# CSS (Cascading Style Sheets)

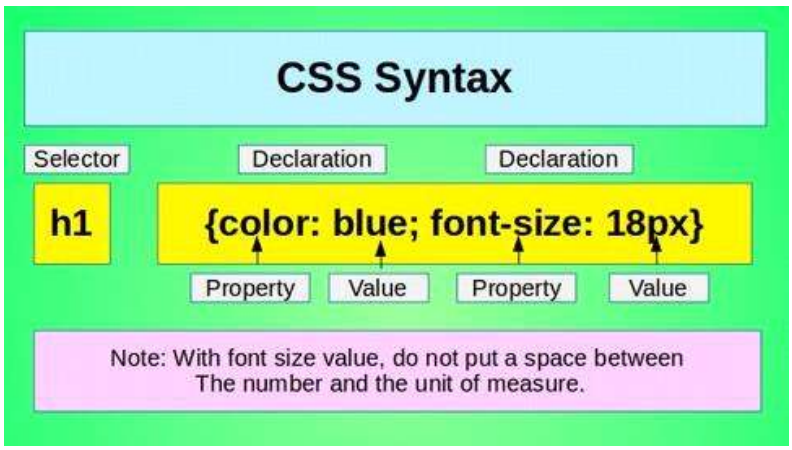
## ☑ selector

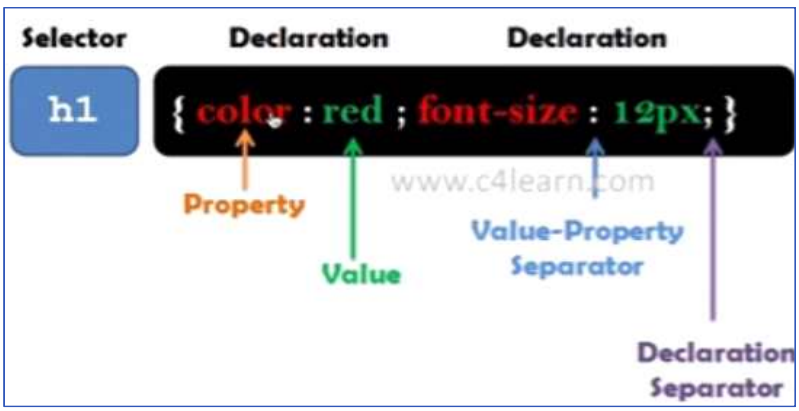
### ❖ selector와 declaration(선언)

어떤 tag를 디자인하기 위해서는 디자인하려는 tag를

1. 선택하고 (selector)
2. 선택한 대상에게 효과를 줘야 합니다. (선언)

이번 시간에는 CSS에서 가장 기본적인 문법이라고 할 수 있는 이 selector(selector)와 선언(declaration)에 대해서 학습한다. 가장 중요한 내용이라고 할 수 있다.




**[ index01.html ]**

```
<!DOCTYPE html>
<html>
  <head>
```

```
    <style>
      li{
        color: red;
        text-decoration: underline;
      }
    </style>
  </head>
  <body>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
  </body>
</html>
```

## ❖ selector의 종류

### ☞ selector의 타입들

다양한 형태의 selector를 알아본다.

☞ tag selector

☞ id selector

☞ class selector

3가지의 selector를 학습할 것이다. 이미 1가지 selector를 학습하였으며, 2개만 더 학습하면 된다.

☞ tag selector

tag를 선택이다. 아래 코드는 이 문서의 모든 li tag 라는 뜻이다.

```
li{color: red}
```

☞ id selector

아이디 속성의 값에 해당하는 tag를 선택하는 selector 이다. 아래의 코드는 이 문서에서 id값이 select인 tag라는 뜻이다.

```
#select{font-color: 50px;}
```

☞ class selector

클래스 속성의 값에 해당하는 tag들을 선택하는 selector 이다.

[ index02.html ]

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      li{
        color: blue;
        text-decoration: underline;
      }
      #select{
        font-size: 30px;
      }
     .deactive{
         text-decoration: line-through;
     }
    </style>
  </head>
  <body>
    <ul>
      <li class="deactive">HTML</li>
      <li id="select">CSS</li>
      <li class="deactive">JavaScript</li>
    </ul>
  </body>
</html>
```


## ❖ parent child selector

어떤 tag의 하위에 있는 tag를 선택하고 싶을 때는 좀 더 복합적인 selector를 사용하게 된다. 여기서는 parent tag 아래에 있는 child tag를 선택하는 몇가지 방법을 학습하도록 한다.

### ☞ ancestor descendant selector

아래의 tag는 ul 밑에 있는 모든 tag를 선택한다.

```
ul li {
    color: red;
}
```

### ☞ parent child selector

아래 selector는 #lecture 바로 밑에 있는 li 만을 선택한다.

```
#lecture > li {
```

```
    border: 1px solid red;
}
```

### ☞ 친구 selector

(이런 용어는 없다)  아래 코드는 ul과 ol을 동시에 선택한다.

```
ul, ol {
    background-color: powderblue;
}
```

**[ index03.html ]**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      ul li {
        color: red;
      }
      #lecture>li{
        border: 1px solid blue;
      }
      ul,ol{
        background-color: rgb(137, 240, 52);
      }
    </style>
  </head>
  <body>
    <ul>
      <li>HTML</li>
      <li>CSS</li>
      <li>JavaScript</li>
    </ul>
    <ol id="lecture">
      <li>HTML</li>
      <li>CSS
        <ol>
          <li>selector</li>
          <li>declaration</li>
        </ol>
      </li>
      <li>JavaScript</li>
    </ol>
  </body>
</html>
```

## ⊙ Descendant combinator(자손 결합자)

보통 한 칸의 공백 ( ) 문자로 표현하는 descendant combinator는 두 개의 selector를 조합하여, 뒤쪽 selector element의 ancestor(parent, parent의 parent, parent의 parent의 parent...)에 앞쪽 selector element가 존재할 경우 선택한다. descendant combinator를 활용하는 selector를 descendant selector(자손 선택자)라고 부른다.

```
/* List items that are descendants of the "my-things" list */
ul.my-things li {
  margin: 2em;
}
```

기술적으로 descendant combinator는 하나 이상의 CSS 공백 문자이므로 스페이스 외에도 캐리지 리턴, 폼 피드, 새 줄, 탭 문자도 해당한다. 또한 combinator를 구성하는 공백 문자는 CSS 주석을 임의의 개수만큼 포함할 수 있다.

**[ Syntax ]**

```
selector1 selector2 {
  /* property declarations */
}
```

**[ index04.html ]**

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      li {
        list-style-type: disc;
      }

      li li {
        list-style-type: circle;
      }
    </style>
  </head>
  <body>
    <ul>
      <li>
        <div>Item 1</div>
        <ul>
          <li>Subitem A</li>
          <li>Subitem B</li>
        </ul>
      </li>
```

```
      <li>
        <div>Item 2</div>
        <ul>
          <li>Subitem A</li>
          <li>Subitem B</li>
        </ul>
      </li>
    </ul>
  </body>
</html>
```

- Item 1
  - Subitem A
  - Subitem B
- Item 2
  - Subitem A
  - Subitem B

## ⊙ [CSS] child, descendant selector 차이점

☞ **s1 s2 : s1 요소에 포함된 s2 요소를 선택한다. (desendant 관계)**

☞ **s1 > s2 : s1 요소의 직계 자식 요소인 s2를 선택한다. (child 관계)**

결론부터 말하자면 child 가 더 작은 범위이다. 즉 child는 바로 선택 tag 바로 아래 계층에 있는 것들이며, desendant은 선택 tag 안에 들어가 있는 모든 것 이라고 생각하면 된다.

**[ index05.html ]**

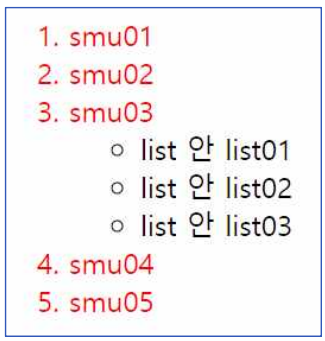
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>child, descendant</title>
</head>
<body>
    <ol>
        <li>smu01</li>
        <li>smu02</li>
        <li>smu03</li>
            <ul>
                <li>list 안 list01</li>
                <li>list 안 list02</li>
                <li>list 안 list03</li>
            </ul>

        <li>smu04</li>
        <li>smu05</li>
    </ol>
</body>
</html>
```

1. smu01
2. smu02
3. smu03
   - list 안 list01
   - list 안 list02
   - list 안 list03
4. smu04
5. smu05

## ❖ CSS child selector (s1>s2)

**[ index06.html ]**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>child, descendant</title>
    <!-- 추가된 부분!! -->
    <style>
      ol > li {
          color:red;
      }
    </style>
</head>
<body>
    <ol>
        <li>smu01</li>
        <li>smu02</li>
        <li>smu03</li>
            <ul>
                <li>list 안 list01</li>
                <li>list 안 list02</li>
                <li>list 안 list03</li>
            </ul>

        <li>smu04</li>
        <li>smu05</li>
    </ol>
</body>
</html>
```




중간에 <ul></ul>로 둘러싸인 부분들(list 안 list)라고 되어있는 부분을 제외하고 글씨가 빨간색으로 바뀌었다. 왜냐하면 그 부분은 <ul></ul>의 child 이기 때문이다.

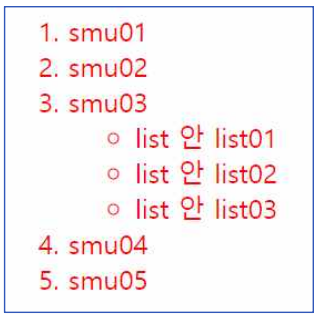
## ❖ CSS descendant selector (s1 s2)

이제 ol의 후손부분인 li를 보라색으로 바꿔보도록 하자. 코드는 아래와 같다.

**[ index07.html ]**

```
<!DOCTYPE html>
```

```html
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>child, descendant</title>
    <!-- 변경된 부분 -->
    <style>
      ol li {
          color: red;
      }
  </style>
</head>
<body>
    <ol>
        <li>smu01</li>
        <li>smu02</li>
        <li>smu03</li>
            <ul>
                <li>list 안 list01</li>
                <li>list 안 list02</li>
                <li>list 안 list03</li>
            </ul>

        <li>smu04</li>
        <li>smu05</li>
    </ol>
</body>
</html>
```




list 안 list는 <ul>의 child 이지만, child의 child는 descendant이기 때문에 함께 바뀌는 것을 확인할 수 있다. 결론적으로 child selector(s1>s2)는 바로 아래 있는 것만, descendant selector(s1 s2)는 아래 있는 해당 tag 모두 선택한다.

## ❖ selector 공부 팁

CSS Diner

CSS selector를 게임의 형식을 통해서 익힐 수 있는 사이트

http://flukeout.github.io/

https://opentutorials.org/module/2367/13516
https://www.youtube.com/watch?v=Me7qCXl94nQ&list=PLuHgQVnccGMDaVaBmkX0qfB45R_bYrV62&index=4

## ❖ Pseudo class selector

pseudo(가상) class selector는 class selector는 아니지만 element들의 상태에 따라서 마치 class selector처럼 여러 element를 선택할 수 있다는 점에서 붙은 이름이다. 여기서는 주요 class selector를 알아본다.

### ☞ 링크와 관련된 pseudo class selector

✓ :link - 방문한 적이 없는 링크
✓ :visited - 방문한 적이 있는 링크
✓ :hover - 마우스를 롤오버 했을 때
✓ :active - 마우스를 클릭했을 때

위의 selector는 순서대로 지정하는 것이 좋다. 특히 visited의 경우는 보안상의 이유로 아래와 같은 속성만 변경이 가능하다.

✓ color
✓ background-color
✓ border-color
✓ outline-color
✓ The color parts of the fill and stroke properties

## ❖ 여러 가지 selector들

여기서는 여러가지 selector를 알아본다. 그런데 지금까지 배운 selector로도 많은 것을 할 수 있다. 만약 CSS 초심자라면 이것을 공부하는 것 보다는 나머지 진도는 나중에 보면 된다.
지루함을 덜기 이해서 아래 게임을 이용해서 학습하도록 한다.

http://flukeout.github.io/

## ☑ 속성을 공부하는 방법

### ❖ 중요한 속성들

CSS의 속성(property)은 약 250개 정도이다. 일상 생활에서 사용하는 자연어의 어휘에 비하면 몇개 되지 않지만, 이것들을 하나씩 열거해서 공부하는 것은 뇌를 혹사시는 일이다.
학습하는 속성들은 가장 빈도수가 높은 CSS의 속성 순으로 공부하고, 빈도가 낮은 것들은 그런 것이 있다는 정도만 확인하고 필요할 때 검색, 질문 등을 이용하는 것이다.

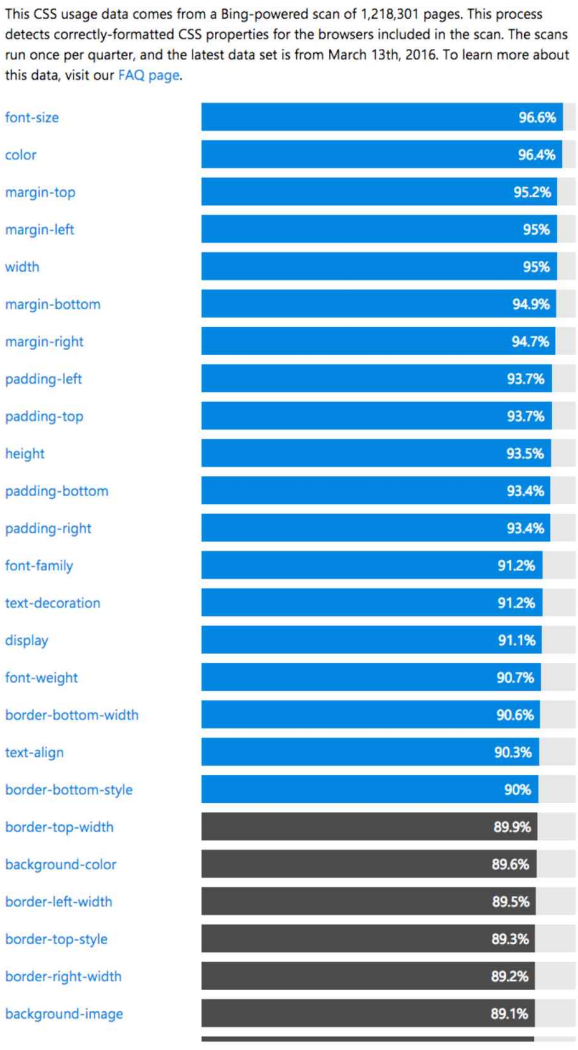
### ❖ 속성에 대한 통계

과학적인 방법으로 빈도수를 조사하기 위해서 마이크로소프트에서 검색엔진 bing.com을 통해서 수집한 전세계 웹페이지의 통계 자료를 인용도록 한다.

https://developer.microsoft.com/en-us/microsoft-edge/platform/usage/

이 자료에 따르면 가장 빈도수가 높은 속성은 아래와 같다.

- ☞ Typography
- ☞ box model
- ☞ color



## ☑ Typography

본 학습의 하위 토픽에서는 서체에 대해서 다룬다. HTML이 처음 등장했을 때는 이미지를 표현하는 기능도 없었다. 단지 문자만이 존재하는 미디어였다. 그렇기 때문에 HTML에서 서체와 관련된 부분은 가장 유서깊은 분야이면서 동시에 속성의 빈도수도 가장 많다. 여기서는 서체에 대해서 다룬다.

### ☞ 초심자를 위한 제안

초심들은 다음 순서로 학습하도록 한다.

☞ font-size
☞ color
☞ text-align

### ❖ font-size

글꼴의 크기를 지정한다. 주요 단위로는 px, em, rem이 있다.

☞ rem
html 태그에 적용된 font-size의 영향을 받는다. html 태그의 폰트 크기에 따라서 상대적으로 크기가 결정되기 때문에 이해하기 쉽다. 가장 바람직한 단위이다. 이것을 사용한다.

☞ px
모니터 상의 화소 하나의 크기에 대응되는 단위이다. 고정된 값이기 때문에 이해하기 쉽지만, 사용자가 글꼴의 크기를 조정할 수 없기 때문에 가급적 사용을 하지 않는 것이 좋다.

☞ em
부모 태그의 영향을 받는 상대적인 크기이다. 부모의 크기에 영향을 받기 때문에 파악하기가 어렵다. rem이 등장하면서 이 단위 역시 사용이 권장되지 않는다.

### ⊙ rem 과 em 의 차이점

HTML 에서 CSS 를 이용해 font나 margin, padding 등의 크기를 지정할 때, 정확한 px 로 지정할 수도 있지만, 상대적인 값을 표현해주는 단위인 rem 이나 em 을 사용하기도 한다.

★ rem
rem의 개념은 기준이 되는 값을 지정된 배수로 변환해 표현한 크기를 의미한다. 여기서 기준이 되는 값

은 최상위 요소(보통은 html 태그)에서 지정된 font-size 의 값을 이야기 한다.
예를 들어 기준이 되는 값, 즉 html 태그의 font-size 값이 이 16px 라면 2rem 은 32px를 의미한다.

```
html { font-size: 16px; }
div {
    font-size: 1.5rem;    /* 24px */
    margin: 2rem;         /* 32px */
    padding: 1.25rem;     /* 20px */
}
```

★ em

em의 개념은 기준이 되는 값을 지정된 배수로 변환해 표현한 크기를 의미한다. 여기서 기준이 되는 값이란 현재 스타일 지정 요소의 font-size 값을 의미한다.
예를 들어 기준이 되는 값, 즉 현재 요소의 font-size 값이 16px 이라면 2em 은 32px 을 의미한다.

```
div { font-size: 16px; }
div {
    font-size: 1.5em;     /* 24px */
    margin: 2em;          /* 32px */
    padding: 1.25em;      /* 20px */
}
```

차이점을 간단하게 이야기 하자면 rem 과 em 은 기준이 되는 값이 다르다는 것이다. rem은 **최상위 태그의 font-size 값이 기준**이며, em은 **현재 요소의 font-size 값이 기준**이라는 것이다. 만약 em을 사용해 스타일을 지정한 요소에 따로 font-size 값이 지정되지 않았다면, 해당 요소는 부모 요소로부터 font-size 값을 상속(inherit) 받을 것이며, em은 그 상속 받은 값을 기준으로 삼게된다.

예를 들어, 최상위에 html 태그가 있고, 그 밑에 body 태그로 본문 영역이 표시되며, container 라는 클래스를 가진 div 요소 하위에 content1 과 content2 라는 클래스의 div 요소가 자리 잡고있다.

```
html { font-size: 16px; }
body { font-size: 2em; }
div.container { font-size: 2em; }
div.content1 { font-size: 2rem; }
div.content2 { font-size: 2em; }
```

이 경우 content1의 font-size는 32px 이지만, content2의 font-size는 16px * 2 * 2 * 2 이므로 128px 로 지정된다. 둘의 차이를 정확해 이해 하지 못하고 있으면, 웹 페이지를 스타일링 하면서 절대 풀리지 않는 숙제에 빠진다거나 곤란한 상황에 처할 수 있으니 반드시 숙지해야 한다.

**[ index08.html ]**

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>child, descendant</title>

  <style>
    #px{ font-size: 16px; }
    #rem{ font-size: 2rem; }
  </style>
</head>
<body>
  <div id="px">PX</div>
  <div id="rem">REM</div>
</body>
</html>
```


## ❖ color

글꼴의 컬러를 지정하는 방법에 대해서 학습한다. 컬러의 다양한 종류는 아래의 페이지를 참고한다.

https://www.w3schools.com/css/css_colors.asp
https://www.w3schools.com/colors/colors_picker.asp

## ❖ text-align

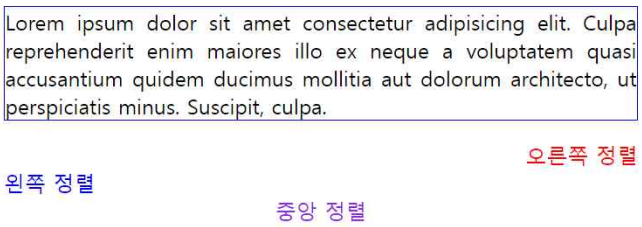
text-align 속성은 텍스트의 정렬 방향을 의미한다.

- ☞ left: 왼쪽 정렬
- ☞ right: 오른쪽 정렬
- ☞ center: 중앙 정렬
- ☞ justify: 양쪽 정렬 (자동 줄바꿈시 오른쪽 경계선 부분 정리)

**[ index09.html ]**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>child, descendant</title>
```

```
<style>
  p{
    text-align: justify;
    border:1px solid blue;
  }
  #box1 { text-align: right; color: red;}
  #box2 { text-align: left; color: blue;}
  #box3 { text-align: center; color: blueviolet; }
</style>
</head>
<body>
  <p>
    Lorem ipsum dolor sit amet consectetur adipisicing elit. Culpa reprehenderit enim maiores illo ex neque a voluptatem quasi accusantium quidem ducimus mollitia aut dolorum architecto, ut perspiciatis minus. Suscipit, culpa.
  </p>
  <div id="box1">오른쪽 정렬</div>
  <div id="box2">왼쪽 정렬</div>
  <div id="box3">중앙 정렬</div>
</body>
</html>
```




## ❖ font

### ☞ font-family

font-family는 서체를 지정하는 속성이다. 아래와 같은 방법으로 한다.

```
h1{
    font-family: "Times New Roman", Times, serif;
}
```

위 코드의 의미는 h1 태그를 Times New Roman을 지정한다. 그런데 사용자의 컴퓨터에 폰트가 없으면 Times를 사용하게 된다. 이 때 마지막 폰트는 포괄적인 폰트로 지정한다. 아래와 같은 것이 있다.

- ✓ serif (장식이 있는 폰트)
- ✓ sans-serif
- ✓ cursive (흘림체)
- ✓ fantasy
- ✓ monospace (고정폭)

## ❖ 서체

### ☞ 고정폭과 가변폭

서체는 크게 두가지 방법으로 구분할 수 있다. 고정폭(monospaced)과 가변폭 이다. 아래는 두가지 방식의 차이를 보여주는 이미지 이다.

### ☞ serif vs sans serif

serif는 글꼴에서 사용하는 장식을 의미한다. sans는 부정의 표현이다. 즉 sans serif는 serif가 아니다는 뜻이다.


출처 : http://www.w3schools.com/css/css_font.asp

### ☞ 폰트 랭킹

많이 사용하는 폰트의 랭킹을 알려주는 사이트가 있다. 이 사이트를 통해서 가장 많이 사용하는 폰트는 무엇인지를 알아보는 것도 재미있을 것 같다.

http://www.fontreach.com/#top

### ☞ 국내폰트

네이버에서 운영하는 D2 Coding 폰트이다다.
https://github.com/naver/d2codingfont

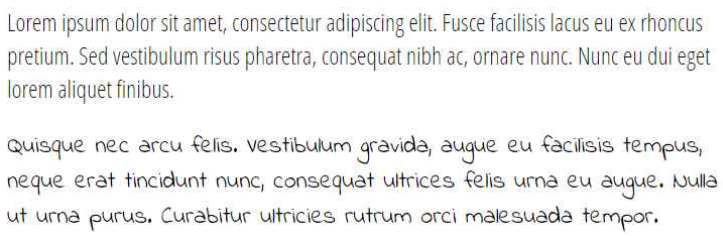
## ❖ 웹폰트

웹폰트는 사용자가 가지고 있지 않은 폰트를 웹페이지에서 사용할 수 있는 방법이다. 폰트를 서버에서 다운로드하는 방식이라고 할 수 있다. 구글에서 제공하는 무료 웹폰트 서비스인 google fonts를 이용해서 웹폰트를 설명한다.

**[ index10.html ]**

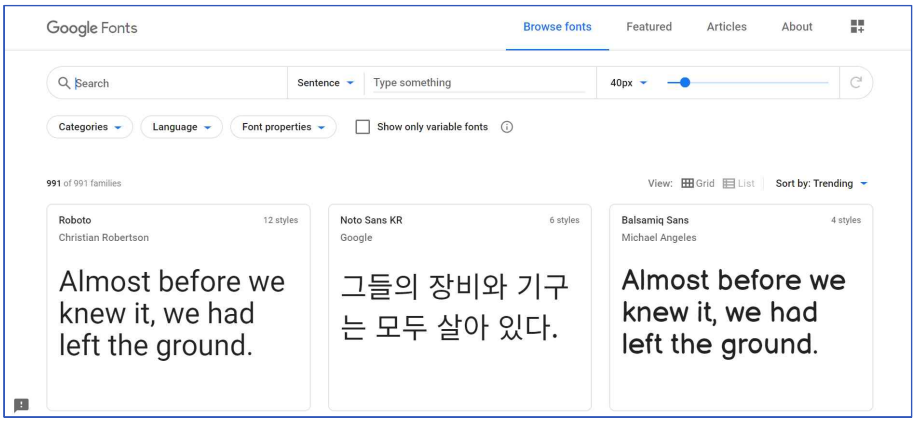
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>child, descendant</title>
```

```
  <link href="https://fonts.googleapis.com/css?family=Indie+Flower|Londrina+Outline|Open+Sans+Condensed:300"
rel="stylesheet">
  <style>
      #font1{
        font-family: 'Open Sans Condensed', sans-serif;
      }
      #font2{
        font-family: 'Indie Flower', cursive;
      }
    </style>
  </head>
  <body>
    <p id="font1">
      Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce facilisis lacus eu ex
rhoncus pretium. Sed vestibulum risus pharetra, consequat nibh ac, ornare nunc. Nunc eu dui eget
lorem aliquet finibus.
    </p>
    <p id="font2">
      Quisque nec arcu felis. Vestibulum gravida, augue eu facilisis tempus, neque erat
tincidunt nunc, consequat ultrices felis urna eu augue. Nulla ut urna purus. Curabitur ultricies
rutrum orci malesuada tempor.
    </p>
  </body>
</html>
```



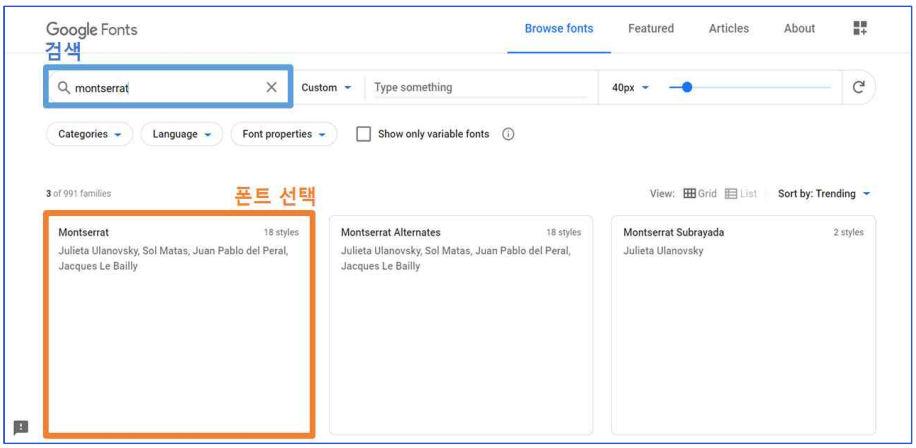
## ⊙ Goolge Fonts (구글 폰트) 사용 방법

### 1. 사이트 들어가기 (https://fonts.google.com/)




### 2. 원하는 폰트 선택하기

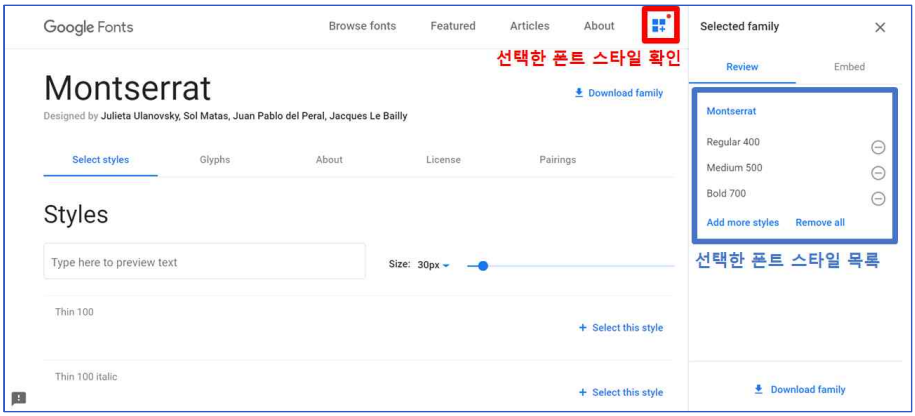
구글 폰트 사이트에 들어갔다면, 원하는 폰트를 선택하면 된다. 맘에 드는 폰트를 선택해도 되고 검색을 통해 원하는 폰트를 찾을 수도 있다. 예시로 montserrat 폰트를 검색해서 선택해 본다.




### 3. 폰트 스타일 선택하고 추가하기

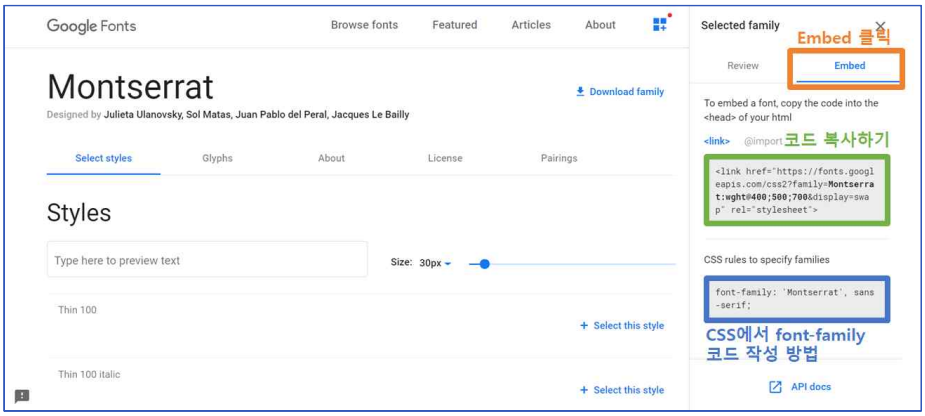
폰트를 정했다면, 그 폰트의 다양한 스타일(글씨 굵기 또는 기울임체 등) 중에서 사용할 스타일을 선택해야 한다. + Select this style 을 누르면 해당 폰트 스타일이 추가된다.

선택한 폰트 스타일은 아래 그림에 표시된 버튼을 누르면 확인할 수 있다. Regular 400, Medium 500, Bold 700 스타일을 선택했다.




#### 4. Embedding 코드 복사하기

이제 해당 폰트를 HTML 파일에 embedding 하기 위한 코드를 복사해서 HTML 파일에 코드를 추가하면 된다. Embed 탭을 클릭해서 <link> 코드를 복사한 뒤 HTML <head> 부분에 추가해 준다. 링크된 폰트를 CSS에서 사용하는 방법은 아래 그림의 파란색 박스에 나와있다.




## ☑ 조화

하나의 엘리먼트는 다양한 요소의 영향을 받게된다. 그렇기 때문에 여러 효과가 엘리먼트를 두고 대치하고 있을 때 브라우저는 어떤 효과의 손을 들어줘야 하는가에 대한 결정을 해야한다. 이를 위한 여러 규칙들이 있다. 그 규칙을 이해하지 못하면 CSS 디자인이 다소 혼란스럽게 될 것이다. 하위 학습에서는 이런 규칙들에 대해서 알아본다.

### ❖ 상속

상속은 부모 엘리먼트의 속성을 자식 엘리먼트가 물려받는 것을 의미한다. 상속은 CSS에서 생산성을 높

이기 위한 중요한 기능이다.

아래와 같은 코드가 있다고 가정해보자.

**[ index11.html ]**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Inhritance</title>
 </head>
  <body>
    <div id="container">
      <h1>수업순서</h1>
      <ul>
        <li>HTML5</li>
        <li>CSS3</li>
        <li>JavaScript</li>
      </ul>
      <h1>수업참가자</h1>
      <ul>
        <li>smu01</li>
        <li>smu02</li>
        <li>smu03</li>
        <li>smu04</li>
      </ul>
    </div>
  </body>
</html>
```


그 결과는 다음과 같다.
만약 위에서 서체의 색상을 지정해야 한다면 어떻게 하면 될까? 모든 엘리먼트의 색상을 하나 하나 지정해야 할까? 그것 보다 효율적인 방법이 있다. 바로 상속을 이용하는 것이다.
아래처럼 html의 서체 색상만 조정하면 하위에 있는 모든 엘리먼트의 색상이 자동으로 바뀌게 됩니다. 이것은 각 엘리먼트가 상위 엘리먼트의 서체 색상 값을 물려 받기 때문입니다.

```
html{color: blue;}
```

이러한 기능을 상속(inheritance)이라고 한다.

### ☞ 상속을 하는 속성과 하지 않는 속성

하지만 모든 속성이 상속을 지원하는 것은 아니다. 상속을 하면 오히려 비효율적인 속성들이 있다. 아래 문서를 보면 상속하는 속성과 상속하지 않는 속성의 목록을 볼 수 있다.

https://www.w3.org/TR/CSS21/propidx.html

이 표에서 아래 컬럼이 상속 여부를 알려주는 내용입니다.

## Appendix F. Full property table

*This appendix is informative, not normative.*

| Name | Values | Initial value | Applies to (Default: all) | Inherited? | Percentages (Default: N/A) | Media groups |
|---|---|---|---|---|---|---|
| 'azimuth' | <angle> \| [[ left-side \| far-left \| left \| center-left \| center \| center-right \| right \| far-right \| right-side ] \|\| behind ] \| leftwards \| rightwards \| inherit | center | | yes | | aural |
| 'background-attachment' | scroll \| fixed \| inherit | scroll | | no | | visual |
| 'background-color' | <color> \| transparent \| inherit | transparent | | no | | visual |
| 'background-image' | <uri> \| none \| inherit | none | | no | | visual |
| 'background-position' | [ [ <percentage> \| <length> \| left \| center \| right ] [ <percentage> \| <length> \| top \| center \| bottom ]? ] \| [ [ left \| center \| right ] \|\| [ top \| center \| bottom ] ] \| inherit | 0% 0% | | no | refer to the size of the box itself | visual |
| 'background-repeat' | repeat \| repeat-x \| repeat-y \| no-repeat \| inherit | repeat | | no | | visual |

## ❖ stylish

누구나 기초가 중요하다고 말한다. 그러나 기초만으로도 많은 것을 할 수 있다는 말은 잘 하지 않는다. 지금까지 배운 것으로도 많은 것을 할 수 있다. 또한 Web Site를 만드는 거대하고 거창한 목표가 아니라도 이미 만들어진 웹사이트의 디자인을 조금 수정해서 자신의 취향에 따라서 변경할 수 있는 것도 CSS를 통해서 할 수 있는 일이라고 생각한다. 여기서는 웹 페이지의 생산자가 아니라 소비자 입장에서 CSS를 알았을 때 할 수 있는 것을 알아본다.

### ☞ stylish

stylish는 Web Site의 디자인을 사용자 마음대로 수정할 수 있는 기능이다. 이 기능을 이용해서 주어진 데로 웹 페이지를 소비하는 것이 아니라 자신의 취향을 반영할 수 있다. 또 다른 사람이 만든 디자인을

적용해서 간편하게 사이트의 디자인을 변경할 수도 있다. 이 도구의 사용법을 알아본다.

https://userstyles.org/

## ❖ Cascading

Cascade는 폭포라는 의미가 있다. 처음 html이 등장했을 때는 CSS가 없었다. 웹의 사용자들은 곧 디자인을 요구하기 시작한다. 웹의 고안자들은 html을 꾸며주는 언어의 필요성에 공감하기 시작한다. 그렇게해서 등장한 것이 CSS 이다.

CSS는 Cascading Style Sheet의 약자이다. Cascading은 CSS의 첫번째 글자로 등장할 만큼 가장 중요한 기능 다시 말해 철학이라는 의미가 있다. Cascading을 사전에서 찾아보면 폭포라는 의미가 있다. 즉 웹 페이지의 디자인이 Web Browser의 기본 디자인과 브라우저 사용자의 디자인 그리고 Web Page 저자의 디자인이 결합될 수 있다는 점에 착안하고 있다고 할 수 있을 것 같다.

즉, Web Browser, 사용자, 컨텐츠 생산자의 조화를 중요한 철학으로 삼고 있다고 생각된다. 여기서는 하나의 엘리먼트에 대해서 다양한 효과가 영향력을 행사하려고 할 때 우선순위를 어떻게 설정하는가에 대한 규칙인 Cascading에 대해서 알아본다.

**[ index12.html ]**

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>CSS Inheritance</title>
  <style>
    li{color:red;}                // !important
    #idsel{color:purple;}
    .classsel{color:green;}
  </style>
</head>
<body>
  <ul>
    <li>html</li>
    <li id="idsel" class="classsel" style="color:blue">css</li>
    <li>javascript</li>
  </ul>
  <ol>
    <li>style attribute</li>
    <li>id selector</li>
    <li>class selector</li>
```

```
    <li>tag selector</li>
  </ol>
</body>
</html>
```

- html
- css
- javascript

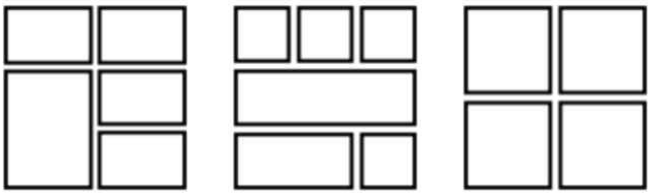
1. style attribute
2. id selector
3. class selector
4. tag selector

## ☑ layout

정보를 잘 정리 정돈해서 일관된 모습으로 보여지도록 하는 것은 디자인에서 매우 중요한 주제이다. 구획을 나누고 적절히 정보를 배치하는 것을 layout(layout)이라고 한다. 안타깝게도 CSS를 이용해서 layout을 잡는 것은 다소 어려운 것이 사실이다. 하위 토픽에서는 layout을 잡기 위해서 필요한 기본적인 개념들을 알아보고 실제로 사용하는 모습도 살펴보도록 한다.

아래 내용을 먼저 학습하고 나머지는 나중에 학습할 것을 추천한다.

- ☞ Inline level vs. block level
- ☞ Box Model
- ☞ **마진겹침 현상**
- ☞ **포지션**
- ☞ float



### ❖ inline level vs block level

html 엘리먼트들은 크게 두가지로 구분된다.

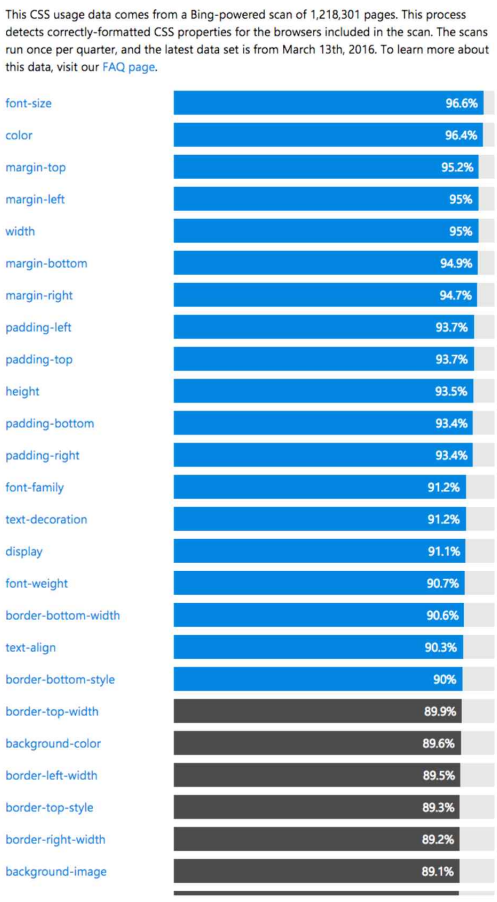
- ☞ **화면 전체를 사용하는 태그** => block level element
- ☞ **화면의 일부를 차지하는 태그** => inline level element

inine level element와 block level element의 차이점을 학습면서 display라는 중요한 속성에 대해서도 배우게 된다.

[ index13.html ]

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>inline vs. block level</title>
  <style>
    h1, a{border:1px solid red;}
/*  h1{display: inline;}
    a{display: block;}
*/
</style>
</head>
<body>
<h1>Hello world</h1>
안녕하세요. <a href="http://www.semyung.ac.kr">세명대학교</a>입니다.
</body>
</html>
```


**※ 참고 : ,  tag 비교**



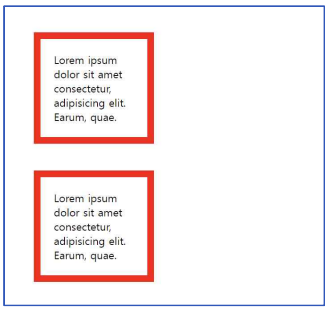

## ❖ Box Model

margin-top, margin-left, width, margin-bottom, margin-right, padding-left, padding-top, height, padding-bottom, padding-right, display, border-bottom-style, border-top-width (70% 이상)

Box Model은 각각의 tag들이 Web Page에 표현될 때 그 tag의 부피감을 결정한다. 여백이라든지 위치라든지 이러한 것들을 결정하는 것을 Box Model 이라고 한다.
Box Model 이라고 한 이유는 각각의 tag들을 그 tag 바깥쪽에 사각형의 Box와 같은 모양으로 둘러쌓여 있다고 하여 Box 라는 표현을 하고 있다.

**[ index14.html ]**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Box Model</title>
  <style>
    p, a{
      border: 10px solid red;
      padding: 20px;
      margin:  40px;
      width: 120px;
    }
</style>
</head>
<body>
  <p>
    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Earum, quae.
  </p>
  <p>
    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Earum, quae.
  </p>
</body>
</html>
```




이러한 Box Model이 inline level에서는 조금 다르게 동작을 한다.

위와 같이 이 a tag 주변부에 있는 text와 간격이 떨어진 것을 볼 수가 있다. 즉, margin이 적용된 것을 알 수가 있다. 또한 content와 테두리 사이의 간격이 생겼기 때문에 padding이 적용된 것을 알 수가 있다. 그러나 width 값을 적용이 되지 않는 것을 볼 수가 있다. inline 방식에서는 CSS가 width나 height 값을 무시하는 것을 알 수가 있다.

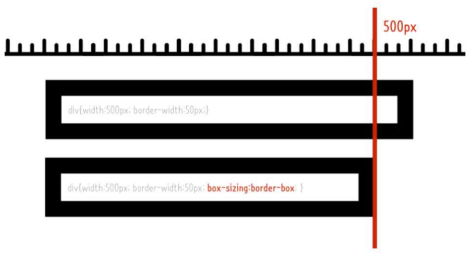
**[ index15.html ]**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Box Model</title>
  <style>
    p, a{
      border: 10px solid red;
      padding: 20px;
      margin:  40px;
```

```
      width: 120px;
    }
</style>
</head>
<body>
  <p>
    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Earum, quae.
  </p>
  <p>
    Lorem ipsum dolor sit amet consectetur, adipisicing elit. Earum, quae.
  </p>
  안녕하세요.
  <a href="http://www.semyung.ac.kr">세명대학교</a>입니다.
</body>
</html>
```

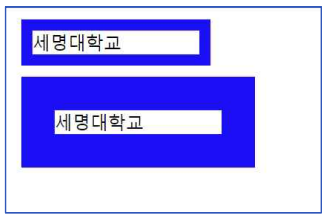

## ❖ box-sizing

box-sizing은 박스의 크기를 화면에 표시하는 방식을 변경하는 속성이다. width와 height는 엘리먼트의 컨텐츠의 크기를 지정한다. 따라서 테두리가 있는 경우에는 테두리의 두께로 인해서 원하는 크기를 찾기가 어렵다. box-sizing 속성을 border-box로 지정하면  테두리를 포함한 크기를 지정할 수 있기 때문에 예측하기가 더 쉽다. 최근엔 모든 엘리먼트에 이 값을 지정하는 경우가 늘고 있다.




**[ index16.html ]**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>box-sizing</title>
  <style>
    *{
        box-sizing: border-box;
    }
    div{
        margin: 10px;
```



```
        width: 150px;
    }
    #small{
        border: 10px solid blue;
    }
    #large{
        border: 30px solid blue;
    }
</style>
</head>
<body>
  <div id="small">세명대학교</div>
  <div id="large">세명대학교</div>
</body>
</html>
```


이유는 element의 크기라고 하면 테두리 바깥쪽의 크기라고 생각하기 쉽다. 그러나 CSS 초기에는 content 영역의 padding과 border가 빠진 하얀색의 content 영역의 크기를 width값으로 지정하기로 하였기 때문에 이 두 개의 element의 content 크기는 같은 것을 알 수가 있다.
이러한 문제를 해결하기 위하여 CSS의 속성이 box-sizing 이라는 속성이다. 이 속성의 기본값은 content-box 이다. 즉 content의 크기만큼 width와 height값이 지정이 된다는 것인데 이것을 border-box로 변경하면 이 두 개의 element는 boder의 두께가 다름에도 불구하고 boder의 경계의 크기가 동일해 진다. 따라서 대부분의 경우에는 이것이 이해하기 쉽고 예측 가능한 결과이다. 이때 사용하는 것이 border-box 이다.
이렇게 CSS를 사용할 때 모든 element에 대하여 크기를 지정할 때 border를 기준으로 지정할 수 있게 이 Web Page에 등장하는 모든 tag라는 의미의 * selector를 사용하여 box-sizing: border-box; 를 지정하여 모든 element가 border의 width와 height의 값으로 사용할 수 있게 하여 CSS 사용을 쉽게 하는 방법을 채택한다.

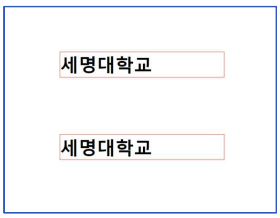
## ❖ margin-collapsing(마진겹침) 현상

마진겹침(margin-collapsing) 현상은 상하 margin값이 어떤 상황에서 사라지는 현상을 의미한다. 이 현상은 CSS 처음 접하는 사람들은 중요한 내용이 아니기 때문에 지나치면 된다. 그러나 이것을 이해하지 못하면 실무를 할 때 이해할 수 없는 CSS의 동작으로 인해서 깊은 화남이 생길 수 있으니까 언젠가는 학습하도록 한다.

**[ index17.html ]**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
```

```
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>margin-collapsing</title>
  <style>
    h1{
      border:1px solid red;
      margin:100px;
    }
  </style>
</head>
<body>
<h1>세명대학교</h1>
<h1>세명대학교</h1>
</body>
</html>
```




위에 있는 tag와 아래있는 tag의 margin 값 중에 더 큰 값이 두 개의 tag 사이의 간격이 된다. 이것이 가장 전형적인 margin-collapsing 현상이다.
li tag 가 3개 있을 때, 만약 margin-collapsing 현상이 없었다면 li tag 사잉의 간격이 너무 넓게 되기 때문에 matgin-collapsing 현상이 있으면 유리하게 된다.

## ❖ 포지션

엘리먼트의 위치를 지정하는 4가지 방법이 있다.

- ☞ static
- ☞ relative
- ☞ absolute
- ☞ fixed

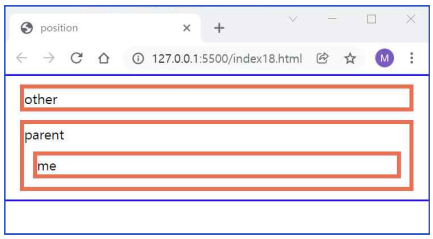
이 4가지 방법을 정확하게 이해하고 사용하는 것이 css를 자유자재로 이용하는데 중요하다.

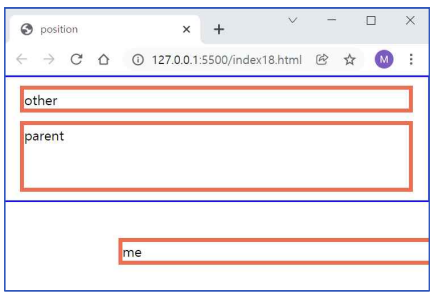
### ☞ static vs. relative

**[ index18.html ]**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
```

```
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>position</title>
  <style>
    html{border: 2px solid blue;}
    div{
        border:5px solid tomato;
        margin:10px;
    }
    /*
    #me{
      position: relative;
      left: 100px;
      right: 100px;
      bottom: 100px;
      top: 100px;
    }
    */
  </style>
</head>
<body>
  <div id="other">other</div>
  <div id="parent">
    parent
  <div id="me">me</div>
  </div>
</body>
</html>
```




이제부터 me element를 왼쪽, 오른쪽, 위, 아래로 이동시키도록 한다.
left와 right가있을 때 left가 우선이 되고, 위쪽으로 올리고 싶다면 bottom이라고 하면 되고, 만약 top과 bottom이 있을 때는 순서와 상관없이 top이 우선이 된다.

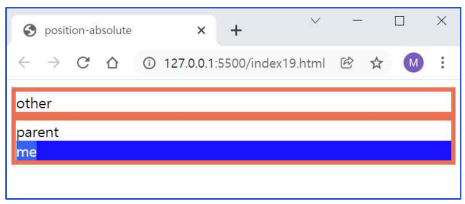
그러나 position: relative;를 삭제하면 left와 top 값이 무시되는 것을 볼 수가 있다. 이것은 기본적으로 CSS의 각각의 element들은 position의 기본값을 가지고 있다. 그 값이 static 이다.

만약 position: static; 으로 하면, left, right, top, bottom 과 같은 offset값을 무시하고 본래 위치하고자 하는 위치에 정적으로 위치하게 되는 것이다. 즉, 부모 위치에서 상대적으로 위치를 이동시키고자할 때는 position: relative; 로 지정하면 된다.
따라서 left, top과 같은 offset을 사용하기 위해서는 최소 relative 라고 하는 position type을 지정할 필요가 있다. 만약 position type을 지정하지 않았을 경우에는 그것은 static 이 된다. 즉, 위치와 관련된 설정을 하지 않은 상태를 말한다.

☞ absolute

[ index19.html ]

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>position-absolute</title>
  <style>
    #parent, #other, #grand{
      border:5px solid tomato;
    }
    #me{
      background-color: blue;
      color:white;
    }
  </style>
</head>
<body>
  <div id="other">other</div>
    <div id="parent">
      parent
    <div id="me">me</div>
  </div>
</body>
</html>
```

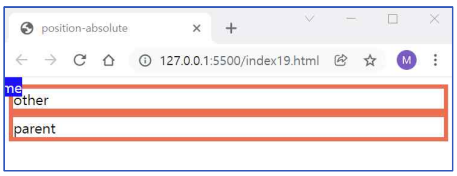



여기에서 position: relative; 와 left와 top으로 부모에 대하여 상대적으로 위치를 지정할 수 있었다.

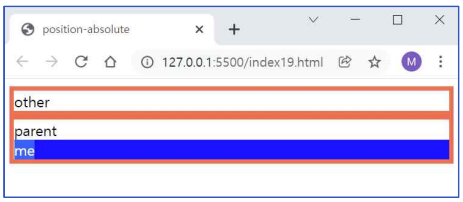
```
  <style>
    #parent, #other, #grand{
      border:5px solid tomato;
    }
    #me{
      background-color: blue;
      color:white;
      position: relative;
      left: 50px;
      top: 50px;
    }
  </style>
```

그러나 경우에 따라 parent가 아니라 Web Page의 가장 가장자리에 있는 경계의 html element를 기준으로 하여 위치를 지정할 필요가 있다. 이럴 경우에는 다음과 같이 지정해 주면 된다.

```
<style>
  #parent, #other, #grand{
    border:5px solid tomato;
  }
  #me{
    background-color: blue;
    color:white;
    position: absolute;
    left: 0;
    top: 0;
  }
</style>
```

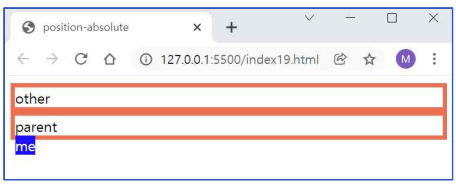



만약 position: relative 였다면, parent element 기준으로 해서 offset이 0, 0 이기 때문에 다음과 같이 된다.



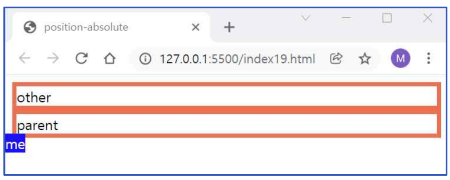

position: absolute; 로 하고, 만약 offset 값을 없애면 <me> tag가 parent element 기준으로 자기의 위치가 생기게 된다.

```
  #me{
    background-color: blue;
    color:white;
    position: absolute;
  }
```




그러나 left: 0; 으로 설정하면 html element 기준으로 된다.

```
  #me{
    background-color: blue;
    color:white;
    position: absolute;
    left: 0;
  }
```

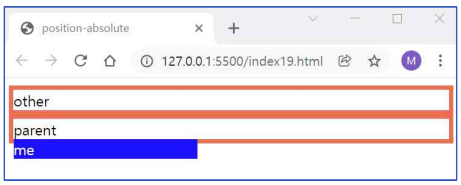



absolute로 지정을 하게 되면, aboslute의 left와 top은 0px, 0px 이 아니고 parent element를 기준으로 해서 자기가 원래의 위치해야 될 그 위치에 기본값으로 left와 top값으로 지정되기 때문이다.
이 상태에서 left: 0, top: 0을 주면 그 때 이제 html을 기준으로 해서 위치가 바뀌게 된다. 기본값은

자기가 원래 있기로 기대되는 곳이 그 기본값이다.

또한 어떤 element를 absolute로 지정하게 되면, 그 element는 더 이상 parent의 소속이 아니다. me는 id가 parent인 tag에 child로 소속되어 있는데 position을 absolute로 하게 되면 link가 끊기면서 parent와는 상당히 무관해 진다.
따라서 me라는 tag가 마치 parent의 child가 아닌 것처럼 parent 크기가 위와 같이 된다. me는 link가 끊기면서 parent의 크기를 사용할 수 없게 되어 자신의 content 크기만하게 크기가 작아지게 되는 것이다.
이제 me의 크기를 지정하고 싶다면, width: 200px; 처럼 직접 지정해 주어야 한다.

```
#me{
  background-color: blue;
  color:white;
  position: absolute;
  width: 200px;
}
```

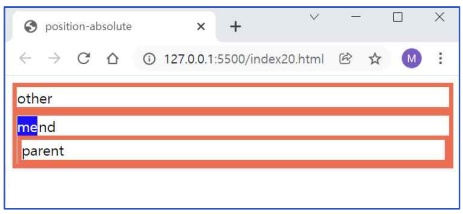



me를 absolute로 하게 되면 me가 html element를 기준으로 해서 위치가 결정되는 것이라고 하였지만, 정확한 표현은 다음과 같이 parent라는 element의 위치를 reletive로 지정하게 되면 저 me는 parent의 위치를 기준으로 해서 자신의 offset값을 설정하게 된다. 만약  tag가 바깥에 또 있다면, 이 me는 parent는 현재 어떤 position type도 지정하지 않은 상태이다. 따라서 그대로 통과된다.
grand의 겨우에는 지금 position을 relative로 주었기 때문에 grand에서 멈추어서 그 grand를 기준을 해서 position 즉, offset(left와 top)값을 할당하게 된다.

**[ index20.html ]**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>position-absolute</title>
  <style>
    #parent, #other, #grand{
        border:5px solid tomato;
    }
    #grand{
        position: relative;
    }
    #me{
        background-color: blue;
        color:white;
        position: absolute;
        left:0;
```

```
        top:0;
    }
  </style>
</head>
<body>
  <div id="other">other</div>
    <div id="grand">
      grand
      <div id="parent">
        parent
      <div id="me">me</div>
    </div>
  </div>
</body>
</html>
```




여기 grand position값을 static이 아닌 다른 값을 주면, me의 absolute로 되어있는 이 element는 static이 아닌 parent가 나타날 때까지 무시하다가 static이 아닌 parent가 나타나면 그 parent의 위치를 기준으로 해서 offset값을 지정한다.
상대적인 것은 parent를 기준으로 하여 위치가 정해지는 것이고, 절대적인 것은 parent 중에 position type이 지정된 parent를 기준으로 해서 위치가 지정이 되고 parent와의 관계성이 끊기기 때문에 자신의 크기는 자신의 content 크기만 해진다. 그리고 parent 역시도 child를 없는 것으로 취급한다.

## ❖ flex

CSS의 flex는 엘리먼트들의 크기나 위치를 쉽게 잡아주는 도구이다. 지금까지 layout과 관련된 다양한 속성들이 있었지만 그리 효과적이지 않았다. flex를 이용하면 layout을 매우 효과적으로 표현할 수 있다.

layout은 지금까지 아름답지 않은 역사를 가지고 있다. 아직까지 layout을 코드로 잘 표현하는 방법을 찾지 못하고 있었다. 먼저 table이다. table은 grid(격자) 모양으로 생겼다. 이러한 점을 착안해서 개발자와 디자이너들이 표에 해당하는 table tag를 이용해서 layout을 잡곤 하였다.
그러나 이것은 심각한 문제가 있었다. 이 표라는 것은 구조화된 정보 또는 많은 정보를 정리 정돈하기 위한 정보로써의 의미를 가지고 있는데 그것을 의미가 없는 layout을 사용하게 되니까 검색 엔진이나 스크린 리더나 여러가지 html을 해석하는 여러가지 소프트웨어의 입장에서는 어떤 table을 만났을 때 그 table이 표로 사용된 것인지 아니면 layout으로 사용된 것인지를 구분하기가 어려워졌다.
또한 table이 복잡하고 코드 양이 많기 때문에 Web Page 안에서 table이 꽉차게 되면 그것을 만드는 것은 쉽지만 나중에 그것을 변경하는 것은 어렵게 된다.
따라서 table로 layout을 만드는 것의 문제점에을 인식하고 이렇게 하지 말자는 운동도 있었고 그 대안들도 등장하게 된다.

그 다음으로 많이 사용하였던 기능들이 position 통해서 각각의 element들을 적당히 옮겨놓는 방식도 많이 사용하게 되었다.
floating으로 image를 삽화같이 표현할 때 image 옆에 글씨가 흘러가게 하는 효과를 이용하여 layout을 잡는데도 많이 사용을 하였다. 그러나 floating은 이해하기도 어렵고 layout을 만들기 위해서 고안된 것이라고 보기는 어렵다. 따라서 아주 오랜 시간 동안 layout을 코드로 표현하는 부분에서 많은 시간을 보내게 된다.

그리고 궁극의 layout 방법이 등장한 것이 flex 이다. 이 flex는 알고 나면 아주 쉽게 layout을 만드는데 사용할 수 있는데, 이 flex는 여러 가지 flex와 관련된 속성들이 다양하게 존재하기 때문에 그것을 다 알기는 쉽지 않고 또한 그것들이 결합되어 어떤 효과가 일어나는지 파악하는 것도 쉽지가 않지만 이해하고 나면 많은 장점들이 있기 때문에 반드시 학습하도록 한다.

먼저 flex를 사용하기 위해서는 tag가 두 단계가 필요하다. 다시말하면 <li> tag는 반드시 <ul>이나 <ol> tag 같은 patent tag가 필요하듯이 정렬하고자 하는 각각의 item 들은 그 parent에 해당되는 container가 필요하다.

```
<container>
    <item></item>
    <item></item>
</container>
```

다음과 같이 container 역할을 하는 tag와 그 child 역할을 하는 item tag가 있다는 것이다. 즉, 바깥쪽에 있는 container tag와 정열의 대상이 되는 item들은 이렇게 container에게 부여해야 되는 속성들이 있고, container 안쪽에 있는 item 들에게 부여해야 되는 속성이 구분되어 있다.

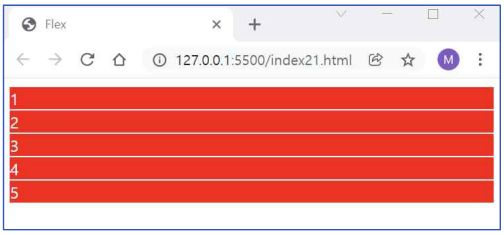
| container | item |
|---|---|
| display | order |
| flex-direction | flex-grow |
| flex-wrap | flex-shrink |
| flex-flow | flex-basis |
| justify-content | flex |
| align-items | align-self |
| align-content | |

### ☞ flex의 기본

flex를 사용하기 위해서는 container tag에 **display: flex;** 속성을 부여해야 한다. 또한 flex-direction을 이용해서 정렬방향을 바꿀 수 있다. 기본은 row 이다.

[ index21.html ]

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flex</title>
  <style>
    .container{
      background-color: blue;
      /*
      display:flex;
      flex-direction: row;
      */
    }
    .item{
      background-color: red;
      color:white;
      border:1px solid white;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
    <div class="item">5</div>
  </div>
</body>
</html>
```





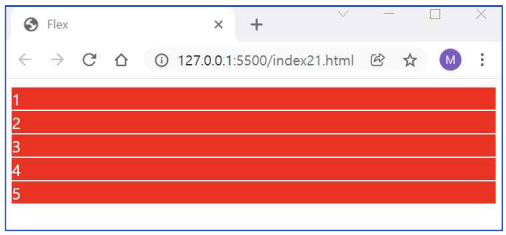
parent에 display: flex;를 사용하면 child들은 무엇인가 달라지게 된다. 이것이 flex의 기본이다. flex는 이 속성들이 여러 가지 있기 때문에 조금 까다롭다.

flex는 각각의 tag들을 정렬하는 방법인데 반대쪽에 위치하고자 하면 다음과 같이 하면 된다.

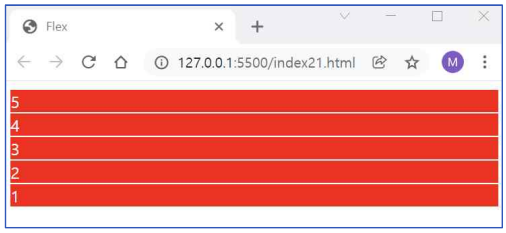
```
    .container{
      background-color: blue;
      display:flex;
      flex-direction: row-reverse;
    }
```

flex-direction을 지정하지 않으면 기본적으로 row 라고 하는 기본값을 갖게 된다.

flex-direction: row;는 행의 방향으로 정렬시키겠다는 의미이다. 그러면 이 방향을 column(수직) 방향으로 정렬시키고자 하면 flex-direction: column;과 같이 하면 된다.




반대방향은 flex-direction: column-reverse; 로 하면 된다.




그런데 이것은 이 2가지를 삭제한 것과 크게 차이자 없다.

```
  display: flex;
  flex-direction: column;
```

그러나 차이점은 container에 height 값을 주면 차이점이 나타난다.

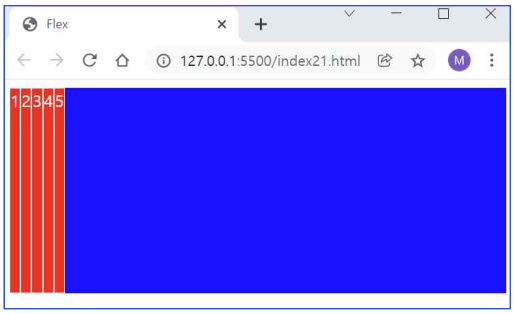
```
.container{
  background-color: blue;
  height: 200px;
  /*
  display: flex;
  flex-direction: column;
  */
}
```


또한 flex-direction: column-reverse; 로 하면 이때는 조금 달라진다.

```
.container{
  background-color: blue;
  height: 200px;
  display: flex;
  flex-direction: column-reverse;
}
```

따라서 flex를 사용하기 위해서는 바깥쪽에 있는 container에 display 값을 flex로 주고, 기본적으로 flex-direction을 하지 않으면 flex-direction row; 를 한 것과 같다. column으로 변경하면 flex는 수직 방향으로 각 item tag를 정렬하는 것을 볼 수가 있다.

flex의 또 다른 특징은 각각의 tag가 화면 전체를 차지하게 된다. row를 하였을 때 각각의 element들이 수직으로 화면 전체를 쓰는 것을 볼 수가 있다.



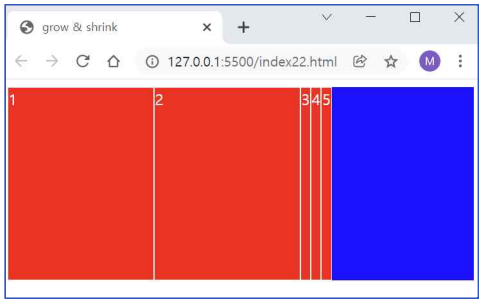

☞ grow & shrink

item은 container의 크기에 따라서 작아지기도 하고 커지기도 한다. 이 때 작아지고, 커지는 비율을 지정하는 방법이 바로 grow & shrink 이다. 여기서는 이 속성들에 대해서 학습하도록 한다.

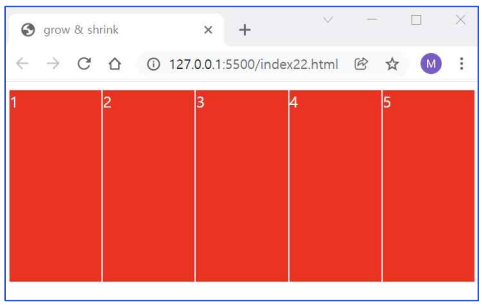
[ index22.html ]

```
<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>grow & shrink</title>
  <style>
    .container {
      background-color: blue;
      height: 200px;
      display: flex;
      flex-direction: row;
    }

    .item {
      background-color: red;
      color: white;
      border: 1px solid white;
    /* flex-grow: 1; */
    }

    .item: nth-child(1) {
      flex-basis: 150px;      // 기본
      flex-shrink: 1;
    }
```

```
    .item: nth-child(2) {
      flex-basis: 150px;      // 기본
      flex-shrink: 2;
    }
  </style>
</head>

<body>
  <div class="container">
    <div class="item">1</div>
    <div class="item">2</div>
    <div class="item">3</div>
    <div class="item">4</div>
    <div class="item">5</div>
  </div>
</body>

</html>
```




flex-basis는 flex의 방향에 해당되는 element의 크기를 지정하는 것이다.
flex-grow는 조금 이해하기가 어렵다. 위 화면 상의 blue가 표시되는 이유는 container에 지정한 color가 blue 였고, element들의 부피를 제외한 여백에 해당되는 것이다.
이렇게 item들을 배치하는 것도 가능하지만 이 item들이 container를 채우는 것을 원할 때도 있다. 어떻게 하면 이 item들이 여백을 다 채울 수 있게 할려면 flex-grow: 0;을 주어도 아무런 변화가 없다.
이유는 각각의 element는 grow 값을 주든지 안주든지 기본값이 0이기 때문이다. 그러나 이 값을 1로 주면 다음과 같이 변하게 된다. 즉, 각각의 element 가 여백을 다 채우게 된다.
각각의 element들에게 grow를 시키면 1을 주게 되면 blue 영역이 각각의 element에 할당이 된다.




그러나 2번째 element가 2배의 여백을 나누어 갖고자 할 때는 flex-grow: 2;를 지정하면 된다.

```
    .item:nth-child(2) {
      flex-grow: 2;
    }
```


이렇게 되면 cascading에 의해서 이것이 우선 순위가 높기 때문에 item 2는 grow 값이 2가 되는 것이다. 만약 .itemm{}에 flex-grow: 0;을 주게 되면 item2를 제외한 모든 것은 0으로 되고 나머지 영역은

item2가 다 차지하게 된다.

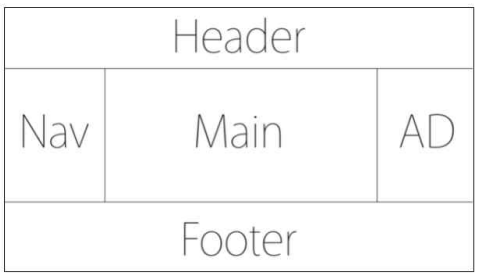
이번에는 grow를 지우고, 두번째 item에 flex-basis: 450px을 주고 이 상태에서 화면을 줄이면 더 이상 여백이 없어지는 순간이 있다. 이 순간 flex는 화면의 container 보다는 content들이 작아지는 경향성을 갖고 있기 때문에 여백이 없어지는 순간에 item들이 작아지게 된다.
이때 1, 3, 4, 5는 더 이상 content를 제외한 나머지 여백이 없기 때문에 작아지지만 2의 경우에는 basis로 500 pixel을 주었지만 여백이 존재하기 때문에 화면이 작아지게 된다.
화면이 작아졌을 때 원래 container 상태보다 작아졌을 때 그 크기는 2번이 부피에서 빼고 있는 것이다. 이때 2번이 부피를 빼지 않게 하는 방법이 flex-grow의 반대 기능인 flex-shrink: 0; 이다.
이 때는 container 크기가 item 보다 작어졌음에도 불구하고 item2는 줄어들지 않는다. 그러나 flex-shrink: 1; 로 설정하면 그 때부터는 같이 줄어든다.

다음으로 item1도 추가하여 실습해 본다.

shrink가 사용되기 위해서는 기본적으로 어떤 특정한 item이 basis값을 가지고 있을 때 그 basis값을 줄이는 것을 통해서 이러한 현상이 일어나게 된다.

### ☞ Holy Grail layout

Holy Grail은 성배라는 뜻이다. 많은 사람들이 성배를 찾기 위해서 노력했지만 찾지 못한 것처럼 많은 사람들이 아래와 같은 layout을 만들기 위해서 노력했지만 완벽한 방법을 찾지 못했다. 이것에 비유해서 이런 layout을 성배 layout이라고 부르곤 한다. flex는 아주 세련된 방법으로 이 문제를 간편하게 해결한다. 여기서는 플랙스를 이용해서 성배 layout을 만드는 법을 알아본다.

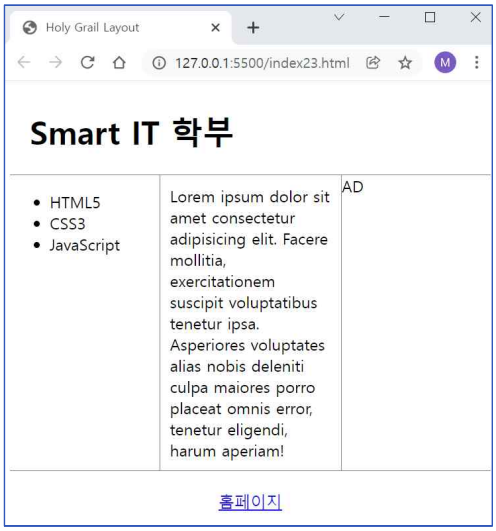



**[ index23.html ]**

```
<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Holy Grail Layout</title>
  <style>
    body {
```

```
      display: flex;
      align-items: center;
      justify-content: center;
    }
    .container {
      display: flex;
      flex-direction: column;
      width: 800px;
      border: 1px solid gray;
    }
    header {
      border-bottom: 1px solid gray;
      padding-left: 20px;
    }
    footer {
      border-top: 1px solid gray;
      padding: 20px;
      text-align: center;
    }
    .content {
      display: flex;
    }
    .content nav {
      border-right: 1px solid gray;
    }
    .content aside {
      border-left: 1px solid gray;
    }
    nav, aside {
      flex-basis: 150px;
      flex-shrink: 0;
    }
    main {
      padding: 10px;
    }
  </style>
</head>

<body>
  <div class="container">
    <header>
      <h1>Smart IT 학부</h1>
    </header>
    <section class="content">
      <nav>
        <ul>
          <li>HTML5</li>
          <li>CSS3</li>
          <li>JavaScript</li>
        </ul>
      </nav>
      <main>
        Lorem ipsum dolor sit amet consectetur adipisicing elit. Facere mollitia, exercitationem
suscipit voluptatibus tenetur ipsa. Asperiores voluptates alias nobis deleniti culpa maiores
```

```
porro placeat omnis error, tenetur eligendi, harum aperiam!
      </main>
      <aside>
        AD
      </aside>
    </section>
    <footer>
      <a href="http://www.semyung.ac.kr">홈페이지</a>
    </footer>
  </div>
</body>

</html>
```




## ☞ flex의 여러 속성들

### [ Properties for the flex container ] => container에게 주어지는 속성들

★ flex-direction

✓ row: 왼쪽에서 오른쪽으로 수평 방향(기본값)

✓ row-reverse: 오른쪽에서 왼쪽으로 수평 방향

✓ column: 위에서 아래로 수직방향

✓ column-reverse: 아래에서 위로 수직 방향

★ flex-wrap

✓ nowrap: 기본값

✓ wrap: container의 크기보다 item들의 크기의 합이 크다면, 그 item은 줄바꿈이 되서 아래줄로 내려간다.

✓ wrap-reverse: wrap 반대 방향으로 보이게 된다.

★ align-items => 수직과 관련된 정렬 (container 밑에 있는 item 전체를 정렬)

✓ flex-start: 각각의 item들이 자신의 content 크기만 하게 된다. (위에서 부터 정렬)

✓ flex-end: 아래부터 정렬

✓ center: 중앙 정렬

✓ baseline: content들이 밑줄로 정렬

✓ stretch: 기본값이며, container의 item들이 있고 그 container가 flex가 되는 순간에 그 item들은 container의 높이값과 같아진다. 이유는 item들이 stretch 상태이기 때문이다.

★ justify-content => 수평과 관련된 정렬

✓ flex-start: 기본값 (왼쪽 부터 정렬)
✓ flex-end: 오른쪽부터 정렬
✓ space-between: item 들을 일정한 간격으로 정렬
✓ center: 가운데 정렬
✓ space-around: padding을 주면서 item 들을 일정한 간격으로 정렬

★ align-content => align-items 와 비슷

✓ flex-start: 각각의 item들이 자신의 content 크기만 하게 된다. (위에서 부터 정렬)
✓ flex-end: 아래부터 정렬
✓ center: 중앙 정렬
✓ space-between: 각 행에 있는 item 들을 그룹으로 하여 그룹과 그룹사이에 일정한 간격으로 정렬
✓ space-around: padding을 주면서 각 행에 있는 item 들을 그룹으로 하여 그룹과 그룹사이에 일정한 간격으로 정렬
✓ stretch: 기본값

[ Properties for the flex items ] => item에게 주어지는 속성들

★ align-self => align-items 중에서 어느 특정한 부분만 다르게 동작함

✓ auto: 기본값
✓ flex-start:
✓ flex-end:
✓ center:
✓ baseline:
✓ stretch:

★ flex-grow

✓ element 1
✓ element 2
✓ element 3
✓ element 4
✓ element 5

★ flex-shrink

✓ element 1

✓ element 2
✓ element 3
✓ element 4
✓ element 5

★ flex => .item { flex: flex-grow [flex-shrink] [flex-basis]; } 를 축약한 것

✓ element 1
- flex-basis
- flex-grow
- flex-shrink

✓ element 2
- flex-basis
- flex-grow
- flex-shrink

★ order

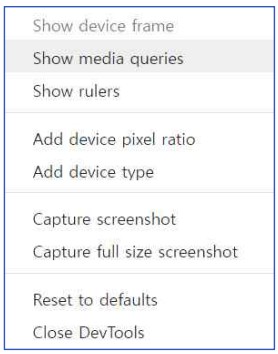
✓ element 1
✓ element 2
✓ element 3
✓ element 4
✓ element 5

## ❖ media query

media query는 화면의 종류와 크기에 따라서 디자인을 달리 줄 수 있는 CSS의 기능이다. 특히 최근의 트랜드인 반응형 디자인의 핵심 기술이라고 할 수 있다.

**[ index24.html ] => cascading 주의**

```
<!DOCTYPE html>
<html lang="ko">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Holy Grail Layout</title>
  <style>
    @media (max-width: 600px) {
      body {
        background-color: green;
      }
    }

    @media (max-width: 500px) {
      body {
        background-color: red;
      }
    }

    @media (min-width: 601px) {
      body {
        background-color: blue;
      }
    }
    li {
      color: white;
    }
  </style>
</head>

<body>
  <ul>
    <li>~500px : red</li>
    <li>501~600px : green</li>
    <li>601px~ : blue</li>
  </ul>
</body>
</html>
```



[ index25.html ]

```
<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <!-- Mobile 에서 동작하기 위하여 -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Holy Grail Layout</title>
  <style>
    .container {
      display: flex;
      flex-direction: column;
    }

    header {
      border-bottom: 1px solid gray;
      padding-left: 20px;
    }

    footer {
      border-top: 1px solid gray;
      padding: 20px;
      text-align: center;
    }

    .content {
      display: flex;

    }

    .content nav {
      border-right: 1px solid gray;
    }

    .content aside {
      border-left: 1px solid gray;
    }

    @media(max-width: 500px) {
      .content {
        flex-direction: column;
      }
      .content nav, .content aside {
        border: none;
        flex-basis: auto;
      }
      main {
        order: 0;
      }
      nav {
        order: 1;
```

```
      }
      aside {
        order: 2;
        display: none;
      }
    }

    nav, aside {
      flex-basis: 150px;
      flex-shrink: 0;
    }

    main {
      padding: 10px;
    }
  </style>
</head>

<body>
  <div class="container">
    <header>
      <h1>Smart IT 학부</h1>
    </header>
    <section class="content">
      <nav>
        <ul>
          <li>HTML5</li>
          <li>CSS3</li>
          <li>JavaScript</li>
        </ul>
      </nav>
      <main>
        Lorem ipsum dolor sit amet consectetur adipisicing elit. Fuga expedita error sint blanditiis voluptatem, accusamus sapiente vero repellat magni distinctio doloribus eligendi minus optio! Deserunt omnis numquam totam sint voluptatum!
      </main>
      <aside>
        AD
      </aside>
    </section>
    <footer>
      <a href="http://www.semyung.ac.kr">홈페이지</a>
    </footer>
  </div>
</body>

</html>
```

https://www.w3schools.com/cssref/css3_pr_mediaquery.asp

## ❖ float

Float는 편집 디자인에서 이미지를 삽화로 삽입할 때 사용하는 기법이다. 또한 layout을 잡을 때도 사용하는 기능이기 때문에 꽤 중요하다.

**[ index26.html ]**

```
<!DOCTYPE html>
<html lang="ko">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <!-- Mobile 에서 동작하기 위하여 -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Holy Grail Layout</title>
  <style>
    img {
      width: 300px;
      float: left;
      margin: 20px;
    }

    p {
      border: 1px solid blue;
    }
  </style>
  </head>

  <body>
    <img src="image01.jpg" alt="">
    <p>
      Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate minus, obcaecati quia eaque perspiciatis! Vero cum libero architecto. Odit, et. Totam expedita
    </p>
    <p style="clear:both;">
      Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate minus, obcaecati quia eaque perspiciatis! Vero cum libero architecto. Odit, et. Totam expedita Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate minus, obcaecati quia eaque perspiciatis! Vero cum libero architecto. Odit, et. Totam expedita Lorem ipsum dolor sit amet, consectetur adipisicing elit. Voluptate minus, obcaecati quia eaque perspiciatis! Vero cum libero architecto. Odit, et. Totam expedita
    </p>
  </body>

  </html>
```

## ❖ 다단(multi column)

다단(multi column)은 아래 신문처럼 화면을 분할해서 좀 더 읽기 쉽도록 만든 레이아웃을 의미한다. CSS에서는 이러한 레이아웃을 쉽게 구현할 수 있는 기능을 제공한다.

## ☑ 그래픽

### ❖ 배경(background)

CSS를 이용하면 엘리먼트의 배경을 지정할 수 있다. 여기서는 배경과 관련된 중요속성들을 학습한다.

- ☞ background-color : red
- ☞ background-image : url("bg.png")
- ☞ background-repeat : repeat, no-repeat, repeat-x, repeat-y
- ☞ background-attachment : scroll, fixed
- ☞ background-position : left top  or x% y% or x y
- ☞ background-size : 100px 100px or cover or contain

## ❖ 필터(filter)

필터는 이미지에 다양한 효과를 추가하는 방법이다.

## ❖ 혼합(blend)

블랜드는 이미지와 이미지를 혼합해서 새로운 이미지를 만들어내는 기법이다.

## ❖ 변형(transform)

transform은 엘리먼트의 크기, 위치, 모양을 변경하는 속성이다.

## ❖ SVG

SVG(Scalable Vector Graphics)는 백터(vector) 이미지를 표현하기 위한 포맷으로 w3c에서 만든 백터 이미지 표준이다. SVG 자체는 CSS가 아닙지만 CSS를 이용해서 다양한 효과를 줄 때 SVG를 활용하는 경우가 많기 때문에 여기서는 SVG에 대해서 간략하게 언급만 한다.

## ☑ 모션그래픽

CSS의 최신버전에서는 포토샵이나 플래쉬와 같은 프로그램으로 해야 했던 일을 CSS로도 할 수 있게 되었다. CSS의 코드를 통해서 그래픽 작업이나 에니메이션 작업을 하는 방법에 대해서 학습하도록 한다.

### ❖ 전환(transition)

전환은 효과가 변경되었을 때 부드럽게 처리해주는 CSS의 기능이다. 이와 관련된 것으로는 아래와 같은 속성들이 있다.

- ☞ transition-duration
- ☞ transition-property
- ☞ transition-delay
- ☞ transition-timing-function
- ☞ transition

## ☑ 유지보수

### ❖ link와 import

똑같은 CSS를 적용해야 하는 웹페이지가 1000개가 있을 때 CSS의 내용이 바뀌었다면 어떻게 해야할까? 아마 천개의 웹페이지를 모두 수정해야 할 것이다. 이것은 CSS의 수정을 소극적으로 만들고 그것은 곧 아름다움을 억압하는 암담한 장애물이 될 것이다. 여기서는 이런 문제를 근본적으로 해결할 수 있는 방법인 link 태그와 @import에 대해서 학습하도록 한다.

외부로 파일을 모듈화하는 방법은 크게 두가지이다.

1. `<link rel="stylesheet" href="style.css">`

2. `<style>@import url("style.css")</style>`

## ❖ 코드 경량화(minify)

CSS는 네트워크를 통해서 전송된다. 자연스럽게 CSS의 크기가 커지면 컨텐츠의 생산자와 소비자 모두에게 손해이다. 코드의 크기를 줄이는 것을 통해서 이런 문제를 완화해주는 도구가 minify 도구이다. 여기서는 코드를 경량화하는 방법을 학습하도록 한다.

- ☞ http://adamburgess.github.io/clean-css-online/
- ☞ clen-css
- ☞ nodejs 설치

## ❖ CSS 뛰어넘기 (preprocessor)

CSS는 뛰어난 언어이다. 하지만 CSS가 모든 면에서 좋을 수는 없다. 따라서 어떤 이들은 CSS에 편리한 기능을 더한 새로운 언어를 만들고 이 언어에 따라서 작성한 코드를 어떤 프로그램으로 실행시키면 결과적으로 CSS를 만들어주는 도구들을 개발되었다. 이런 도구를 preprocessor이라고 한다. 여기서는 이런 도구들의 개념과 간단한 사용법을 학습하도록 한다.

아래는 대표적인 preprocessor들이다.

- ☞ http://lesscss.org/ (온라인 변환기)
- ☞ http://sass-lang.com/
- ☞ http://stylus-lang.com/ (온라인 변환기)

아래는 이런 도구들에 대해서 비교한 사이트입니다.

https://csspre.com/compile/

## ☑ library

라이브러리(library)는 도서관이라는 뜻이다. 도서관이란 공용으로 책을 공유하는 공간/체계를 우리는 도서관이라고 부른다. 소프트웨어에서 라이브러리는 도서관의 이런 역할을 비유로 채택한 것이다. 즉 많은 곳에서 공통적으로 사용될 수 있는 코드를 부품화한 것이 라이브러리 이다. CSS에도 이런 라이브러리가 많다. 여기서는 중요한 라이브러리들의 사용법을 학습하도록 한다.

## ❖ fontello

Dingbat(딩벳) 폰트는 폰트 대신 어떤 문자에 해당하는 이미지로 이루어진 폰트를 의미한다. fontello는 딩벳이나 아이콘을 폰트로 제공하는 여러 서비스를 모아둔 서비스이다. 특히 SVG 파일을 업로드하면 폰트로 만들어주기도 한다.

https://fontello.com/

여기서는 폰트텔로를 이용해서 아이콘을 웹페이지에 삽입하는 방법을 알아본다.

## ❖ buttons

디자인 된 간단한 버튼 CSS 라이브러리 이다.

☞ **Buttons 홈페이지**
☞ **직접 3D 효과의 버튼을 만들기**

## ❖ Semantic-UI

Semantic ui 라이브러리는 웹페이지를 구축할 때 사용하는 라이브러리 이다. 여러 기능성을 가지고 있고, 매우 세련된 기본 디자인을 가지고 있기 때문에 웹페이지 구축에 큰 도움을 준다.