

## CSS Selector 개념과 종류

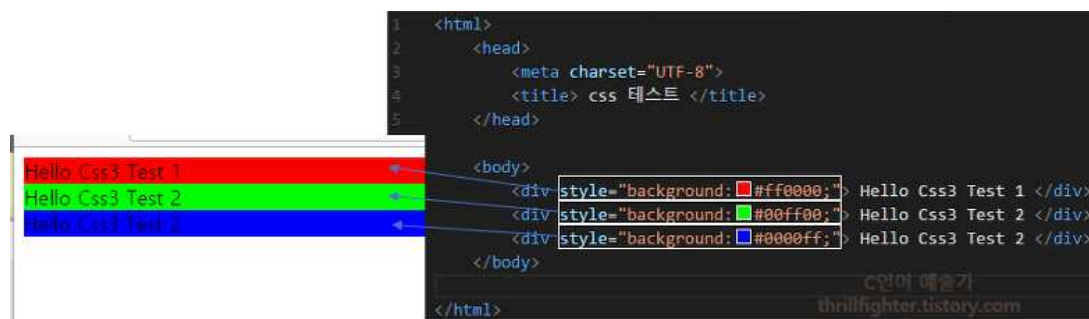
CSS는 HTML과 쌍을 이룬다. 최근 Web Page는 html 문서와 CSS 문서가 같이 사용되는데 html이 건축물이라면, CSS는 건축물을 꾸미는 인테리어라고 볼 수 있다. HTML 문서만 있더라도 Web Page가 굴러가는데는 충분하다. 하지만 건축을 해놓고 인테리어를 안해 놓으면 그 건물에 입주를 하지 않는다. 따라서 이들은 항상 같이 다닌다고 생각하면 된다.

### ☑ HTML 문서에 CSS 적용 방법 3가지

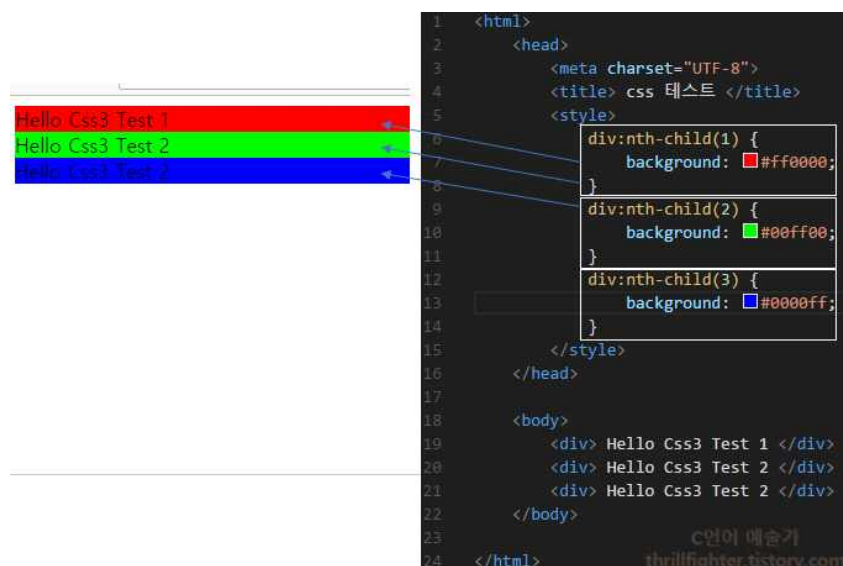
CSS를 HTML 문서에 적용하는 방법은 3가지가 있다.

#### ❖ HTML 문서 내부에서 처리하는 경우

① html tag에 style 속성을 주어 직접 적용하는 방법 (Inline Style Sheet)



② <head> tag 내에서 <style> tag로 사용하는 방법 (Internal Style Sheet)



## ❖ HTML 문서 외부에서 처리하는 경우 (External Style Sheet)

③ .CSS 확장자를 가진 파일을 html 문서에서 <link> tag로 불러들여서 CSS를 적용할 수 있다.



href 속성에 CSS 파일명만 적어줬다. html 파일과 CSS 파일이 있는 곳의 위치가 같기 때문이다. CSS 파일의 내용은 ②의 <style> tag의 내용을 그대로 옮겨온 것이다. Web Browser 실행 결과는 동일하므로 생략한다.

①은 이해가 가더라도 아직 ②, ③의 CSS 문법은 어색할 것이다. div나 nth-child와 같은 것 때문일텐데 이것은 Selector(Selector)라는 것이다. 말 그대로 스타일을 적용할 타겟을 선택한다는 것이다. 좀 구체적으로 알아보도록 하자.

## ☑ Selector 란?

CSS는 html 문서에 인테리어를 한다. 건축물이라면 침실을 이렇게 꾸미고, 화장실은 저렇게 꾸미고, 좀 더 세부적으로 침실의 전등은 Led 등으로 화장대는 어디 제품으로 등등.. 꾸미고자 하는 타겟이 있어야 한다.

Selector는 타겟이 어떤 것인지를 알려주는 역할을 한다. 앞서 div라든지 nth-child는 선택될 대상을 알려주는 Selector다. div는 html tag 이름이다. 이렇게 tag를 선택할 수 있는데 이를 tag Selector라고 한다. div tag Selector가 쓰이면 div tag에 스타일을 적용하라는 뜻이다.

div Selector 뒤에 :nth-child 역시 Selector 인데 좀 더 세부적인 Selector이다. 이 Selector는 해당 tag(여기서는 div)의 순서를 말한다.

div: nth-child(1) div tag 중에 첫 번째에 적용

div: nth-child(2) div tag 중에 두 번째에 적용

div: nth-child(3) div tag 중에 세 번째에 적용

사실 nth-child Selector를 div tag Selector와 같이 쓰지는 않는데 우선 설명의 편의상 이렇게 사용했다. nth-child Selector는 리스트 tag의 <li> tag와 같이 일련의 목록 tag와 같이 쓰인다.

Selector를 사용하여 스타일을 주는 방법은 다음과 같다.

```
Selector {  
    속성 : 속성값;  
}
```

이제는 앞 예제의 CSS 문법이 눈에 들어올 것이다.

## ☑ Selector의 종류

Selector는 tag Selector 말고도 다양하다. 속성 Selector라는 것도 있고, 속성값 Selector도 있다. 링크 Selector, 부정 Selector, 문자 Selector 등등 다양하다.

다음은 대표적인 Selector들이다. (구체적인 속성과 속성값은 생략한다.)

### ❖ Universal Selector

```
* { 속성: 속성값; }
```

html 문서 전체에 적용할 때 사용한다.

### ❖ tag Selector

```
p {color:#ff0000;}
```

선택된 tag에만 스타일을 적용할 때 사용한다.

### ❖ id Selector

```
#idname { 속성: 속성값; }
```

id 속성이 idname인 tag에 스타일을 적용한다.

### ❖ class Selector

```
.classname { 속성: 속성값; }
```

class 속성 값이 classname인 tag에 스타일을 적용한다.

### ❖ Child(하위, Descendant) combinator

```
div p { 속성: 속성값; }
```

tag와 tag 사이에 띄어 쓰기가 되어 있다. 먼저 쓰인 tag는 뒤에 쓰인 tag의 상위 tag다. div tag 내부에 있는 p tag에 적용한다. 띄어쓰기 대신 다음과 같이 사용해도 된다.

```
div > p { 속성: 속성값;}
```

예를들어 보자.



두 개의 테이블을 만들었는데 각각의 테이블에 class 속성을 주었다. 첫 번째 테이블은 class="mytable1", 두 번째 테이블은 class="mytable2"이다. 두 테이블의 첫 번째 행은 class="row1"의 속성을 갖는다. 이제 CSS 파일을 보자.

**.mytable1 .row1 td:nth-child(1)**

해석하면 .mytable1 클래스 속성을 가진 곳에서 row1 클래스 속성이 있는 곳을 찾는다. 그 다음 하위 Selector로 tag td를 찾는데 :nth-child(1) Selector가 붙어있으므로 첫 번째 td tag의 내용에 빨간색 배경 속성을 적용한다.

#### ❖ General sibling combinator

**p~div { 속성: 속성값;}**

p tag와 동일한 레벨에 있는 div tag에 속성을 적용한다. 이 때 p tag에는 스타일이 적용되지 않음.  
p~div라고 하면 p tag와 동일 레벨의 첫 번째 div tag.

이 외에도 다음과 같은 Selector들이 있다.

#### ❖ Attribute Selector

**a[href] { 속성: 속성값; }**

a tag의 href 속성이 있는 경우 스타일 적용

#### ❖ 속성값 Selector

**a[href="http://thrllfighter.tistory.com"] { 속성: 속성값;}**

속성값이 일치하는 경우 스타일 적용

### ❖ 문자 Selector

```
#idname:first-letter { font-size : 30px; }
```

id속성이 idname인 tag의 내용의 첫 번째 문자에 스타일 적용

### ❖ 라인 Selector

```
.classname:first-line { color : red; }
```

class 속성의 값이 classname인 tag의 첫 줄 글씨 색을 빨간색으로 한다.

### ❖ 반응 Selector

```
.mytable1 .row1 td:nth-child(1):hover {  
    background: #0000ff;  
}
```

:hover Selector는 마우스 포인터가 올라가 있을 때만 반응하는 스타일을 적용한다.

### ❖ 상태 Selector

focus, checked Selector.

## font attribute

CSS font에 관련된 주요 속성은 다음과 같은 것들이 있다.

font-family, font-size, font-style, font-variant, font-weight

다음 역시 폰트에 관련된 속성이다.

letter-spacing, word-spacing, line-height

속성이 이름만 봐도 대략 느낌이 온다. 이제 하나씩 학습하도록 한다.

### ☑ font-family

```
<div style="font-family: Nofont, 'Malgun Gothic', 굴림, 돋움;">
<p> 폰트 테스트, font Test, 1234567890 </p>
</div>
```

속성 이름 font-family에서 family는 집합을 의미한다. 코드를 보면 폰트(글꼴)이 여러 개가 지정되어 있음을 알 수 있다. font-family의 속성 값들은 왼쪽에 있는 것이 우선순위를 갖는다. 위에서는 Nofont라는 폰트가 최 우선순위지만, 이런 폰트가 없다면 다음 우선순위인 Malgun Gothic 폰트가 사용된다. 물론 font-family를 설정하지 않아도 기본 글꼴로 사용된다.

'Malgun Gothic' 과 같이 폰트이름에 띄어쓰기가 있다면 따옴표로 묶어주면 된다. 외부 따옴표가 "쌍따옴표이므로 이와 구별되기 위해 내부에서는 '홀따옴표'를 사용했다.

### ☑ font-size

폰트 크기를 지정한다.

```
<div style="font-size: 16px">
<p> 폰트 테스트, font Test, 1234567890 </p>
</div>
<div style="font-size: xx-large">
<p> 폰트 테스트, font Test, 1234567890 </p>
</div>
...
```

폰트 크기를 나타내는 단위는 두가지 종류로 나눌 수 있다.

#### A. 절대 크기

font-size가 지정되지 않으면 16px의 크기를 갖는다.(아래 참고) 또는 다음과 같이 절대적인 크기로 지정할 수 있다.

xx-large	32px	24pt	2pc
x-large	24px	-	-
large	18px	-	-
medium	16px	12pt	1pc
small	13px	-	-
x-small	10px	-	-
xx-small	9px	-	-

## ※ 단위

- ☞ px : 픽셀(pixel) 단위다. font-size가 지정되지 않으면 기본 사이즈인 16px로 표시된다.
- ☞ pt : 일반적인 문서 프로그램에서 사용되는 폰트 크기 16px = 12pt

그 밖에 in(인치), cm(센티미터), mm(밀리미터), pc(1pc = 12pt) 등이 있다.

## B. 상대 크기 단위 (em, %)

말 그대로 상대적인 크기를 나타낸다. 따라서 기준이 되는 font-size 속성이 앞에서 지정되거나 또는 감싸는 tag에 지정되어 있어야 한다. (이런 속성을 상속된 속성이라고 한다.)

만약 앞에서 지정된 font-size 속성이 없다면 디폴트인 16px를 기준(1em = 16px)으로 한다.

```
<div style="font-size: 20px">
  <p> 20px </p>
  <p style="font-size: 1.2em"> 20px 의 1.2배 </P>
  <p style="font-size: 0.5em"> 20px 의 0.5배 </P>
  <p style="font-size: 1em"> 20px </P>
</div>
```

```
<div style="font-size: 20px">
  <p> 20px </p>
  <p style="font-size: 50%"> 20px 의 50% </P>
  <p style="font-size: 150%"> 20px 의 150% </P>
  <p style="font-size: 100%"> 20px 의 100% </P>
</div>
```

## ☑ font-style

글꼴의 기울임에 관련된 속성이다.

```
<div style="font-family: 'Times New Roman', Times, serif; font-size: 20px; font-style: italic;">
  <p style="font-style: normal"> font-style, 폰트 스타일, 12345 </P>
  <p style="font-style: oblique"> font-style, 폰트 스타일, 12345 </P>
  <p style="font-style: italic"> font-style, 폰트 스타일, 12345 </P>
```

```
<p style="font-style: inherit"> font-style, 폰트 스타일, 12345 </P>
</div>
```

- ☞ normal : 글꼴의 기본 형태로 표시
- ☞ oblique : 글꼴을 기운형태로 표시
- ☞ italic : italic은 기운 글꼴이지만 oblique와는 다를 수 있다. 필기체와 같은 모양으로 글꼴에 따라서 표현이 될 수도 있고 안될 수도 있다. 만약 글꼴이 italic을 지원하지 않으면 oblique로 표현한다.
- ☞ inherit : 앞에서 설정된 또는 감싼 tag가 설정한 font-style을 따른다.

## ☑ font-variant

```
<div>
<p style="font-variant: normal"> abcdEFG </p>
<p style="font-variant: small-caps"> abcdEFG </p>
</div>
```

font\_variant의 속성값을 small-caps로 하면 소문자를 작은 대문자로 변경한다. 특이한 속성이다.

## ☑ font-weight

폰트 굵기를 지정한다.

```
<div style="font-weight: 700">
<p style="font-weight: normal;"> Hello CSS3, 하이~!, 1234 </p>
<p style="font-weight: bold;"> Hello CSS3, 하이~!, 1234</p>
<p style="font-weight: bolder;"> Hello CSS3, 하이~!, 1234</p>
<p style="font-weight: lighter;"> Hello CSS3, 하이~!, 1234</p>
<p style="font-weight: 100;"> Hello CSS3, 하이~!, 1234</p>
<p style="font-weight: 400;"> Hello CSS3, 하이~!, 1234</p>
<p style="font-weight: 700;"> Hello CSS3, 하이~!, 1234</p>
<p style="font-weight: 900;"> Hello CSS3, 하이~!, 1234</p>
</div>
```

font-weight는 100, 200, 300, .... 900 까지 값을 가질 수 있다.

- ☞ normal : 기본 굵기 (400)
- ☞ bold : 굵은 굵기 (700)
- ☞ bolder : "상대적으로 굵게"를 의미한다. 위 코드는 div tag의 font-weight:700 을 상속받으므로 700보다 굵은 값을 의미한다.
- ☞ lighter : 위 코드에서는 700 보다 가는 값



## ☑ letter-spacing

글자 간의 간격, 즉 자간을 지정한다.

```
<div>
<p style="letter-spacing: normal;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: -20px;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: -10px;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: -3px;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: 0px;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: 3px;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: 10px;"> Hello CSS3, 하이~!, 1234</p>
<p style="letter-spacing: 20px;"> Hello CSS3, 하이~!, 1234</p>
</div>
```

단위는 픽셀(px), cm, in 등 폰트 사이즈 단위를 쓸 수 있는데 글자간격은 세밀히 조정해야하므로 픽셀(px)을 쓰도록 한다. 음수는 자간이 좁아지고, 양수는 자간이 넓어진다.  
normal과 0px은 동일하고 폰트의 기본 글자 간격이다.

## ☑ word-spacing

단어간의 간격이다. 띄어쓰기를 기준으로 간격을 조정한다.

```
<div >
<p style="word-spacing: 16px"> 안녕하세요, Hellow!, 1234 </P>
<p style="word-spacing: 16pt"> 안녕하세요, Hellow!, 1234 </P>
<p style="word-spacing: 1in"> 안녕하세요, Hellow!, 1234 </P>
<p style="word-spacing: 1pc"> 안녕하세요, Hellow!, 1234 </P>
<p style="word-spacing: 1cm"> 안녕하세요, Hellow!, 1234 </P>
</div>
```

letter-spacing과 마찬가지로 여러 단위로 사용할 수 있지만 세세한 조정을 위해 px(pixel)로 사용한다.

## 가상 클래스 Selector(Link, visited, active, hover, focus, nth-child)

Selector란 tag, id, class 등과 같이 html 문서에 존재하는 것들을 지정하는 것이다. 반면에 가상 클래스 Selector는 어떤 상태를 지정한다. 이런 상태는 tag, id, class 처럼 html문서 상에 있는 것이 아니다. 따라서 가상 Selector라고 한다.

예를들어 상태를 지정하는 가상 Selector에는 마우스를 올려 놓은 상태를 의미하는 hover Selector, 방문하지 않은 링크를 나타내는 link Selector, 이미 방문했던 링크를 선택하는 visited Selector 등이 있다.

가상 클래스 Selector를 사용해서 상태에 따른 스타일을 지정하는 것을 학습하도록 한다.

### ☑ link, visited 가상 Selector



웹페이지에 있는 링크들은 기본적으로 파란색이다. 사용자가 링크를 클릭을 하면 오른쪽과 같이 보라색으로 변한다. 클릭 전과 후의 링크의 상태가 변한 것이다. 우리는 클릭 전후의 상태를 상태Selector로 선택해서 스타일을 입힐 수 있다.



☞ : link

방문하지 않은 링크를 선택

☞ : visited

방문한 링크를 선택

☞ a:link { color:darkslategrey;}

설명 a tag 중에서 링크를 방문하지 않은 tag의 내용을 색을 darkslategrey로 지정

☞ `a:visited { color:aquamarine;}`

설명 a tag 중에서 방문한 tag의 내용을 색을 aquamarine으로 지정

## ☑ active, hover 가상 Selector



위 예의 문장에 마우스를 올려놓았을 때 빨간 색으로 바뀌도록 추가하려면 다음과 같이 하면 된다.

```
<style>
  p:active{color: gold;}
  p:hover{color: red;}
</style>
```

그런데 이렇게 하면 active 가상 Selector는 동작하지 않는다. 두 개의 Selector가 동시에 동작하면 나중에 있는 것만 적용되기 때문이다. 주의하도록 하자.

☞ `: active`

마우스로 클릭한 상태

☞ `:hover`

마우스를 위에 올린 상태

☞ `p:active{color: gold;}`

p tag의 내용을 클릭하면 색을 gold 한다.

☞ `p:hover{color:red;}`

p tag의 내용 위에 마우스를 올려 놓으면 색을 red로 한다.

## ☑ focus 가상 Selector

```
1 <html>
2 <head>
3 <style>
4   p:active{color: gold;}
5   input:focus{ background: green;}
6 </style>
7 </head>
8 <body>
9   <p>클릭해보세요.</p>
10  <input type="text">
11 </body>
12 </html>
```



☞ : focus

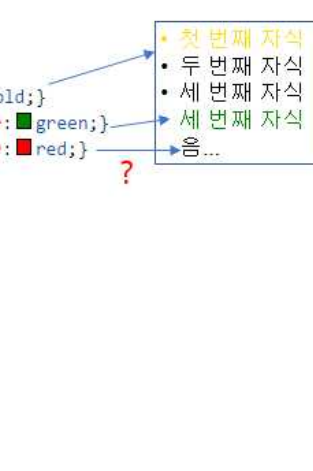
포커스를 가질 때(예를들어 input 박스에서 텍스트를 입력 가능한 상태)

☞ input:focus { background : green;}

input 박스에 포커스가 있을 때 색을 green으로 바꿈

## ☑ first-child, last-child, nth-child(n), nth-last-child(n) 가상 Selector

```
1 <html>
2 <head>
3 <style>
4   li:nth-child(1){color: gold;}
5   li:nth-last-child(2){color: green;}
6   li:nth-last-child(1){color: red;}
7 </style>
8 </head>
9 <body>
10 <ul>
11   <li>첫 번째 자식</li>
12   <li>두 번째 자식</li>
13   <li>세 번째 자식</li>
14   <li>네 번째 자식</li>
15   <div>음...</div>
16 </ul>
17 </body>
18 </html>
```



☞ :first-child

첫 번째 tag 요소 선택

☞ :last-child

마지막 tag 요소 선택

☞ :nth-child(n)

n 번째 tag 요소 선택

☞ `:nth-last-child(n)`

끝에서 부터 n 번째 tag 요소 선택

☞ `li:nth-child(1)`

li tag가 있을 때 이와 동일 레벨에 있는 tag들 중에 1번째 요소를 선택한다.

☞ `li:nth-last-child(2)`

li tag가 있을 때 이와 동일 레벨에 있는 tag들 중에 마지막 에서 2번째 요소를 선택한다.

☞ `li:nth-last-child(1)`

li tag가 있을 때 이와 동일 레벨에 있는 tag들 중에 마지막 에서 1번째 요소를 선택한다. 위 예에서 li tag와 동일 레벨에 있는 마지막 tag가 p tag다 이 tag가 순서에 포함되지만 선택되지는 않는다 따라서 위 코드에서 적용이 안된 것이다.

참고로 `li:nth-last-of-type(1)` 이라고 하면 p tag는 세지 않고 li tag에 대해서만 순서를 따진다. `li:last-of-type`과 동일하다. `li:nth-first-of type(1)` 역시 마찬가지다. `li:first-of-type`과 동일하다.

또한 가상 엘리먼트(element) Selector라는 것도 있다. CSS로 문서 내용을 삽입할 때 사용하는데 (`::before`, `::after`) 우선 이런 것이 있다.

## Layout 레이아웃 속성(display, inline, block)

html 문서에 문자나 이미지 등의 콘텐츠가 위치될 때 다양한 tag들로 둘러싸인다. 이런 tag들은 display 속성이 갖는 속성값(inline, block)에 따라서 콘텐츠의 표시되는 방식에 차이가 생기게 된다.

inline 속성값을 디폴트로 갖는 tag는 <a> tag가 있고, block 속성값을 디폴트로 갖는 tag는 <div>, <p> 등의 tag가 있다.

설명에 앞서 이 tag들을 사용한 간단한 예제를 보자.



코드와 결과를 보면< div> tag로 둘러싸인 내용(block) 앞뒤로 줄바꿈이 되어 있다. block 속성을 가진 tag는 욕심쟁이라서 한 라인 전체를 혼자서 다 쓰는 성질을 갖기 때문이다. 반면에 <a> tag는 한 라인 내에서 다른 콘텐츠들과 어울어져 있다.

이런 기본적인 성질 외에도 inline 속성과 block 속성을 갖는 tag들은 몇가지 구별되는 차이점을 가진다.

### ❖ display 속성이 inline인 tag의 성질

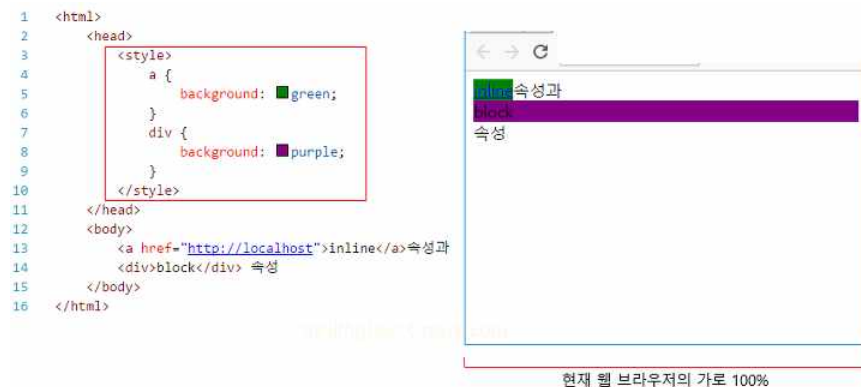
- ☞ width, height, margin 속성들을 가질 수 없다.(이런 속성을 지정하더라도 무시된다.)
- ☞ 줄 간격의 조정은 line-height 속성을 사용한다.
- ☞ inline 속성을 갖는 tag가 연속으로 위치할 때는 두 내용을 구별할 수 있을 정도의 간격이 생긴다.

### ❖ display 속성이 block인 tag의 성질

- ☞ width, height, margin 속성을 가진다.

이런 성질로 인해서 inline 속성값을 갖는 tag 안에 block 속성값을 갖는 tag가 쓰일 수 없다. (반대로 block 속성값을 갖는 tag 안에 inline 속성값을 갖는 tag는 쓰일 수 있다.)

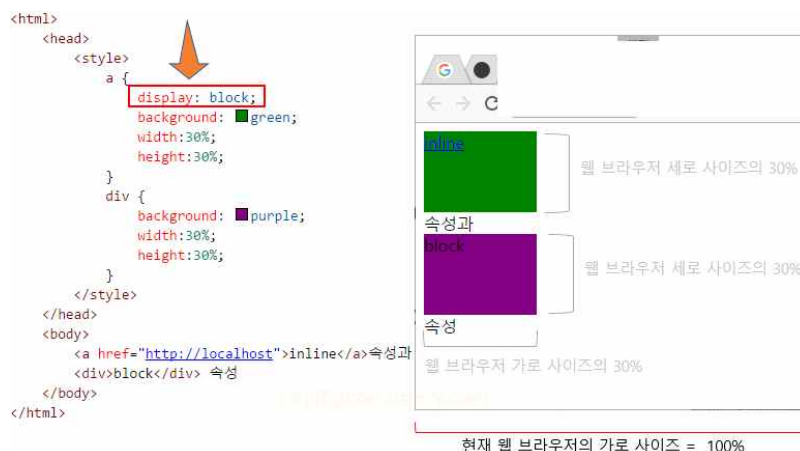
다음은 a tag와 div tag에 background 속성을 입혀서 두 tag가 문서 내에서 차지하는 공간을 비교한 것이다. block 속성을 갖는 tag는 기본적으로 width:100%의 값을 갖는다 이 값은 현재 웹 Browser 창의 가로 사이즈를 나타낸다.



다음처럼 a tag와 div tag에 width와 height 속성을 주면, a tag에는 width와 height 속성이 적용되지 않음을 알 수 있다. margin 속성 역시 div tag에만 적용되고 a tag에는 무시될 것이다.



만약 a tag에 width와 height 속성을 주고 싶다면 display 속성을 추가한다.

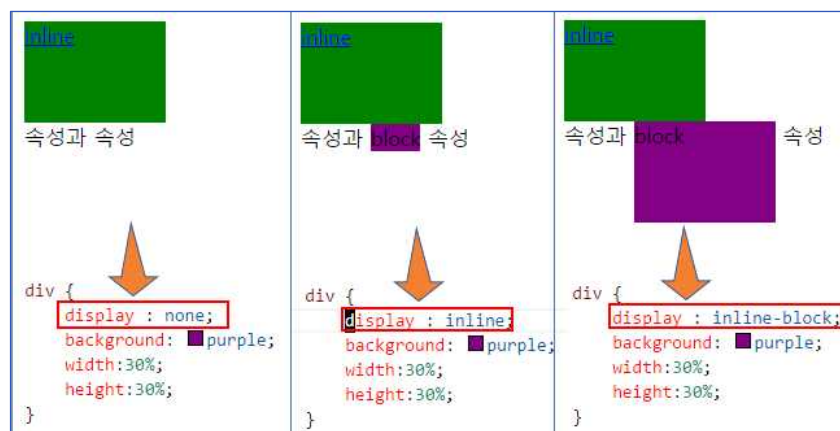


## ◎ display 속성값

- ☞ none : 해당 요소를 제거
- ☞ inline : 해당 요소를 inline 으로 바꿈
- ☞ block : 해당 요소를 block 으로 바꿈
- ☞ inline-block : inline 속성(한 줄 내에서 다른 요소들과 연결됨)과 block 속성(width, height, margin 등의 속성)을 동시에 가짐

레이아웃에 관련된 내용은 매우 중요한데 위 내용 뿐 아니라 box-sizing, 마진 겹침 등에 대한 내용을 제대로 이해해야 정확한 레이아웃을 설계할 수 있다.

간혹 레이아웃이 겹치거나 보기에 이상한 경우가 생기는 사이트들은 이런 기본적인 레이아웃 설계를 잘 못한 것이다.





## 상속, parent, child 용어 정리

CSS를 학습하면서 몇가지 정리되지 않은 용어들을 정리하도록 한다.

### ☑ html element

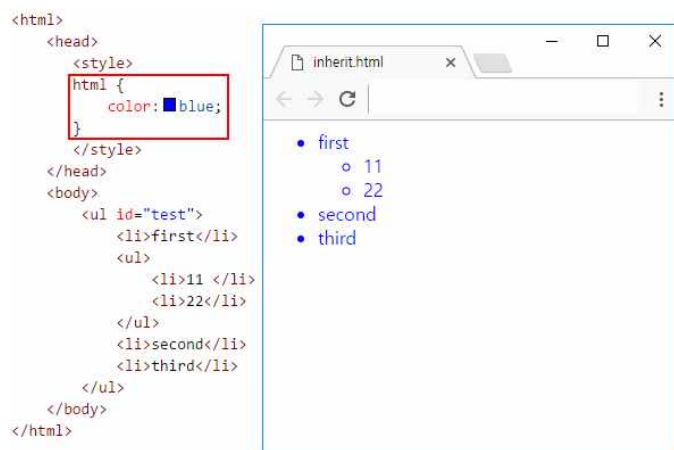
html 요소라고도 하며 tag를 의미한다. 나중에 다루게 될 Box Model에 대해서 간단히 언급하면, html 요소는 각각 박스(box)로 둘러싸여져 있음을 말하며 박스의 크기를 결정하는 속성들을 CSS를 사용해 설정, 변경할 수도 있다. 각 html 요소의 박스 크기에 따라 다른 html 요소와의 상호작용을 통해 문서에 배치되는 모습이 결정된다.

예를들어 html tag, body tag, p tag, div tag들은 박스로 이루어져 있으며 다른 요소들의 내부에 배치될 수도 있고 다른 요소와 나란히 배치될 수도 있다.

### ☑ parent, child

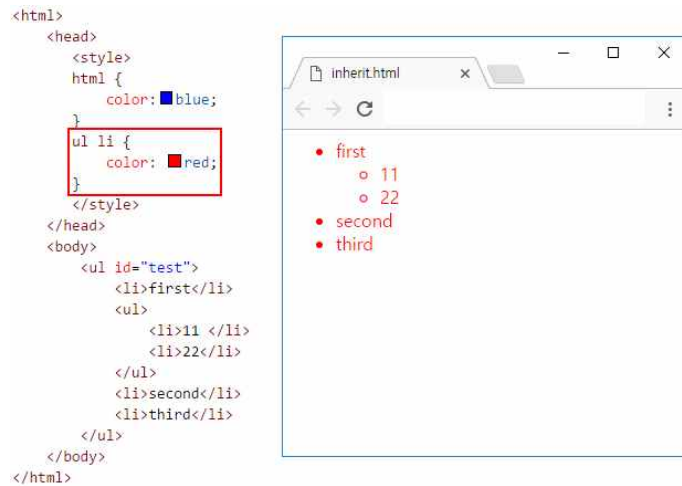
html 요소들간에는 parent, child관계가 형성되는데 이런 관계는 상대적이다. 예를들어 parent는 포함하는 tag, child은 포함되는 tag를 말한다. 따라서 html tag는 최상위 tag이므로 모든 tag의 parent가 된다. 반면에 body tag는 html tag의 child이지만 body 내부에 있는 tag들에 대해서 parent가 된다. 그리고 parent, child관계를 통해 속성값의 상속이 가능하게 된다. 속성값은 parent로 부터 child으로 전파된다. 속성값이 상속되면 어떤 일이 일어나는지 예제를 통해서 알아보자.

### ❖ parent의 속성값이 child에게 상속되는 예



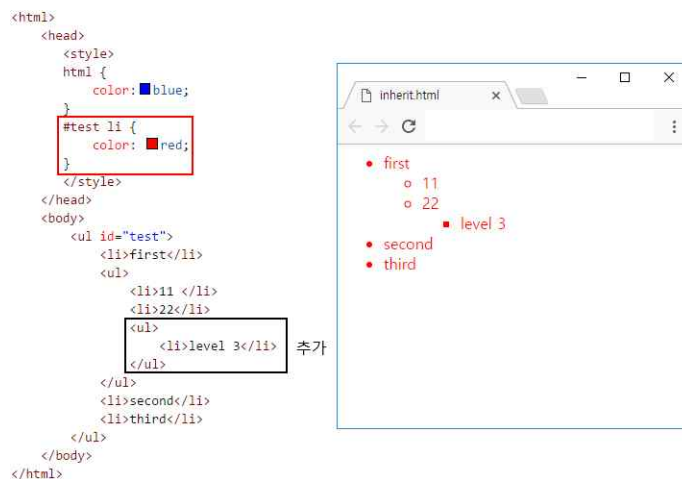
html Selector에 color 속성을 blue로 줬는데 문서의 글씨가 모두 blue로 변했다. parent의 속성값 blue가 child에게 상속됐기 때문이다.

이 코드에 li tag의 색을 바꾸는 코드를 추가해보자.



이렇게 직접 속성값을 주면 parent로 부터 상속된 값은 무시된다. 다시말해 속성값을 따로 주지 않은 경우에만 상속된 값이 적용된다.

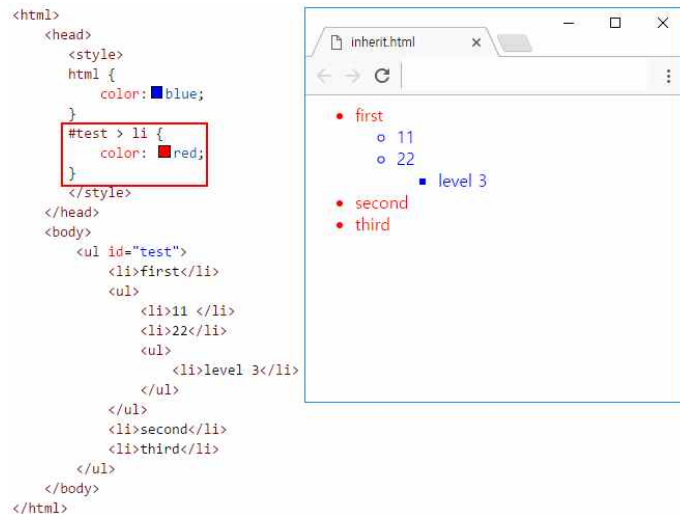
이번에는 id Selector를 사용하여 색을 바꿔보자.



id가 test 인 tag의 직계 child으로 세 개의 litag가 있다. 문서에 표시된 first, second, third에 해당한다. CSS에서는 이 세 개의 litag에 속성값 red를 적용했는데 보시다시피 (손자?, 증손자?) tag 까지 속성값이 적용되었다. 왜냐면 litag에 적용된 속성값이 그 하위 tag까지 상속되기 때문이다.

여기에서 빨간 박스의 코드를 다음과 같이 바꿔보자.

```
#test > li {
  color: red;
}
```



child Selector에 > 를 사용하면 직계 child 만을 뜻한다. 그래서 위 코드는 직계 child 보다 더 아래 자손들에게는 속성값 red가 상속되지 않게된다. 그림을 보면 더 하위 후손은 html Selector로 지정한 색 blue를 상속받아 파란 색 글씨가 된다.

### ❖ 상속이 되지 않는 속성

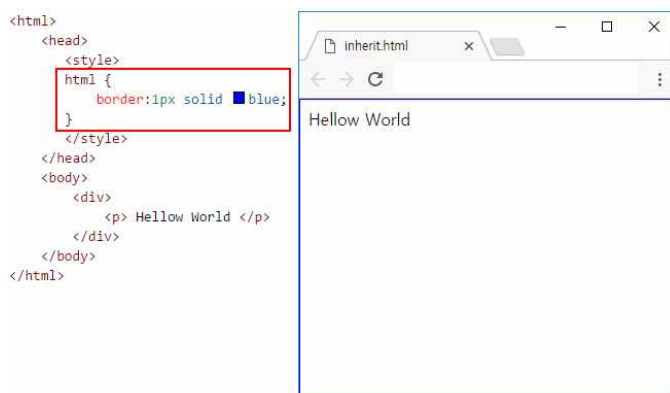
하지만 모든 속성이 color처럼 상속되는 것은 아니다. 예를들어 border 속성은 상속되지 않는다. 상속되는 속성과 상속되지 않는 속성은

<https://www.w3.org/TR/CSS2/propidx.html>

위 사이트의 표를 참고하면 된다. (Inherited? 가 yes 면 상속되는 것)

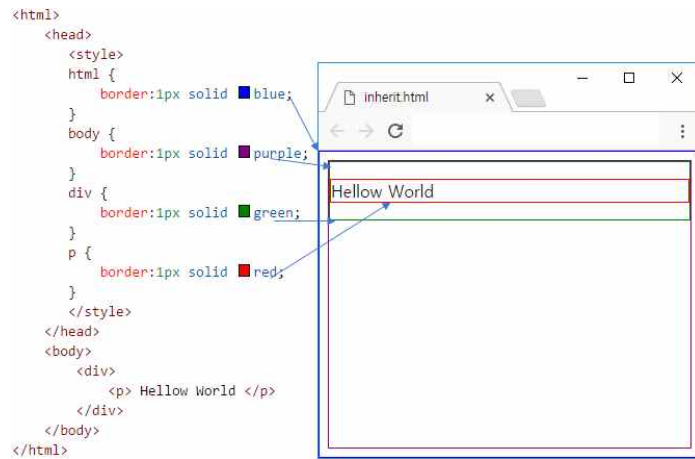
CSS에는 많은 속성이 있으므로 상속 유무를 외우려 하면 지칠 것이다. 자주 쓰는 몇개의 속성만 알아두고 감각적으로 익히면 된다. 웹 애플리케이션을 만드는 것은 0, X를 맞추는 퀴즈가 아니므로 코드를 작성하고 결과를 확인해 가면서 상속여부를 알아낼 수도 있다.

예를들어 border 속성 의 경우를 살펴보자.



html tag에 border 속성을 줬지만 child tag인 body, div, p 에는 적용되지 않았다. 만약 border 속성이 상속된다고 가정해 보면 꽤 골치아픈 일이 일어날 것이다. (다음 예제 참고)

다음은 child tag에 border 속성을 직접 줬본 것이다.



이렇게 border 속성은 상속이 안되고 직접 지정해야 하는 속성이다. 그리고 이 그림에서 알 수 있는 것은 모든 요소(tag)는 box로 이루어져 있다는 것이다.

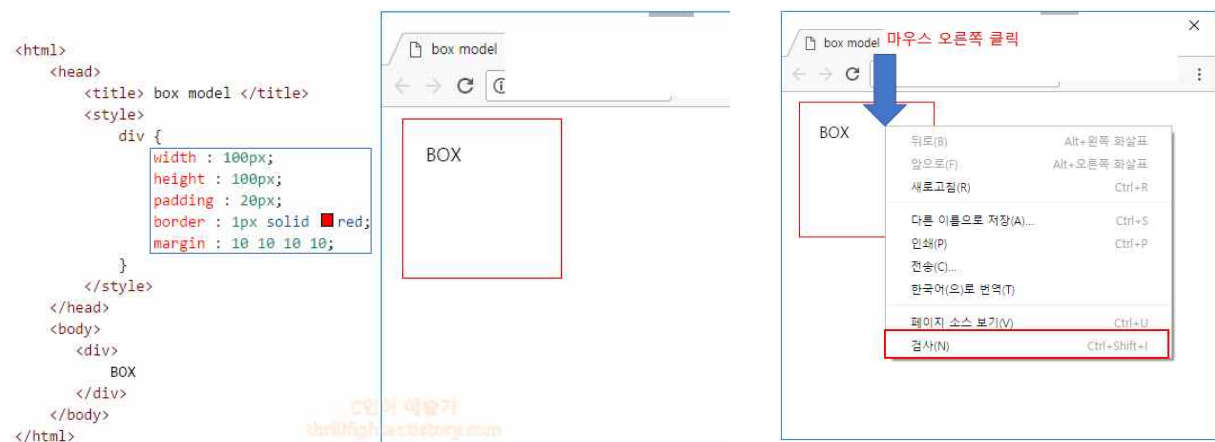
## Box Model (width, height, padding, border, margin 속성)

Box Model은 Web Page를 만드는데 있어 기본 중에 기본이라고 할만큼 중요한 내용이다. 그럼에도 소홀해지기 쉬운 내용인데 제대로 이해하고 있지 못하면 Web Page의 레이아웃을 설계하다 문제가 발생했을 시에 원인 파악이 힘들어질 수 있다.

Box Model은 html 요소(element)들이 문서 내의 공간을 차지하는 부피에 관한 규칙이다. 차지하는 모양이 네모 박스(box) 모양이라서 Box Model이라고 한다.

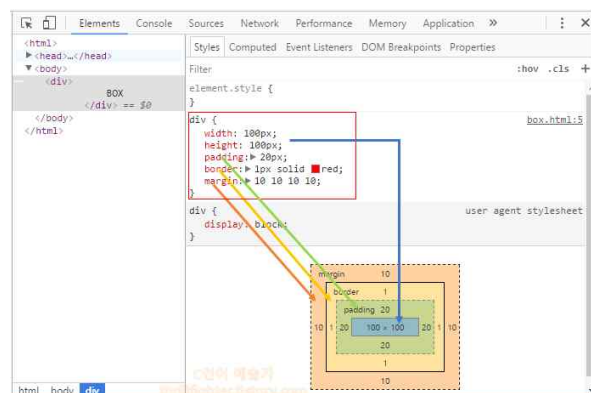
tag들은 내용(content)를 감싸고 있고 내용은 문자 또는 이미지 등이 있으며 문서에서 일정한 부피를 박스 모양으로 차지한다. 그리고 내용들 간에 간격이 정해지는 방식도 존재한다. 이제부터 하나씩 알아가보자.

### ☑ Box Model



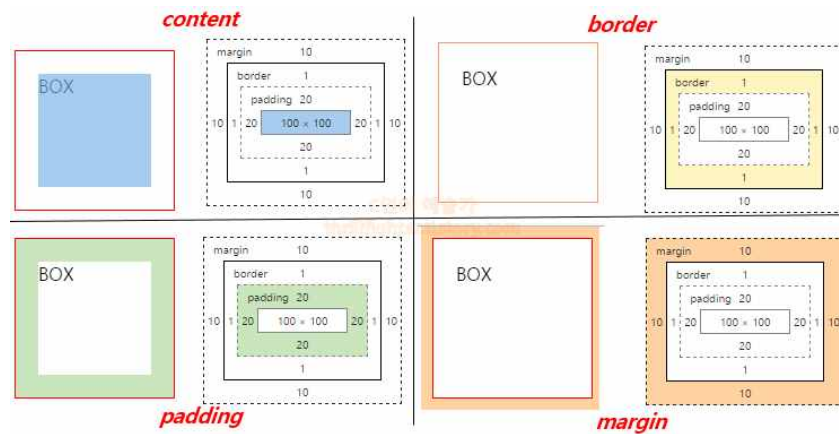
그림은 html 코드와 Browser에서 코드를 실행한 모습이다. div tag에 CSS 스타일로 5가지 속성을 주었다. 이 속성은 divtag 요소의 박스의 모습을 결정하는 속성들인데 하나하나 따져볼 필요가 있다. 이 때 크롬 Browser를 사용하면 매우 편리하게 각 속성을 살펴볼 수 있다.

크롬 Browser로 앞서 코드를 실행한 후에 div tag 안쪽 위치에서 마우스 오른쪽 클릭을 한 후 검사(N)를 클릭하면 다음과 같은 창이 열린다.



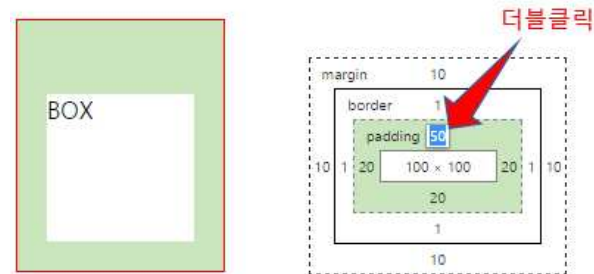
여러 층으로 구성된 박스들이 보이는데 이것은 div tag에 대한 박스로 4개의 층으로 이루어져 있다. 각각의 층은 div tag에 적용된 style에 대응되는데 그림에서 화살표로 표시해 놓았다.

각각의 박스(box)에 마우스를 갖대 대보면 Browser에 아래와 같이 표시가 된다.



각 속성들에 의해서 각 박스가 어떻게 결정되는지 정확히 알 수 있다.

한번 padding의 위쪽 부분을 넓혀보자.



content와 border 사이에 위쪽 공간이 늘어난 것을 확인할 수 있다. 이 것을 CSS 스타일의 속성 코드로 표현하면 다음과 같다.

`padding: 50px 20px 20px 20px;`

### ❖ padding

padding은 내용(content)과 border 사이의 간격을 결정하는 속성이다. 속성값은 top(위)쪽부터 시작해서 시계방향으로 순서대로 대응되는데 속성값이 생략될 때는 규칙이 있다.

ex) `padding : 50px`

top, left, bottom, right 모두 50px 만큼의 간격을 갖는다.

ex) `padding : 50px 20px`

top, left, bottom, right 이 각각 50, 20, 50, 20의 간격을 갖는다.

ex) padding : 50px 20px 30px

top, left, bottom, right 이 각각 50, 20, 30, 20 의 간격을 갖는다. 부연 설명하자면 bottom 값이 생략되면 top과 같은 값을, left의 값이 생략되면 right와 같은 값을 갖는다.

일반적으로 위와 같이 padding 속성으로 한번에 패딩(padding)을 정하나 다음과 같은 속성이 기본 속성이다. padding은 약식 속성인 것이다.

padding-top, padding-right, padding-bottom, padding-left

각각 패딩(padding)의 상, 우, 하, 좌에 대한 속성이다.

## ❖ border

border는 요소를 둘러싸는 테두리에 대한 속성이다. border 역시 padding과 마찬가지로 약식 속성이다.

ex) border: 1px solid red

1px의 두께, 실선, 붉은 색의 테두리를 갖는다.

원래 속성은 다음과 같다.

border-width : 1px;

border-style : solid;

border-color : red;

☞ border-width

값 : thin, medium(기본값), thick, 수치px 등등..

☞ border-style

값 : solid(실선), dashed(짧은선), dotted(점선), double(실선 두개), groove(음각), ridge(양각), inset(안쪽 입체감), outset(밖쪽 입체감), none(기본값), hidden(숨김)

☞ border-color

값 : 색상 코드

## ☞ margin

margin은 외부 테두리(border) 밖갈쪽의 공간을 결정한다. 이 공간은 다른 tag 요소가 침범할 수 없다. margin 속성 역시 padding과 border와 마찬가지로 약식 속성이다.

ex) `margin : 10px 20px 30px 40px;`

border 속성과 규칙은 같다. border 속성을 참고하자.

원래 속성은 다음과 같다.

`margin-top : 10px;`

`margin-right : 20px;`

`margin-bottom : 30px;`

`margin-left : 40px;`

`margin-top, margin-right, margin-bottom, margin-left`

마진(margin)의 상, 우, 하, 좌 에 대한 속성이다.

여기까지 기본적인 Box Model을 설명했는데 추가적으로 알아두어야할 box-sizing과 마진겹칩 이다.



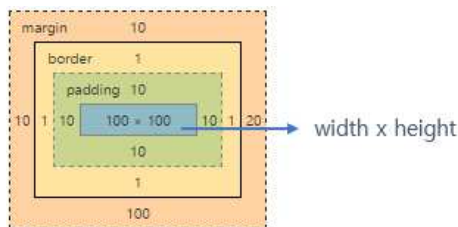
## box-sizing 속성과 margin 겹침 현상

앞장에서 Box Model에 대한 HTML tag 요소들이 문서 내에서 어떻게 배치되는지에 대해서 살펴보았다. 이를 결정하는 속성 다섯가지(width, height, padding, border, margin)에 대해 살펴보았다.

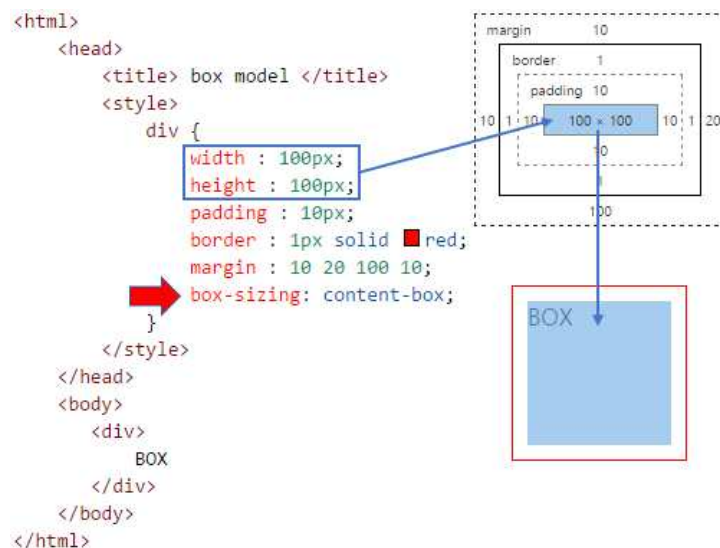
이 장에서는 이 규칙의 기준을 바꿀 수 있는 box-sizing이라는 속성을 살펴보고, 더 나아가 tag 요소들 간에 margin이 어떻게 적용되는지에 대해서도 학습한다. 앞장은 매우 중요한 내용이므로 반드시 이해하고 넘어가야 한다.

### ☑ box-sizing 속성

Box Model에서 tag 요소들을 둘러싼 4개의 층으로 된 박스들은 앞에서 언급한 5가지 속성에 의해서 결정된다. 그 중 width와 height 속성은 가장 안쪽에 있는 내용(content)의 가로와 세로 사이즈를 결정하는 속성이다.



크롬 Browser의 요소검사 기능(지난 포스팅 참고)을 통해서 width와 height 속성을 가시적으로 살펴볼 수 있다.



border(빨간 테두리)안쪽에 padding이 10px만큼 있고 그 안쪽에 파란 박스(100 x 100)의 콘텐츠(content) 영역이 있다. 이 영역은 CSS 속성 width와 height로 정해진다.

## ❖ box-sizing

코드를 보면 box-sizing이라는 속성 추가했는데 이 속성이 갖을 수 있는 속성값은 다음과 같다.

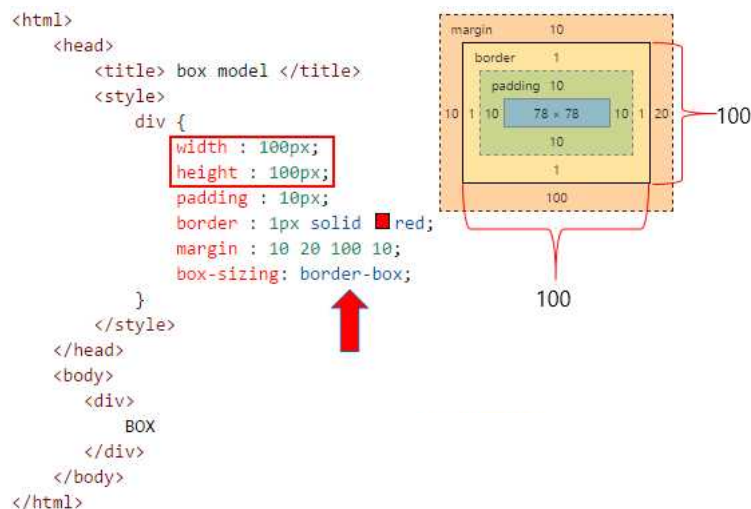
### ☞ content-box

width와 height 속성이 content의 가로와 세로의 길이를 설정하도록 한다. (기본값)

### ☞ border-box

width와 height 속성이 content와 padding과 border의 두께까지 합친 값으로 설정한다.

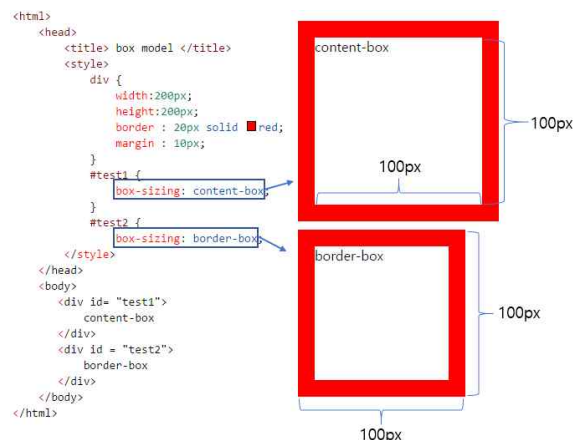
이해가 잘 안될 수 있으니 다음 설명을 보자.



box-sizing 속성의 값으로 border-box를 주었다. 따라서 width와 height는 content의 사이즈가 아닌 **content + padding + border**의 두께를 모두 합친 사이즈가 된다. 위 그림에서 width와 height를 각각 100으로 했지만, content의 가로 세로 사이즈가 78 x 78인 것을 확인하도록 하자.

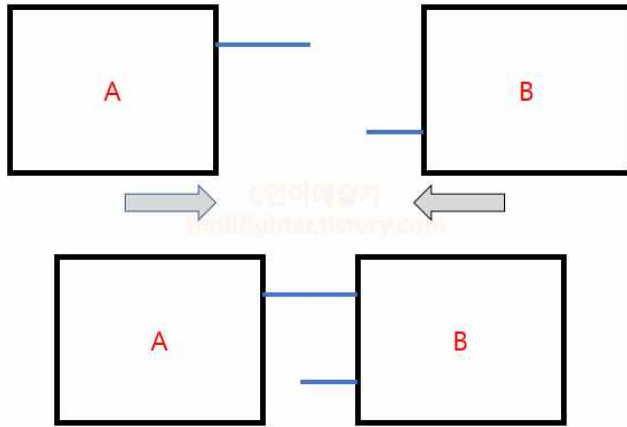
다음은 좀 더 알기 쉽게 패딩(padding) 값을 주지 않고 content와 border 값을 동일하게 준 후에 box-sizing 속성만 다르게 해서 비교한 것이다.

레이아웃을 정할 때 box를 content에 맞추지 border에 맞추지를 먼저 결정한 후에 통일된 레이아웃을 설계하는 것이 중요하다. 그러기 위해서는 box-sizing 속성을 제대로 이해하고 있어야 한다.



## ☑ margin 겹침 현상

margin은 tag 요소간의 간격을 설정하는 속성이다. margin 속성이 있는 두 tag 간에 간격이 결정될 때는 마진 겹침 현상이 일어나는데 다음 그림을 통해서 마진겹침 현상을 이해해 보기로 한다.



A와 B라는 html tag 요소가 있을 때 각 tag 요소에 파란색 막대기가 달려있다고 하자. 그리고 A와 B가 점점 가까워질 때 어떤 순간에 멈출지 생각해보자. A tag 요소에 달린 막대기가 B tag 요소의 막대기보다 길기 때문에 두 tag 요소들 간의 간격은 A가 가진 막대기 길이 이상 가까워질 수 없다. 그리고 이때 B tag 요소가 가진 짧은 막대기의 길이만큼은 막대기 끼리 겹쳐져 있다고 생각할 수 있다.

이제 이 파란 막대기를 margin 속성으로 생각해도 별 무리가 없다. 바로 tag 요소가 가진 막대기 중에 긴 녀석에 의해서 요소간 간격이 정해진다는 것이 마진 겹침 현상의 전부라고 할 수 있다.

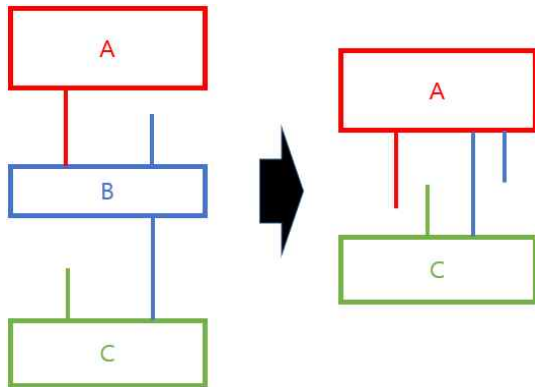
아래 코드와 Browser에 나타나는 결과를 보자. 가상의 파란막대기를 그려보았다. 긴 것이 100px, 짧은 것이 70px의 길이를 갖는다. 이 것이 바로 마진이며 두 tag 요소간의 간격을 긴 막대기(margin)로 결정한다.

여기서 id가 testB인 tag 요소의 마진(margin)을 80px 또는 90px로 바꿔도 두 tag 요소간의 간격은 100px로 그대로일 것이다. 100px 이상을 설정해야 간격이 벌어질 것이다.

```
<html>
<head>
  <style>
    #testA {
      border:10px solid blue;
      margin-bottom:100px;
    }
    #testB {
      border:10px solid black;
      margin-top:70px;
    }
  </style>
</head>
<body>
  <div id="testA">
    A
  </div>
  <div id="testB">
    B
  </div>
</body>
</html>
```

### ❖ 시각적 요소와 마진 겹침 현상

이번에는 시각적인 요소의 유무에 따라서 마진 겹침현상이 어떻게 일어나는지 살펴보도록 한다. 먼저 그림을 보자.



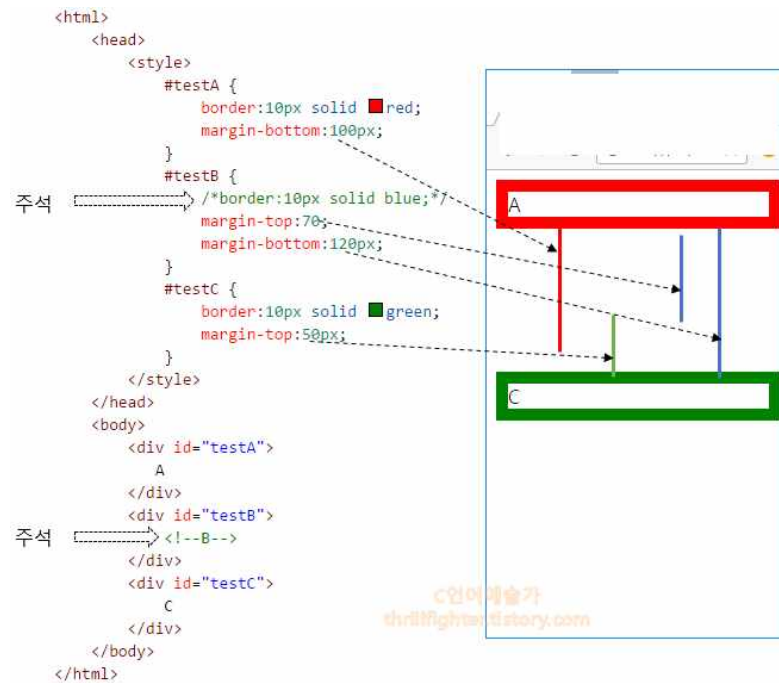
3개의 tag 요소가 있고 각 요소들의 margin을 가상의 막대기로 대신 그려놓았다. 이 때 중간에 있는 tag 요소의 시각적인 부분만 제거했다고 해보자.(tag는 그대로 있다.)

그러면 오른쪽 그림과 같이 마진들이 모두 겹쳐지는 현상이 발생한다. 재미있는 것은 시각적으로 보이지 않게된 tag의 top과 bottom 마진조차 서로 겹쳐진다는 것이다. 이렇게 시각적인 요소를 제거한 tag는 실체는 없으니 margin만 남게 되는 것이다. 그리고 오른쪽 그림처럼 4개의 margin값 중에 가장 큰 margin 값에 따라서 A와 C의 간격이 결정된다.

다음은 이에 대한 실제 코드와 실행 결과다.

```
<html>
  <head>
    <style>
      #testA {
        border:10px solid red;
        margin-bottom:100px;
      }
      #testB {
        border:10px solid blue;
        margin-top:70px;
        margin-bottom:120px;
      }
      #testC {
        border:10px solid green;
        margin-top:50px;
      }
    </style>
  </head>
  <body>
    <div id="testA">
      A
    </div>
    <div id="testB">
      B
    </div>
    <div id="testC">
      C
    </div>
  </body>
</html>
```

이제 id값이 testB인 tag 요소의 시각적인 요소를 제거해 보겠다.



중간에 있는 tag 요소의 시각적인 요소를 모두 주석처리한 결과를 보여준다. 이렇게 두 tag 요소간의 간격을 보이지 않는 요소를 중간에 두어서 결정할 수도 있을 것이다. 어쨌든 이런 현상들에 대해서 이해하는 것이 Web Page의 정교한 레이아웃을 만드는데 필요한 기초가 된다.

## CSS position 속성과 relative, absolute, fixed, static

이 장에서는 tag 요소의 위치를 설정하는 속성에 대하여 학습하도록 한다. 앞서 Box Model에서 tag 요소간에 상호작용에 의해서 위치가 결정된다고 했다. 맞긴 하지만 지금까지는 tag 요소를 위치하는 방법 중에 한가지만 살펴본 것이다. 위치를 결정하는 속성은 position인데 이 속성을 지금까지 설정하지는 않았다. position 속성을 주지 않으면 디폴트로 **static**이라는 속성값이 부여된다. 그러면 지금까지 static 속성값으로 위치를 정한 것이 된다.

### ☑ position 속성

position 속성이 가질 수 있는 속성값에는 다음과 같은 4가지가 있다.

#### ☞ static

position 속성을 부여하지 않았을 때 가지는 디폴트 값.

#### ☞ relative

현재 위치에서 상대적인 offset 속성을 줄 수 있다.

#### ☞ absolute

parent 중에 static이 아닌 요소의 위치를 기준으로 상대적인 offset 속성을 줄 수 있다. 하지만 현재 위치가 변하는 것은 아니다. 특히 relative와 absolute의 공통점은 offset 속성을 가질 수 있다는 것이지만, absolute의 경우는 offset 속성의 기준이 되는 위치가 조건에 따라서 달라질 수 있다는 점을 기억하자. 이 속성의 이름이 absolute라서 혼동될 수 있다.

#### ☞ fixed

Browser에 대해서 위치를 잡는다. 스크롤에 영향을 받지 않고 고정된 위치를 가짐.

### ❖ offset 속성

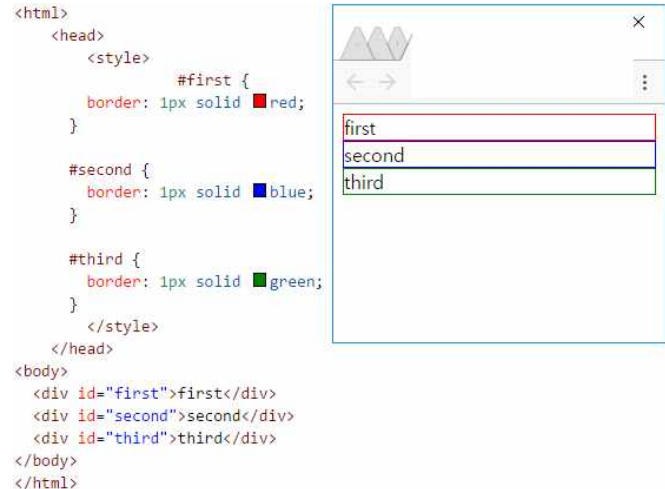
기준이 되는 곳으로부터 얼마만큼 떨어졌는지를 정한다. top, right, bottom, left

예를들어 top : 10px; 는 기준이 되는 곳(top)으로 부터 아래로 10px 만큼 떨어진 곳을 의미한다.

## 👉 static

static은 position 속성을 주지 않을 때의 디폴드 값이다. 위 코드는 3개의 div tag 요소의 position 값이 static 이다.

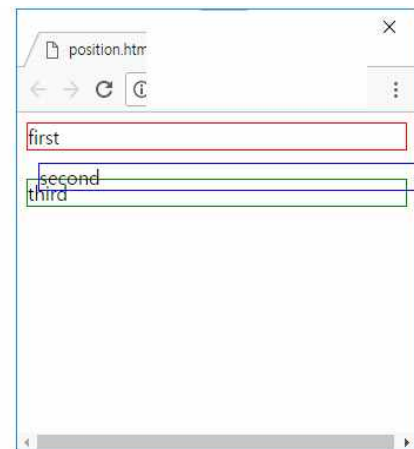
이 예를 변형하면서 나머지 속성값들에 대해서 설명한다.



## 👉 relative

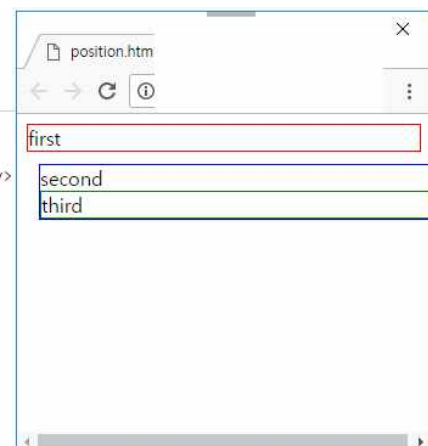
두 번째 tag요소의 position을 relative로 변경하고 아래로 10px 오른쪽으로 10px 만큼 offset을 주었다. 이 때 다른 tag요소들은 변함이 없다. 왜냐하면 position이 static이기 때문이다.

```
#second {
  border: 1px solid blue;
  position: relative;
  top: 10px;
  left: 10px;
}
```



물론 third를 second의 child으로 만들면 다음과 같이 된다. border를 얇게 줘서 잘 안보이는데, second의 파란 테두리가 third까지 감싸고 있다. parent요소가 움직이면 child 요소도 그만큼 움직인다. 혼동하지 말자.

```
<div id="second">second
  <div id="third">third</div>
</div>
```



## 🔗 absolute

이번에는 두 번째 tag요소(second)의 position을 absolute로만 바꿔본다. 이 때 offset 속성은 주지 않겠다.

이 결과는 무엇을 뜻할까? 혼란스러울 수도 있겠는데 알고보면 간단하다.

absolute 속성을 주는 순간 이 tag요소는 다른 tag 요소들과 다른 층으로 옮겨 갔다고 이해하면 된다. 3차원 좌표계라면 z축이 변경되었고 이로 인해서 남은 빈 공간에 third가 채워져 들어간것 이다. 이 때 (x, y) 의 위치는 변하지 않은 것에 주목하시기 바란다.

이번에는 offset 값을 입력하도록 한다. 0, 0을 입력해 보자.

자 원점이 Browser에 맞추어 졌다. 여기서 한가지 짚고 넘어가야할 것이 있다.

현재 id가 second인 tag 요소는 parent tag가 없다. 이런 경우는 Browser의 좌 상단의 좌표가 0, 0이 된다.

그러나 parent tag가 있는 경우는 다음 조건에 따라서 달라진다.

- ① parent의 position 속성의 값이 모두 static이라면 Browser의 좌 상단의 좌표가 0, 0이 된다.
- ② parent의 position 속성의 값 중에 static이 아닌 값이 있다면 해당 parent의 위치가 0, 0이 된다.

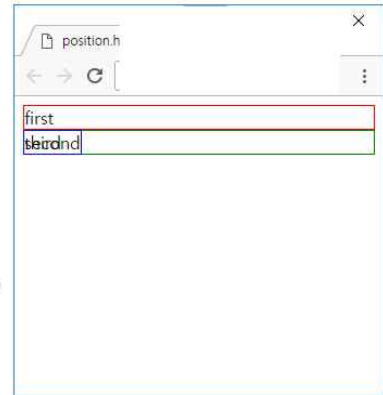
복잡하나 생각될 것이다. 복잡한 것 같지만 **parent가 있을 때, 그리고 parent가 static이 아닐 때** 외에는 Browser의 좌상단이 0,0이 되므로 어렵지 않게 외울 수 있다.

예) second의 parent가 first고 first의 position이 **relative**일 때

```
<html>
<head>
  <style>
    #first {
      border: 1px solid red;
    }

    #second {
      border: 1px solid blue;
      position: absolute;
    }

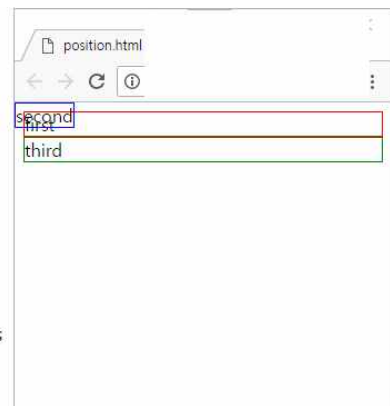
    #third {
      border: 1px solid green;
    }
  </style>
</head>
<body>
  <div id="first">first</div>
  <div id="second">second </div>
  <div id="third">third</div>
</body>
</html>
```



```
<html>
<head>
  <style>
    #first {
      border: 1px solid red;
    }

    #second {
      border: 1px solid blue;
      position: absolute;
      top: 0;
      left: 0;
    }

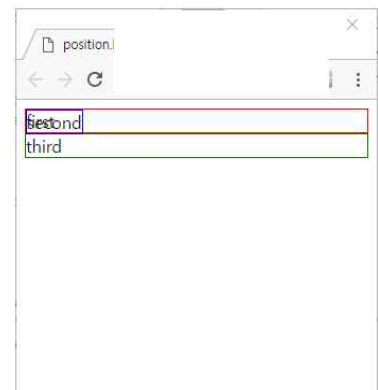
    #third {
      border: 1px solid green;
    }
  </style>
</head>
<body>
  <div id="first">first</div>
  <div id="second">second </div>
  <div id="third">third</div>
</body>
</html>
```



```
<html>
<head>
  <style>
    #first {
      border: 1px solid red;
      position: relative;
    }

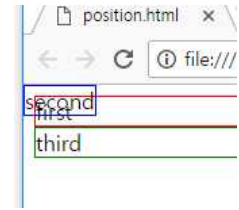
    #second {
      border: 1px solid blue;
      position: absolute;
      top: 0;
      left: 0;
    }

    #third {
      border: 1px solid green;
    }
  </style>
</head>
<body>
  <div id="first">first
    <div id="second">second </div>
  </div>
  <div id="third">third</div>
</body>
</html>
```



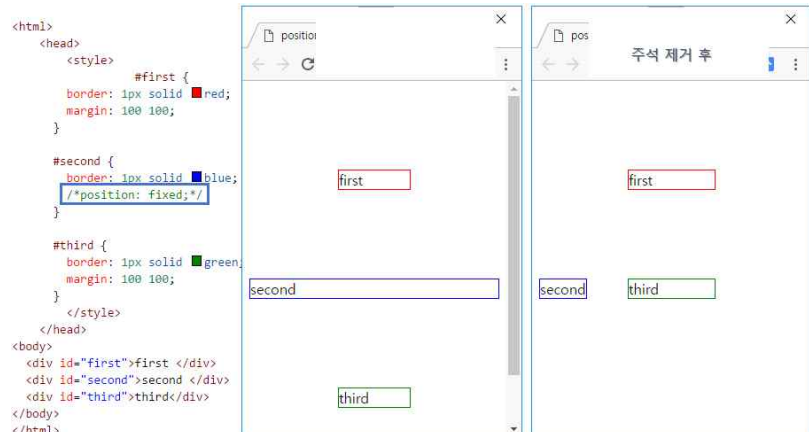


second를 first의 child로 만들고, second의 offset을 0, 0으로 했다. 이 때 first의 position이 **relative**인 것을 기억한다. 만약 first의 position이 static 이라면 결과는 다음과 같다.



## fixed

fixed 역시 absolute처럼 z축 속성이 변한다. Browser 내의 중력이 위쪽으로 작용한다고 생각해보자. first와 second와 third는 위치 속성값이 static 일 때는 서로 맞물려 있다. 이 상태에서 second에 absolute속성값을 주어 공중으로 띄웠다고 생각하면 third가 중력에 의해서 위쪽 빈 공간으로 이동할 것이다.

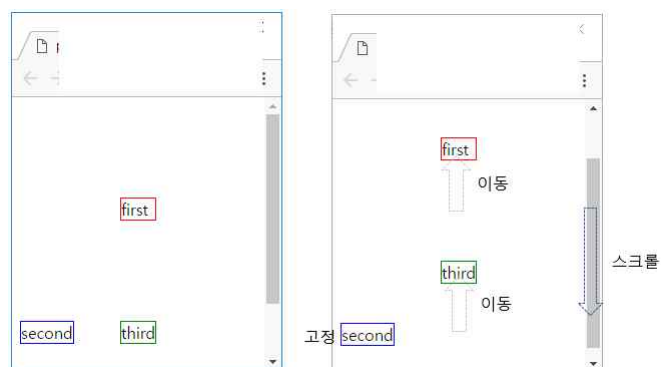


fixed 역시 똑같은 현상이 일어난다. 그렇다면 absolute 속성과 어떤 차이점이 있는지 알아야 한다. fixed는 absolute와 달리 무조건 Browser의 좌표를 따른다. 게다가 스크롤도 무시하므로 Browser에 고정되어 보인다.

fixed 속성값을 주어 second를 z축 방향으로 올리니 third가 자리를 매꾼다. 이 때 second의 border 테두리가 내용에 맞게 변경된 것에 주목하자. 이 현상은 absolute의 경우도 동일하게 일어난다.

이 상태에서 Browser에 스크롤 바가 생기도록 창의 크기를 줄여본 후에 스크롤을 해보자.

fixed 속성값을 가지고 있는 second는 Browser에 고정되어 스크롤을 해도 움직이지 않는다.



### ☞ 마진 겹침 현상과 absolute, fixed

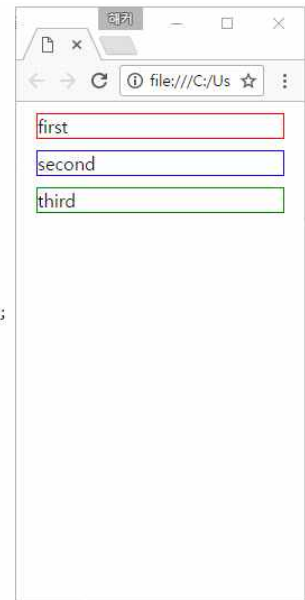
absolute 또는 fixed 속성값을 가지는 요소에 마진(margin)값을 줄 때는 주의한다.

요소들 사이의 거리는 마진겹침 현상에 의해서 10px이 된다.

```
<html>
<head>
  <style>
    #first {
      border: 1px solid red;
      margin: 10;
    }

    #second {
      border: 1px solid blue;
      margin: 10;
    }

    #third {
      border: 1px solid green;
      margin: 10;
    }
  </style>
</head>
<body>
  <div id="first">first </div>
  <div id="second">second </div>
  <div id="third">third</div>
</body>
</html>
```



그럼 second의 position 속성을 absolute 또는 fixed로 바꿔보자.

margin이 없을 때는 몰랐는데 margin이 있을 때는 second가 좀 어긋나는 듯 보인다. 이 결과에서 first와 second의 간격이 20px 이다. 마진겹침 현상이 없어지고 서로의 margin이 다 적용되었다. 물론 이 두 요소가 현재 서로의 위치에 영향을 주는 것은 아니지만 position 속성이 absolute나 fixed로 바뀐 요소는 위쪽에 있는 요소와 마진겹침 현상이 없어진 위치로 위치된다.

```
<html>
<head>
  <style>
    #first {
      border: 1px solid red;
      margin: 10;
    }

    #second {
      border: 1px solid blue;
      margin: 10;
      position: absolute;
    }

    #third {
      border: 1px solid green;
      margin: 10;
    }
  </style>
</head>
<body>
  <div id="first">first </div>
  <div id="second">second </div>
  <div id="third">third</div>
</body>
</html>
```



사실 position 속성을 absolute나 fixed로 변경할 경우는 offset 속성을 정하기 때문에 위 현상을 깊게 이해할 필요는 없다.

## html CSS float 속성 정리

Box Model을 공부한 후부터 지금까지 사이트 레이아웃을 만드는데 필요한 중요한 속성들을 계속 살펴보고 있다. 이번에 다룰 float 속성도 레이아웃을 잡는데 쓰이는 속성으로 기본 개념은 단순하지만 기본 개념만으로는 동작방식을 예측하기 힘든 경우가 있어서 케이스별로 정리해서 이해할 필요가 있다.

### ☑ float 속성

float의 뜻은 "뜨다", "흐르다" 뭐 이런 뜻이다. 단어의 의미가 이 속성과 딱 매치된다고 말은 못한다. 그래도 단어의 뜻으로 이 속성을 설명해 보자면 이렇다. "float 속성을 갖는 요소는 html 문서에서 공간을 차지하되 다른 요소의 배치에 영향을 안주는 요소가 된다."

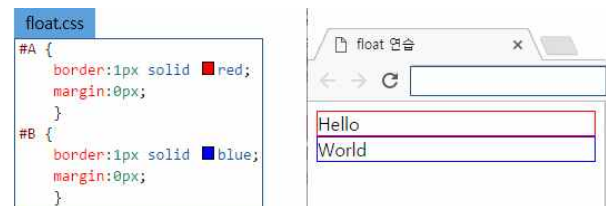
이렇게 float 속성을 갖는 요소는 html 내에서 공간은 차지하되 붕 떠 있다고 보면 된다. 예제를 통하여 케이스별로 살펴보자.

html 문서를 기본으로 하여 CSS 코드만 바꿔 보면서 테스트해보자. (아직 CSS 파일에 아무런 스타일도 적용하지 않았다. div tag 요소 두 개가 있다. div 속성은 display 속성 기본값이 block이므로 한 줄을 혼자 차지한다.)



### ❖ float 속성이 없을 때

마진(margin)을 0으로 두고 테두리를 주었다. 여기까지는 쉽다.



### ❖ 앞에 요소가 float 속성을 가질 때

id가 A인 요소에 float: left 속성을 준다.

이 부분이 중요하다. float: left 속성을 주는 것은 이렇게 명령하는 것과 같다.

Hello라는 내용(content)만큼만 공간을 차지하고 다른 요소에 대해서 상대적으로 왼쪽(left)으로

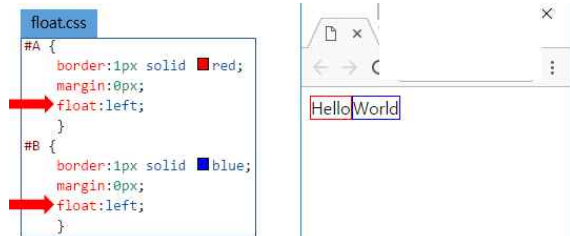
배치되고 무시되어라. 다시말해 다른 요소들이 float 속성을 지닌 tag 요소의 존재를 무시하는 것이다. 그러면 다음에 배치될 id가 B인 tag가 Hello 만큼의 공간을 제외하고 바로 옆으로 배치된다. 원래 div tag는 display 속성이 block이므로 한 줄을 혼자 차지하는 것이 정상이다. 그러나 위 두 요소는 나란히



배치되었다. 위 코드에서 앞에 요소가 float 속성을 가짐과 동시에 무시되어 block 속성값을 가진 다음 요소가 앞에 요소를 무시한채 나란히 배치될 수 있는 것이다. (물론 이미 차지하고 있는 공간은 제외한다.)

## ❖ 두 요소가 모두 float 속성을 가질 때

내용(content)만큼의 공간만 차지한 채로 나란히 배치된다. div tag인데 말이다. 지금 살펴본 성질을 이용해서 block 속성을 가진 요소의 옆에 다른 요소들을 배치할 수 있다.



## ❖ 그림 옆에 다른 요소 배치하기

이미지 옆에 글씨가 배치되는 경우를 본 적이 있을 것이다. 이미지를 삽입하기 위한 img tag는 display: inline; 이 기본 속성이므로 다른 inline 속성을 가진 요소들과 한줄에 나란히 배치된다. 문제는 이미지와 글씨를 같이 배치해도 한 줄 이상 배치할 수 없다는 것이다.



이 문제를 해결하려면 img tag 요소에 float 속성을 주면 된다.

img tag에 float 속성을 주었으므로 이미지 만큼의 공간을 차지하되 다른 요소들에 의해 무시됨에 따라서 문자열들이 이미지 공간만 제외한 채로 상대적으로 오른쪽에 배치되며 이미지의 존재가 무시된 채로 배치된다. (원래는 이전 그림처럼 한 줄 안에 이미지와 문자열이 나란히 배치되어야 한다.)

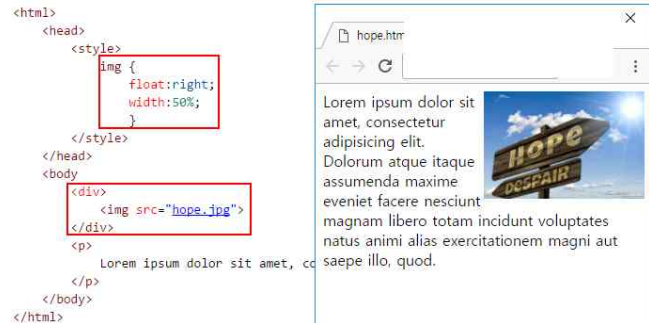


## ❖ clear 속성

float는 left나 right 속성값을 가질 수 있는데 왼쪽 또는 오른쪽에 배치된다. 이 속성은 상대적인 속성으로 다음과 같이 동작한다.

left 속성값을 가지는 요소의 다음에 오는 요소들은 상대적으로 오른쪽에 배치되고, right 속성값을 갖는 요소의 다음에 오는 요소들은 상대적으로 왼쪽에 배치된다.

참고 : img tag에 float: right 속성을 주고 div tag로 다시 감쌌는데 데도 역시 계속해서 무시된다. 원래 ptag는 block 속성이므로 div tag와 한 줄에 배치되지 않는게 정상이다.



이 때 만약 p tag가 원래처럼 동작하도록 하려면 clear 속성을 주면 된다.

clear: right; 속성을 가진 요소는 float: right; 속성의 영향을 받지 않는다. 따라서 위와 같은 결과가 나온다.

clear: left;를 설정하면 float: left 속성의 영향을 받지않고 clear: both; 라고 하면 모든 float 속성의 영향을 받지 않는다.



따라서 보통은 clear: right; 나 clear: left; 대신 clear: both; 를 사용한다.

float 속성은 아쉽게도 center라는 속성값이 없다. 만약 center 값이 있다면 이 경우 상대적 배치를 어떻게 할까? 아마도 그런 의미에서 center 라는 속성값이 없는 것 같다.

## CSS 우선순위 규칙 Cascading

Cascading은 작은 계단모양의 폭포를 말한다. 폭포의 전후로 물의 높낮이가 바뀐다. 연속적인 물의 흐름에서 높낮이 차이를 우선순위에 비유한 것인 듯 하다.

CSS(Cascading style sheets)의 C는 Cascading의 약자이다. CSS의 뜻을 살펴보면 우선순위가 있는 스타일 시트라는 뜻이다. 구체적으로 하나의 tag에 여러가지 방법으로 동일한 속성의 스타일을 적용하는 경우 우선순위에 따라서 적용될 스타일이 결정된다는 뜻이다. 속성을 적용하다보면 하나의 tag에 본의 아니게 같은 속성이 겹쳐 적용될 때가 있을 것이다.

단편적인 예제를 통해서 CSS를 공부할 때는 이런 경우가 그다지 발생하지 않으므로 Cascading에 대해서 깊게 생각하지 않아도 될지 몰라도 완성도 있는 HTML 페이지를 만들기 위해 Cascading에 대한 이해는 필수이다. 그렇다면 스타일을 적용할 때 우선순위가 어떻게 달라지는지 학습하도록 한다.

### ☑ 캐스케이딩(Cascading)

#### ◎ 캐스케이딩(우선순위)를 결정하는 요소

- ① 중요도
- ② 명시도
- ③ 코드의 순서

#### ☞ 중요도

저작자 !important > 저작자 CSS > 사용자 CSS > Browser CSS

웹 디자인너(프로그래머)는 저작자에 해당하고, 사용자나 Browser가 설정한 기본 CSS 보다 우선순위가 높다. Browser가 가장 우선순위가 낮는데, 저작자가 아무런 CSS 속성도 지정하지 않았을 때 디폴트로 보여지는 모습이 Browser에 기본으로 설정된 CSS이다. 같은 페이지라도 Chrome과 Internet Explorer(IE)에서 약간 차이가 있게 보이는 경우가 각 Browser의 기본 CSS의 차이 때문이다.

그리고 사용자가 CSS 속성을 지정하는 것은 흔하지는 않기 때문에 넘어가도록 한다.

저작자 !important는 CSS 속성을 줄 때 !important를 뒤에 붙이는 것을 말한다. 이렇게 하면 우선순위가 가장 높아진다. 나중에 예제로 학습하도록 한다. 다음부터가 중요한 내용이다.

#### ☞ Selector에 따른 명시도

tag 요소에 스타일 속성을 줄 때 Selector를 사용한다. 그런데 Selector의 종류에 따라 명시도의 차이가 있다. 그리고 명시도는 더 구체적이라는 뜻이고 바로 우선순위를 뜻한다.

style > id > class > tag

예를들어 id Selector는 class Selector보다 명시도가 높고 class Selector는 tag Selector보다 명시도

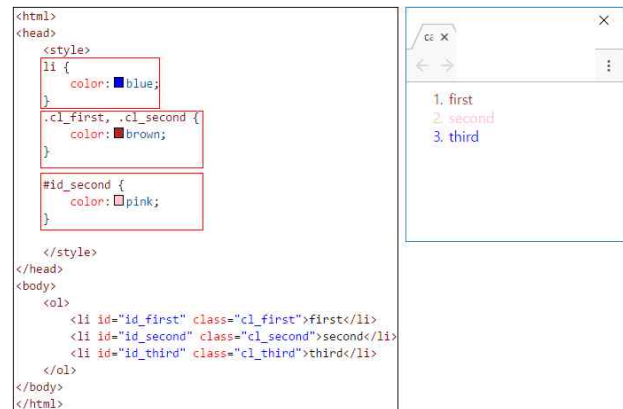
가 높다. 그리고 같은 tag Selector 끼리라면 뒤에 추가적인 Selector가 더 붙는 것이 더 자세하므로 명시도가 높다.

예를 들어 살펴보자.

순서있는 리스트에 3개의 아이템이 있다. (li tag 안에) 그리고 li tag에 폰트색을 blue로 주었다. 결과를 보면 third에만 blue가 적용되었다. 우선순위에서 밀린 것이다. 바로 다음에 나오는 class Selector와 id Selector가 tag Selector보다 우선순위가 높기 때문이다.

다음은 클래스 Selector로 리스트의 첫 번째, 두 번째 아이템에 brown색을 주었다. 하지만 첫 번째 아이템만 적용되었다. 왜냐면 다음에 나오는

id Selector가 두 번째 아이템에 pink색을 주었기 때문에 우선순위에서 밀린 것이다. 이렇게 중복해서 속성이 적용될 때는 Selector의 종류에 따라서 적용 우선순위가 달라진다.



참고로 CSS Selector로 속성을 주는 것 보다 다음과 같이 tag에 직접 style을 주는 경우가 우선순위가 더 높다.

`<li style="color:yellow">`

이번에는 같은 종류의 Selector일 때 우선순위를 정하는 규칙에 대한 예제이다.

**같은 Selector 끼리라면 깊이가 깊을수록(구체적일 수록) 명시도(우선순위)가 높다.**

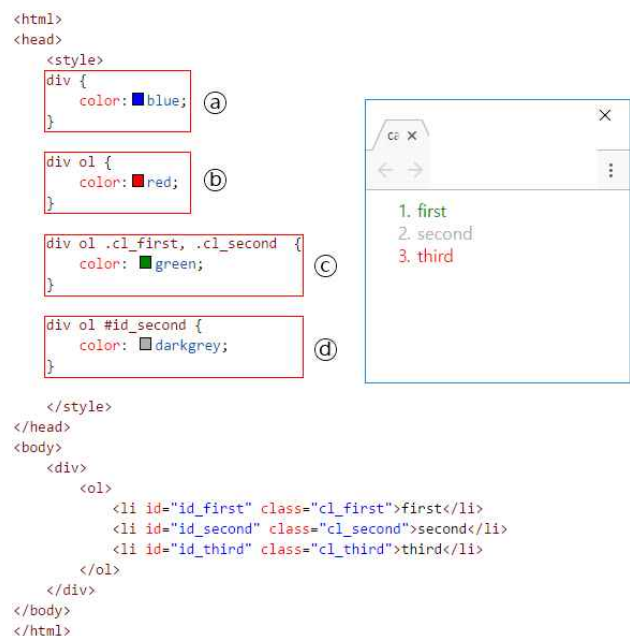
①, ②, ③은 모두 같은 div tag Selector로 시작하지만 Selector의 깊이가 다르다. ① 보다 ②가 더 구체적이다. 이럴 때는 ②가 우선순위가 높으므로 ①은 적용되지 않고 ②가 적용된다.

그런데 결과를 보면 리스트의 세 번째 아이템만 빨간색이 적용되었다. ③은 ②보다 더 구체적으로 class Selector까지 명시했으므로 우선순위가 더 높기 때문이다. 그런데 두 번째 아이템은 green이 아닌 darkgrey가 적용되었다.

③과 ④가 같은 깊이지만 ④의 세 번째 Selector가 id Selector로 ③의 세 번째

Selector class Selector보다 우선순위가 높기 때문이다

같은 만원짜리 지폐라도 현 지폐보다 새 지폐가 선호도가 높다. 비슷한 원리라고 기억하면 이해하기 쉬울 듯 하다.

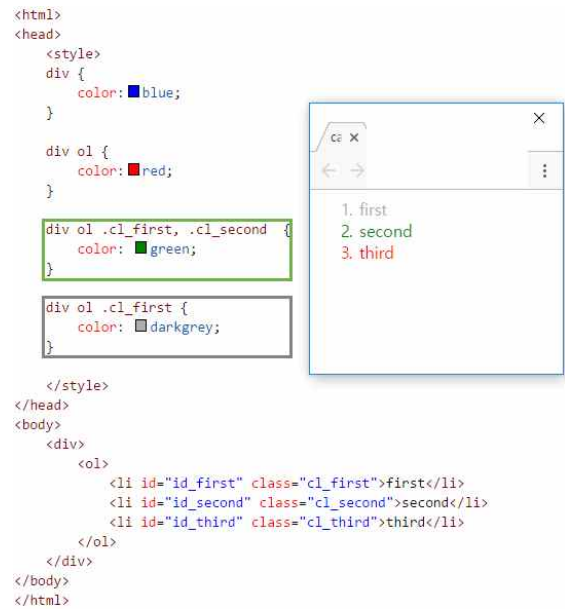




## ☞ 코드 순서

앞의 내용과 연결해서 이해할 수 있다. Selector의 종류와 깊이가 같을 때 우선순위의 결정 방식은 코드의 순서이다. 다음 예제를 보자.

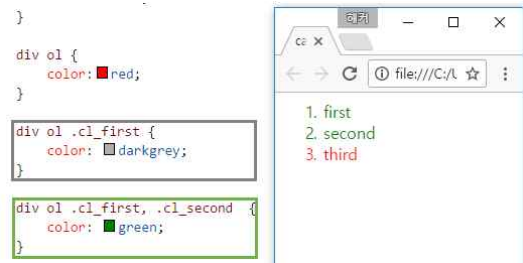
CSS에서 박스로 표시된 부분을 보면 .cl\_first에 대해서 각각 color 속성을 지정했다. 이 때 명시도가 같아서 명시도로 우선순위를 가릴 수 없으므로 코드 상에서 나중에 오는 경우가 우선순위가 높도록 설정된다. 비유하자면 캔버스에 색을 칠할 때 먼저 칠했던 색을 다른 색으로 덮어씌우기를 하는 거라고 생각하면 된다. 코드의 해석은 위에서 아래방향의 순서대로 해석되니 아래쪽에 있는 코드가 나중에 적용되는 코드이다.



두 코드의 위치를 바꾸면 다음과 같이 .cl\_first가 green 색으로 적용된다.

## ※ 주의

같은 깊이라도 (,)콤마로 나열된 경우 우선순위가 생긴다.



div ol .cl\_first, .cl\_second 보다 div ol .cl\_second 가 .cl\_second에 대해서 우선순위가 높다.