

웹프로그래밍 기초

황교찬

eeidro@gmail.com

010-4736-5204

2강

CSS 선택자

- id 선택자
 - HTML 태그에 '이름'을 지어 선택하는 방식

```
html
<body>
  <h1>type 선택자</h1>
  <h2 id="bg">id 선택자</h2>
</body>
```

```
css
h1 {
  color: red;
}
#bg {
  background-color: yellow;
}
```

- 실행 결과

type 선택자

id 선택자

html

```
<body>
  <h1>type 선택자</h1>
  <h2 id="bg">id 선택자</h2>
  <h3 class="size">class 선택자</h3>
</body>
```

CSS

```
.size {
  font-size: 50px;
}
```

- 실행 결과

type 선택자

id 선택자

class 선택자

- id 속성과 class 속성의 차이점
 - id 속성은 속성값을 하나만 사용할 수 있고,
class 속성은 속성값을 여러 개 사용할 수 있다

```
<body>  
  <h1>type 선택자</h1>  
  <h2 id="bg">id 선택자</h2>  
  <h3 class="size color">class 선택자</h3>  
</body>
```

html

```
.size {  
  font-size: 50px;  
}  
.color {  
  color: blue;  
}
```

CSS

- 실행 결과

type 선택자

id 선택자

class 선택자

부모 자식 간의 css 상속

- 부모 자식 간의 CSS 상속
 - HTML의 부모-자식 관계: 태그가 태그를 포함하는 구조
 - 상위에 있는 <header> 태그를 '부모'라고 하고, 하위에 있는 <h1>과 <p> 태그를 '자식'이라고 함
 - <h1>과 <p> 태그를 서로 '형제'라고 표현
 - 상속 : 부모가 가진 CSS 속성이 자식에게 영향을 미치는 것

```
<!DOCTYPE html>
<html>
<head>.....</head>
<body>
  <header>
    <h1>header h1</h1>
    <p>header p</p>
  </header>
</body>
</html>
```

html

```
header {
  color: red;
}
```

CSS

- 실행 결과

header h1

header p

1 | 부모 자식 간의 css 상속

- <header> 태그의 color: red 속성을 상속받은 상태에서 <h1>과 <p> 태그에 다시 color속성값을 적용하면?

```
header {  
    color: red;  
}  
h1 {  
    color: blue;  
}  
p {  
    color: blue;  
}
```

CSS

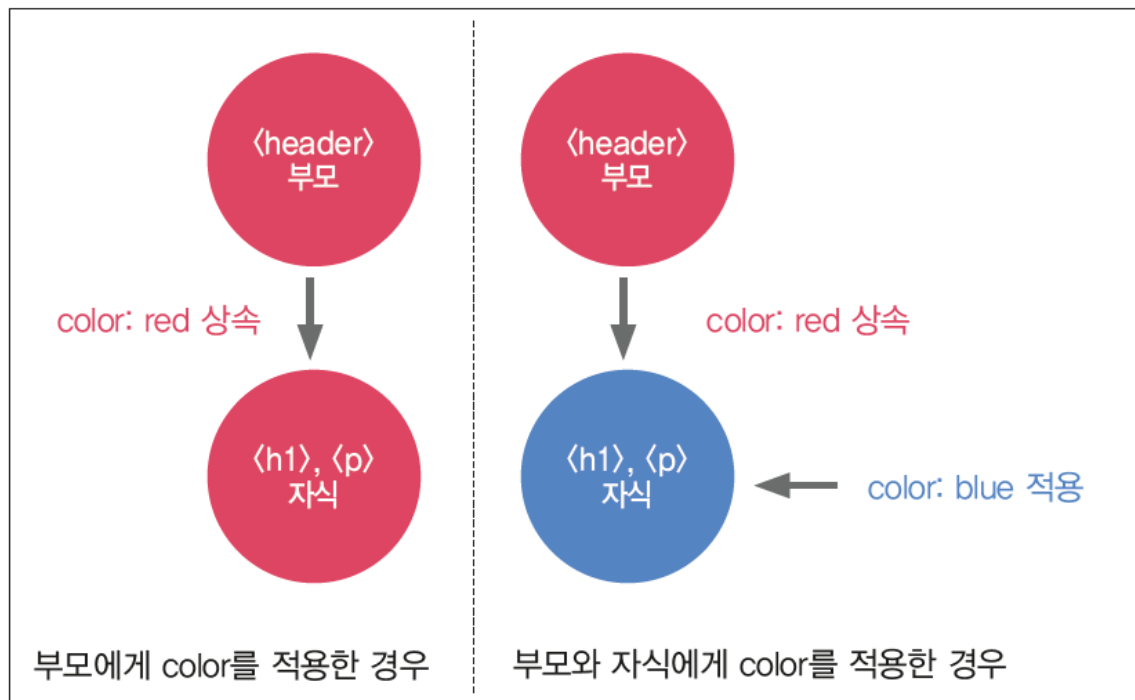
- 실행 결과

header h1

header p

1 | 부모 자식 간의 css 상속

- 부모 자식 간의 CSS 상속
 - 같은 CSS 속성을 사용해도 자식의 유전자가 부모의 유전자보다 우선순위가 높다
 - 따라서 글자 색이 <h1>과 <p> 태그에 직접 지정한 color 속성값인 파란색으로 바뀜



CSS 속성의 우선순위, 캐스케이딩

- 캐스케이딩 : 같은 영역에 같은 디자인을 적용했을 때 어느 디자인을 우선해서 적용하는지를 나타냄
- 캐스케이딩 우선순위에 영향을 미치는 것
 - CSS 속성이 입력된 순서
 - 선택자를 구체적으로 입력했는지 여부
 - type · id · class 선택자

CSS 속성의 우선순위, 캐스케이딩

- CSS 속성이 입력된 순서에 따른 우선순위

```
<!DOCTYPE html>
<html>
<head>.....</head>
<body>
  <p>Hello World</p>
</body>
</html>
```

html

```
p {
  color: red;
}
/* 나중에 적용된 CSS 속성값의 우선순위가
높음 */
p {
  color: blue;
}
```

CSS

- 실행 결과

Hello World

CSS 속성의 우선순위, 캐스케이딩

- 선택자를 구체적으로 입력했는지 여부에 따른 우선순위

```
<body>
  <header>
    <h2>Nice to meet you</h2>
  </header>
</body>
```

html

```
/* 선택자를 구체적으로 작성할수록 우선순위가 높음 */
header h2 {
  color: blue;
}
h2 {
  color: red;
}
```

CSS

CSS 속성의 우선순위, 캐스케이딩

- 실행 결과



Nice to meet you

그림 5-8

- 지금처럼 자식인 <h2> 태그가 부모인 <header> 태그 안에 포함되어 있다면, 선택자로 h2만 쓰는 것보다 header h2 형식으로 부모를 함께 써 주는 것이 우선순위가 높다
- 따라서 글자는 빨간색이 아닌 파란색으로 출력됨

- 선택자의 종류에 따른 우선순위

```
<body>
```

html

```
  <h3 class="color">Welcome to CSS</h3>
```

```
</body>
```

```
.color {
```

```
  color: red;
```

```
}
```

```
h3 {
```

```
  color: green;
```

```
}
```

CSS

- 실행 결과

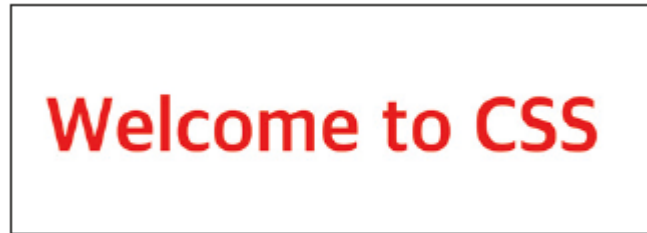


그림 5-9

- .color는 class 선택자이고 h3는 type 선택자
- 이러한 상황에서는 순서에 상관없이 class 선택자(.color)가 type 선택자 (h3)보다 우선순위가 높다
- 따라서 글자는 빨간색으로 출력됨

```
<body>
  <h3 id="color" class="color">Welcome to CSS</h3>
</body>
```

html

```
#color {
  color: blue;
}
.color {
  color: red;
}
h3 {
  color: green;
}
```

CSS

- 실행 결과

Welcome to CSS

- id 선택자는 class 선택자보다 우선순위가 높아서 글자가 파란색으로 바뀜

- 인라인 방식 vs id 선택자

```
<body>
  <h3 style="color: pink;" id="color" class="color">Welcome to CSS</h3>
</body>
```

html

- 실행 결과



Welcome to CSS

그림 5-11

- 인라인 방식으로 입력했을 때가 id 선택자를 지정했을 때보다 우선순위가 높다는 걸 알 수 있다

- CSS 속성의 우선순위

인라인 방식 > id 선택자 > class 선택자 > type 선택자

width와 height 속성

- width와 height 속성
 - width와 height 속성은 공간을 만들 때 사용
 - 웹 사이트를 작업할 때 각 영역의 크기를 지정할 수 있다
 - HTML 태그는 배경색의 기본값이 투명이라 공간이 제대로 만들어졌는지 확인하기가 어려움
 - 확인하고 싶다면 background-color 속성을 사용하여 배경색을 설정해 본다

```
<div class="place"></div>
```

index.html

```
.place {  
  width: 500px;  
  height: 500px;  
  background-color: yellow;  
}
```

style.css

- 실행 결과



font- 속성

- width와 height 속성
 - 글자 크기, 글꼴 종류, 글자 모양, 글자 굵기 등을 변경할 때 사용

```
<p class="font">Hello World</p>
```

html

```
.font {  
  font-size: 50px;  
  font-family: Arial, Times, sans-serif;  
  font-style: italic;  
  font-weight: bold;  
}
```

CSS

1

2

3

4

- 실행 결과

Hello World

font- 속성

- 실행 결과

Hello World

- 1 font-size 속성은 글자 크기를 지정 - px(픽셀) 단위를 가장 많이 사용
font-family 속성은 글꼴을 지정
- 2 Arial 글꼴을 지원하지 않으면 Times 글꼴을 적용하고,
Times 글꼴도 지원하지 않으면 sans-serif 글꼴을 적용하라는 의미
- 3 font-style 속성은 글자 모양을 지정 - 속성값에는 normal, italic, oblique 등이 있다

normal	기본값으로 기본 글꼴을 적용합니다.
italic	글꼴을 이탤릭 모양으로 바꿉니다.
oblique	글꼴을 비스듬하게 바꿉니다.

font-style 속성의 속성값

4 font-weight 속성은 글자 굵기를 지정- 속성값에는 normal, bold, bolder, lighter, 100 ~ 900 등이 있다

normal	기본값으로 보통 두께를 적용합니다.
bold	글자를 굵게 지정합니다.
bolder	글자를 bold보다 더 굵게 지정합니다.
lighter	normal보다 얇게 지정합니다.
100, 200, 300, 400, 500, 600, 700, 800, 900	숫자가 클수록 글자가 굵어집니다. 400은 normal, 700은 bold와 같은 굵기입니다.

font-weight 속성의 속성값

background- 속성

- 배경색을 지정하거나 이미지를 삽입하고 이미지의 위치를 변경할 때 사용

```
<div class="background"></div>
```

html

```
.background {  
  width: 500px;  
  height: 500px;  
  background-color: yellow;  
  background-image: url(rice.png);  
  background-repeat: no-repeat;  
  background-position: left;  
}
```

①

②

③

④

CSS

- background-color 속성은 배경색을 지정,
 ① 속성값은 yellow처럼 색상의 영어명을 입력해도 되지만 RGB 값이나 # 기호로 시작하는 HEX 값을 입력도 가능
- ② background-image 속성은 선택된 영역에 배경 이미지를 적용 url()의 괄호 안에는 이미지 파일의 경로를 입력
- ③ background-repeat 속성은 삽입된 배경 이미지에 반복 효과를 적용 속성값에 따라 x축이나 y축으로만 반복 효과를 적용할 수도 있다

속성값	repeat	repeat-x	repeat-y	no-repeat
역할	x축과 y축으로 모두 반복 효과를 적용	x축으로 반복 효과를 적용	y축으로 반복 효과를 적용	반복 효과를 적용하지 않고 이미지를 하나만 적용
결과				

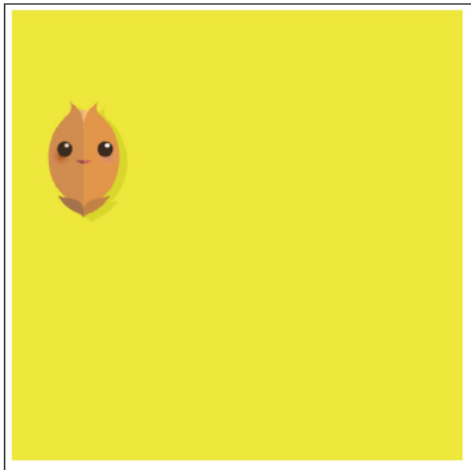
- 4 background-position 속성은 이미지의 좌푯값을 지정
속성값에는 영어 표현인 top, left, bottom, right, center를 넣을 수도 있고
구체적인 좌푯값을 넣을 수도 있다

좌푯값은 띄어쓰기를 기준으로 첫 번째 숫자가 x축이고 두 번째 숫자가 y축

```
background-position: 40px 100px;
```

현재 노란색 박스를 기준으로 x축으로 40px, y축으로 100px만큼 background-image를 이동시킨다는 뜻의 코드

- 실행 결과



background- 속성

- background-로 시작하는 네 가지 속성을 다음과 같이 한 줄로 정리해서 사용할 수 있다

```
background: yellow url(rice.png) no-repeat left;
```

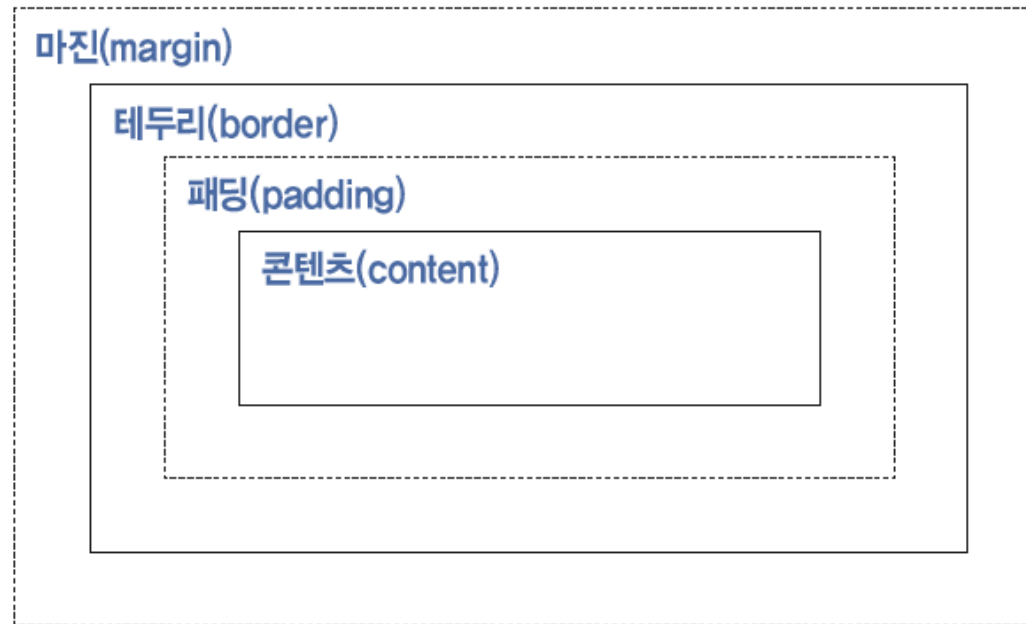
- 코드를 한 줄로 줄이면 CSS 문서 용량이 그만큼 줄어들므로 브라우저가 해당 파일을 불러오는 시간이 단축됨

- 여백

- HTML 태그는 우리 눈에 보이지 않는 여러 개의 박스가 겹쳐져 있다

- 여백의 크기와 박스 간격을 정의하는 네 가지 요소

- : margin, border, padding, content



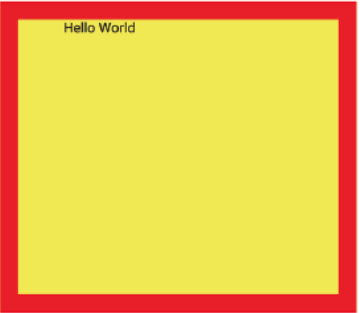
```
<div id="box_model">  
  <span>Hello World</span>  
</div>
```

html

```
#box_model {  
  width: 300px;  
  height: 300px;  
  background-color: yellow;  
  border: solid 20px red;  
  margin-left: 50px;  
  padding-left: 50px;  
}
```

CSS

- HTML 태그 선택 후 CSS 속성 확인



HTML 태그 클릭

<div> 태그에 해당하는 CSS 속성

```
<!DOCTYPE html>
<html>
  <#shadow-root (open)>
  <head>...</head>
  <body>
    <div id="box_model">...</div>
  </body>
</html>
```

Styles

Filter

element.style {

#box_model {

width: 300px;


height: 300px;

background-color: yellow;

border: solid 20px red;

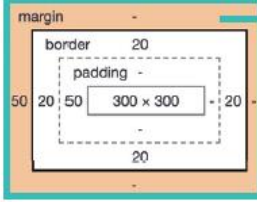
margin-left: 50px;

padding-left: 50px;



margin 영역이 주황색으로 바뀜

1 마우스 포인터 올리기



margin -

border 20

padding -

300 x 300

20

```
<!DOCTYPE html>
<html>
  <#shadow-root (open)>
  <head>...</head>
  <body>
    <div id="box_model">...</div>
  </body>
</html>
```

html body div#box_model

```
#box_model {  
  width: 300px;  
  height: 300px;  
  background-color: yellow;  
  
  border: solid 20px red;  
  
  margin-left: 50px;  
  padding-left: 50px;  
}
```

- margin 속성도 한 줄로 정리할 수 있다

```
/* 12시를 기준으로 시계 방향으로 top, right, bottom, left 속성을 의미 */  
margin: 40px 30px 20px 10px;
```

- border 속성
 - 공간의 테두리를 생성
 - solid 속성값: 테두리 종류 중 하나인 실선을 의미
 - 20px 속성값: 선의 굵기, red 속성값: 선의 색상
 - border 속성의 속성값은 어떤 순서로 입력해도 상관없음

```
#box_model {  
  width: 300px;  
  height: 300px;  
  background-color: yellow;  
  
  border: solid 20px red;  
  
  margin-left: 50px;  
  padding-left: 50px;  
}
```


마진 병합 현상

- 인접한 공간에 margin-bottom과 margin-top 속성을 적용할 경우에 두 속성 중 큰 속성값이 작은 속성값을 병합하는 현상
- 형제간에 발생하는 마진 병합 현상, 부모 자식 간에 발생하는 마진 병합 현상
- Block 요소의 성격을 갖고 있는 태그에서만 발생

마진 병합 현상

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

html

```
#first {  
  width: 100%;  
  height: 200px;  
  background-color: yellow;  
  margin-bottom: 100px;  
}  
#second {  
  width: 100%;  
  height: 200px;  
  background-color: blue;  
}
```

CSS



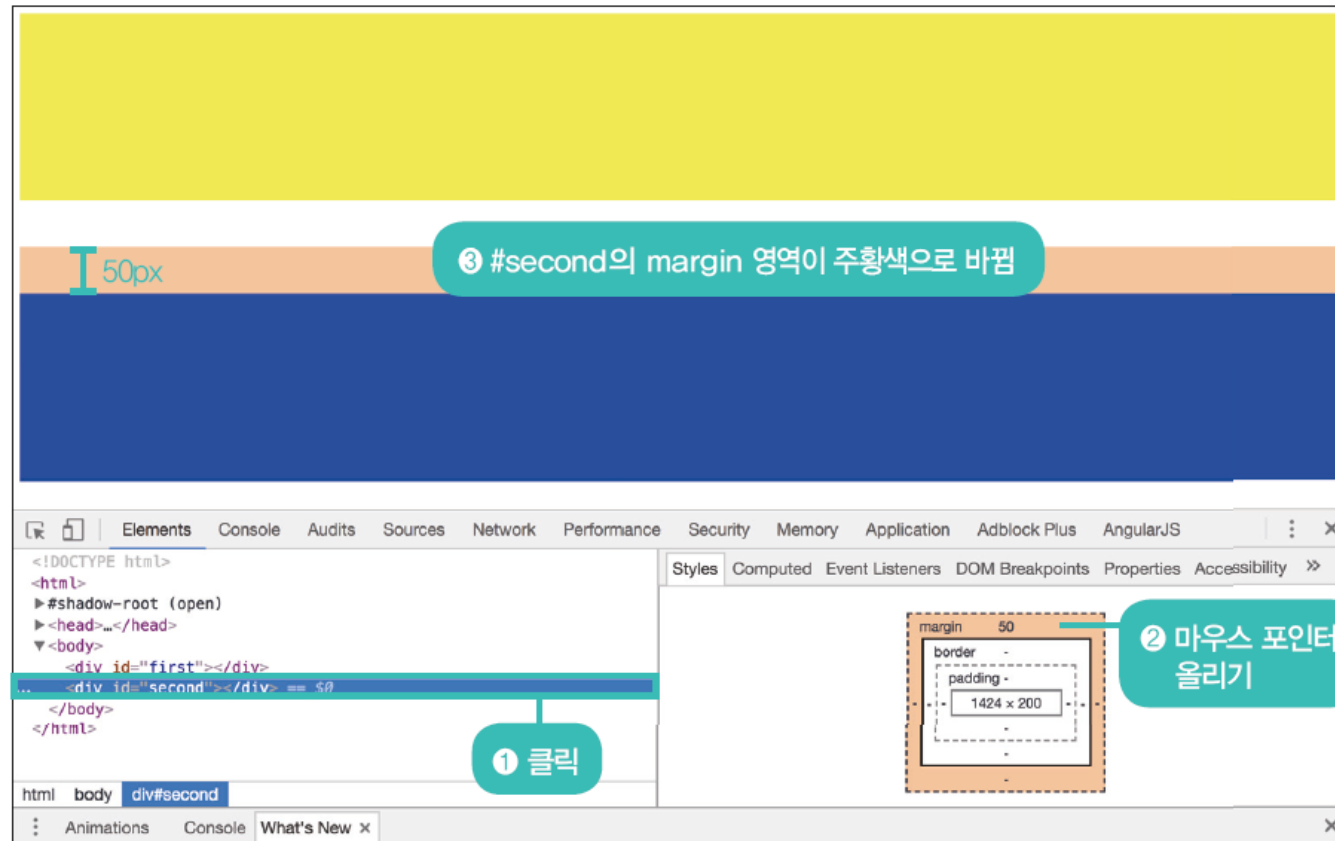
- 노란색 박스 아래에 공백이 100픽셀 생김

```
#first {  
  width: 100%;  
  height: 200px;  
  background-color: yellow;  
  margin-bottom: 100px;  
}  
#second {  
  width: 100%;  
  height: 200px;  
  background-color: blue;  
  margin-top: 50px;  
}
```

- #first와 #second의 margin 영역을 살펴보면 겹치는 것을 확인할 수 있다
- HTML 문서에서 #first 선택



- HTML 문서에서 #second 선택



마진 병합 현상

- 부모 자식 간에 발생하는 마진 병합 현상
 - 자식의 margin-top 속성이 부모에게 영향을 미치는 것

```
<div id="parent">  
  <div id="child"></div>  
</div>
```

html

```
#parent {  
  width: 100%;  
  height: 500px;  
  background-color: yellow;  
}  
#child {  
  width: 300px;  
  height: 300px;  
  background-color: blue;  
  margin-top: 100px;  
}
```

CSS

- 파란색 박스는 물론 노란색 박스까지 함께 내려간다



- #child 안에 position: absolute;을 추가하면 부모 자식 간에 발생하는 마진 병합 현상이 사라지고 파란색 박스만 아래로 움직임

```
#child {  
    position: absolute;  
    width: 300px;  
    height: 300px;  
    background-color: blue;  
    margin-top: 100px;  
}
```

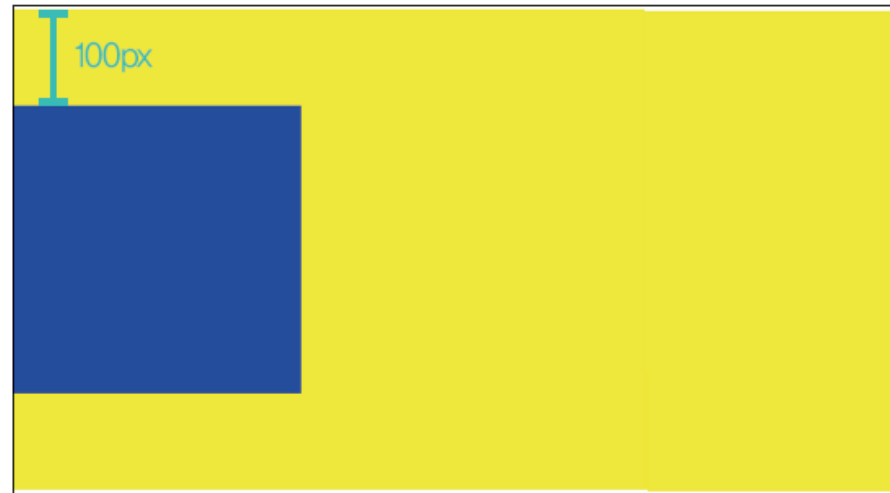


그림 7-16

CSS의 5가지 position 속성 값 기본 정리

position 속성

CSS에서 `position` 속성은 HTML 문서 상에서 요소가 배치되는 방식을 결정합니다. 많은 경우, `position` 속성은 요소의 정확한 위치 지정을 위해서 `top`, `left`, `bottom`, `right` 속성과 함께 사용됩니다.

position: static

`position` 속성을 별도로 지정해주지 않으면 기본값인 `static` 이 적용됩니다. `position` 속성이 `static` 인 요소는 HTML 문서 상에서 원래 있어야 하는 위치에 배치됩니다.

이 말은 요소들이 HTML에 작성된 순서 그대로 브라우저 화면에 표시가 된다는 것을 뜻하며, 따라서 `top`, `left`, `bottom`, `right` 속성값은 `position` 속성이 `static` 일 때는 무시됩니다.

예를 들어, 다음과 같이 `<main>` 요소 아래에 3개의 `<div>` 요소가 있다면 맨 위에 첫 번째 요소, 중간에 두 번째 요소, 제일 아래에 세 번째 요소가 나란히 순서대로 배치됩니다.

```
<main>
  <div>첫 번째 요소</div>
  <div>두 번째 요소</div>
  <div>세 번째 요소</div>
</main>
```

```
main {  
  width: 300px;  
  height: 400px;  
  background: tomato;  
}
```

```
div {  
  width: 200px;  
  height: 100px;  
  border: 1px solid;  
  background: yellow;  
  line-height: 100px;  
  text-align: center;  
}
```

```
div:nth-of-type(2) {  
  position: static;  
  background: cyan;  
  opacity: 0.8;  
}
```

```
<main>  
  <div>첫 번째 요소</div>  
  <div>두 번째 요소</div>  
  <div>세 번째 요소</div>  
</main>
```

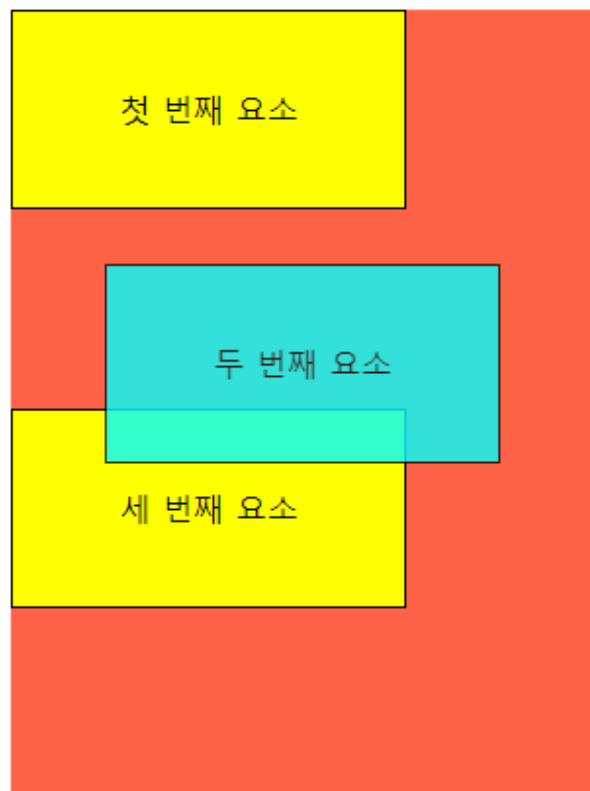


position: relative

`position` 속성을 `relative` 로 설정하게 되면, 요소를 원래 위치에서 벗어나게 배치할 수 있게 됩니다. 요소를 **원래 위치**를 기준으로 상대적(relative)으로 배치해준다고 생각하시면 이해가 쉬울 것 같은데요. 요소의 위치 지정은 `top`, `bottom`, `left`, `right` 속성을 이용해서, 요소가 원래 위치에 있을 때의 상하좌우로 부터 얼마나 떨어지게 할지를 지정할 수 있습니다.

예를 들어, 두 번째 `<div>` 요소의 `position` 속성을 `relative` 로 변경하고, 요소의 원래 위치로 부터 위에서 `28px`, 왼쪽에서 `48px` 떨어지도록 `top` 과 `left` 속성을 설정해보겠습니다.

```
div:nth-of-type(2) {  
  position: relative;  
  top: 28px;  
  left: 48px;  
  background: cyan;  
  opacity: 0.8;  
}
```



position: absolute

`position` 속성값 중 가장 난해하고 주의해서 사용해야하는 속성값은 `absolute` 입니다. 아마도 `absolute` 라는 영단어의 의미 때문일텐데 `relative` 속성의 정반대 개념이라고 많이 오해를 합니다. 오히려 `absolute` 속성값은 `relative` 속성값과 함께 사용되는 경우가 많은 데 말입니다.

`position` 속성을 `absolute` 로 지정하면 사실 전혀 절대적(absolute)으로 요소를 배치해주지 않습니다. 오히려 배치 기준이 상황에 따라 굉장히 달라질 수 있는데요. 흥미롭게도 `position` 속성이 `absolute` 일 때 해당 요소는 배치 기준을 자신이 아닌 상위 요소에서 찾습니다. DOM 트리를 따라 올라가다가 `position` 속성이 `static` 이 아닌 첫 번째 상위 요소가 해당 요소의 배치 기준으로 설정되는데요. 만약에 해당 요소 상위에 `position` 속성이 `static` 이 아닌 요소가 없다면, DOM 트리에 최상위에 있는 `<body>` 요소가 배치 기준이 됩니다.

알고리즘이 상당히 복잡하게 느껴지죠? 하지만 실제로 `absolute` 속성값을 사용할 때 이러한 복잡한 특성을 활용하는 경우는 드뭅니다. 대부분의 경우, 부모 요소(가장 가까운 상위 요소)를 기준으로 `top`, `left`, `bottom`, `right` 속성을 적용해야하기 때문입니다. 따라서 어떤 요소의 `position` 속성을 `absolute` 로 설정하면, 부모 요소의 `position` 속성을 `relative` 로 지정해주는 것이 관례입니다.

```
main {  
  position: relative;  
  width: 300px;  
  height: 400px;  
  background: tomato;  
}
```

그 다음 두 번째 `<div>` 요소의 `position` 속성을 `absolute` 로 변경하고, 부모 요소를 기준으로 하단에서 `8px` , 우측에서 `16px` 떨어지도록 `bottom` 과 `right` 속성을 설정해주겠습니다.

```
div:nth-of-type(2) {  
  position: absolute;  
  bottom: 8px;  
  right: 16px;  
  background: cyan;  
  opacity: 0.8;  
}
```

```

main {
  position: relative;
  width: 300px;
  height: 400px;
  background: ■ tomato;
}

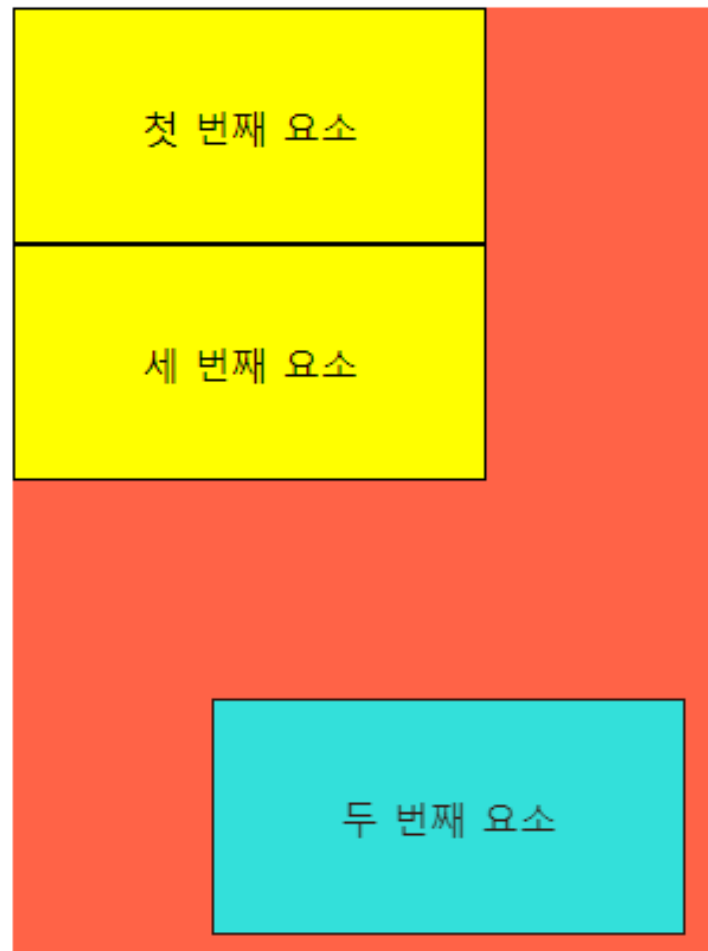
div {
  width: 200px;
  height: 100px;
  border: 1px solid;
  background: ■ yellow;
  text-align: center;
  line-height: 100px;
  box-sizing: border-box;
}

```

```

div:nth-of-type(2) {
  position: absolute;
  bottom: 8px;
  right: 16px;
  background: ■ cyan;
  opacity: 0.8;
}

```



여기서 꼭 짚고 넘어가야하는 부분이 있는데요. 바로 `position: absolute` 인 요소는 HTML 문서 상에서 독립되어 앞뒤에 나온 요소와 더 이상 상호작용을 하지 않게 된다는 것입니다. 따라서 위에서 보이는 것처럼, 첫 번째 요소 아래에 바로 세 번째 요소가 배치되었습니다.

position: fixed

화면을 위아래로 스크롤하더라도 브라우저 화면의 특정 부분에 고정되어 움직이지 않는 UI를 본 적이 있으신가요? 보통 라이브 채팅 버튼을 구현할 때 많이 쓰이는 기법인데요. `position` 속성을 `fixed` 로 지정하면 이렇게 요소를 항상 고정된(fixed) 위치에 배치할 수 있습니다.

이게 가능한 이유는 `fixed` 속성값의 배치 기준이 자신이나 부모 요소가 아닌 뷰포트(viewport), 즉 브라우저 전체화면이기 때문인데요. `top`, `left`, `bottom`, `right` 속성은 각각 브라우저 상단, 좌측, 하단, 우측으로 부터 해당 요소가 얼마나 떨어져있는지를 결정합니다.

두번째 `<div>` 요소의 `position` 속성을 `fixed` 로 변경하고, 뷰포트 기준으로 하단에서 `8px`, 우측에서 `16px` 떨어지도록 `bottom` 과 `right` 속성을 설정해주겠습니다.

`position: fixed` 인 요소도 `position: absolute` 인 요소와 마찬가지로 HTML 문서 상에서 독립되어 앞뒤에 나온 요소와 더 이상 상호작용을 하지 않습니다.

```
div:nth-of-type(2) {  
  position: fixed;  
  bottom: 8px;  
  right: 16px;  
  background: cyan;  
  opacity: 0.8;  
}
```

position: sticky

`position` 속성의 `sticky` 값은 CSS에서 비교적 최근에 추가된 속성값인데요. 특이하게도 브라우저 화면을 스크롤링할 때 효과가 나타납니다.

사실 말로 장황하게 설명하는 것 보다는 예제를 보는 게 이해가 빠를 수 있는데요.

먼저, `<div>` 요소의 부모인 `<main>` 요소의 높이를 줄이고 스크롤링이 가능해지도록 `height` 외 `overflow` 속성을 조정해줍니다.

```
main {  
  width: 300px;  
  height: 120px;  
  overflow: auto;  
  background: tomato;  
}
```

그 다음, 두번째 `<div>` 요소의 `position` 속성을 `sticky` 로 변경하고, `top` 속성을 `0` 으로 설정해주겠습니다.

```
div:nth-of-type(2) {  
  position: sticky;  
  top: 0;  
  background: cyan;  
  opacity: 0.8;  
}
```