CSS Selector 30 ตัวที่ควรจำ

คุณคิดว่าคุณได้เรียนพื้นฐานเรื่อง id, class และ descendant selector แล้วคิดว่าพอ? ถ้าคำตอบนั้นคือใช่ นั่นแปลว่าคุณพลาดในเรื่องความยืดหยุ่นจำนวนมากที่คุณจะได้ใช้แล้ว ขณะที่ selector จำนวนมากในบทความนี้อยู่ใน CSS3 specification และใช้ได้เฉพาะบราวเซอร์รุ่นใหม่ล่าสุดเท่านั้น บางทีคุณอาจจะต้องบอกกับตัวคุณเองทีหลังว่า คุณน่าจะจำมันได้ตั้งนานแล้ว

1. *

```
1    * {
2     margin: 0;
3     padding: 0;
4    }
```

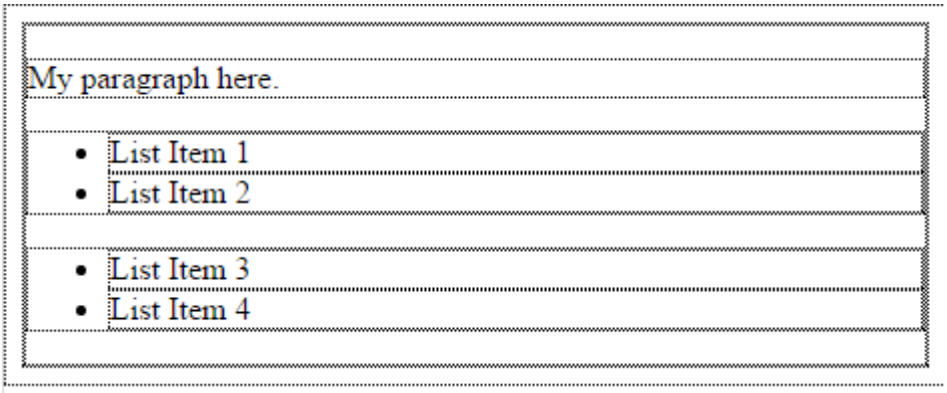
เริ่มง่ายๆ ตัวแรกเลยสำหรับมือใหม่ ก่อนที่คุณจะเจอ selector ขั้นสูงกว่านี้ต่อ

สัญลักษณ์ดอกจันทร์ (*) นั้น select ทุกๆ element บน page นักพัฒนาจำนวนมากจะใช้เทคนิคนี้ในการ Set ค่า `margin` และ `padding` ให้เท่ากับ 0 ขณะที่มันโอเคเลยที่จะใช้ในการเขียนทดสอบ แต่ผมก็ขอแนะนำว่าอย่าได้ใช้มันใน production code จะดีที่สุด เพราะมันทำให้ browser นั้น *ทำงานหนัก* ในการประมวลผลโดยไม่จำเป็น

`*` สามารถใช้ร่วมกับ child selector ได้ ตัวอย่างเช่น

```
1    #container * {
2     border: 1px solid black;
3    }
```

ตัวอย่างนี้จะ select ทุกๆ element ที่เป็นลูกของ `#container` `div` โปรดฟังอีกครั้งหนึ่ง พยายามอย่าใช้เทคนิคนี้ถ้าไม่จำเป็น



2. #X

```
1    #container {
2      width: 960px;
3      margin: auto;
4    }
```

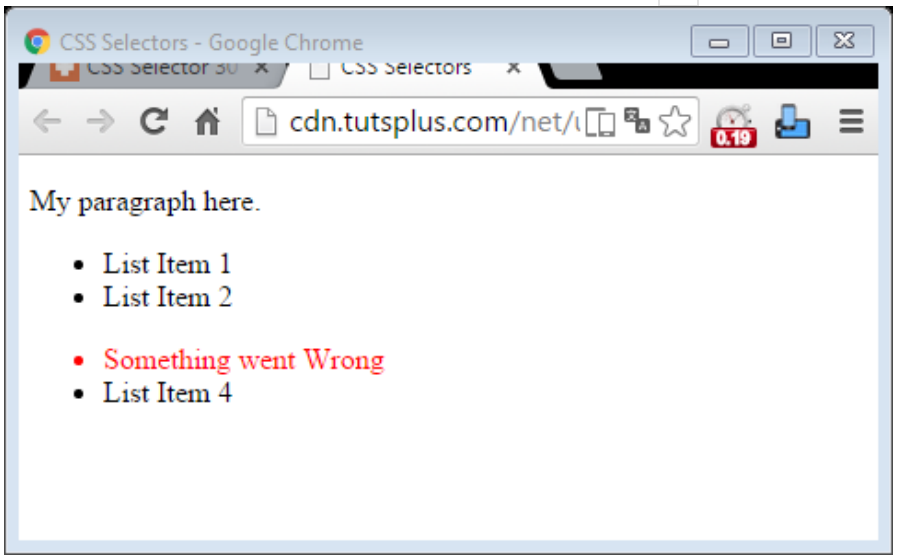
ใส่เครื่องหมายสี่เหลี่ยม (#) ไว้ด้านหน้านั้นแปลว่าให้เรา select โดยใช้ `id` โดยทั่วไปมันง่ายที่จะใช้ แต่อย่างไรก็ตามเราควรระวังในการใช้ id selector

id selector นั้นไม่ยืดหยุ่น มันไม่อนุญาตให้ใช้ซ้ำ ถ้าเป็นไปได้ เริ่มแรกให้พยายามใช้ tag name ของ HTML5 ซึ่งเป็น element แบบใหม่ หรือใช้ pseudo-class แทน (หมายเหตุจากผู้แปล Tag Name ก็เช่น section,article เป็นต้น)

3. .X

```
1    .error {
2      color: red;
3    }
```

นี่คือ class selector ความต่างระหว่าง id และ class นั้นคือคุณสามารถระบุ element ได้หลายตัว โดยการใช้ class เมื่อคุณต้องการใส่ style เข้ากับกลุ่มของ element แต่ใช้ id เพื่อระบุ element ที่เจาะจงไปเลย

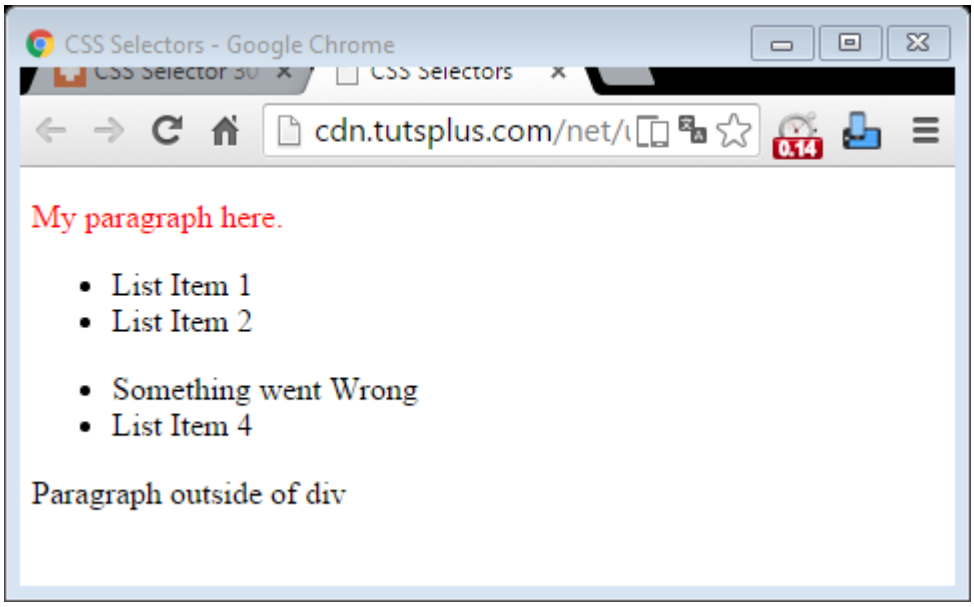


4. X Y

```
1    li a {
2      text-decoration: none;
3    }
```

ถัดไปคือ selector ที่ต้องช่วยใส่ใจกันให้มากๆ มันชื่อ descendant selector ใช้เมื่อคุณต้องการระบุ selector ของคุณให้เฉพาะเจาะจงยิ่งขึ้น ยกตัวอย่าง อะไรจะเกิดขึ้นถ้า แทนที่คุณจะระบุทุกๆ tag a (anchor tags) ทั้งหมด แต่จริงๆ แล้วคุณต้องการระบุ tag a เฉพาะที่อยู่ใต้ unordered list เท่านั้น แน่นอนคุณสามารถระบุได้โดยใช้ descendant selector

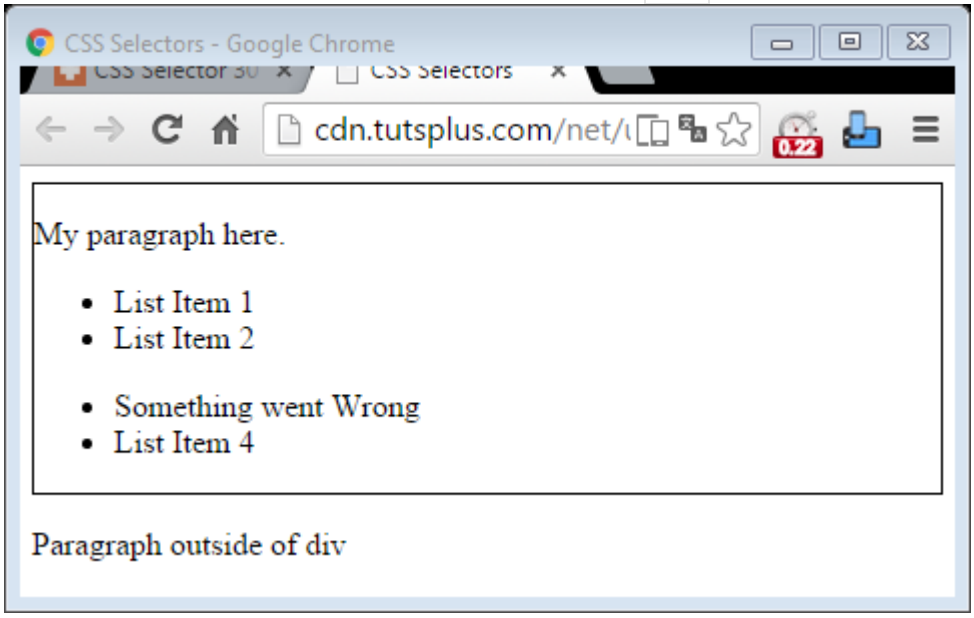
Pro-tip - ถ้า selector ที่คุณเขียนเป็นแบบนี้เช่น X Y Z A B.error คุณว่าคุณทำไรผิดไปรึเปล่า ถามตัวคุณเองก่อนเสมอว่าจำเป็นรึที่จะต้องระบุ ละเอียด ขนาดนั้น?



5. X

```
1    a { color: red; }
2    ul { margin-left: 0; }
```

แล้วถ้าคุณต้องการระบุทุกๆ element บน page โดยใช้ `type` แทนที่จะใช้ `id` หรือชื่อ `class` หล่ะ ง่ายๆ เลย ใช้ type selector ถ้าคุณต้องการระบุ unordered list ทุกตัว ใช้ `ul {}`

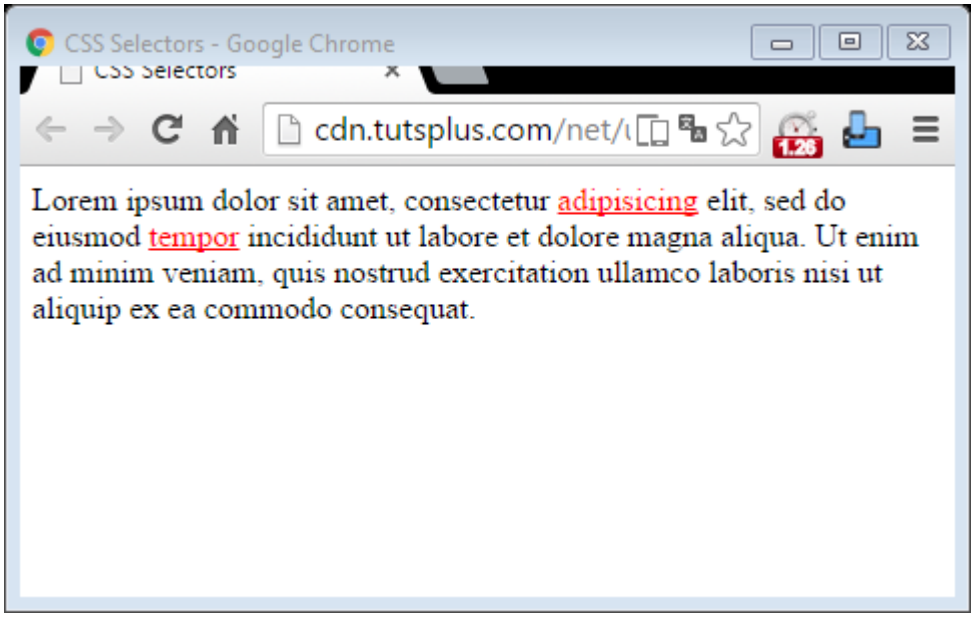


6. X:visited and X:link

```
1    a:link { color: red; }
2    a:visted { color: purple; }
```

เราใช้ `:link` pseudo-class เพื่อที่จะระบุทุกๆ tag a ที่คุณยังไม่ได้ไปคลิก

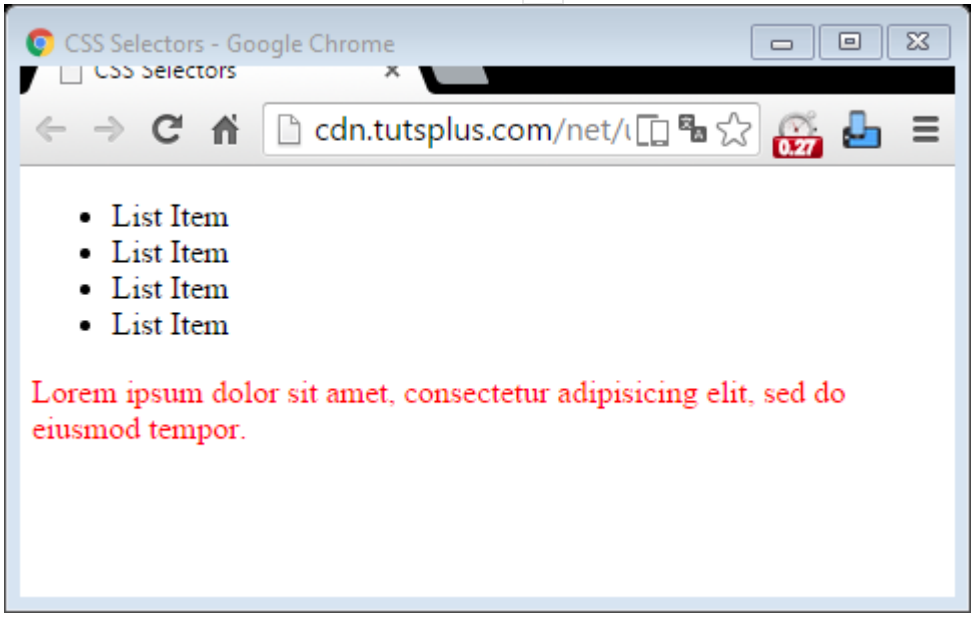
นอกจากนี้ เราใช้ `:visited` pseudo class ที่คุณก็เดาได้ว่ามันจะไป style กับ tag a บน page ที่คุณ *เคยไปคลิก* หรือ *เคยไปเยี่ยมชม* แล้วนั่นเอง



7. X + Y

```
1    ul + p {
2      color: red;
3    }
```

นี่เรียกว่า adjacent selector มันจะระบุ*เฉพาะ* element ที่อยู่ชิดถัดจาก element ต้นทาง (อยู่ชิดติดกันจริงๆ) ในกรณีนี้ เฉพาะ paragraph แรกที่อยู่ชิดต่อหลังจาก `ul` แต่ละตัว จะแสดงตัวอักษรสีแดง



8. X > Y

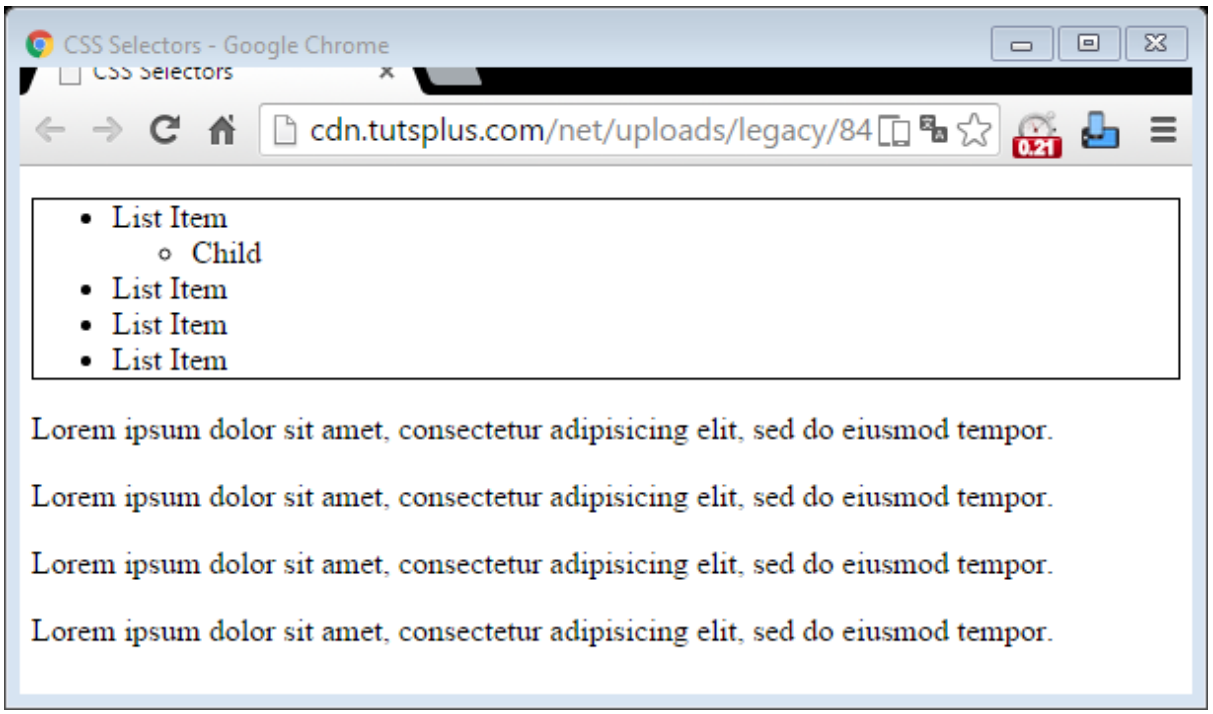
```
1    div#container > ul {
2      border: 1px solid black;
3    }
```

ความแตกต่างระหว่าง `X Y` กับ `X > Y` คือ ตัวหลังจะเลือกเฉพาะ element ลูกที่อยู่โดยตรงลงไปหนึ่งชั้นเท่านั้น ตัวอย่าง ลองดู markup ต่อไปนี้

```
01    <div id="container">
02      <ul>
03        <li> List Item
04          <ul>
05            <li> Child </li>
06          </ul>
07        </li>
08        <li> List Item </li>
09        <li> List Item </li>
10        <li> List Item </li>
11      </ul>
12    </div>
```

selector `#container > ul` จะเลือกเฉพาะ `ul` ที่เป็นลูกโดยตรงของ `div` ที่มี `id` ชื่อ `container` เท่านั้น มันจะไม่ระบุ `ul` ที่อยู่ภายใน `li` ตัวแรก

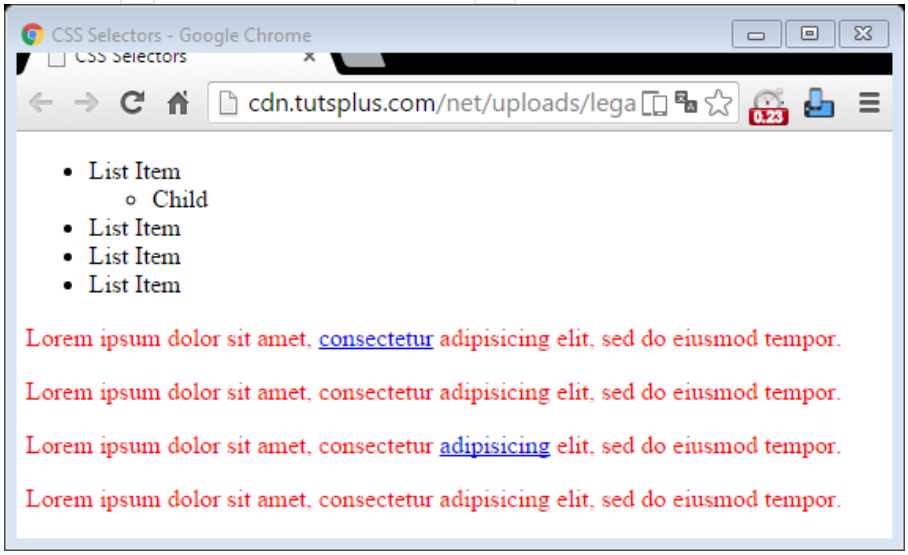
ด้วยเหตุนี้ จะก่อให้ส่งผลให้โค้ดทำงานได้มีประสิทธิภาพมากขึ้นในการใช้ child combinator อันนี้ ในความเป็นจริง มันเหมาะอย่างยิ่งในกรณีที่คุณทำงานกับ CSS engine ที่อยู่ในภาษา JavaScript



9. X ~ Y

```
1    ul ~ p {
2        color: red;
3    }
```

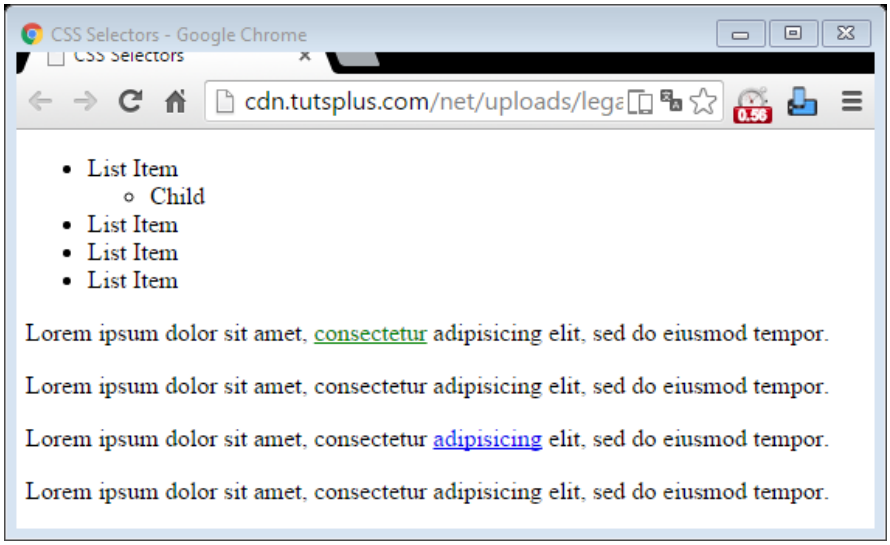
นี่คือ sibling combinator ที่ดูคล้ายกับ X + Y อย่างไรก็ตามมันจะยืดหยุ่นกว่า ขณะที่ adjacent selector (`ul + p`) จะเลือกเฉพาะ element ที่อยู่ต่อและติดกับ selector ก่อนหน้า*โดยทันที*แต่อันนี้จะทำงานได้กว้างกว่าที่ จากตัวอย่างมันจะเลือก `p` element ทุกตัวเลยที่อยู่ต่อจาก `ul`



10. X[title]

```
1    a[title] {
2        color: green;
3    }
```

อันนี้เรียกว่า *attribute selector* ในตัวอย่างด้านบน มันจะเลือกเฉพาะ tag a ที่มีแอตทริบิวต์ `title` เท่านั้น Tag a อื่นๆ จะไม่ได้รับผลกระทบ อธิบายง่ายๆ ก็แค่คุณต้องการเลือกให้เฉพาะเจาะจงมากขึ้น เท่านั้นแหละ...



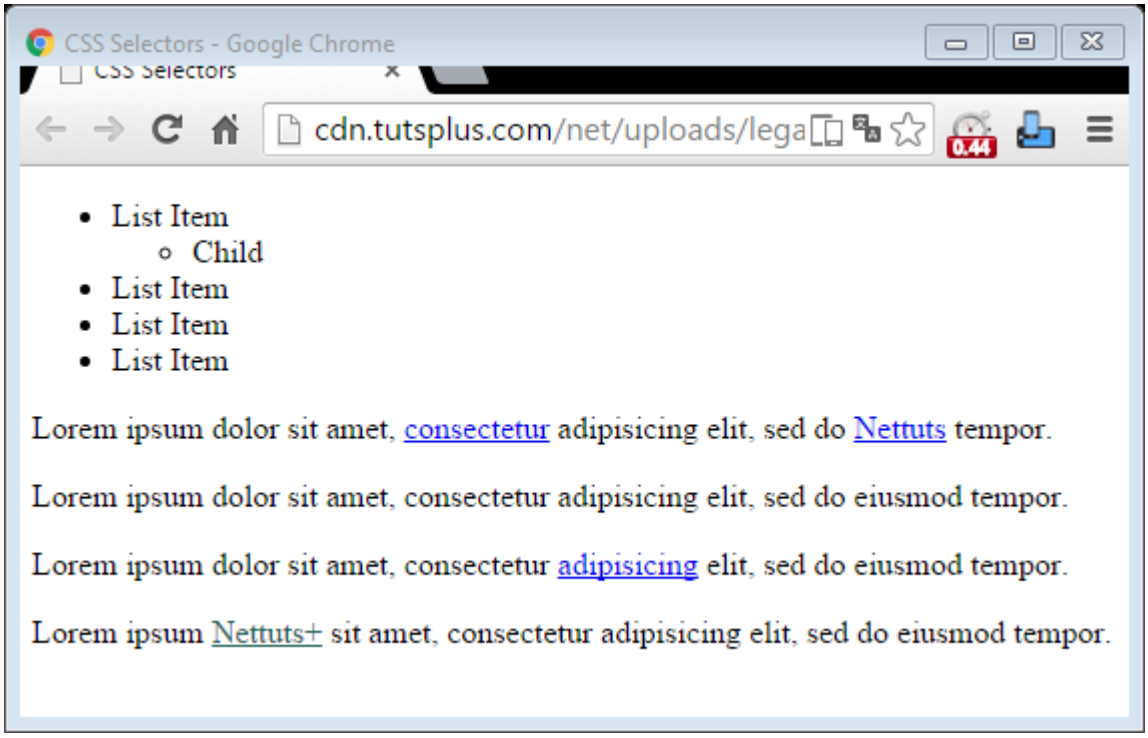
11. X[href="foo"]

```
1    a[href="http://net.tutsplus.com"] {
2      color: #1f6053; /* nettuts green */
3    }
```

โค้ดด้านบนนั้นจะ style ทุกๆ tag a ที่ link ไปที่ *http://net.tutsplus.com* มันจะทำให้ลิ้งค์นั้นมีสีเขียวเข้ม tag a อื่นๆ นั้นจะไม่ได้รับผลกระทบ

สังเกตว่าเราจะนำค่าใส่ไว้ในเครื่องหมาย " " จำไว้ว่าต้องทำแบบนี้ด้วยถ้าคุณใช้ CSS selector ที่มาจาก engine ของภาษา JavaScript ถ้าเป็นไปได้ ให้ใช้ CSS3 selector ไปเลยโดยตรงแทนที่จะใช้ unofficial method เหล่านี้

มันทำงานได้ดี แม้ว่า ค่อนข้างตรงไปหน่อย ถ้าลิ้งที่โดยตรงไปยัง Nettuts+ แต่เขียน path ในรูปแบบ *nettuts.com* แทนที่จะเป็น url แบบเต็มหล่ะ ถ้าเป็นแบบนั้น เราสามารถใช้ syntax ของ regular expression แทนได้

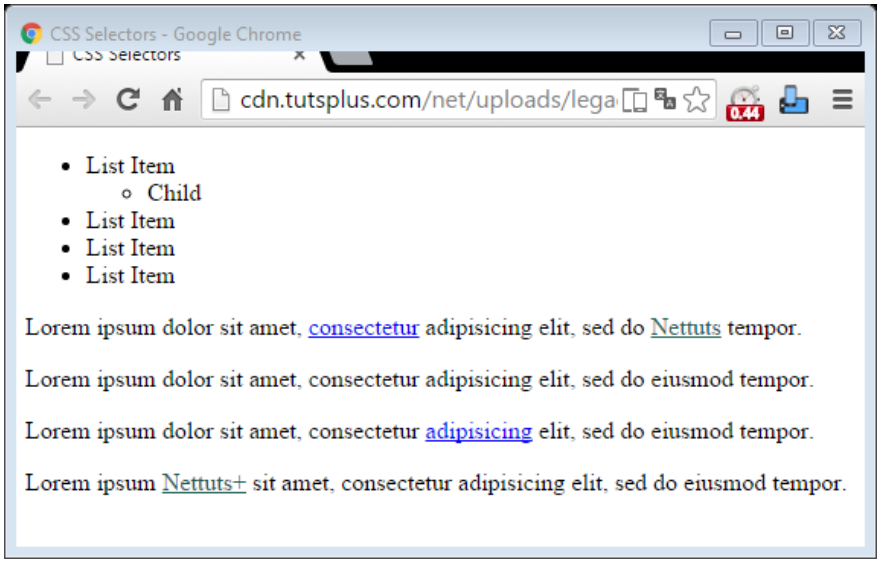


12. X[href*="nettuts"]

```
1    a[href*="tuts"] {
2      color: #1f6053; /* nettuts green */
3    }
```

นั่นแหละ ที่เราต้องการ เครื่องหมายดอกจันทร์ระบุว่า ค่านั้นจะ *ปรากฏที่ไหน* ก็ได้ใน attribute โดยวิธีนี้เอง ทำให้การระบุครอบคลุมถึง *nettuts.com net.tutsplus.com* และก็ *tutsplus.com*

จำไว้ว่าการใช้แบบนี้ค่อนข้างกว้าง ถ้า tag a นั้น link ไป site ที่ไม่ใช่ Envato ถึงแม้มี tuts นั้นอยู่ใน url หล่ะ มันก็โดนเลือกด้วย ฉะนั้นเมื่อคุณต้องการระบุให้ชัดเจนเข้าไปอีก ก็ให้ใช้ `^` กับ `$` เพื่อระบุถึงจุดเริ่มต้นและจุดสิ้นสุดของ string ตามลำดับ



13. X[href^="http"]

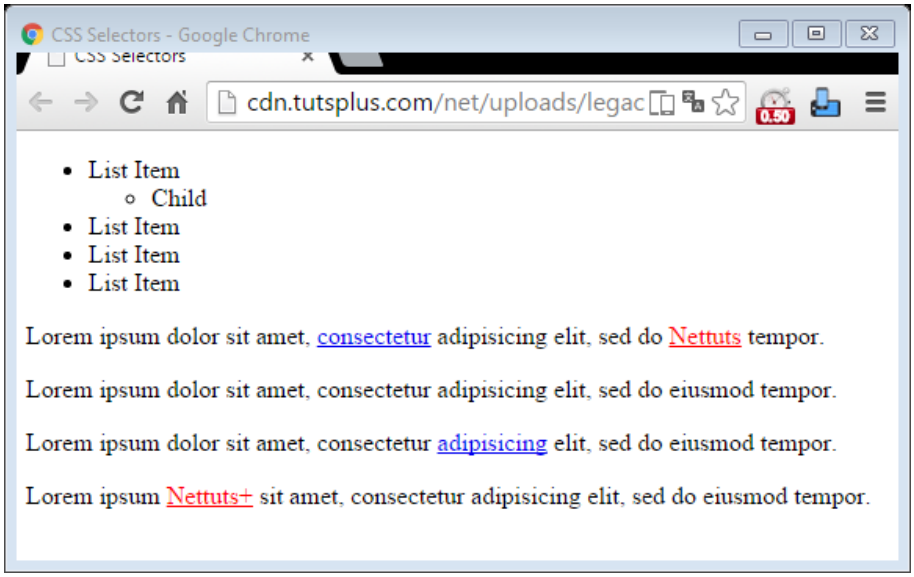
```
a[href^="http"] {
    background: url(path/to/external/icon.png) no-repeat;
    padding-left: 10px;
}
```

เคยแปลกใจไหมว่าบางเว็บไซต์นั้นสามารถแสดง icon เล็กๆ ถัดจาก link ที่ออกไปยังเว็บไซต์ภายนอก ผมมั่นใจว่าคุณต้องเคยเห็นมาก่อน มันเป็นตัวแจ้งเตือนที่ดีที่บอกว่า คุณจะออกไปเว็บอื่นแล้วนะ

สัญลักษณ์ ^ อันนี้ มันถูกใช้โดยทั่วไปใน regular expression เพื่อบ่งบอกจุดเริ่มต้นของ string ถ้าเราต้องการระบุ tag a ทั้งหมดที่มี `href` ที่เริ่มต้นด้วย `http` เราควรใช้ selector ประมาณโค้ดด้านบน

สังเกตว่าเราไม่หา `http://` ตรงๆ มันไม่จำเป็น เพราะมันยังมี url แบบอื่นด้วย ตัวอย่างเช่นการขึ้นต้นด้วย `https://` เป็นต้น

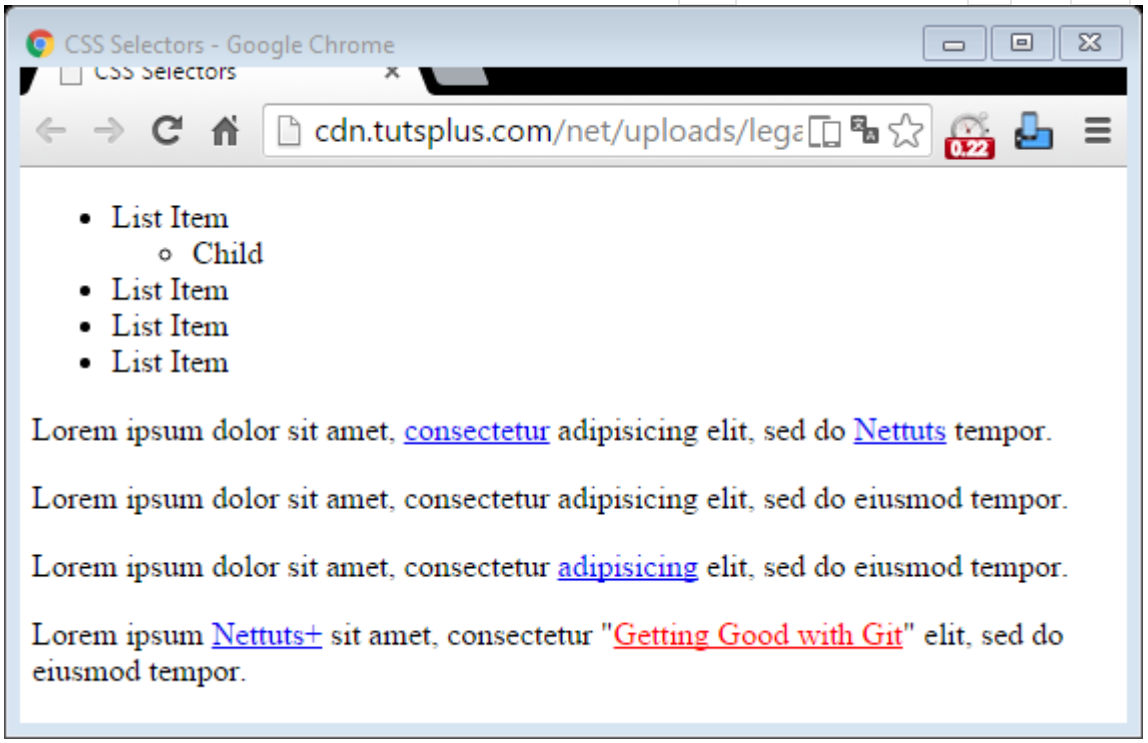
และก็ ถ้าคุณต้องการ style เจ้า tag a ทั้งหมดที่ link ไปยังรูปภาพ ในกรณีนี้ให้ใช้การค้นหาแบบ *จุดสิ้นสุด* ของ string แทน (ซึ่งจะกล่าวถัดไป)



14. X[href$=".jpg"]

```
a[href$=".jpg"] {
    color: red;
}
```

อีกครั้ง เราใช้สัญลักษณ์จาก regular expression `$` เพื่อบ่งบอกถึงจุดสิ้นสุดของ string ในกรณีนี้ เรากำลังค้นหาทุกๆ tag a ที่ link ไปยัง image หรืออย่างน้อย url ใดๆ ที่ลงท้ายด้วย `.jpg` จำไว้ว่ากรณีนี้จะใช้กับ `gif` และ `png` ไม่ได้



15. X[data-*="foo"]

```
a[data-filetype="image"] {
    color: red;
}
```

อ้างอิงกลับไปยังข้อที่สิบสี่ด้านบน แล้วถ้าเราต้องการเอาทุกๆ image type หล่ะ (`png` , `jpeg` , `jpg` , `gif`) ถ้าอย่างนั้น เราต้องสร้าง multiple selector แบบนี้

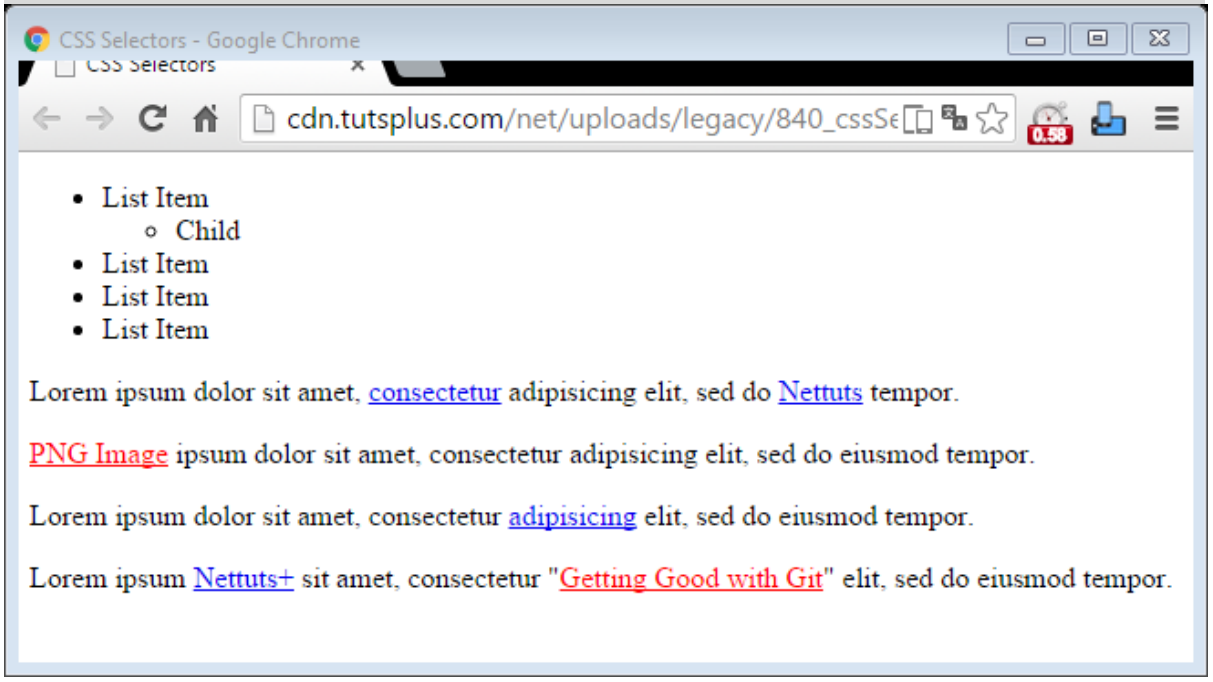
```
a[href$=".jpg"],
a[href$=".jpeg"],
a[href$=".png"],
a[href$=".gif"] {
    color: red;
}
```

แต่นั่นมันดูแสนเหนื่อยใจและไร้ประสิทธิภาพ ความเป็นไปได้อื่นๆ เช่นการใช้ custom attribute แล้วถ้าเราเพิ่ม data-filetype attribute ของเราเองไปที่ tag a แต่ละอันที่ link ไปยังรูปภาพหล่ะ

```
<a href="path/to/image.jpg" data-filetype="image"> Image Link </a>
```

จากนั้น เมื่อเข้าที่เข้าทางแล้ว เราสามารถใช้ attribute selector ทั่วไปในการระบุ tag a เหล่านั้น

```
a[data-filetype="image"] {
    color: red;
}
```



16. X[foo~="bar"]

```
a[data-info~="external"] {
    color: red;
}

a[data-info~="image"] {
    border: 1px solid black;
}
```

มันเป็น selector แบบพิเศษแบบหนึ่งที่จะทำให้เพื่อนคุณประทับใจ มีคนไม่กี่คนที่รู้ trick นี้ สัญลักษณ์ตัวหนอน หรือ tilda symbol (`~`) ช่วยให้เราระบุ list ของ attribute ที่ซึ่งคั่นด้วยเครื่องหมายเว้นวรรค

ไปที่ custom attribute ข้อ 15 ด้านบน เราสามารถสร้าง `data-info` attribute ที่ซึ่งครอบคลุมรายการที่คั่นด้วยการเว้นวรรคใดๆ ได้ ในกรณีนี้ เราจะสามารถระบุ external link และ link ไปยังรูปภาพต่างๆ ได้ -- นี่คือตัวอย่าง

```
"<a href="path/to/image.jpg" data-info="external image"> Click Me, Fool </a>
```

ด้วยการใส่ค่าไปใน markup เราสามารถจับ tag ใดๆ ที่มีค่าเหล่านั้น ด้วยการใช้เทคนิคจาก attribute selector อันนี้

```
/* Target data-info attr that contains the value "external" */
a[data-info~="external"] {
    color: red;
}
```

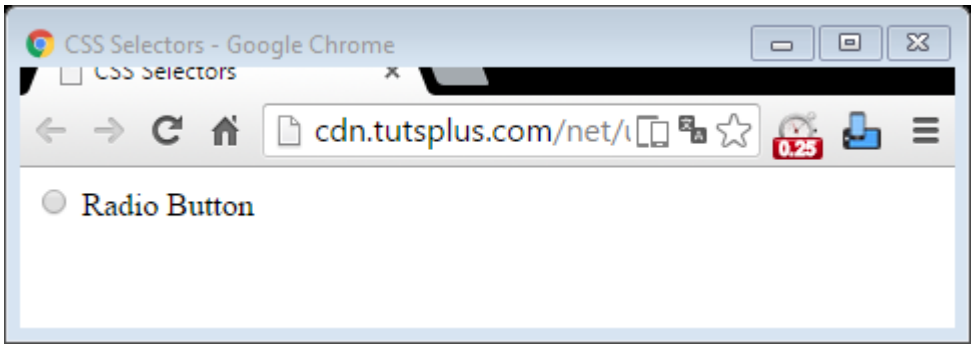
```
5
6     /* And which contain the value "image" */
7     a[data-info~="image"] {
8       border: 1px solid black;
9     }
```

ง่ายใช่ไหมหล่ะ?

17. X:checked

```
1     input[type=radio]:checked {
2       border: 1px solid black;
3     }
```

pseudo class นี้เพียงแค่ระบุ element ของ user interface ที่ได้รับ*การเลือก (checked) เช่น radio button หรือ checkbox* มันก็น่าจะเข้าใจง่ายๆ ตามนั้น...



18. X:after

pseudo class `before` และ `after` นั้นเจอเต็มๆ ทุกๆ วัน เหมือนว่าผู้คนสามารถค้นหาวิธีที่สร้างสรรค์ใหม่ๆ เพื่อที่จะใช้มันให้เกิดประสิทธิภาพมากที่สุด พวกมันทำงานง่ายๆ แค่การสร้าง content รอบๆ element ที่ถูกเลือก

หลายคนเห็นพวกมันครั้งแรกเมื่อต้องเจอกับเทคนิคที่เรียกว่า clear-fix hack

```
.clearfix:after {
    content: "";
    display: block;
    clear: both;
    visibility: hidden;
    font-size: 0;
    height: 0;
    }

.clearfix {
   *display: inline-block;
   _height: 1%;
}
```

clear-fix hack นี้ใช้ pseudo class `:after` ในการสร้างอักขระเว้นวรรค หรือ space หลังจาก element จากนั้นทำการ clear มัน มันช่างเป็นเทคนิคที่ยอดเยี่ยมที่คุณต้องมีและเก็บใส่หัวไว้เลย โดยเฉพาะในกรณีที่ไม่สามารถใช้คำสั่ง `overflow: hidden;` ได้

สำหรับการใช้งานแบบสร้างสรรค์อื่นๆ http://net.tutsplus.com/tutorials/html-css-techniques/quick-tip-getting-clever-with-css3-shadows/

อ้างอิง CSS3 Selector specification คุณควรใช้เทคนิคนี้บน pseudo element ด้วย syntax ที่ใช้โคล่อน 2 ตัวติดกัน `::` *อย่างไรก็ตาม เพื่อความเข้ากันได้ ควรใช้โคล่อนอันเดียวจะดีกว่า ในความเป็นจริง มันดีกว่าที่จะใช้เวอร์ชั่น colon อันเดียวในโปรเจคของคุณ*

19. X:hover

```
div:hover {
  background: #e3e3e3;
}
```

โอ้ และก็มาถึง คุณก็รู้อันนี้ มันเป็น official term ที่เรียกว่า `user action pseudo class` ชื่อนั้นทำให้สับสน แค่ต้องการใส่ style เฉพาะเจาะจงไปบน element ทุกชนิดเมื่อคุณเอาเม้าส์ไปวางหรือ hover นี่แหละมันใช่เลย

โปรดจำไว้ว่า Internet Explorer เวอร์ชันเก่าๆ ไม่ตอบสนองกับ `:hover` *pseudo class ที่ใส่กับ element อื่นๆ ที่ไม่ใช่ tag a (anchor tag)*

คุณมักจะได้ใช้ selector ตัวนี้บ่อยๆ เช่นการทำ `border-bottom` เมื่อคุณเอาเม้าส์ hover มัน

```
a:hover {
 border-bottom: 1px solid black;
}
```

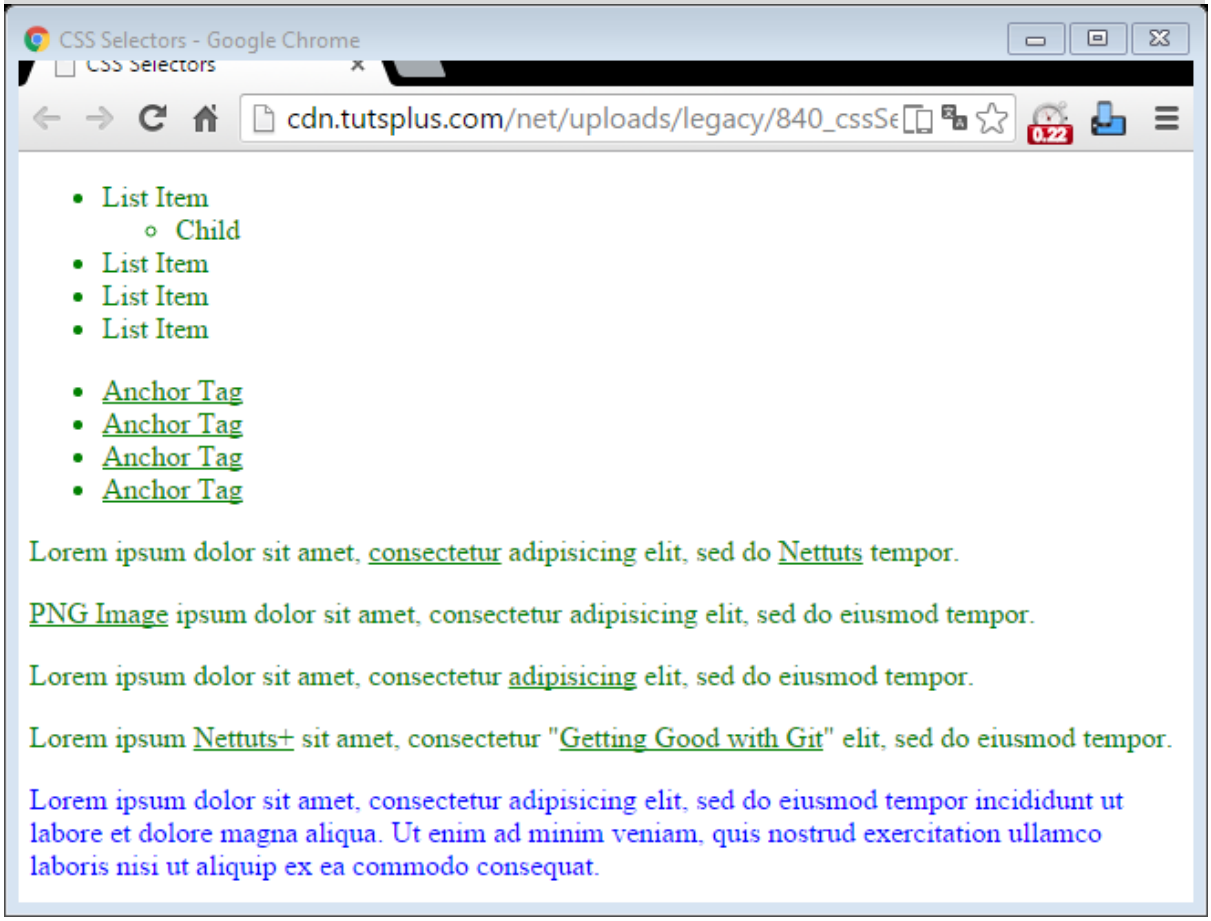
Pro-tip - `border-bottom: 1px solid black;` *นั้นดูดีกว่า* `text-decoration: underline;`

20. X:not(selector)

```
1    div:not(#container) {
2      color: blue;
3    }
```

pseudo class แบบ `negation` ค่อนข้างใช้ได้ดี ลองคิดว่าผมต้องการเลือก tag div ทั้งหมด ยกเว้นแค่ตัวที่มี `id` ที่ชื่อ `container` โค้ดด้านบนสามารถทำได้ดีเลยทีเดียว หรือ ถ้าผมต้องการเลือกทุกๆ element ยกเว้น tag paragraph เราจะใช้โค้ดนี้จัดการ (ซึ่งแน่นอน ไม่แนะนำให้ใช้)

```
1    *:not(p) {
2      color: green;
3    }
```



21. X::pseudoElement

```
1    p::first-line {
2      font-weight: bold;
3      font-size: 1.2em;
4    }
```

เราสามารถใช้ pseudo element (ที่แสดงโดยสัญลักษณ์ ::) เพื่อ style บน element เช่นใช้กับบรรทัดแรก หรืออักษรตัวแรก จำไว้ว่าต้องใส่ที่ block level element เท่านั้นเพื่อที่จะให้มันแสดงผล

pseudo-element นั้นเกิดจากสัญลักษณ์โคล่อนสองตัว `::`

ระบุตัวอักษรตัวแรกของ Paragraph

```
1    p::first-letter {
2       float: left;
3       font-size: 2em;
4       font-weight: bold;
5       font-family: cursive;
6       padding-right: 2px;
7    }
```

โค้ดด้านบนบอกว่า เราจะไปที่ทุกๆ paragraph บน page จากนั้นระบุแค่ตัวอักษรตัวแรกของ element paragraph เท่านั้น

มันใช้บ่อยเหมือนกับสไตล์ของบทความในหนังสือพิมพ์ที่มักทำให้ตัวอักษรตัวแรกดูใหญ่

ระบุบรรทัดแรกของ Paragraph

```
1    p::first-line {
2       font-weight: bold;
3       font-size: 1.2em;
4    }
```

คล้ายๆกัน pseudo element `::first-line` นั้นเดาได้เลยว่าจะ style บรรทัดแรกของ element เท่านั้น

เพื่อความเข้ากันได้กับ style sheet ที่มีอยู่แล้ว ผู้ใช้ต้องยอมรับว่า colon 1 อัน ใช้สำหรับ pseudo-element ที่มีอยู่ใน CSS level 1 และ 2 (อันที่ชื่อว่า :first-line, :first-letter, :before และ :after) ความเข้ากันได้อันนี้จะไม่สามารถใช้ได้กับ pseudo-element แบบใหม่ใน specification นี้ - ที่มา

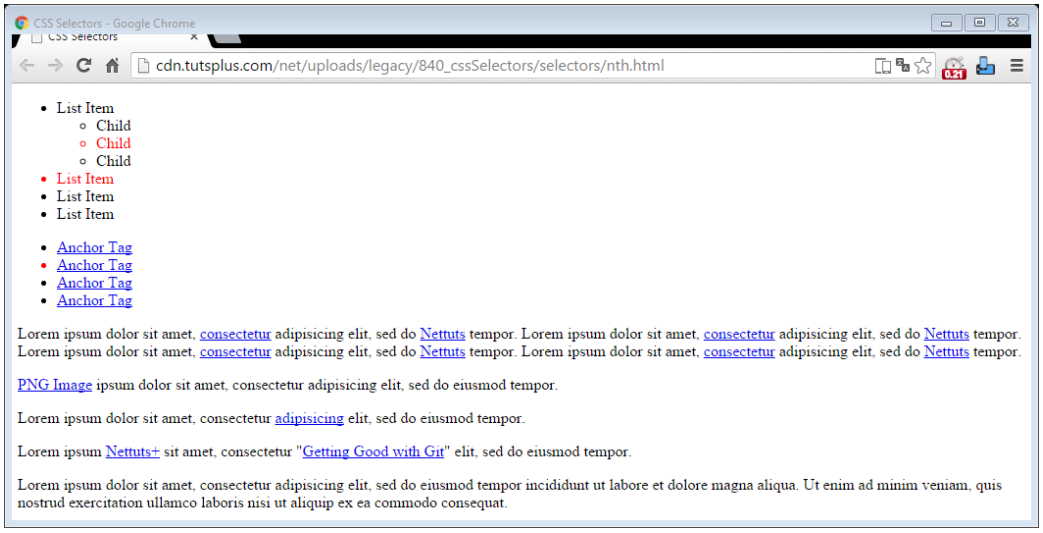
22. X:nth-child(n)

```
li:nth-child(3) {
    color: red;
}
```

จำได้ไหมว่าเราไม่มีวิธีที่จะระบุ element ในกลุ่มที่เราเลือกออกมา `nth-child` นั้นจะเข้ามาแก้ปัญหาให้

สังเกตว่า nth-child นั้นจะรับค่า integer ที่ parameter ไม่ใช่ zero-based เช่น ถ้าคุณต้องการระบุ item อันที่สอง ก็ใช้ li:nth-child(2) ตรงๆ ไปเลย

คุณยังสามารถใช้ตัวแปรเพื่อ select ได้อย่างเช่น เราใช้ li:nth-child(4n) เพื่อ select ทุกๆ list item อันที่ 4

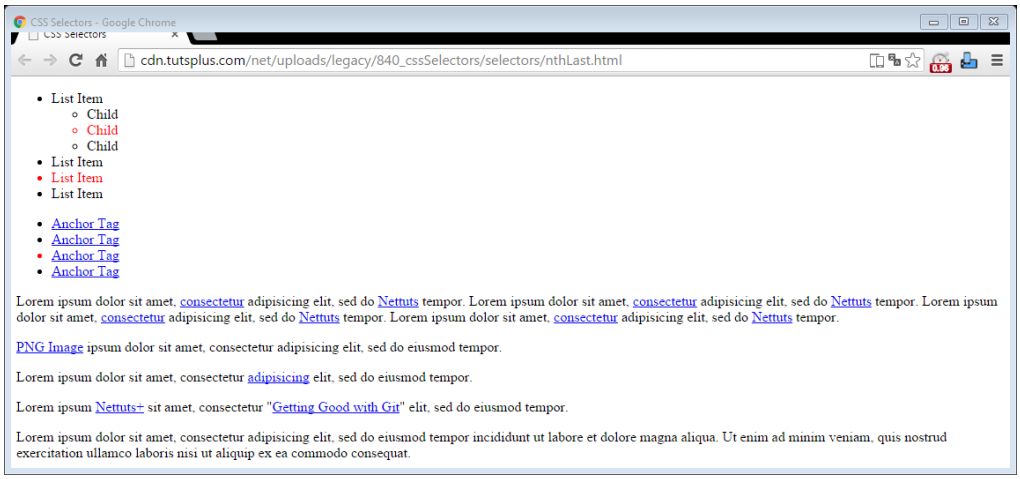


23. X:nth-last-child(n)

```
li:nth-last-child(2) {
    color: red;
}
```

แล้วถ้าเรามี list ขนาดใหญ่ที่มี item อยู่เป็นจำนวนมากแหละ สมมุติเราต้องการ item อันที่ 2 จากท้ายที่อยู่ใน `ul` แทนที่จะใช้ `li:nth-child(397)` คุณควรใช้ pseudo class `nth-last-child` แทน

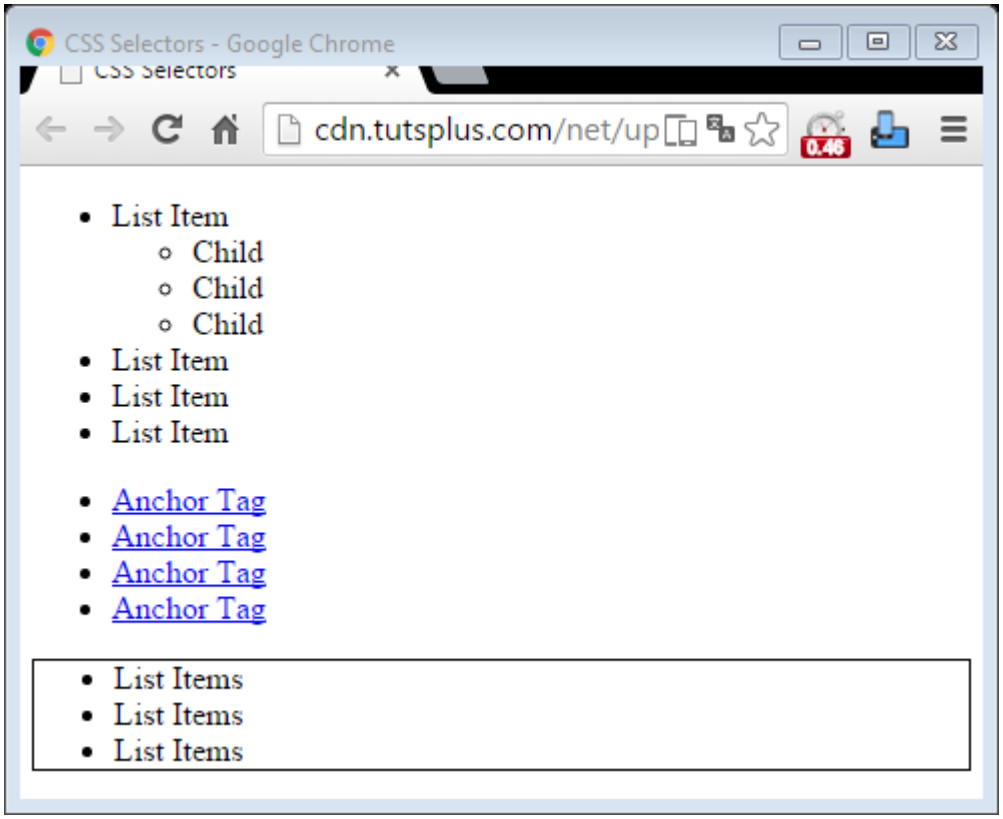
เทคนิคนี้ใช้ได้เกือบเหมือนกับข้อ 22 ด้านบน อย่างไรก็ตามเทคนิคนี้เริ่มจากข้างหลังแทน



24. X:nth-of-type(n)

```
1    ul:nth-of-type(3) {
2        border: 1px solid black;
3    }
```

จะมีบางครั้งแทนที่คุณจะ select โดยใช้ `child` แต่คุณจำเป็นต้องใช้ `type` ในการ select element จินตนาการว่า mark-up นั้นประกอบไปด้วย unordered list 5 อัน ถ้าคุณต้องการ style เพียงแค่ ul ตัวที่สาม และไม่มี id เฉพาะเกี่ยวข้องด้วย คุณควรใช้ nth-of-type(n) pseudo-class ในโค้ดด้านบน เพียงแค่ `ul` ตัวที่สามนั้นจะแสดง border รอบๆ มัน



25. X:nth-last-of-type(n)

```
1    ul:nth-last-of-type(3) {
2        border: 1px solid black;
3    }
```

และใช่ เพื่อให้คู่กัน เราสามารถใช้ nth-last-of-type เพื่อเริ่มด้านหลัง ไปที่ element ที่เราต้องการแทน

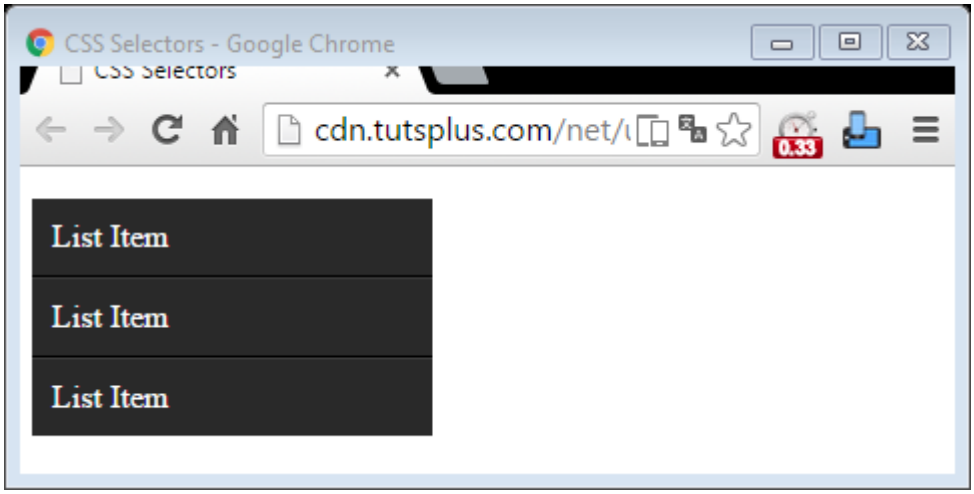
26. X:first-child

```
1    ul li:first-child {
2        border-top: none;
3    }
```

pseudo class แบบนี้จะทำให้เราระบุแค่ child ตัวแรกของ element parent บ่อยครั้งคุณจะใช้มันในการเอา border ออกจาก item ตัวแรกและตัวสุดท้าย

ตัวอย่าง ลองคิดว่าถ้าคุณมี list ของแถวจำนวนหนึ่ง แต่ละอันมี style `border-top` และ `border-bottom` ด้วยวิธีนี้ แถวแรกกับแถวสุดท้ายจะดูประหลาด

นักออกแบบจำนวนมากใส่ class `first` และ `last` เพื่อจัดการมัน นอกจากนี้ คุณสามารถใช้ pseudo class เพื่อมาช่วยคุณแทน



27. X:last-child

```
1    ul > li:last-child {
2      color: green;
3    }
```

ตรงข้ามกับ first-child โดยที่ last-child จะระบุ item ตัวสุดท้ายของ element parent แทน

ลองดูตัวอย่างง่ายๆ ที่จะอธิบายความเป็นไปได้ของการใช้ class เหล่านี้ เราจะทำการสร้างและทำการ style list item

Markup

```
1    <ul>
2      <li> List Item </li>
3      <li> List Item </li>
4      <li> List Item </li>
5    </ul>
```

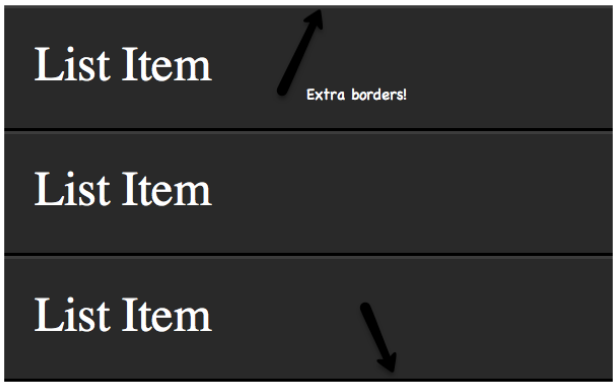
ไม่มีอะไรพิเศษ แค่ list ธรรมดา

CSS

```
01    ul {
02      width: 200px;
03      background: #292929;
04      color: white;
05      list-style: none;
```

```
06      padding-left: 0;
07    }
08
09    li {
10      padding: 10px;
11      border-bottom: 1px solid black;
12      border-top: 1px solid #3c3c3c;
13    }
```

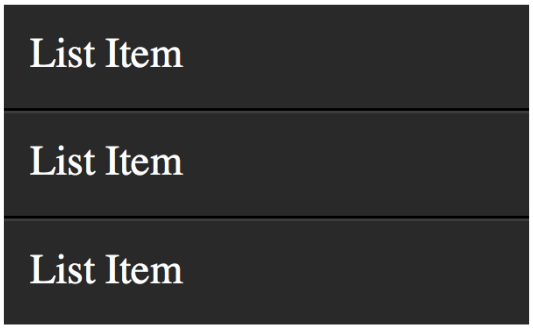
style ที่เขียนนี้จะใส่สี background เอา padding ที่เป็น browser-default บน element `ul` ออก และใส่ border เข้าไปกับ `li` แต่ละตัวเพื่อให้ดูมีมิติ



เพื่อเพิ่มมิติไปยัง list ของคุณ ให้ใส่ `border-bottom` ไปที่แต่ละ `li` นั่นจะทำให้เกิดเฉดสีดำมากกว่าสี background ของ `li` จากนั้น ใส่ `border-top` ที่ทำให้เฉดสีดูสว่างขึ้น

ปัญหาเดียวสำหรับภาพด้านบนก็คือ border จะถูกใส่ทุกๆ top และ bottom ของ unordered list ด้วย ซึ่งไม่ใช่ที่เราต้องการ ให้ใช้ pseudo `:first-child` และ `:last-child` เพื่อมาแก้ไขมัน

```
1   li:first-child {
2       border-top: none;
3   }
4
5   li:last-child {
6       border-bottom: none;
7   }
```



28. X:only-child

```
1    div p:only-child {
2        color: red;
3    }
```

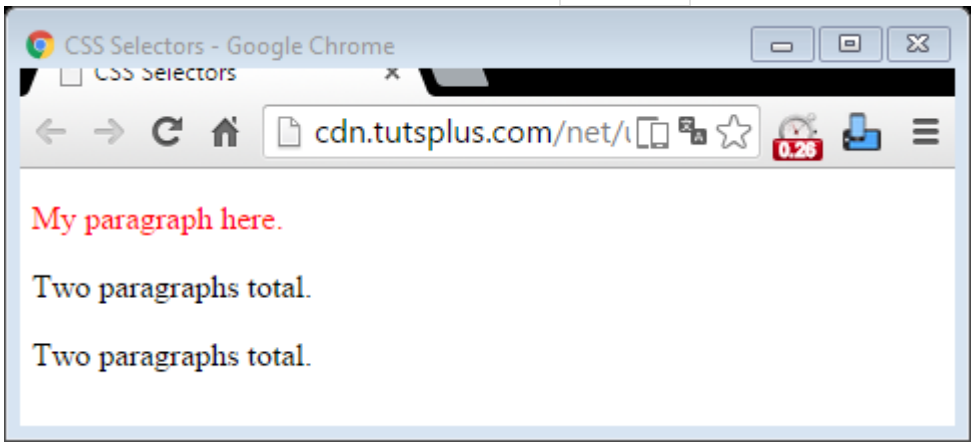
ในความเป็นจริง คุณจะไม่ค่อยพบว่าคุณใช้ pseudo class `only-child` นี้บ่อยนัก ไม่หรอก ในเมื่อมันมี บางทีคุณอาจต้องใช้มัน

มันช่วยให้คุณระบุ element ที่เป็นลูกตัวเดียวของ parent เท่านั้น (ตามชื่อ only child) ตัวอย่าง ลองดูโค้ดด้านบน เพียงแค่ paragraph ที่เป็นลูกตัวเดียวของ div นั้นจะมีสีแดง

ลองดู markup ต่อไปนี้ก็ได้

```
1    <div><p> My paragraph here. </p></div>
2
3    <div>
4        <p> Two paragraphs total. </p>
5        <p> Two paragraphs total. </p>
6    </div>
```

ในกรณีนี้ paragraph ภายใน `div` ตัวที่สองจะไม่ถูกเลือก จะแสดงผลแค่ใน `div` ตัวแรกเท่านั้น ตราบใดที่คุณมี child แค่ตัวเดียวใน element นั่นแหละ pseudo class `only-child` จะได้ทำงาน



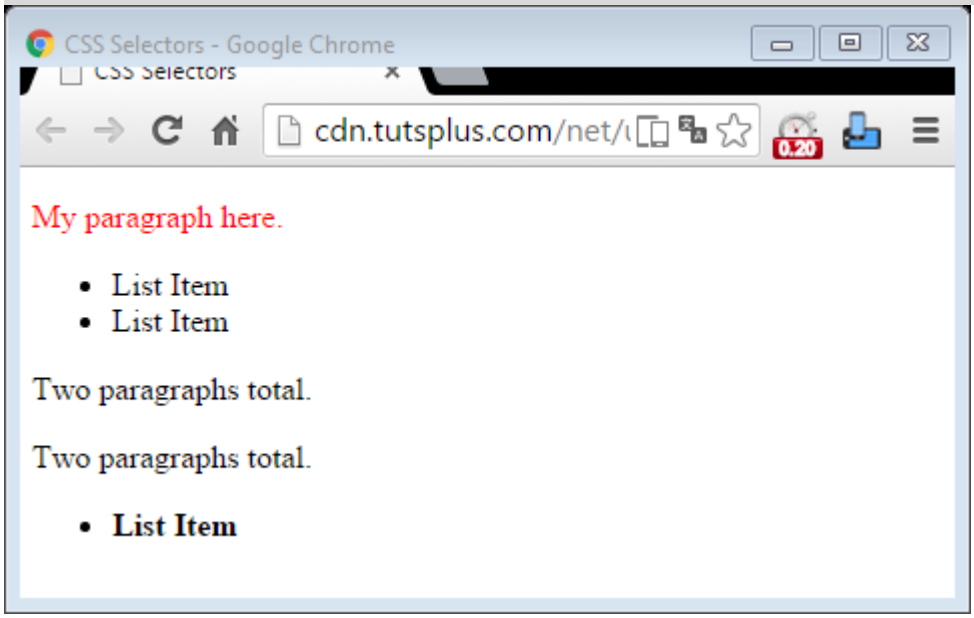
29. X:only-of-type

```
1    li:only-of-type {
2        font-weight: bold;
3    }
```

pseudo class อันนี้สามารถใช้ได้หลากหลาย มันจะระบุ element ที่ไม่มีพี่น้อง (sibling) ภายใน parent ที่ครอบมัน จากตัวอย่าง ลองระบุ ul ทั้งหมดที่มีแค่ list อันเดียว

แรกเลย ถามตัวคุณเองก่อนว่าจะทำได้อย่างไร คุณสามารถใช้ `ul li` แต่ว่ามันจะเลือก list*ทั้งหมด* ไม่ต้องเดาละมีคำตอบเดียวเลยคือใช้ `only-of-type`

```
ul > li:only-of-type {
  font-weight: bold;
}
```



30. X:first-of-type

pseudo class first-of-type ทั้งจะช่วยให้เราเลือกพี่น้องแรก (first sibling) ภายใน type ของมัน เพื่อให้เข้าใจมากขึ้น เราจะมาทดลองกัน Copy mark-up ต่อไปนี้ไปที่ code editor ของคุณ

```
<div>
  <p> My paragraph here. </p>
  <ul>
    <li> List Item 1 </li>
    <li> List Item 2 </li>
  </ul>

  <ul>
    <li> List Item 3 </li>
    <li> List Item 4 </li>
  </ul>
</div>
```

และก็ อย่าเพิ่งอ่านอะไรถัดไปนะ ลองเดาว่าจะทำอย่างไรจึงจะเลือกเฉพาะ *"List item อันที่ 2"*ได้ เมื่อคุณพบคำตอบแล้ว (หรือยอมแพ้ 555+) อ่านต่อเลย

<u>เฉลยวิธีที่ 1</u>

มีวิธีการมากมายที่จะแก้ปัญหา เราจะ review แค่บางอัน เริ่มจากการใช้ `first-of-type`

```
ul:first-of-type > li:nth-child(2) {
  font-weight: bold;
}
```

โค้ดสั้นกระชับนี้บอกว่า "ค้นหา unordered list ตัวแรกใน page จากนั้นหาเฉพาะที่มีลูกที่เป็น list ตามมา จากนั้นเอาแค่ตัวที่สองที่อยู่ในนั้น

เฉลยวิธีที่ 2

ตัวเลือกอื่นคือใช้ adjacent selector

```
1    p + ul li:last-child {
2        font-weight: bold;
3    }
```

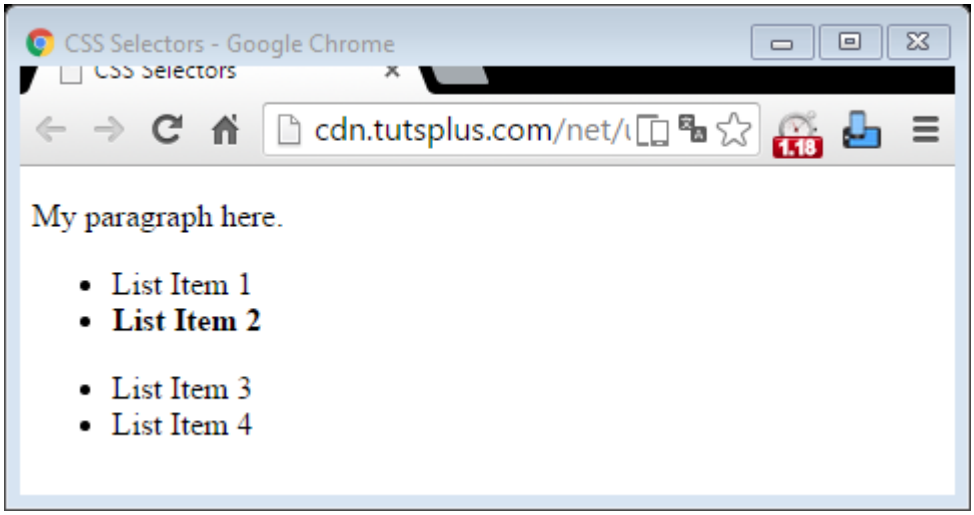
ในเหตุการณ์นี้ เราจะหา `ul` ที่ตามติดมาด้วย `p` tag และค้นหา element ที่เป็นตัวสุดท้าย

เฉลยวิธีที่ 3

เราสามารถรู้สึกได้ทั้งรำคาญหรือสนุก? ตราบที่เราต้องการ selector เหล่านี้

```
1    ul:first-of-type li:nth-last-child(1) {
2        font-weight: bold;
3    }
```

ครั้งนี้ เราเลือก ul ตัวแรกบน page จากนั้นค้นหา list item ตัวแรก แต่ว่าเริ่มจากท้ายนะ! :)



สุดท้าย

ถ้าคุณกังวลเกี่ยวกับ browser รุ่นเก่าๆ เช่น Internet Explorer 6 คุณก็ต้องระวังในการใช้ selector ใหม่ๆ เหล่านี้ด้วย แต่อย่างว่า ถ้ามันทำให้คุณถูกขวางกั้นจากการเรียนรู้สิ่งเหล่านี้ หรือเหมือนคุณไม่ค่อยสบายใจ ไงก็ลองเช็ค browser-compatibility จากที่นี่ดู หรือว่าคุณอาจใช้ Dean Edward's excellent IE9.js script เพิ่มเข้าไปในโค้ดของคุณเพื่อทำให้ browser รุ่นเก่าๆ support เจ้า selector เหล่านี้ได้

จากนั้น เมื่อคุณทำงานกับไลบารี่ Javascript เช่น jQuery ที่แสนโด่งดัง จำไว้เสมอ ถ้าเป็นไปได้ให้พยายามใช้ native CSS3 selector เพียวๆก่อน แทนที่จะใช้จากตัว library มันจะทำให้ code ของคุณทำงานได้ไวขึ้น เพราะ engine ของ selector สามารถถูก parse ได้โดยตรง (native parsing)

เอาหล่ะ! ขอบคุณที่อ่านและติดตาม ผมหวังว่าคุณจะได้ trick ดีๆ ไปบ้าง ไม่มากก็น้อยหล่ะ!