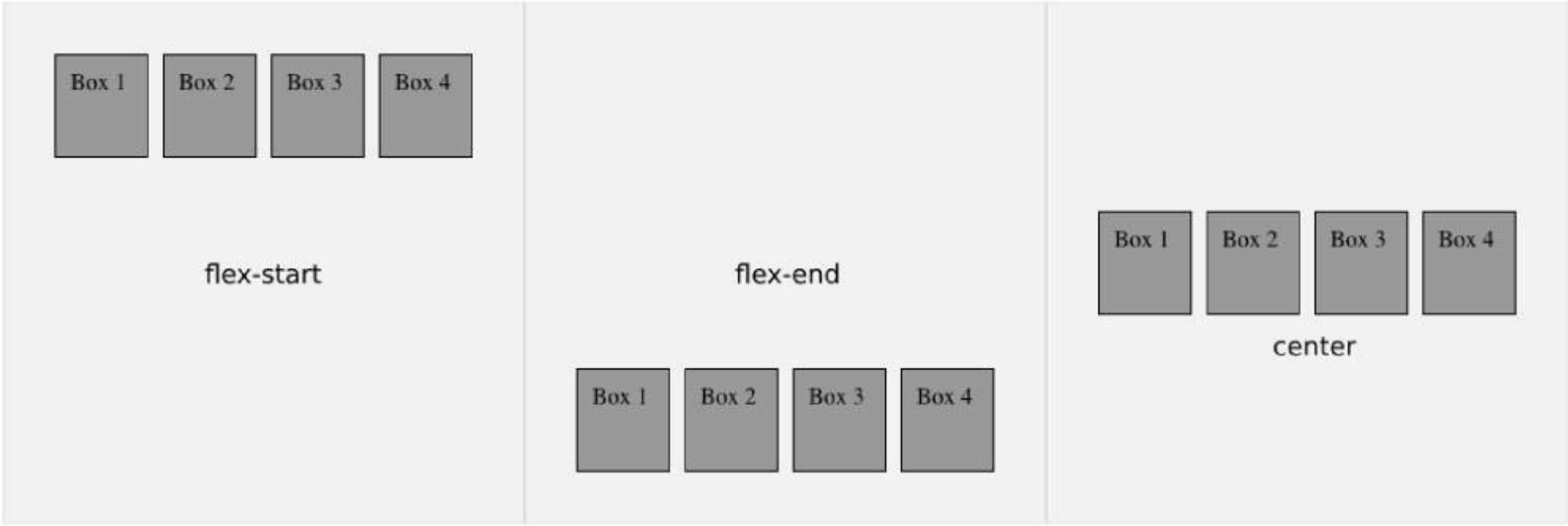


Property ต่าง ๆ ของ Flex Container

align-items:

เป็นการกำหนดการจัดตำแหน่งของ Flex Item ในแนวตั้ง ภายใน Flex Container

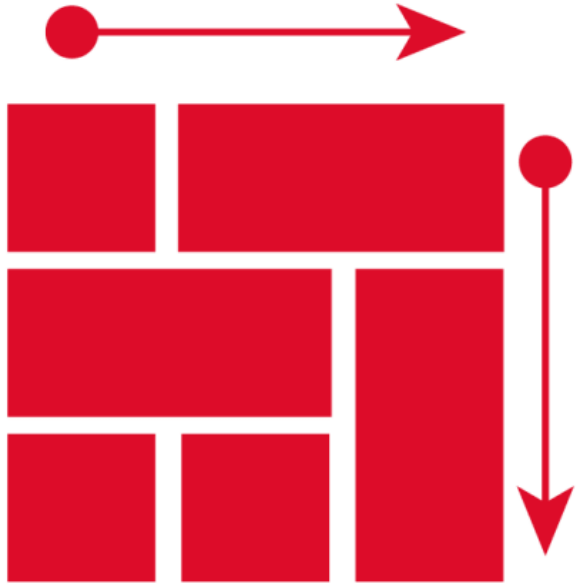
- **flex-start**: จัด Flex Item ให้อยู่บนสุดใน Flex Container
- **center**: จัด Flex Item ให้อยู่ตรงกลางใน Flex Container
- **flex-end**: จัด Flex Item ให้อยู่ล่างสุดใน Flex Container



Grid CSS

Grid เป็นตัวช่วยในการจัด Layout ให้กับเรา ซึ่ง Grid เป็น โครงสร้างแบบ 2 มิติ (Two Dimensions) ที่จะช่วยจัด Layout ได้ทั้ง **Row** และ **Column** ถ้าเป็นเมื่อก่อนเราอาจจะต้องใช้ *Float* และ *Position* เข้ามาช่วย แต่ในปัจจุบันได้มีการใช้ Grid มาช่วยในส่วนนี้แล้ว ดังนั้นจึงไม่จำเป็นที่จะต้องใช้ **Float** และ *Position* เลย

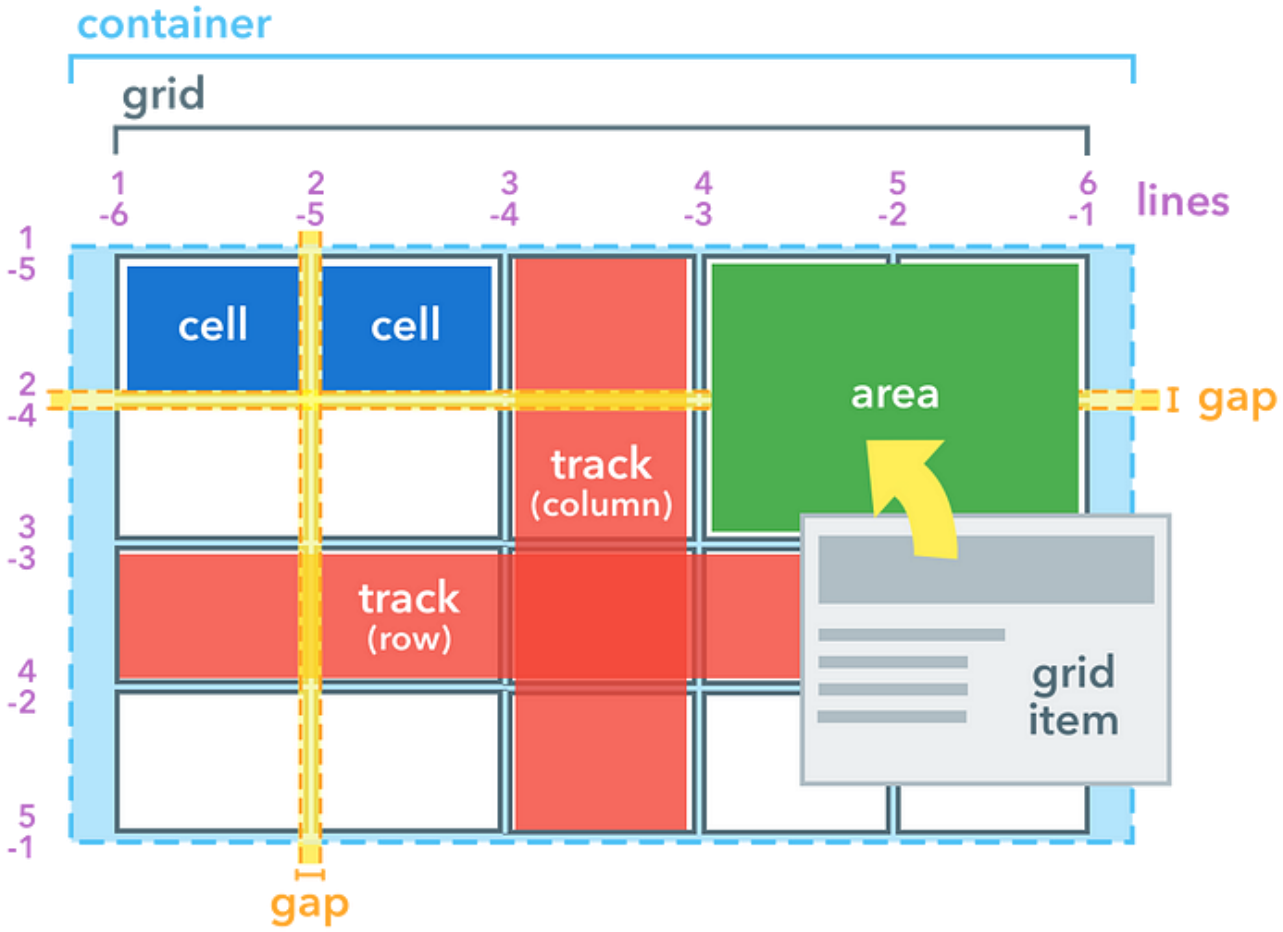
- Grid vs Flexbox คืออะไร?
- CSS : Grid Layout



Grid Layout

 css Grid

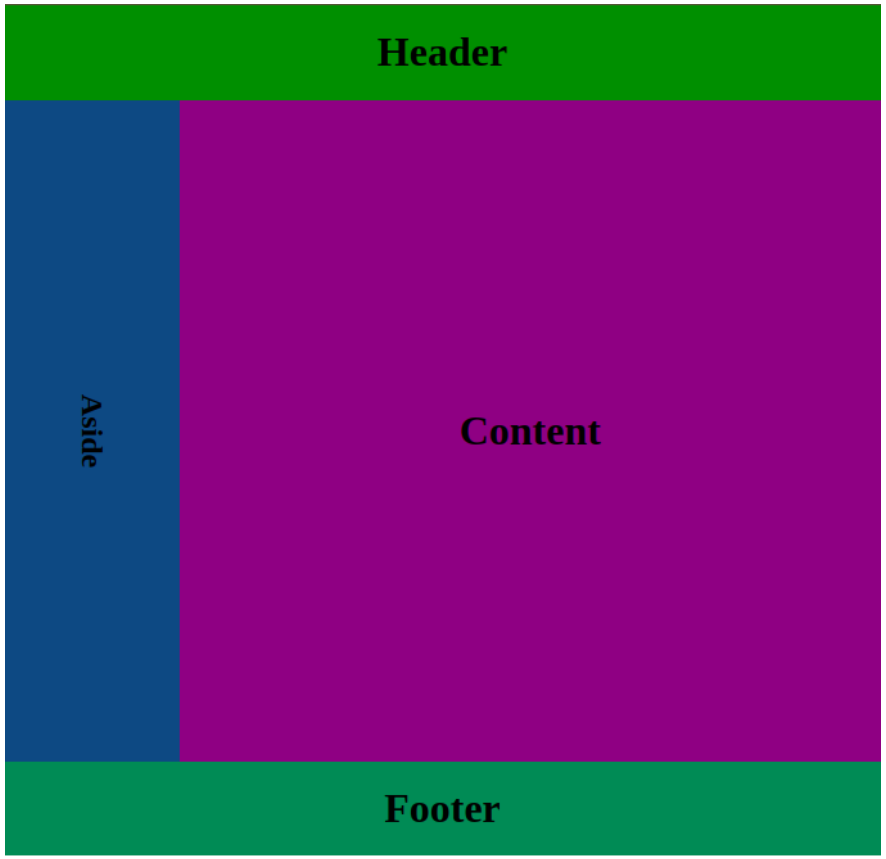
Grid Layout นั้นเป็นตัวการเด่นสำหรับเรื่องการจัดวาง layout เลยก็ว่าได้
โดยที่ตัวมันเนี้ยมีการจัดวางรูปแบบการแสดงผลแบบ 2 Dimenation (สองทิศทาง) ซึ่งจะอิสระกว่า Flex box

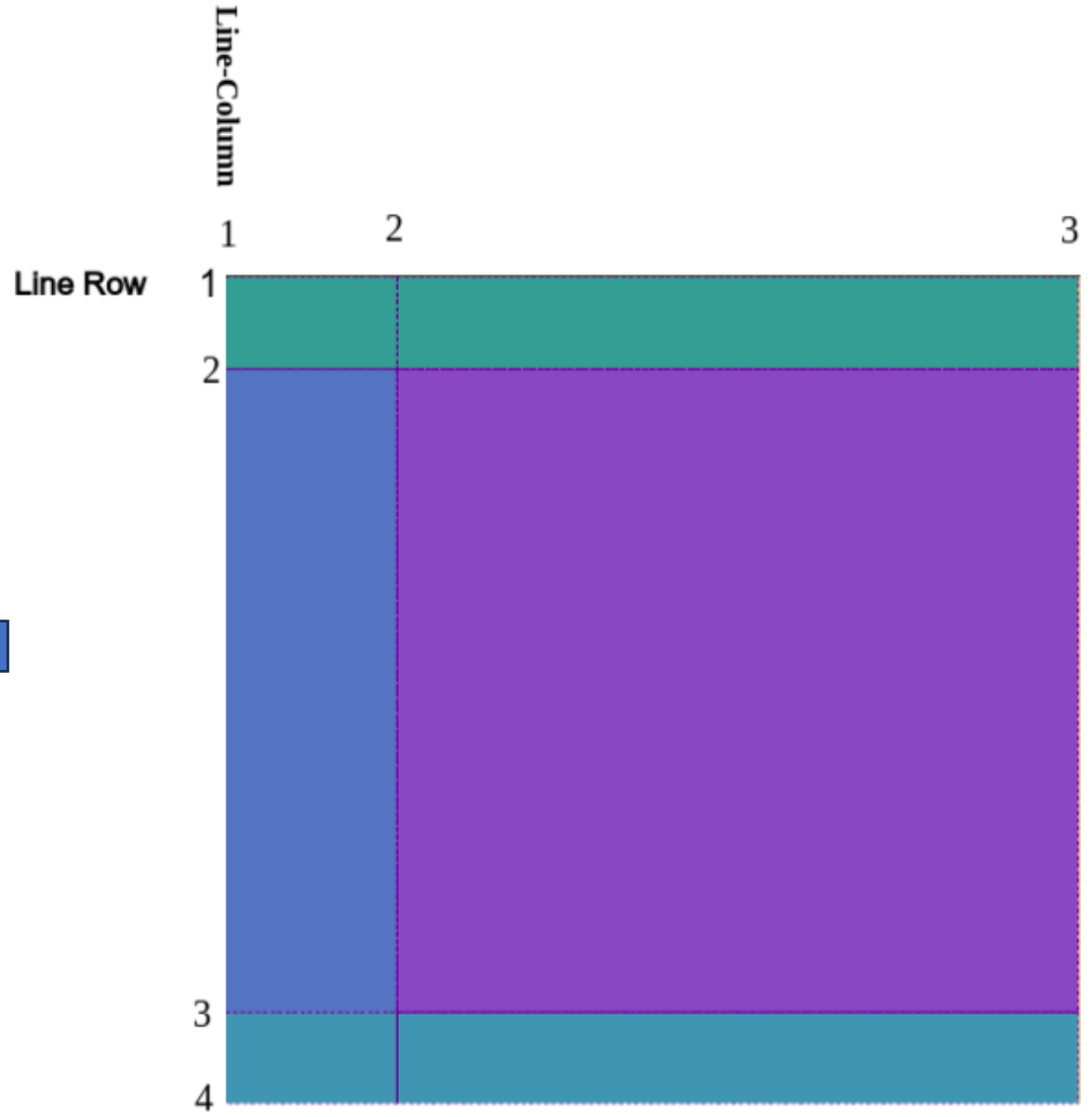


- สิ่งที่อยู่ใน Grid Container นับว่าเป็น **Grid Item**
- เส้นแต่ละเส้นตาม แนวนอน เรียกว่า **Grid Line Row**
- เส้นแต่ละเส้นตาม แนวตั้ง เรียกว่า **Grid Line Column**
- Grid Line Row สองเส้นที่ตัดกันจนเกิดพื้นที่เราเรียก **Grid Track Row หรือ Grid Row**
- Grid Line Column สองเส้นที่ตัดกันจนเกิดพื้นที่เราเรียก **Grid Track Column หรือ Grid Column**
- พื้นที่ที่เกิดจากการตัดกันของ Grid Row และ Grid Column เรียกว่า **Grid Cell**
- พื้นที่ที่เกิดจากการรวมของ Grid Cell ใกล้ๆเราเรียกว่า **Grid Area**
- ช่องว่างภายนอกระหว่าง Grid Cell เราเรียกว่า **Grid Gap**

Grid Layout

css Grid





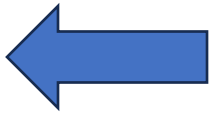
Grid Layout

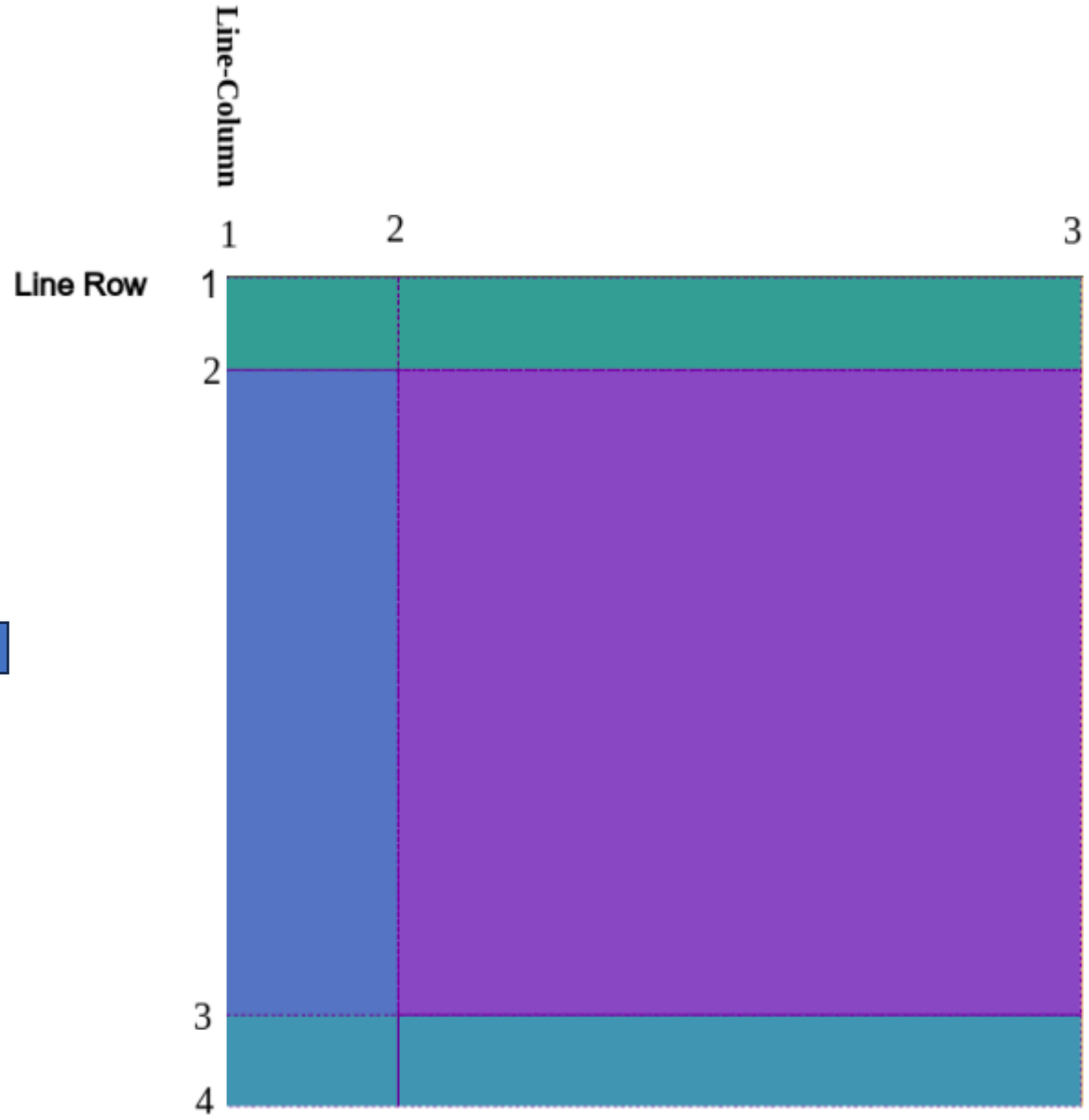
css Grid

จากภาพคือการแบ่ง layoutเป็นส่วนต่างๆ

ให้ง่ายต่อการจัดวาง เราสามารถมองเป็น grid ได้ดังนี้

- **Grid Line Row** 4 เส้น
- **Grid Line Column** 3 เส้น
- **Grid Track Row** 3 ช่อง
- **Grid Track Column** 2 ช่อง





สังเกตุได้ว่า

Grid Item ไม่จำเป็นต้องอยู่ใน Cell เดียวเสมอ

เพราะอย่าง Header และ Footer อยู่ในพื้นที่ 2 Grid Track Column

Grid Layout

เริ่มต้นการใช้ Grid

ขั้นตอนแรกเลย ให้ทำการสร้างไฟล์ index.html และ style.css ตามตัวอย่างในรูป
สร้าง div ขึ้นมา 1 ตัว ที่ชื่อว่า container กำหนด CSS display: grid;
ดังนั้น ตัว Element ที่อยู่ภายใน div ตัวนี้จะถูกจัดวาง layout อยู่ในรูปแบบ *grid* เป็นที่เรียบร้อย

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample</title>
    <link rel="stylesheet" href="styless.css">
</head>
<body>
    <div class="container">
        <div class ="grid-item">Home</div>
        <div class ="grid-item">About us</div>
        <div class ="grid-item">Portfolio</div>
        <div class ="grid-item">Contact Us</div>
        <div class ="grid-item">Privacy Policy</div>
      </div>
</body>
</html>
```

style.css

```css
.container {
    display: grid;
}
```

```css
.container {
    display: grid;
    grid: auto auto / auto auto auto;
    background-color: mediumturquoise;
    grid-gap: 10px;
    padding: 10px;
}

.grid-item{
   background-color: white;
   text-align: center;
   padding: 25px 0;
   font-size: 30px;
}
```

Grid Layout

css Grid

```css
.container {
    display: grid;
    grid: auto auto / auto auto auto;
    background-color: mediumturquoise;
    grid-gap: 10px;
    padding: 10px;
}

.grid-item{
   background-color: white;
   text-align: center;
   padding: 25px 0;
   font-size: 30px;
}
```

- **grid:** เพื่อกำหนดว่า ต้องการจำนวน Row และ Column เท่าไหร่

  จากตัวอย่าง เราต้องการ Row 2 แถว และ Column 3 แถว

- **background-color:** กำหนดสีพื้นหลัง

- **grid-gap:** เป็นรูปย่อของการใช้งาน *grid-row-gap* และ

  ***grid-column-gap* กรณี** ที่ช่องว่างทั้งแนวแถวและแนวหลักเท่ากัน

  เป็นการจัดพื้นที่ให้กับ element นั่นเอง

- **padding:** เพื่อตั้งค่าระยะห่างของขอบภายใน

การแสดงผล

Home	About us	Portfolio
Contact Us	Privacy Policy	

Grid Layout

css Grid

การกำหนด Property แบบปกติ

```css
.grid-container{
    display: grid;
    grid-template-columns: 200px 800px;
    grid-template-rows:70px 530px 70px;
}
```

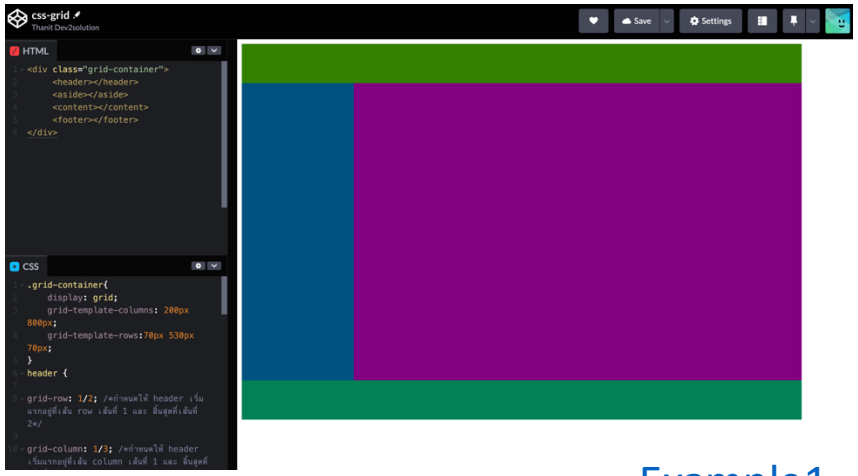
ถ้าเรากำหนด ว่า Grid-item ต่างๆ เริ่มต้นจากเส้นนี้ไปจนถึงเส้นนี้

ทำแบบนี้ไปเรื่อยๆ อาจจะทำให้สับสนได้ว่าเส้นไหนเป็นเส้นไหน

```css
header{
    grid-row: 1/2; /*กำหนดให้ header เริ่มแรกอยู่ที่เส้น row เส้นที่ 1 และ สิ้นสุดที่เส้นที่ 2*/
    grid-column: 1/3; /*กำหนดให้ header เริ่มแรกอยู่ที่เส้น column เส้นที่ 1 และ สิ้นสุดที่เส้นที่ 3*/
    background-color: rgb(52, 129, 0);/*ใส่สีตามชอบ*/
}
aside{
    grid-row: 2/3; /*กำหนดให้ aside เริ่มแรกอยู่ที่เส้น row เส้นที่ 2 และ สิ้นสุดที่เส้นที่ 3*/
    grid-column: 1/2; /*กำหนดให้ aside เริ่มแรกอยู่ที่เส้น column เส้นที่ 1 และ สิ้นสุดที่เส้นที่ 2*/
    background-color: rgb(0, 82, 129);/*ใส่สีตามชอบ*/
}
content{
    grid-row: 2/3; /*กำหนดให้ content เริ่มแรกอยู่ที่เส้น row เส้นที่ 2 และ สิ้นสุดที่เส้นที่ 3*/
    grid-column: 2/3; /*กำหนดให้ content เริ่มแรกอยู่ที่เส้น column เส้นที่ 2 และ สิ้นสุดที่เส้นที่ 3*/
    background-color: rgb(129, 0, 129);/*ใส่สีตามชอบ*/
}
footer{
    grid-row: 3/4; /*กำหนดให้ footer เริ่มแรกอยู่ที่เส้น row เส้นที่ 3 และ สิ้นสุดที่เส้นที่ 4*/
    grid-column: 1/3; /*กำหนดให้ footer เริ่มแรกอยู่ที่เส้น column เส้นที่ 1 และ สิ้นสุดที่เส้นที่ 3*/
    background-color: rgb(0, 129, 86);/*ใส่สีตามชอบ*/
}
```

html file

```html
<div class="grid-container">
    <header></header>
    <aside></aside>
    <content></content>
    <footer></footer>
</div>
```



Example1

Grid Layout


```css
.grid-container{
    display: grid;
    grid-template:
    "header header" 70px /*ความสูงของ Grid Track Row ช่องที่ 1 = 70px*/
    "aside  main" 530px  /*ความสูงของ Grid Track Row ช่องที่ 2 = 530px*/
    "footer footer" 70px /*ความสูงของ Grid Track Row ช่องที่ 3 = 70px*/
    /200px   800px;
    /*ความกว้างของ Grid Track Column ช่องที่ 1 = 200px*/
    /*ความกว้างของ Grid Track Column ช่องที่ / = 800px*/
}
```

```css
header{
    grid-area: header; /*header จะอยู่ในพื้นที่ header*/
    background-color: rgb(52, 129, 0);/*ใส่สีตามชอบ*/
}
aside{
    grid-area: aside; /*aside จะอยู่ในพื้นที่ aside*/
    background-color: rgb(0, 82, 129);/*ใส่สีตามชอบ*/
}
content{
    grid-area: main; /*content จะอยู่ในพื้นที่ main*/
    background-color: rgb(129, 0, 129);/*ใส่สีตามชอบ*/
}
footer{
    grid-area: footer; /*footer จะอยู่ในพื้นที่ footer*/
    background-color: rgb(0, 129, 86);/*ใส่สีตามชอบ*/
}
```

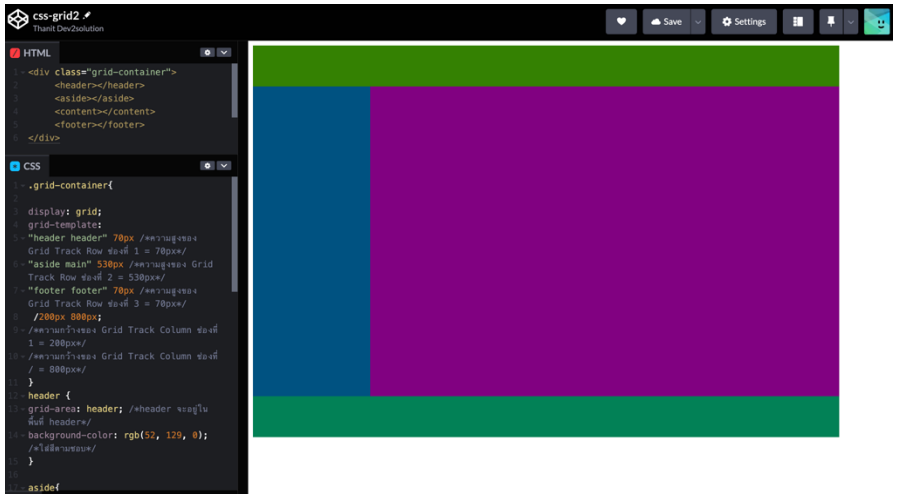
ถ้ากำหนดชื่อให้พื้นที่นั้น ๆ..จะเกิดอะไรขึ้น

ภาพทางซ้ายมือเป็นวิธีที่**ง่ายต่อการเข้าใจ**...

เพียงแต่ถ้าเราใช้ grid-templateเราสามารถใส่ชื่อให้พื้นที่

Grid Cell ได้ซึ่งมันง่ายต่อการจะไปกำหนดพื้นที่ให้กับ Grid Item เพราะเราใช้ชื่อเป็นตัวกำหนดพื้นที่ได้เลย

Example2



การใช้ CSS Grid และ CSS Flexbox ร่วมกัน

สร้าง div ขึ้นมา 1 ตัว ที่ Class ชื่อว่า menu-container กำหนด **CSS display: grid;** ดังตัวอย่างในรูป

อยากให้ความกว้างของ Logo มีความกว้าง 200px อีก คอลัมน์ให้คำนวณระยะเอาเป็น 1fr Grid Column 1 กว้าง 200px และ Grid Column 2 กว้าง 1fr

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sample</title>
    <link rel="stylesheet" href="styless.css">
</head>
<body>
    <div class="menu-container">
        <div class="logo">
            <h1>MyLogo</h1>
        </div>
        <ul>
            <li><a href="#">Home</a></li>
            <li><a href="#">About Us</a></li>
            <li><a href="#">Portfolio</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
      </div>
</body>
</html>
```

style.css

```css
.menu-container {
    display: grid;
    grid-template-columns: 200px 1fr;
}

.logo {
    text-align: center;
}

ul {
    display: flex;
    align-items: center;
}

```

การแสดงผล

MyLogo

- HomeAbout UsPortfolioContact

การแสดงผล ใหม่ หลังปรับ CSS

การใช้ CSS Grid และ CSS Flexbox ร่วมกัน

- **text-decoration: none;**

  Text Decoration คือการขีดเส้น none คือ **ไม่มีขีดเส้น**

- **align-items: center;** จัดกึ่งกลางตัวเมนู

- **margin** เพื่อตั้งค่าระยะห่างของขอบภายนอก

```css
body {
    margin: 0;
    padding: 0;
}

.menu-container {
    display: grid;
    grid-template-columns: 200px 1fr;
    background: mediumslateblue;
    color: white;
}

.logo {
    text-align: center;
}

ul {
    display: flex;
    align-items: center;
}

li {
    margin: 0 20px;
}

a {
    color: white;
    text-decoration: none;
}
```

สรุป

- **CSS Grid** เราจะใช้ในการทำให้ Layout ที่ซับซ้อนนั้น ทำได้ง่ายขึ้น ถ้าอยากจะทำเป็น Layout Website แนะนำให้ใช้ **CSS Grid**
- ส่วน **Flexbox** จะช่วยในการจัด Content เคลื่อนย้ายพวก div ต่าง ๆ และ Flexbox จะทำงานได้ดี

  ใน มิติเดียว (One Dimension)

เราจะต้องเลือกก่อนว่า จะใช้ Flexbox ในการส่วนของ **Row Column** จะช่วยประหยัดเวลาในการทำงานมาก ๆ เมื่อเราใช้ CSS Grid และ CSS Flexbox ร่วมกันได้

References

CSS เบื้องต้น

พื้นฐาน CSS3 สำหรับการพัฒนาเว็บแอพพลิเคชั่น

การจัดตำแหน่งโดย CSS ด้วย float และ clear

การจัด HTML ให้เป็นระเบียบ เพื่อให้ชีวิตง่ายขึ้น

ความแตกต่างระหว่าง Div และ Span

http://www.bus.tu.ac.th/usr/wanchai/is216/sheet/IS216_Part3_for_B&W_print.pdf

เริ่มต้นเขียน CSS ต้องรู้อะไรบ้าง

บทเรียนออนไลน์/บทเรียน-css

References

สอน HTML5 & CSS3 สำหรับผู้เริ่มต้น [Phase1]

https://www.youtube.com/watch?v=HcInSUzhaUc

https://www.youtube.com/watch?v=0hfeNPM7piw

สอน HTML5 & CSS3 สำหรับผู้เริ่มต้น [Phase2]

https://www.youtube.com/watch?v=EdbWjGkPlUo

https://www.youtube.com/watch?app=desktop&v=vH3i3pfPbBs&t=0s

Let’s practice by your own