บทที่ 7 CSS

Computer Graphics Design 1

ทส 105 การออกแบบคอมพิวเตอร์กราฟิก 1

Lecture Miss Nongkran Khomwichai
Information Technology Division
Faculty of Science , Maejo University
Chiangmai ,50290 Thailand.
E-mail : it.nongkran@gmail.com

Contents

Contents

รู้จักกับ CSS

คำเตือน*ก่อนจะเริ่มเรียน CSS คุณควรที่จะมีความรู้ในภาษา HTML / XHTML อยู่พอสมควร

- CSS ย่อมาจาก Cascading Style Sheets
- ใช้กำหนด "การแสดงข้อมูล" ของหน้าเว็บเพจ
- กำหนดโดยเก็บไว้ในหน้าเว็บเพจ **หรือ** เก็บไว้เป็นไฟล์ภายนอกก็ได้
- การเก็บ CSS เป็นไฟล์ภายนอกทำให้สามารถนำไปใช้ร่วมกับเว็ปไซต์อื่นๆได้

การทำงานของ CSS ไม่ใช่การสร้างข้อมูลใหม่ แต่จะเป็นการปรับปรุงการแสดงข้อมูลของไฟล์ html เช่น

```
<p>this is text</p>
<h1>this is header</h1>
```

ซึ่งโดยปรกติแล้ว เราจะใช้แท็ก <font> ในการแสดงลูกเล่นของข้อความ เช่น การเปลี่ยนสีข้อความ

```
<font color="red">.....</font>
```

ซึ่งการทำในรูปแบบนี้จะทำให้เสียเวลาและเสียพื้นที่ในการทำงานมาก ลองคิดถึงกรณีที่เรามีโค้ดโปรแกรมซัก **1000** บรรทัดดู

ด้วยเหตุนี้จึงได้มีการสร้าง CSS(cascading style sheets) ขึ้นมาเพื่อช่วยลดปัญหาและความยุ่งยากในการแสดงผลของ html

ตัวอย่างเว็บไซต์ที่มีการใช้ CSS

ไวยากรณ์ของ CSS

- ไวยากรณ์ของ CSS ประกอบขึ้นจาก 3 ส่วนคือ
  - Selector
  - Property
  - value

```
Selector { property1 : value1 ; proerty2 : value2 ; ... }
```

Selector คือ **อีลีเม้นท์** หรือ **แท็ก** ทั่วไปของภาษา HTML เช่น p{...} ของแท็ก <p> หรือ h1{...} ของแท็ก <h1>
Property คือ attribute ที่คุณอยากจะเปลี่ยน เช่น color(สีข้อความ), font-size(ขนาดตัวอักษร)
Value คือ ค่าของ attribute ที่เราต้องการให้เป็นเช่น color : red (เปลี่ยนข้อความเป็นสีแดง), font-size : 15px (เปลี่ยนข้อความให้มีขนาดเท่ากับ 15 pixel)

การใส่ property หลายอย่างให้ selector เดียวจะคั่นระหว่าง property ด้วย ' ; ' และคั่นระหว่าง property กับ value ด้วย ' : '

```
p { color : red ; font-size : 15px ; text-align : center }
```

ไวยากรณ์ของ CSS : Class selector

- การประการ property ให้กับ selector หลายตัวพร้อมกัน ก็สามารถทำได้เช่นกัน

```
 p, h1, h3, span, tr, td
{
        color : red;
        text-align : center;
}
```

โค้ดตรงนี้จะเป็นการใส่ property ของ color และ text-align ให้กับแท็ก p, h1, h3, span, tr, td พร้อมๆกัน

- การเรียกใช้ด้วยคลาส(class selector)

การเรียกใช้วิธีนี้สามารถทำให้เรากำหนดการทำงานต่างกันให้กับแท็กเดียวกันได้ เช่น

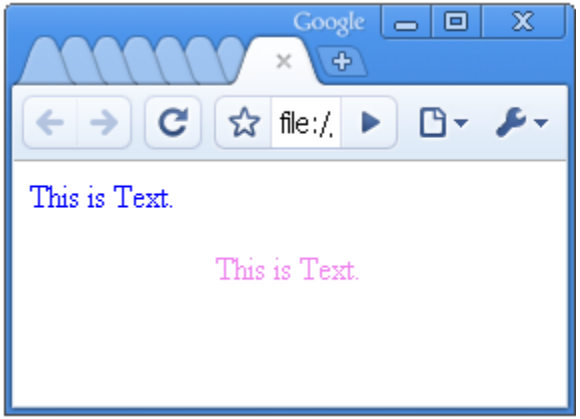
```
<p class="first_p">This is Text.</p>
<p class="second_p">This is Text.</p>
```

สังเกตุว่าแท็ก <p> ทั้งสองอันนั้นจะมีตัวบอกลักษณะหรือคลาส(class) ซึ่งมีชื่อที่ต่างกัน ดังนั้นเราสามารถเรียกใช้ได้ดังนี้

ไวยากรณ์ของ CSS : Class Selector

HTML

```
<body>
<p class="first_p">This is Text.</p>
<p class="second_p">This is Text.</p>
</body>
```



CSS

```
<style type="text/css">
p.first_p {
        color : blue;
        font-size : ;
}
p.second_p {
        color : violet;
        text-align :
        center;
}
</style>
```

ให้สังเกตในไฟล์ **CSS** ให้ดีว่ามีการเรียกใช้ **.classname** ตามหลังชื่อแท็กด้วย

ไวยากรณ์ของ CSS : Class Selector

- การใช้งาน class selector อีกแบบหนึ่ง คือการเรียกใช้จากชื่อ class โดยตรงเลย

HTML

```
<body>
<p class="class_selector">This is Text.</p>
<h1 class="class_selector">This is Text.</h1>
</body>
```

CSS

```
<style type="text/css">
.class_selector{
        color : navy;
        text-align : center;
}
</style>
```

ครานี้เป็นการเรียกใช้ **class selector** เพื่อกำหนดรูปแบบให้กับแท็กต่างกัน แต่มีคลาสชื่อเดียวกัน

การใส่รูปแบบให้กับแท็กแบบเจาะจง attribute

- การใส่รูปแบบให้กับแท็กแบบเจาะจง **attribute** สามารถทำได้โดย

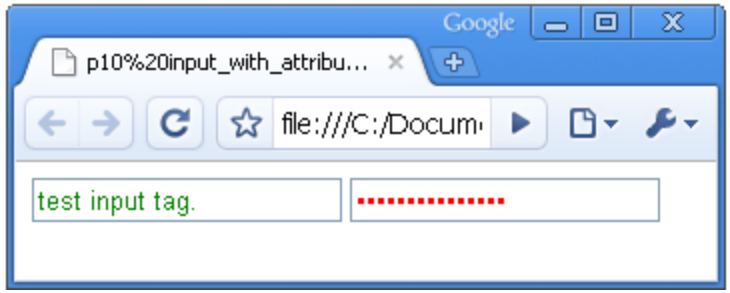
HTML

```
<body>
<input type="text" value="test input tag."/>
<input type="password" value="test input tag."/>
</body>
```

CSS

```
<style type="text/css">
input[type="text"]{
        color : green;
}
input[type="password"]{
        color : red;
}
</style>
```

การใช้งานในรูปแบบนี้ค่อนข้างยุ่งยาก และซับซ้อน กว่ามาก ส่วนมากจึงใช้เป็นการกำหนด **class** และ **id** มากกว่า



การอ้างถึงแท็กด้วยไอดี : id selector

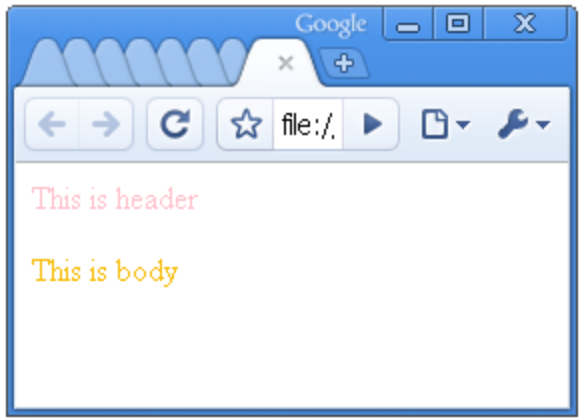
- การอ้างถึงด้วยวิธีนี้จะคล้ายกับแบบ class selector แต่จะใช้เครื่องหมาย '#' เป็นตัวอ้างอิงแทน

HTML

```
<body>
<p id="header">This is header</p>
<p id="body">This is body</p>
</body>
```

CSS

```
<style type="text/css">
#header{
        color : pink;
}
#body{
        color : #f8bb01;
}
</style>
```



*ข้อแตกต่างระหว่าง **class selector** กับ **id selector** คือ **class selector** สามารถใช้อ้างอิงได้หลายอีลีเม้นท์ ในขณะที่ **id selector** ใช้อ้างอิงได้เพียงแค่อีลีเม้นท์เดียว

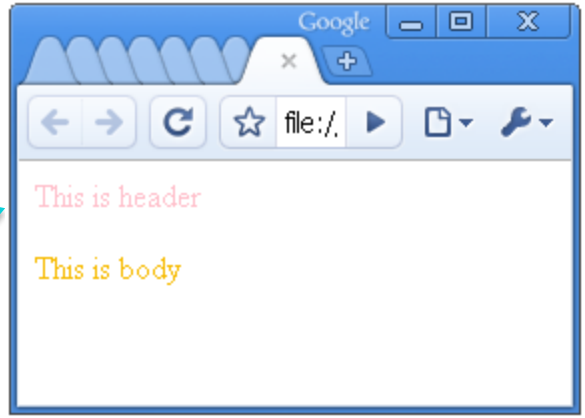
การอ้างถึง **id selector** สามารถทำได้อีกวิธีหนึ่ง ชื่อแท็ก#ชื่อไอดี เช่น **p#header{...}** จะเป็นการอ้างถึงแท็ก **p** ที่มีไอดีเป็น **'header'**

การใส่คอมเม้นท์ใน CSS

- การใส่คอมเม้นท์เรามักจะทำเพื่อเป็นการเตือนความจำ ให้รู้ว่าส่วนไหนทำงานยังไง หรือว่ายังมีปัญหาอะไรอยู่หรือปล่าว เนื่องจากในการทำงานจริงๆแล้ว มักจะมีการเขียนโค้ดมากกว่าพันบรรทัดขึ้นไป หากไม่เขียนคอมเม้นจะทำให้เสียเวลาในการแก้ไขปรับปรุง

CSS

```
<style type="text/css">
#header{
        /*comment herre*/
        /*this css is for header*/
        color : pink;
}
#body{
        color : #f8bb01;
        /*color : blue;*/
}
</style>
```



การใส่ **comment** รูปแบบนี้ใน **css** จะไม่มีผลกระทบใดๆ กับตัวโปรแกรมทั้งสิ้น

ในกรณีที่เรามีโค้ดที่ไม่ต้องการใช้งาน เราสามารถใส่คอมเม้นท์ครอบไว้เพื่อให้โค้ดส่วนนั้นไม่ถูกประมวลได้

การใช้งาน CSS

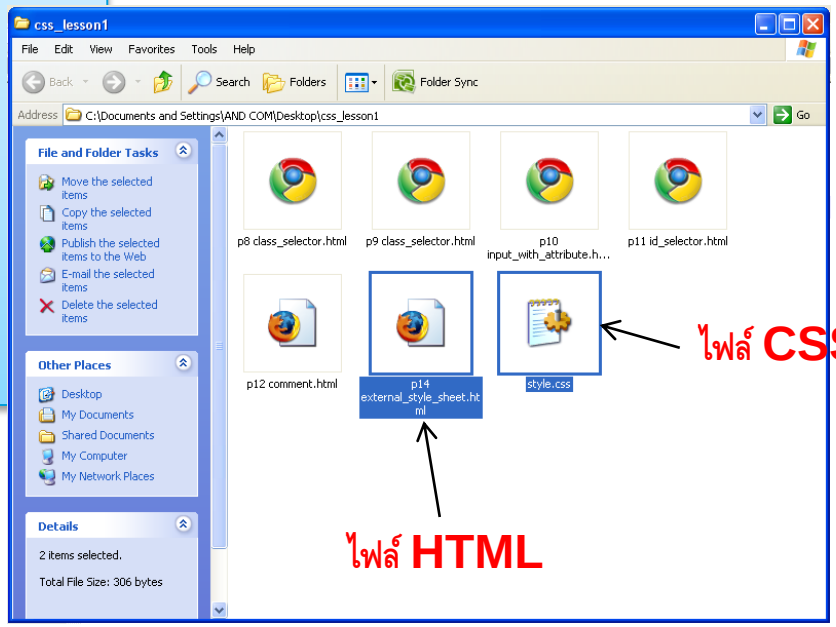
- การเรียกใช้งาน CSS สามารถทำได้ 3 วิธีด้วยกัน
  - การเรียกใช้จากไฟล์ภายนอก (external style sheet)
  - การเรียกใช้จากภายในไฟล์เดียวกัน (internal style sheet)
  - การเรียกใช้ภายในแท็ก (inline style sheet)

การเรียกใช้จากไฟล์ภายนอก (external style sheet)

- การเรียกใช้ด้วยวิธีนี้สามารถใช้ได้หลายเว็บเพจ เนื่องจากเป็นการอ้างถึงไฟล์ที่อยู่ภายนอก เรียกใช้ด้วยแท็ก <link rel="stylesheet" type="text/css" href="ชื่อไฟล์.css"/> ซึ่งประกาศไว้ภายใต้แท็ก <head>

CSS

```
/*style sheet for css example p.14 external link*/
p {
        color : #261de2;
}
p#header{
        font-size:50px;
        font-weight:bold;
}
```



ขั้นตอนแรกสร้างไฟล์ **.css** โดยในตัวอย่างนี้สร้างไว้ใน **folder** เดียวกันกับไฟล์ **HTML**

การเรียกใช้จากไฟล์ภายนอก (external style sheet)

ขั้นตอนที่สอง เรียกใช้ไฟล์ .css โดยอ้างถึงผ่านแท็ก <link>

HTML

```
<head>
        <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
<body>
        <p id="header">This is header</p>
        <p id="body">This is body</p>
</body>
```

จะได้ผลดังนี้

การเรียกใช้จากภายในไฟล์เดียวกัน (internal style sheet)

- การประกาศแบบนี้จะใช้ในกรณีที่ต้องการประกาศ CSS ใช้เฉพาะไฟล์นั้นๆ โดยไม่ต้องการให้ไฟล์อื่นมาเรียกใช้ได้ โดยจะประกาศเป็นแท็ก <style> ไว้ภายใต้แท็ก <head>

```
<head>
<style type="text/css">
p {
        color : #261de2;
}
p#header{
        font-size:50px;
        font-weight:bold;
}
</style>
</head>
<body>
        <p  id="header">This is header</p>
        <p  id="body">This is body</p>
</body>
```

ส่วนโค้ด CSS

ส่วนโค้ด HTML

*สังเกตว่าการประกาศทั้งสองแบบให้ผลลัพธ์เหมือนกัน เพียงแต่ต่างกันตรงที่วิธีการอ้างอิงเท่านั้น

การเรียกใช้จากภายในไฟล์เดียวกัน (internal style sheet)

ข้อสังเกต

การประกาศแท็ก **<style>** ไว้ในแท็ก **<head>** จะใช้รูปแบบนี้เสมอ

```
<style  type="text/css">
p{
        color : red;
}
</style>
```

แต่ภายในไฟล์ **CSS** ที่ประกาศแยกแบบ **external style sheet** จะไม่ต้องมีแท็ก **<stlye>**

```
p{
        color : red;
}
```

การเรียกใช้ภายในแท็ก (inline style sheet)

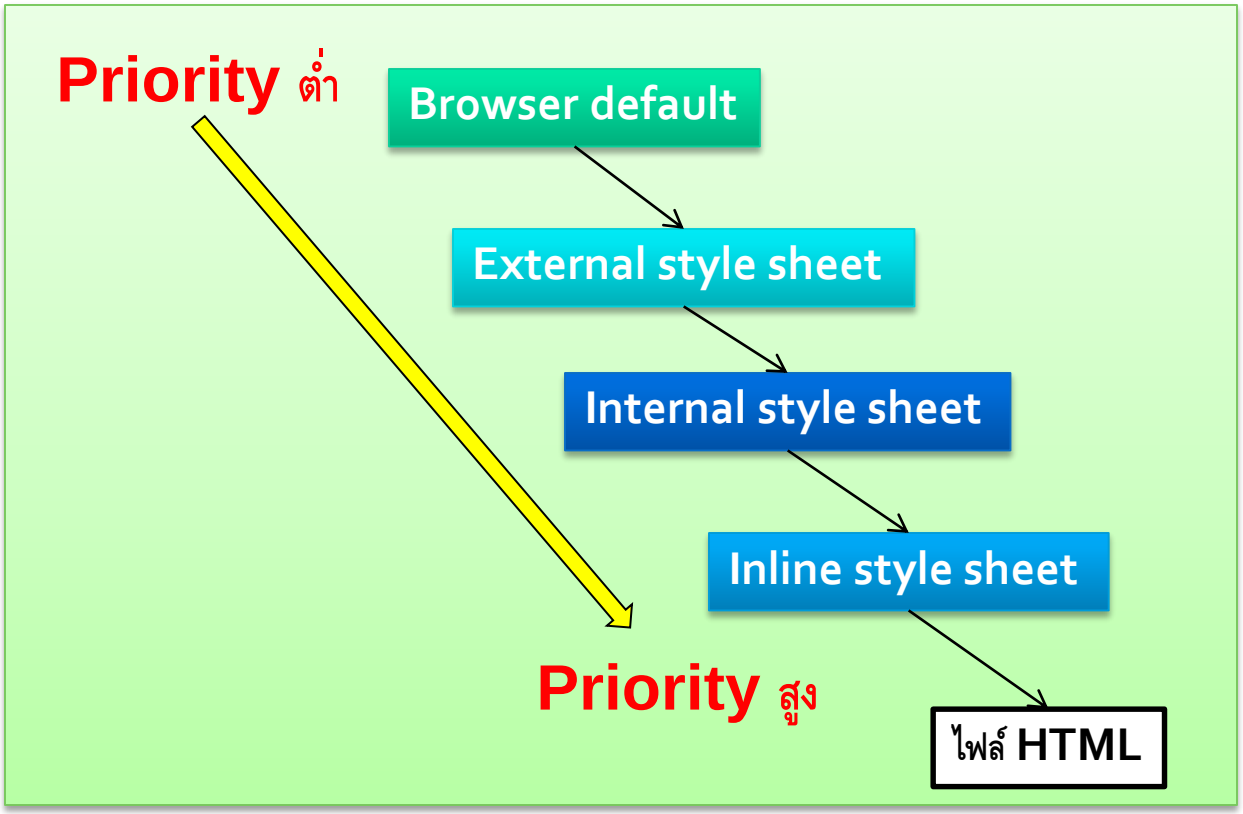
- วิธีการเรียกใช้แบบนี้ไม่แนะนำให้ใช้ถ้าไม่จำเป็น เนื่องจากจะทำให้สูญเสียคุณสมบัติสำคัญ คือ คุณสมบัติในการกำหนดลักษณะแบบทั่วถึง (global) เพราะวิธีนี้จะมีผลเฉพาะกับแท็กที่เรียกใช้เท่านั้น

```
<body>
        <p id="header" style="color:#red;font-size:50px;" >This is header</p>
        <p id="body">This is body</p>
</body>
```

สังเกตว่าแท็ก **<p>** อันที่สองนั้นจะไม่ได้รับผลของการกำหนดลักษณะในแท็ก **<p>** แรกเลย

การเรียกใช้ style sheet หลาย ๆแบบพร้อมกัน

- การเรียกใช้ **style sheet** นั้นสามารถเรียกจากวิธีต่างๆ ได้พร้อมกัน เพียงแต่จะมีลำดับความสำคัญต่างกันไป ดังนี้



ถ้าหากมีการเรียกใช้ style sheet จากทั้งสามวิธีพร้อมๆ กัน selector ใน external style sheet จะถูกแทนที่ด้วย selector ที่มีการเรียกใช้ซ้ำใน internal style sheet และเช่นกัน selector ใน internal style sheet ก็จะถูกแทนที่ด้วย selector ที่มีการเรียกใช้ซ้ำใน inline style sheet

การเรียกใช้ style sheet หลาย ๆ แบบพร้อมกัน

◆ ตัวอย่าง

HTML

```
<head>
    <link rel="stylesheet" type="text/css" href="style.css"/>
    <style type="text/css">
            p#body{
                color:green;
                font-size:30px;
            }
    </style>
</head>
<body>
    <p id="header" style="color:#red;font-size:50px;" >This is header</p>
    <p id="body">This is body</p>
    <p style="font-style:italic;font-size:30px;">This is p-tag</p>
    <p>This is p-tag</p>
</body>
```

External style sheet

Internal style sheet

Inline style sheet

การเรียกใช้ style sheet หลาย ๆแบบพร้อมกัน

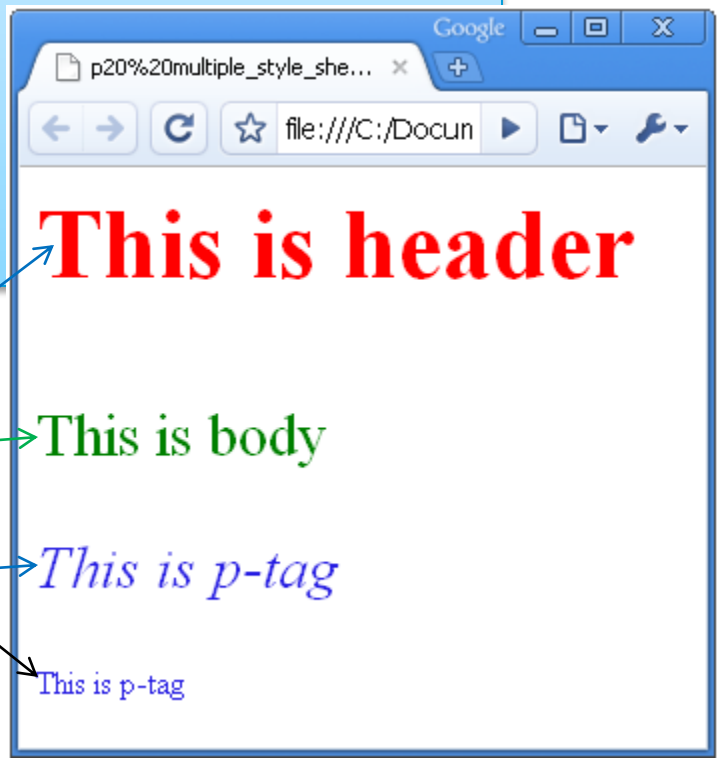
CSS

```
/*style sheet for css example p.14 external link*/
p {
        color : #261de2;
}
p#header{
        font-size:50px;
        font-weight:bold;
}
```

มาจาก **external style sheet**

มาจาก **internal style sheet**

มาจาก **inline style sheet**



จัดการพื้นหลังด้วย CSS

Style sheet สามารถใช้กำหนดคุณสมบัติของพื้นหลังได้ทุกอีลีเม้นท์

Properties ที่ใช้ทำงานกับพื้นหลัง

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

Property background-* อาจใช้กำหนดสีพื้นหลังของตาราง<table> ของช่องตาราง<td> ของแท็ก <p> หรือแท็กใดๆก็ได้ที่มีการแสดงบนหน้าเว็บเพจ

Background-color

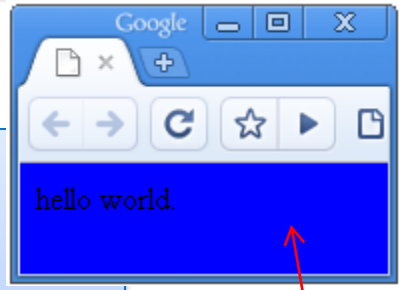
CSS

ใช้สำหรับกำหนดสีของพื้นหลัง เช่น

```
body{
background-color:blue;
}
```

HTML

```
<body>
        <p>hello world.</p>
</body>
```



พื้นหลังของทั้ง body จะเป็นสีน้ำเงิน

จัดการพื้นหลังด้วย CSS

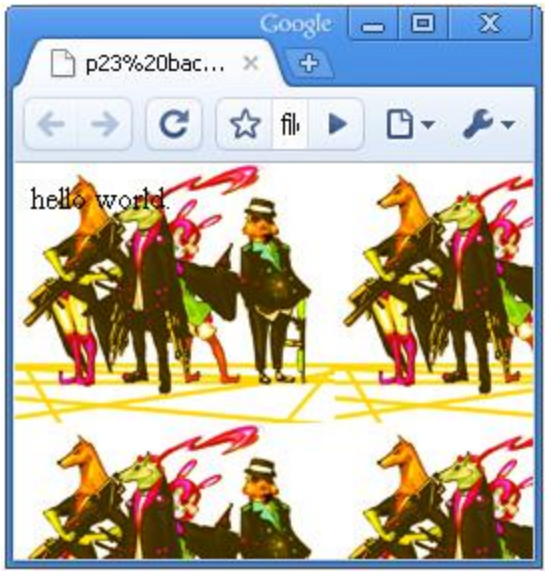
Background-image

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
body{
background-
image:url(band.jpg);
}
```



*การเลือกสีและรูปภาพมาเป็นพื้นหลังควรระวังให้ดีในเรื่องของข้อความที่จะแสดง ดูในตัวอย่างนี้ที่ภาพสีสดจนเกินไปทำให้อ่านข้อความได้ยาก

จัดการพื้นหลังด้วย CSS

Background-repeat

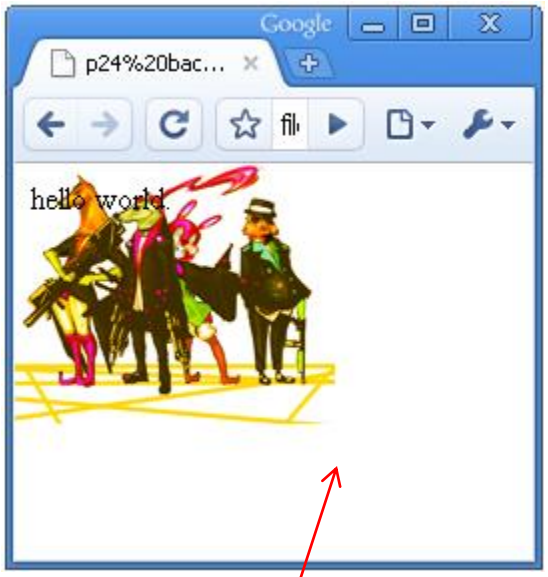
ใช้สำหรับควบคุมการทำซ้ำ(repeat) ของพื้นหลัง ส่วนมากใช้คู่กับรูปภาพ

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
body{
background-image:url(band.jpg);
background-repeat:no-repeat;
}
```



ไม่มีการทำซ้ำของรูปภาพพื้นหลังแล้ว

จัดการพื้นหลังด้วย CSS

ค่าที่เป็นไปได้ของ property 'background-repeat'

ค่าที่เป็นได้	ลักษณะ
Repeat	ทำซ้ำทั้งแนวตั้งและแนวนอน
Repeat-x	ทำซ้ำเฉพาะแนวนอน
Repeat-y	ทำซ้ำเฉพาะแนวตั้ง
No-repeat	ไม่ทำซ้ำเลย
inherit	ทำซ้ำโดยอ้างอิงจากอีลีเม้นท์ที่เป็น parent

จัดการพื้นหลังด้วย CSS

Background-position

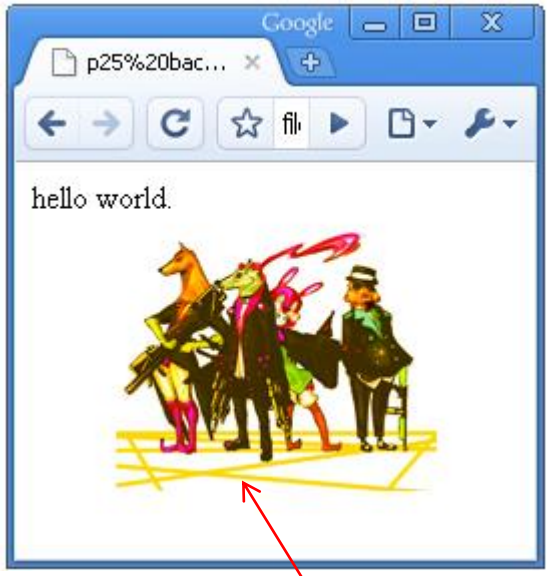
กำหนดตำแหน่งของภาพพื้นหลัง ในกรณีที่ภาพพื้นหลังไม่ได้ทำซ้ำ

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
body{
background-image:url(band.jpg);
background-repeat:no-repeat;
background-position:center;
}
```



ภาพพื้นหลังเลื่อนมาอยู่ตรงกลางแล้ว

จัดการพื้นหลังด้วย CSS

ค่าที่เป็นไปได้ของ property 'background-position'

ค่าที่เป็นได้	ลักษณะ
Top	วางรูปไว้บนสุด
Center	วางรูปไว้ตรงกลาง
Bottom	วางรูปไว้ล่างสุด
Left	วางรูปไว้ชิดขอบซ้าย
Right	วางรูปไว้ชิดขอบขวา
Inherit	วางรูปตามคุณสมบัติของอีลีเม้นท์ parent
X, Y	วางรูปแบบกำหนดระยะห่างตายตัว โดยนับจากมุมซ้ายบนไป x(หน่วยวัด,%) y(หน่วยวัด,%)

ค่าของ property 'background-position' นั้นสามารถใส่ได้สองค่าพร้อมกันเพื่อให้ได้ตำแหน่งที่แม่นยำมากขึ้น เช่น
background-position: top center; คือให้วางรูปไว้ตรงกลางชิดขอบบน
background-position: 50px 100px; คือให้วางรูปไว้ห่างจากขอบบน 50px และห่างจากขอบซ้าย 100px

จัดการพื้นหลังด้วย CSS

Background-attachment

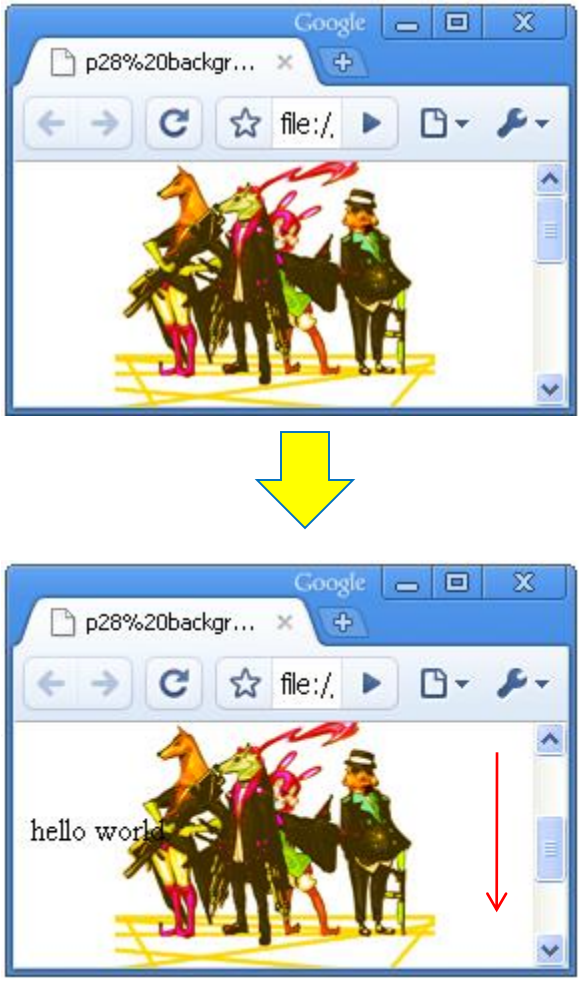
กำหนดภาพพื้นหลังเวลาขยับแถบเลื่อน (scroll bar)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
body{
background-image:url(band.jpg);
background-repeat:no-repeat;
background-position:center;
background-attachment:fixed;
}
```



จัดการพื้นหลังด้วย CSS

ค่าที่เป็นไปได้ของ property 'background-attachment'

ค่าที่เป็นได้	ลักษณะ
Fixed	พื้นหลังไม่ขยับ
Scroll	พื้นหลังเลื่อนตามแถบเลื่อน (scroll bar)
Inherit	พื้นหลังเลื่อนตามลักษณะของ parent

การกำหนดพื้นหลังแบบรวม (background)

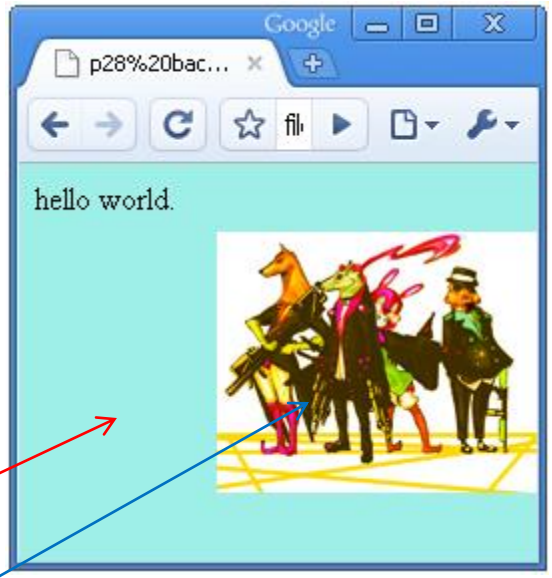
- เราจะใช้ property 'background' ในการกำหนดพื้นหลังเพียงตัวเดียว เนื่องจาก property นี้สามารถจัดการพื้นหลังได้เหมือนทุกอย่างที่กล่าวมาในข้างต้น

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
body{
background: #9defe8      กำหนดสี
        url(band.jpg)    ใส่รูปภาพพื้นหลัง
        center right     กำหนดตำแหน่งรูปภาพพื้นหลัง
        no-repeat ;      กำหนดไม่ให้รูปทำซ้ำ
}
```



สังเกตว่าแต่ละค่าใน property นี้จะไม่มีเครื่องหมาย (,) หรือ (;) มาคั่น

จัดการข้อความด้วย CSS

- เราสามารถใช้ CSS ในการกำหนดลักษณะของข้อความที่แสดงบนหน้าเว็บเพจได้ เช่น กำหนดสีของตัวอักษร กำหนดขนาด กำหนดตัวเข้ม หรือตัวเอียง

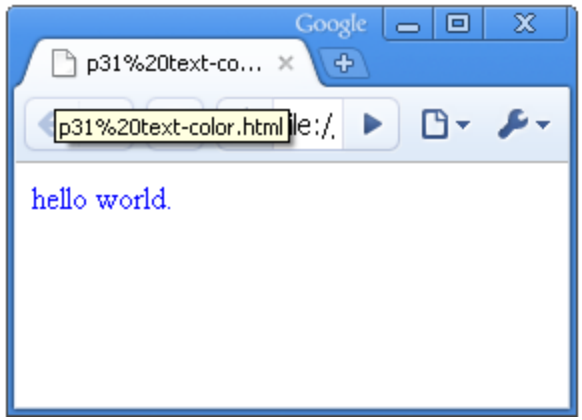
- กำหนดสีของตัวอักษร (color)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        color:blue;
}
```



จัดการข้อความด้วย CSS

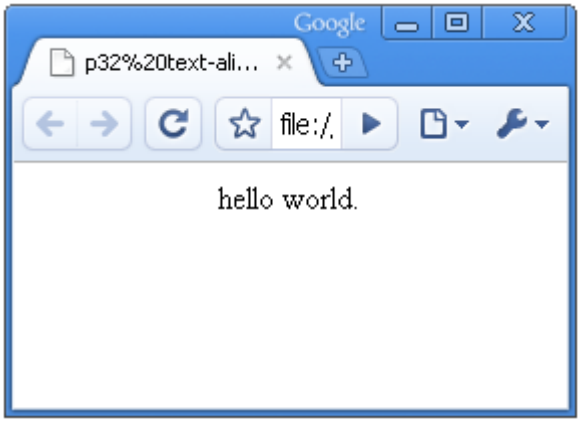
กำหนดตำแหน่งข้อความ (text-align)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        text-align:center;
}
```



ค่าที่เป็นได้	ลักษณะ
Left	วางข้อความชิดขอบซ้าย
Right	วางข้อความชิดขอบขวา
Center	วางข้อความไว้ตรงกลาง
Justify	วางข้อความให้ชิดทั้งขอบซ้ายและขวา เหมือนกับข้อความในคอลัมน์หนังสือพิมพ์

จัดการข้อความด้วย CSS

ตกแต่งข้อความ (text-decoration)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        text-decoration:underline;
}
```

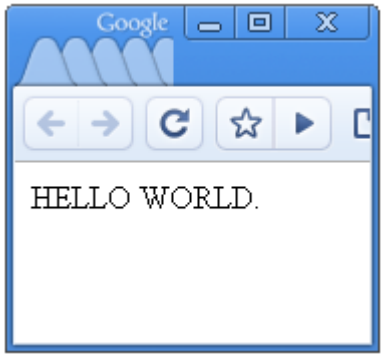
ค่าที่เป็นได้	ลักษณะ
Underline	ขีดเส้นใต้ข้อความ
Overline	ขีดเส้นเหนือข้อความ
Line-through	ขีดเส้นผ่ากลางข้อความ
Blink	ข้อความกระพริบ *ใช้ได้ใน firefox เท่านั้น
None	ไม่มีการตกแต่งข้อความ มักใช้กับลิ้งค์ที่ไม่ต้องการให้แสดงเส้นใต้ข้อความ

จัดการข้อความด้วย CSS

กำหนดตัวพิมพ์ใหญ่-พิมพ์เล็ก (text-transform)

HTML

```
<body>
        <p>hello world.</p>
</body>
```



CSS

```
p{
        text-transform:uppercase;
}
```

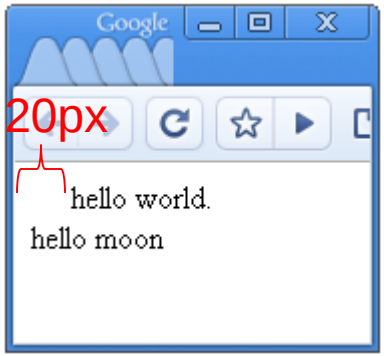
ค่าที่เป็นได้	ลักษณะ
Uppercase	ทำให้ข้อความเป็นตัวพิมพ์ใหญ่ทั้งหมด
Lowercase	ทำให้ข้อความเป็นตัวพิมพ์เล็กทั้งหมด
Capitalize	ทำให้ตัวอักษรตัวแรกเป็นตัวพิมพ์ใหญ่
None	คงค่าตามข้อมูลเดิม

จัดการข้อความด้วย CSS

การเว้นระยะย่อหน้า (text-indent)

HTML

```
<body>
        <p>hello world.<br>
        hello moon</p>
</body>
```



CSS

```
p{
        text-indent:20px;
}
```

ผลของ **property 'text-indent'** จะมีผลกับข้อความบรรทัดแรกของแท็กเท่านั้น โดยจะเว้นช่องว่างของย่อหน้าให้เท่ากับค่าที่เรากำหนด ดังนั้นเมื่อขึ้นบรรทัดใหม่ ข้อความจะกลับไปอยู่ชิดขอบซ้ายเหมือนเดิม

Properties อื่นๆ ที่ใช้จัดการกับข้อความ

property	ลักษณะ	ค่าที่เป็นได้
Direction	กำหนดทิศทางการแสดงข้อความ	ltr Rtl
Line-height	กำหนดระยะห่างของแต่ละบรรทัด	Normal ความยาวเป็นหน่วยวัด %
Letter-spacing	กำหนดระยะห่างของแต่ละตัวอักษร	Normal ความยาวเป็นหน่วยวัด
Vertical-align	จัดตำแหน่งของจ้อความในแนวตั้ง	baseline sub super top text-top middle bottom text-bottom ความยาวเป็นหน่วยวัด %

Properties อื่นๆ ที่ใช้จัดการกับข้อความ

property	ลักษณะ	ค่าที่เป็นได้
White-space	กำหนดว่าจะจัดการกับช่องว่างในแท็กอย่างไร	normal pre nowrap
Word-spacing	กำหนดระยะห่างระหว่างข้อความ	normal ความยาวเป็นหน่วยวัด

จัดการตัวอักษรด้วย CSS

- การจัดการตัวอักษรด้วย CSS นี้จะต่างกับการจัดการข้อความ เพราะเป็นการกำหนดลักษณะพิเศษของตัวอักษรแต่ละตัว เช่น กำหนดขนาดตัวอักษร กำหนดตัวเอียง-ตัวหนา

- กำหนดแบบอักษร (font-family)

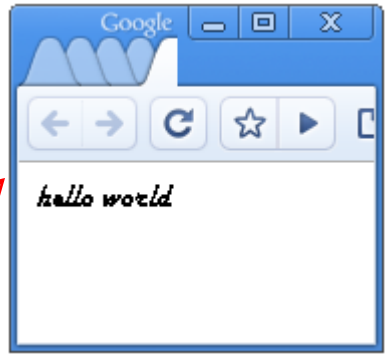
HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
font-family:"Harlow Solid Italic",Georgia,Serif;
}
```

*หากชื่อของแบบอักษรมีช่องว่างให้ใช้เครื่องหมาย (“) คั่นหัว-ท้าย



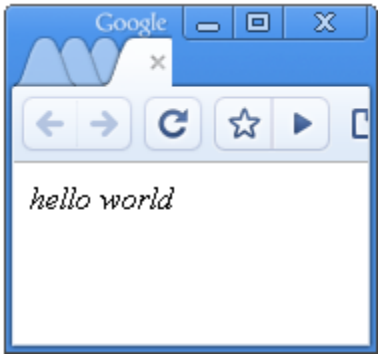
ค่าของตัวอักษรใน property ‘font-family’ นั้น เราสามารถใส่ได้มากกว่าหนึ่งค่า โดยเมื่อผู้ใช้เปิดหน้าเว็บเพจขึ้นมา บราวเซอร์จะดึงค่าของตัวอักษรแบบแรกมาใช้ ถ้าหาไม่เจอก็จะไปดึงค่าของแบบอักษรตัวต่อไปตามลำดับ หรือใช้แบบอักษรปรกติ(default) ของบราวเซอร์ในกรณีที่หาแบบอักษรไม่เจอเลย

จัดการตัวอักษรด้วย CSS

กำหนดสไตล์ของตัวอักษร (font-style)

HTML

```
<body>
        <p>hello world.</p>
</body>
```



CSS

```
p{
        font-style:italic;
}
```

ค่าที่เป็นได้	ลักษณะ
Normal	แบบอักษรธรรมดา
Italic	ตัวอักษรเอียง
Oblique	ตัวอักษรเอียง

จัดการตัวอักษรด้วย CSS

กำหนดขนาดของตัวอักษร (font-size)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        font-size:xx-large;
}
```

ค่าที่เป็นไปได้ของ property font-size

ค่าที่เป็นได้	ลักษณะ
xx-small	เล็กมากมาก
x-small	เล็กมาก
small	เล็ก
medium	ปานกลาง
large	ใหญ่
x-large	ใหญ่มาก
xx-large	ใหญ่มากมาก
smaller	ทำให้ตัวอักษรเล็กลงโดยเทียบจากขนาดอักษรของ parent
larger	ทำให้ตัวอักษรใหญ่ขึ้นโดยเทียบจากขนาดอักษรของ parent
หน่วยวัดความยาว	กำหนดให้ตัวอักษรมีขนาดเท่ากับค่าที่กำหนด(px,em,cm,...)
%	กำหนดขนาดเป็นเปอร์เซ็นต์โดยเทียบจากขนาดเดิมของตัวอักษร
inherit	ตัวอักษรมีขนาดเท่ากับตัวอักษรของ parent

Property อื่นๆ ที่ใช้จัดการกับตัวอักษร

property	ลักษณะ	ค่าที่เป็นได้
Font	กำหนดค่าของตัวอักษรทั้งหมดใน property เดียว เช่น Font : italic bold 50px arial,san-serif;	*font-style* *font-variant* *font-weight* *font-size/line-height* *font-family* caption icon menu message-box small-caption status-bar inherit
Font-variant	กำหนดว่าจะแสดงตัวอักษรเป็นลักษณะ small-cap รึเปล่า	normal small-caps inherit
Font-weight	กำหนดความหนาของตัวอักษร	normal bold bolder lighter 100-900 inherit

โครงสร้างของ box model

- แนวคิดของ box model ใน CSS นั้นจะมองอีลีเม้นท์เป็นกล่องกล่องนึง ซึ่งประกอบไปด้วยกล่องเล็กๆหลายชั้น ดังรูป

Margin – เป็นพื้นที่ที่อยู่รอบนอกของเส้นขอบ(border) ไม่มีพื้นหลัง(background) ของตนเอง และมีลักษณะโปร่งใส
Border – เป็นเส้นขอบของแต่ละอีลีเม้นท์ ปรกติจะไม่แสดงบนหน้าเว็บเพจ ยกเว้นผู้ใช้กำหนดเอง
Paddin – เป็นพื้นที่ที่อยู่ระหว่างเนื้อหาของอีลีเม้นท์(เช่น ข้อความต่างๆ) กับเส้นขอบของอีลีเม้นท์
Content – เป็นพื้นที่ส่วนที่เราใช้แสดงข้อมูลที่เราต้องการขึ้นบนหน้าเว็บเพจ

ข้อควรระวังของการใช้ box model

- **ขนาดของอีลีเม้น**

  บางครั้งที่เรามีการกำหนดขนาดของอีลีเม้นท์เพื่อจัดวางรูปแบบให้สวยงาม เช่น เรากำหนดให้อีลีเม้นท์มีความกว้างเท่ากับ 250px เราจึงควรระวังไม่ให้ขนาดที่เรากำหนดยาวกว่าขนาดที่เป็นได้ โดยคิดจาก ความกว้างของ margin ซ้าย-ขวา + ความกว้างของ border ซ้าย-ขวา + ความกว้างของ padding ซ้าย-ขวา + ความกว้างของ content ให้ทั้งหมดรวมกันไม่เกินค่าที่เรากำหนด(ในที่นี้คือ 250px)

- **ความเข้ากันได้ของบราวเซอร์(browser compatibility)**

  การกำหนดค่าความกว้างบางบราวเซอร์อาจแสดงผลไม่ถูกต้อง เช่น กว้างกว่าหรือแคบกว่าค่าที่เรากำหนด แต่เราสามารถแก้ไขข้อผิดพลาดนี้ได้ด้วยการประกาศ DOCTYPE ไว้ข้างบนของแท็ก <html>

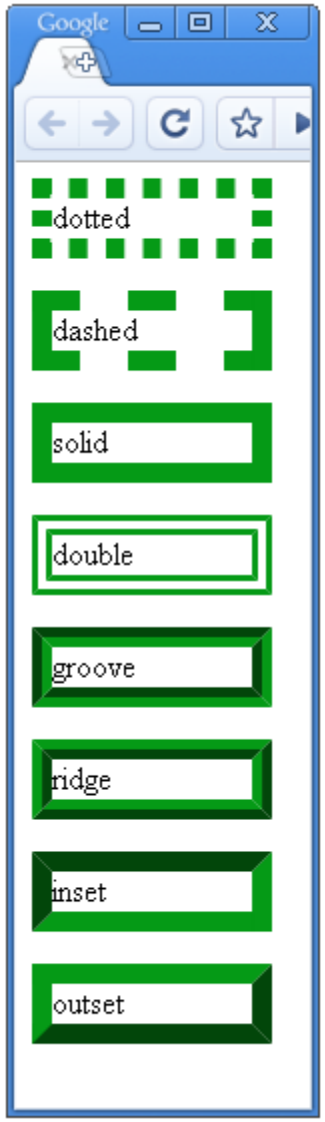
```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

  DOCTYPE ไม่ใช่แท็กของภาษา HTML แต่เป็นการประกาศให้บราวเซอร์ของเรารู้ว่าเว็บเพจของเราใช้ markup language แบบไหน เพื่อให้บราวเซอร์ประมวลผลได้อย่างถูกต้อง

การจัดการเส้นขอบ (border)

กำหนดรูปแบบของเส้นขอบ (border-style)

```
<body>
        <p style="border-style:dotted;">dotted</p>
        <p style="border-style:dashed;">dashed</p>
        <p style="border-style:solid;">solid</p>
        <p style="border-style:double;">double</p>
        <p style="border-style:groove;">groove</p>
        <p style="border-style:ridge;">ridge</p>
        <p style="border-style:inset;">inset</p>
        <p style="border-style:outset;">outset</p>
</body>
```



*ค่าของ property บางค่าจำเป็นต้องใช้ร่วมกับค่าอื่นๆด้วย เช่น double ต้องกำหนดความหนาของเส้นขอบ(border-width)ด้วย ไม่อย่างนั้นเส้นขอบทั้งสองเส้นจะซ้อนทับกัน หรือ groove ที่ต้องกำหนดสีของเส้นขอบ(border-color) ไม่อย่างนั้นเราจะมองไม่เห็นเส้นขอบ

การจัดการเส้นขอบ (border)

กำหนดความหนาของเส้นขอบ(border-width)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        border-style:solid;
        border-width: 10px;
}
```

*ค่าของ **property 'border-width'** เองก็ต้องกำหนดค่าของ **'border-style'** ด้วย ไม่เช่นนั้นจะมองไม่เห็นเส้นขอบ

การจัดการเส้นขอบ (border)

กำหนดสีของเส้นขอบ (border-color)

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        border-style:solid;
        border-color: green;
}
```

การจัดการเส้นขอบ (border)

การกำหนดเส้นขอบแบบหลากหลาย (border-individual sides)

เส้นขอบของอีลีเม้นท์สามารถกำหนดแยกแต่ละด้านได้

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        border-top-style:dashed;
        border-right-style:solid;
        border-bottom-style:double;
        border-left-style:dotted;
}
```

หรือเราสามารถกำหนดไปในบรรทัดเดียวเลยก็ได้

บน ขวา ล่าง ซ้าย

```
border-style : dotted solid double dashed;
```

บน ขวา+ซ้าย ล่าง

```
border-style:dotted solid double;
```

บน+ล่าง ขวา+ซ้าย

```
border-style:dotted solid;
```

Property อื่นๆ ของเส้นขอบ

นอกจากที่กล่าวมาในข้างต้นแล้วเรายังสามารถกำหนดลักษณะของเส้นขอบด้วยวิธีอื่นได้อีกเช่น

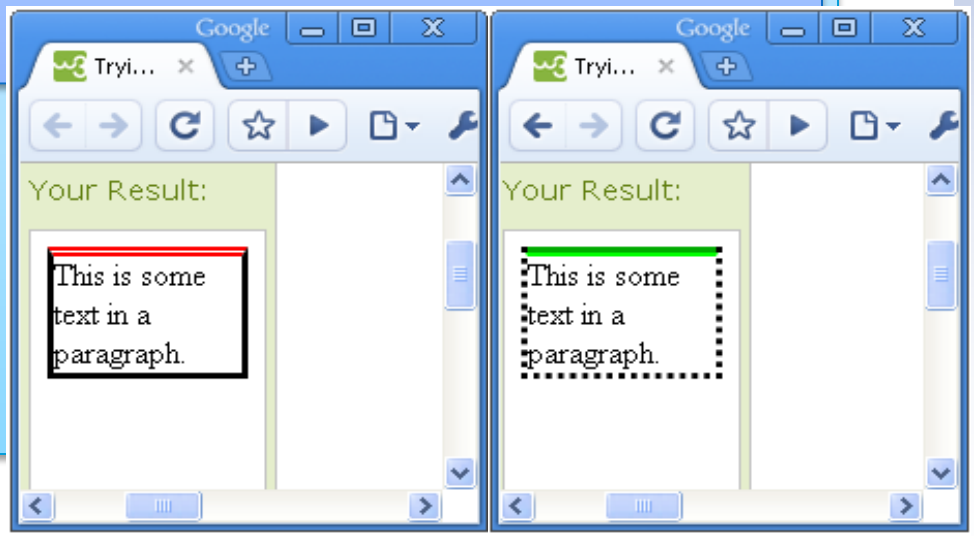
การประกาศแบบสั้น

```
border: 5px solid red;
```

การกำหนดสีหรือความกว้างเฉพาะด้าน

```
border-bottom-color: red;
border-left-width:10px
```

ทั้งนี้ก็ขึ้นอยู่กับความต้องการของผู้ใช้ว่าต้องการแบบไหน



กำหนดเส้นรอบนอกอีลีเม้นท์

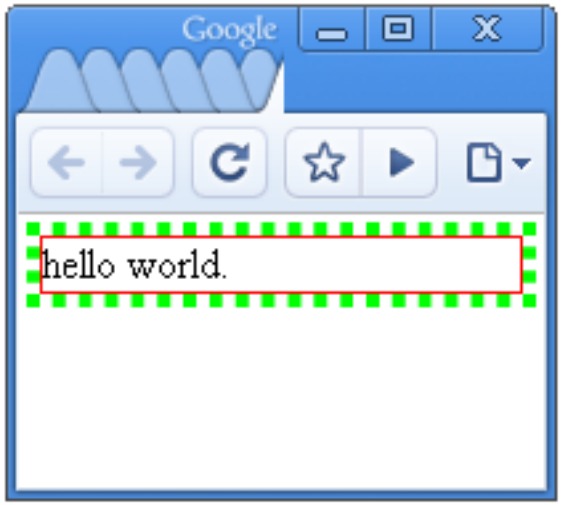
- การกำหนดเส้นรอบนอก (outline) จะคล้ายกับการกำหนดเส้นขอบ (border) ต่างกันแค่ไม่สามารถกำหนดลักษณะเฉพาะด้านได้

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        border:red solid thin;
        outline:#00ff00 dotted thick;
}
```



Property ขอเส้นรอบนอก (outline)

Property	ลักษณะ	ค่าที่เป็นได้
Outline	กำหนดลักษณะต่างๆ ของเส้นรอบนอกแบบสั้น	*outline-color* *outline-style* *outline-width* inherit
Outline-color	กำหนดสีของเส้นรอบนอก	*color_name* *hex_number* *rgb_number* invert inherit

Property ขอเส้นรอบนอก (outline)

Property	ลักษณะ	ค่าที่เป็นได้
Outline-style	กำหนดแบบเส้นของเส้นรอบนอก	none dotted dashed solid double groove ridge inset outset inherit
Outline-width	กำหนดความหนาของเส้นรอบนอก	thin medium thick *ค่าหน่วยวัดความยาว (px,em,cm,...)* inherit

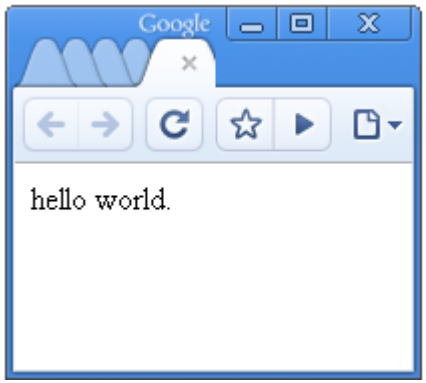
กำหนดพื้นที่นอกเส้นขอบ (margin)

HTML

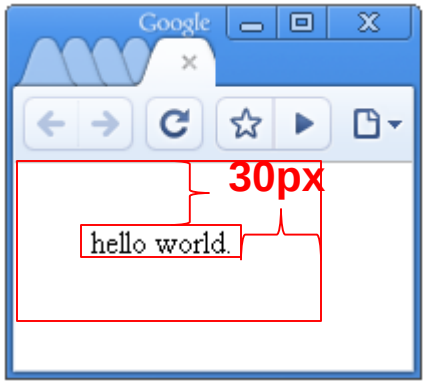
```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        margin : 30px;
}
```



แบบไม่มี **margin**



แบบกำหนด **margin**

การกำหนด **margin** นั้นสามารถกำหนดลักษณะเฉพาะของแต่ละด้านได้ด้วย

- **margin-top:100px;**
- **margin-bottom:100px;**
- **margin-right:50px;**
- **margin-left:50px;**

หรือจะเป็นการประกาศแบบสั้นก็ทำได้เช่นกัน

บน ขวา ล่าง

margin:25px 50px 75px 100px;

ซ้าย

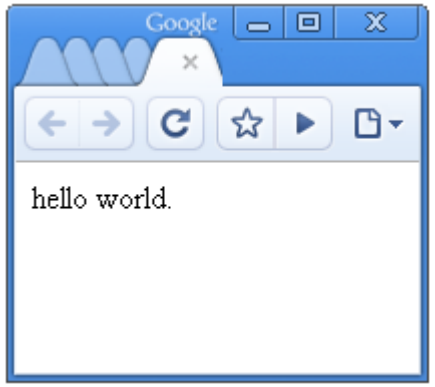
กำหนดพื้นที่ระหว่างเส้นขอบกับข้อมูล

HTML

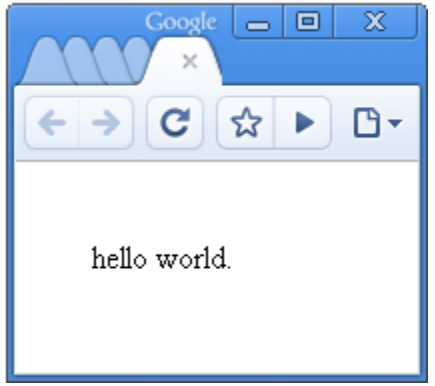
```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
p{
        padding : 30px;
}
```



แบบไม่มี **padding**



แบบกำหนด **padding**

Padding จะคล้ายกับ **margin** มาก สามารถกำหนดลักษณะเฉพาะด้านได้

- **padding-top:25px;**
- **padding-bottom:25px;**
- **padding-right:50px;**
- **padding-left:50px;**

และยังสามารถกำหนดแบบสั้นได้เช่นกัน

บน-ล่าง ขวา-ซ้าย

padding : 25px 50px;

ตกแต่งลิสต์ด้วย CSS (list)

- สำหรับลิสต์นั้น เราสามารถใช้ CSS ในการกำหนดได้ทั้ง ลิสต์แบบไม่เรียงลำดับ (unordered-list) และ ลิสต์แบบเรียงลำดับ (ordered-list)
- การกำหนดสัญลักษณ์หน้าลิสต์จะใช้คำสั่งดังนี้

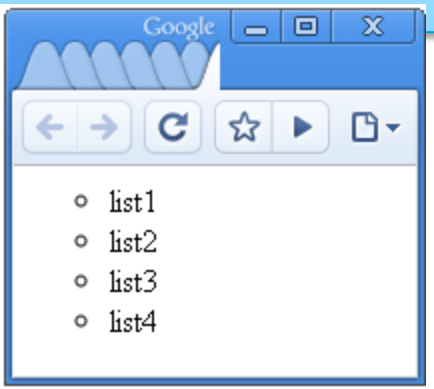
list-style-type : circle; กำหนดให้แสดงสัญลักษณ์เป็นวงกลม

HTML

```
<body>
<ul>
        <li>list1</li>
        <li>list2</li>
        <li>list3</li>
        <li>list4</li>
</ul>
</body>
```

CSS

```
li{
        list-style-type : circle;
}
```



สัญลักษณ์ต่างๆ ของลิสต์แบบไม่เรียงลำดับ (unordered-list)

ค่าที่เป็นได้	ลักษณะ
None	ไม่มีสัญลักษณ์
Disc	*ค่า default* วงกลมทึบ
Circle	วงกลมโปร่ง
Square	สี่เหลี่ยมทึบ

สัญลักษณ์ต่างๆ ของลิสต์แบบเรียงลำดับ (ordered-list)

ค่าที่เป็นได้	ลักษณะ
None	ไม่มีสัญลักษณ์
Disc	*ค่า default* วงกลมทึบ
Circle	วงกลมโปร่ง
Square	สี่เหลี่ยมทึบ
Armenian	ตัวเลข armenian*
Decimal	ตัวเลข
Decimal-leading-zero	ตัวเลขแบบมี ‘0’ นำหน้า เช่น 01, 02, 03, ...
Georgian	ตัวเลข Georgian เช่น an, ban, gan, ...
Lower-alpha	ตัวอักษร alpha ตัวพิมพ์เล็ก เช่น a,b,c,d,...
Lower-greek	ตัวอักษรกรีก เช่น alpha, beta, gamma
Lower-latin	ตัวอักษรลาตินตัวพิมพ์เล็ก เช่น a,b,c,d,...
Lower-roman	ตัวเลขโรมันตัวพิมพ์เล็ก เช่น i, ii, iii, iv, ...

สัญลักษณ์ต่างๆ ของลิสต์แบบเรียงลำดับ (ordered-list)

ค่าที่เป็นได้	ลักษณะ
Upper-alpha	ตัวอักษร alpha ตัวพิมพ์ใหญ่ เช่น A,B,C,D,...
Upper-latin	ตัวอักษรลาตินตัวพิมพ์ใหญ่ เช่น A,B,C,D,...
Upper-roman	ตัวเลขโรมันตัวพิมพ์ใหญ่ เช่น I,II,III,IV ...

1	Ա
2	Բ
3	Գ
4	Դ
5	Ե
6	Զ
7	Է
8	Ը
9	Թ

*ตัวเลข **armenian**

ตกแต่งลิสต์ด้วย CSS (list)

วางตำแหน่งของลิสต์

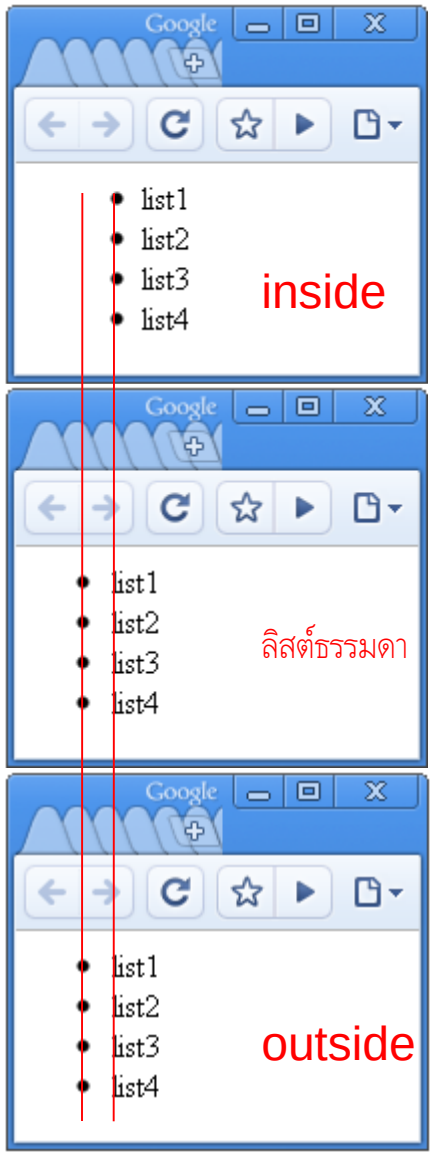
HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
ul li{
        list-style-position : outside;
}
```

*outside จะเป็นค่าปรกติ(default) ของลิสต์อยู่แล้ว

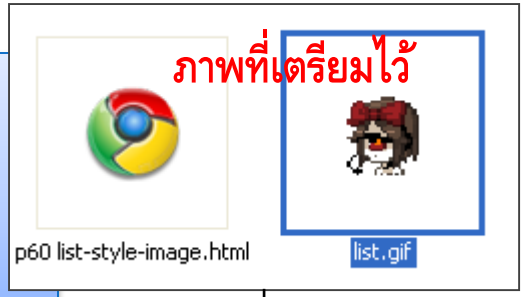


ตกแต่งลิสต์ด้วย CSS (list)

ใช้ภาพแทนสัญลักษณ์(list-style-image)

HTML

```
<body>
        <p>hello world.</p>
</body>
```



ผลที่ได้

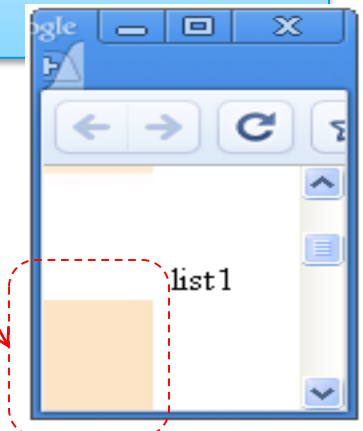
CSS

```
ul li{
        list-style-image:url('list.gif');
}
```

*ข้อควรระวัง

หากรูปภาพที่นำมาใส่ใหญ่เกินไป ภาพอาจจะหลุดขอบ ทำให้มองไม่เห็นได้

ภาพใหญ่เกินไป ทำให้หลุดออกนอกกรอบทางด้านซ้าย



ตกแต่งลิสต์ด้วย CSS (list)

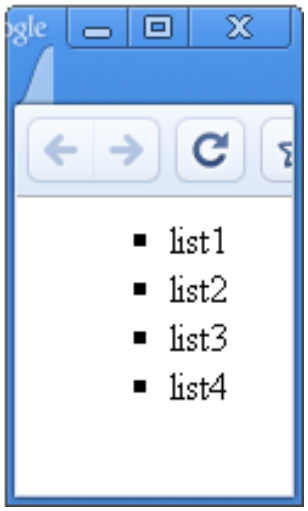
การประกาศลักษณะของลิสต์แบบสั้น

HTML

```
<body>
        <p>hello world.</p>
</body>
```

CSS

```
li{
        list-style:square inside;
}
```



ผลลัพธ์ที่ได้จะเหมือนกันการประกาศใช้

```
li {
list-style-type : square;
list-style-position :
inside;
}
```

ตกแต่งตารางด้วย CSS(table)

- CSS สามารถใช้ตกแต่งตารางได้ เช่นการปรับเส้นขอบตาราง ปรับขนาดช่องว่างในตาราง หรือเลือกตำแหน่งของข้อความกำกับตาราง(caption)

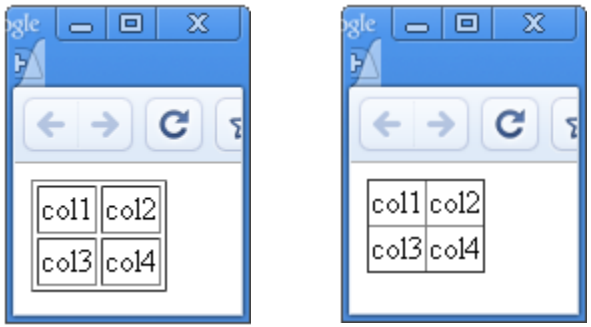
- รวมเส้นขอบตาราง (border-collapse)

HTML

```
<table border="1px">
<tr>
        <td>col1</td>
        <td>col2</td>
</tr>
<tr>
        <td>col3</td>
<td>col4</td>
</tr>
</table>
```

CSS

```
table{
        border-collapse: collapse;
        /*border-collapse: separate;*/
}
```



Property อื่นๆ ของ table

property	ลักษณะ	ค่าที่เป็นได้
Border-spacing	กำหนดระยะห่างของเส้นขอบกับช่องในตาราง	*ความยาวแกนx ความยาวแกนy(px,...)* inherit
Caption-side	กำหนดตำแหน่งที่จะแสดงของข้อความกำกับตาราง (caption)	top bottom inherit
Empty-cells	กำหนดว่าจะให้แสดงเส้นขอบกับพื้นหลังของช่องตารางที่ไม่มีข้อมูลรึเปล่า	show hide inherit
Table-layout	กำหนดลักษณะการแสดงของช่องตาราง	auto – ช่องตารางขยายตามข้อมูลข้างใน fixed – ช่องตารางมีความยาวแบบเจาะจง โดยขึ้นกับความกว้างของคอลัมน์ที่ถูกกำหนดไว้ – หน้าเว็บจะโหลดเร็วกว่าแบบแรก inherit

ประโยชน์ของ CSS

1. การใช้ CSS ในการจัดรูปแบบการแสดงผล จะช่วยลดการใช้ภาษา HTML ในการตกแต่งเอกสารเว็บเพจ ทำให้ code ภายในเอกสาร HTML เหลือเพียงส่วนเนื้อหา ทำให้เข้าใจง่ายขึ้น การแก้ไขเอกสารทำได้ง่ายและรวดเร็ว

2. เมื่อ code ภายในเอกสาร HTML ลดลง ทำให้ขนาดไฟล์เล็กลง จึงดาวน์โหลดได้เร็ว

3. สามารถกำหนดรูปแบบการแสดงผลจากคำสั่ง style sheet ชุดเดียวกัน ให้มีผลกับเอกสาร HTML ทั้งหน้า หรือทุกหน้าได้ ทำให้เวลาแก้ไขหรือปรับปรุงทำได้ง่าย ไม่ต้องไล่ตามแก้ที่ HTML tag ต่างๆ ทั่วทั้งเอกสาร

4. สามารถควบคุมการแสดงผลให้เหมือนกัน หรือใกล้เคียงกัน ได้ในหลาย Web Browser

5. สามารถกำหนดการแสดงผลในรูปแบบที่เหมาะกับสื่อชนิดต่างๆ ไม่ว่าจะเป็นการแสดงผลบนหน้าจอ, บนกระดาษเมื่อสั่งพิมพ์, บนมือถือ หรือบน PDA โดยที่เป็นเนื้อหาเดียวกัน

6. ทำให้เป็นเว็บไซต์ที่มีมาตรฐาน ปัจจุบันการใช้ attribute ของ HTML ตกแต่งเอกสารเว็บเพจ นั้นล้าสมัยแล้ว W3C แนะนำให้เราใช้ CSS แทน ดังนั้นหากเราใช้ CSS กับเอกสาร HTML ของเรา ก็จะทำให้เข้ากับเว็บเบราเซอร์ในอนาคตได้ดี

กรณีที่จัดรูปแบบการแสดงผลด้วยภาษา HTML

```
<html>
    <body>

    <h1><font color="red" face="Arial">วิธีดูแลรักษาสุขภาพ</font></h1>
    <p><font color="black" face="Arial"><b>รับประทานอาหารที่มีประโยชน์ หมั่น
    ออกกำลังกาย และพักผ่อนให้เพียงพอ</b></font></p>

    <h1><font color="red" face="Arial">วิธีกินผลไม้ที่ถูกต้อง</font></h1>
    <p><font color="black" face="Arial"><b>ให้กินผลไม้แค่ทีละอย่าง เช่นจะกิน
    มะม่วงก็มะม่วงอย่างเดียวทั้งมื้อ เพื่อให้ร่างกายจัดเตรียมการย่อยได้ง่าย ไม่สับสน
    นอกจากนี้ยังไม่ควรกินผลไม้ทันทีหลังอาหาร
    ถ้าทานมื้อหลักแล้วควรรออย่างน้อย 20 นาที</b></font></p>

    </body>
    </html>
```

เปลี่ยนมาใช้คำสั่ง style sheet จัดรูปแบบการแสดงผลแทนการใช้ code ภาษา HTML

```
<html>
    <head>
    <style type="text/css">
    h1{color:red; font-family:Arial; }
    p{color:black; font-family:Arial; font-weight:bold }
    </style>
    </head>
    <body>

    <h1>วิธีดูแลรักษาสุขภาพ</h1>
    <p>รับประทานอาหารที่มีประโยชน์ หมั่นออกกำลังกาย และพักผ่อนให้เพียงพอ</p>

    <h1>วิธีกินผลไม้ที่ถูกต้อง</h1>
    <p>ให้กินผลไม้แค่ทีละอย่าง เช่นจะกินมะม่วงก็มะม่วงอย่างเดียวทั้งมื้อ เพื่อให้ร่างกายจัด
    เตรียมการย่อยได้ง่าย ไม่สับสน นอกจากนี้ยังไม่ควรกินผลไม้ทันทีหลังอาหาร
    ถ้าทานมื้อหลักแล้วควรรออย่างน้อย 20 นาที</p>

    </body>
    </html>
```