04

CSS

PNU Mini Bootcamp – Front End

CSS 란

Cascading Style Sheet

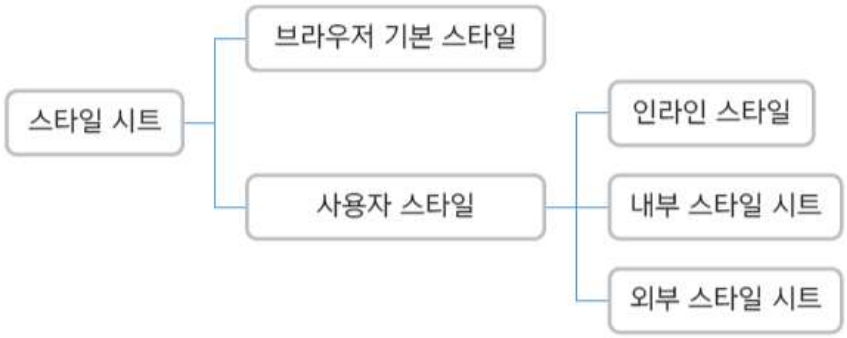
- 웹 문서에 디자인을 적용
- 색상, 사이즈, 배치 방법등

- HTML로는 웹 사이트의 내용을 나열하고 CSS로는 웹 문서의 디자인을 구성
- 스타일을 사용하면 웹 문서의 내용과 상관없이 디자인만 바꿀 수 있다
  - 내용과 디자인의 분리이다.
  - 내용을 수정하려면 html 수정
  - 디자인을 수정하고 싶다면 css 수정

- 다양한 기기에 맞게 탄력적으로 바뀌는 문서를 만들 수 있다
  - 반응형 웹을 적용하기 위함이다.
  - html 태그 내에 디자인 요소가 적용되어 있다면 html 을 여러 개 만들 수 밖에 없다.
  - html 과 css 가 분리되어 각각의 역할로 나누어 졌다면 html 하나에 다양한 css 가 적용되게 해서 반응형 웹을 만들 수 있다.

CSS 란

스타일 시트



브라우저 기본 스타일

- 브라우저에서 기본으로 적용하는 스타일
- 웹 문서에서 아무 스타일도 적용하지 않고 HTML만 사용해도 그 기능에 따라 크기에 맞게 보여줌

CSS 란

인라인 스타일

- 스타일을 적용하고 싶은 태그에 style 속성을 사용해 **style="속성: 속성 값;"** 형태로 스타일 적용
- 태그와 디자인요소가 결합되어 권장하지 않는다.

내부 스타일 시트

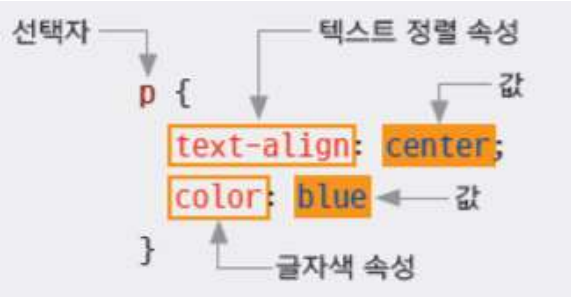
- 웹 문서 안에서 사용할 스타일을 문서 안에 정리한 것
- <style> 태그와 </style> 태그 사이에 작성
- 일반적으로 스타일 정보는 <head> 태그와 </head> 태그 안에서 정의
- body 에 추가해도 되고, 맨 마지막에 추가해도 된다.
- 브라우저가 모든 html 을 파싱한 후에 출력하기 때문에 어디에 있든 상관없다.

외부 스타일 시트

- 여러 웹 문서에서 사용할 스타일을 별도 파일로 저장해 놓고 필요할 때마다 파일에서 가져와 사용
- <link> 태그로 사용해 미리 만들어 놓은 외부 스타일 시트 파일 연결
- 여러 html 에서 재사용 가능하다.

CSS 란

선택자 { 속성1: 속성값1; 속성2: 속성값2; }



- 중괄호({ }) 사이에 스타일 규칙 나열
- 규칙이 여러 개일 경우 세미콜론(;)으로 구분

스타일 주석

- /*와 */ 사이에 주석 내용 입력
- 한 줄 또는 여러 줄을 입력 가능

CSS Selector

전체 선택자

- 페이지에 있는 모든 요소를 대상으로 스타일을 적용할 때 사용
- 웹 브라우저의 기본 스타일을 초기화할 때 사용

```
  <style>
    * {
      margin: 0;
    }
  </style>
</head>
<body>
  <img src="images/cat.jpg">
</body>
```

CSS Selector

태그 선택자

- 문서에서 특정 태그를 사용한 모든 요소에 스타일이 적용됨
- 태그명 { 스타일 규칙 }

- 각 태그의 기본 스타일을 지정할 때 주로 사용된다.
- 태그 중복 사용이 많음으로 특정 영역을 디자인하기 위해서는 사용되지 않는다.
-
- p 태그로 나오는 문자열의 폰트, 사이즈등을 전체 웹 페이지내에서 통일 시키고자 할 때..
- a 태그에 의한 링크 문자열에 밑줄을 제거하고 싶을 때
- li 태그에 의한 블릿을 제거하고 싶을 때 등

CSS Selector

class 선택자

- 요소의 특정 부분에만 스타일 적용
- 마침표(.) 다음에 클래스 이름 지정
- 문서 안에서 여러 번 반복할 스타일이라면 클래스 선택자로 정의
- .클래스명 { 스타일 규칙 }

```
.accent {
  border: 1px solid #000; /* 테두리 */
  padding: 5px;           /* 테두리와 내용 사이의 여백 */
}
.bg {
  background-color: #ddd; /* 배경색 */
}
```

CSS Selector

id 선택자

- 요소의 특정 부분에만 스타일 적용
- 문서 안에서 한번만 사용한다면 id 선택자로 정의
- #아이디명 { 스타일 규칙 }

```
#container {
  width: 500px;              /* 너비 */
  margin: 10px auto;         /* 중앙 배치 */
  padding: 10px;             /* 테두리와 내용 사이 여백 */
  border: 1px solid #000;    /* 테두리 굵기와 색깔 */
}
```

- 특정 요소에만 적용하고 싶을때 class, id 를 이용한다.
- class 는 하나의 문서내에 동일 값을 중복 선언이 가능
- id 는 하나의 문서내에서 유일한 값을 가지는 것이 일반적이다.
- id 가 중복되어도 CSS 에서는 문제가 없다. 모두 적용된다.
- JS 에서 문제가 될 수 있다.

CSS Selector

그룹 선택자

- 같은 스타일을 사용하는 선택자를 한꺼번에 정의
- 쉼표(,)로 구분해 여러 선택자를 나열
- 선택자1, 선택자2 { 스타일 규칙 }

```
h1, p {
  text-align:
}
```

스타일 우선순위

캐스케이딩의 의미

- 캐스캐이딩(Cascading) : '위에서 아래로 흐른다'는 뜻. 즉 계단식으로 적용된다는 의미로 사용.
- 선택자에 여러 스타일이 적용될 때 스타일 충돌을 막기 위해 우선순위에 따라 적용할 스타일을 결정함.

스타일 충돌을 막는(캐스캐이딩)의 원칙

- **스타일 우선순위** - 스타일 규칙의 중요도와 적용 범위에 따라 우선순위가 결정되고 그 우선순위에 따라 위에서 아래로 스타일 적용
- **스타일 상속** - 태그들의 포함 관계에 따라 부모 요소의 스타일을 자식 요소로, 위에서 아래로 전달

스타일 우선순위

- **스타일 상속**

- 부모 태그에 적용된 스타일이 자식 태그에도 그대로 적용된다는 의미이다.

- **스타일 우선순위**

- 각각의 스타일은 우선순위를 가지고 있으며 가장 높은 우선순위의 스타일이 적용된다.
- 중요도, 명시도, 코드 순서에 따라 결정된다.

스타일 우선순위

중요도

- 사용자(user) 스타일 > 작성자(author) 스타일 > 사용자 도구(user agent, 브라우저)

- 우선순위를 높이는 방법
- 스타일 속성 뒤에 !important 로 선언

```
h1 {
  background-color: red !important;
}
```

- important 사용자(user) 스타일 > important 작성자(author) 스타일 > 사용자(user) 스타일 > 작성자(author) 스타일 > 사용자 도구(user agent, 브라우저)
- 여기서 사용자가 css 를 지정하는 경우는 극히 드물기 때문에 무시해도 된다.

스타일 우선순위

명시도

- 명시도는 셀렉더가 적은 범위를 가르킬 수록 조금더 명시적으로 적용될 대상이 지정되었다고 보고 이를 더 높은 우선순위에 둔다는 의미
- 인라인>id>class>태그

코드 순서

- 중요도와 명시도가 같다면 소스 순서에 따라 결정
- 소스에서 나중에 온 스타일이 먼저 온 스타일을 덮어씀

다양한 스타일

font-family 속성

- 글꼴 지정
- 지정한 글꼴이 없을 경우에 대비해 두 번째, 세 번째 글꼴까지 지정.
- 둘 이상의 글꼴 이름을 지정할 때는 쉼표(,)로 글꼴 구분

- font-family 에서 지정한 폰트가 유저 컴퓨터에 있어야 한다. 없다면 적용되지 않는다.
- 이를 위해서 폰트를 여러 개 명시해서 순서대로 적용되게 하거나
- 웹 폰트를 이용하거나
- 폰트파일을 같이 업로드해서 사용자가 이용할 수 있게 해주어야 한다.

```
body { font-family: "맑은 고딕", 돋움, 굴림 }
```

다양한 스타일

font-size 속성

- 키워드를 사용해 글자 크기 지정가능
- xx-small < x-small < small < medium < large < x-large < xx-large
- medium 은 브라우저에서 정한 기본 크기이다.
- xx-small, x-small 등은 medium 에 대한 상대적인 크기이다.

- 직접 숫자로 크기 지정
- px – 물리적인 단위
- em, rem – 상대적인 단위
- em 은 em 이 적용된 태그 혹은 부모 태그의 글자크기에 비례해 크가가 결정
- rem 은 html 태그에 적용된 글자크기에만 영향을 받는다. 즉 다른 태그의 font-size 에는 영향을 받지 않는다.
- 일반적으로 브라우저는 font-size 가 설정되지 않았다면 폰트 기본값을 가지고 있다.
- 대부분 16px 이다.

다양한 스타일

font-style 속성

- 글자를 이탤릭체로 표시하는 속성
- oblique, italic 이외에도 여러 값이 있지만 큰 차이가 없어서 대부분 italic

font-weight 속성

- 글자 굵기를 조절하는 속성
- normal, bold, lighter, bolder 등의 글자로 지정해도 되고
- 100~900 사이의 숫자를 이용해 글자의 굵기 표현
- lighter 는 부모의 글자 굵기보다 얇게
- bolder 는 부모의 글자 굵기 보다 굵게
- bold 는 숫자 700과 동일
- normal 은 숫자 400과 동일

다양한 스타일

웹 폰트란

- 웹 문서 안에 글꼴 정보도 함께 저장했다가 사용자가 웹 문서에 접속하면 글꼴을 사용자 시스템으로 다운로드 시켜 사용하는 글꼴.
- 사용자 시스템에 없는 글꼴이더라도 웹 제작자가 의도한 대로 텍스트를 표시할 수 있다.
- @font-face 속성 사용

- body { font-family: Verdana, Arial, sans-serif; }
- 이렇게 선언하면 먼저 Verdana 를 적용, 없다면 Anal 을 적용, 없다면 sans-serif 를 적용해 준다.
- 글꼴 이름이 스페이스가 포함된 여러 단어라면 ' ' 작은 따음표로 묶어 주어야 한다.
- 왼쪽부터 순서대로 적용된다.

다양한 스타일

직접 웹 폰트 업로드해 사용하기

1. 웹 폰트 파일 준비

   - eot 파일, woff 파일

   - 기존 ttf 파일을 변환해서 사용할 수도 있음

2. 다운로드하기 전에 사용자 시스템에 있는지 확인

   - local(*글꼴이름*)

   (IE8 이하 고려해야 하면 eot 파일 선언)

3. woff 파일 선언

4. 용량이 큰 ttf 파일을 마지막에 선언

다양한 스타일

- @font-face 웹 폰트를 선언하는 것이다. 이곳에 등록된 폰트를 아래의 셀렉터에서 사용해야 한다.

- font-family 는 폰트의 이름이다.
- 실제 이름을 사용하는 것이 권장이지만 가상의 이름을 선언하고 이 가상의 이름으로 셀렉터에서 이용해도 된다.

- src
- 폰트의 위치를 지정한다.
- 사용자의 컴퓨터에 위치한 폰트라면 local()
- 원격지의 폰트라면 url() 로 명시한다.
- 여러 개 선언해도 되며 위에부터 순차적으로 판단한다.
- eot 와 woff 는 브라우저마다 지원사항이 다르다.
- 크롬 브라우저는 eot 를 지원하지 않는다.

다양한 스타일

```
@font-face {
  font-family: NanumSquareWeb;
  src: local(NanumSquareR), /* 첫번째 */
       local(NanumSquare), /* 두번째 */
       url(NanumSquareR.eot), /* 세번째 */
       url(NanumSquareR.woff), /* 네번째 */
       url(NanumSquareR.ttf); /* 다섯번째 */
}
```

- local(NanumSquareR) → local(NanumSquare) → url(NanumSquareR.eot) → url(NanumSquareR.woff)
- 위 처럼 선언하면 불필요한 다운로드가 발생하게 된다.
- 즉 크롬에서 eot 를 지원하지 않음에도 불구하고 eot 를 다운로드 하게 된다.
- 그래서 format 을 같이 선언해주어 불필요한 다운로드가 발생하지 않게 해준다.

다양한 스타일

```
@font-face {
  font-family: NanumSquareWeb;
  src: local(NanumSquareR), /* 첫번째 */
       local(NanumSquare), /* 두번째 */
       url(NanumSquareR.eot) format('embedded-opentype'),
       url(NanumSquareR.woff) format('woff'), /* 세번째 */
       url(NanumSquareR.ttf) format('truetype'); /* 네번째 */
}
```

다양한 스타일

- @font-face 내에 font-style, font-weight 같이 선언 가능
- font-style 과 font-weight 는 직접 태그에 적용해도 되지만 @font-face 에 적용하는 이유는
- 하나의 폰트를 이 font-style 과 font-weight 을 적용해서 마치 여러 폰트가 있는 것처럼 효과를 내기 위해서 이다.

```
@font-face {
  font-family: NanumSquareWeb;
  src: url(NanumSquareL.woff) format('woff');
  font-weight: 300;
}
```

다양한 스타일

구글 폰트 사용하기

1. https://fonts.google.com/ 로 접속
2. 한글 폰트 검색
3. 웹 문서의 <style> 태그 안에 붙여넣음
4. font-family 속성에서 웹 폰트 글꼴 이름 사용

다양한 스타일

color 속성

- 글자 색 지정
- 16진수 값이나 rgb 값, 색상 이름 중에서 사용

text-align 속성

- 텍스트 정렬 방법 지정

text-decoration 속성

- 텍스트에 밑줄을 긋거나 가로지르는 줄 표시
- 텍스트 링크의 밑줄을 없앨 때도 사용

속성 값	설명
none	밑줄을 표시하지 않습니다.*
underline	밑줄을 표시합니다.
overline	영역 위로 선을 그립니다.
line-through	영역을 가로지르는 선(취소 선)을 그립니다.

다양한 스타일

list-style-type 속성

- 순서 없는 목록의 불릿이나 순서 목록의 숫자를 바꾸는 속성
- 블릿을 없애기 위해서도 많이 사용된다

종류	설명	예시
disc	채운 원 모양입니다.	●
circle	빈 원 모양입니다.	○
square	채운 사각형 모양입니다.	■
decimal	1부터 시작하는 10진수입니다.	1, 2, 3, …
decimal-leading-zero	앞에 0이 붙는 10진수입니다.	01, 02, …
lower-roman	로마 숫자 소문자입니다.	i, ii, iii, …
upper-roman	로마 숫자 대문자입니다.	I, II, III, …
lower-alpha 또는 lower-latin	알파벳 소문자입니다.	a, b, c, …
upper-alpha 또는 upper-latin	알파벳 대문자입니다.	A, B, C, …
none	불릿이나 숫자를 없앱니다.	

다양한 스타일

caption-side

- 캡션(설명글)은 기본으로 표 위쪽에 표시됨.
- 이 속성을 이용해 아래쪽에 표시 가능

border

- table 만 border 지정하면 바깥만 지정
- 셀은 td, th 에 따로 지정해야 셀의 border 가 지정된다.

border-collapse

- 표 테두리와 셀 테두리를 합칠 것인지 설정
- border-collapse 를 지정하지 않으면 두겹으로 나온다.

상품 구성

용도	중량	개수	가격
선물용	3kg	11~16과	35,000원
	5kg	18~26과	52,000원
가정용	3kg	11~16과	30,000원
	5kg	18~26과	47,000원

선물용과 가정용 상품 구성

레이아웃

블록 레벨 요소

- 요소를 삽입했을 때 혼자 한 줄을 차지하는 요소
- 요소의 너비가 100%
- <address>, <article>, <aside>, <blockgquote>, <canvas>, <dd>, , <dl>, <hr>, <header>, <form>,<h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <table>, <pre>, <ul>, <p>, <ol>, <video>

인라인 레벨 요소

- 줄을 차지하지 않는 요소
- 화면에 표시되는 콘텐츠만큼만 영역을 차지하고 나머지 공간에는 다른 요소가 올 수 있음
- <a>, <i>, , <abbr>, <img>, , <b>, <input>, <sub>,
, <code>, , <small>, <tt>, <map>, <textarea>, <label>, <sup>, <q>, <button>, <cite>

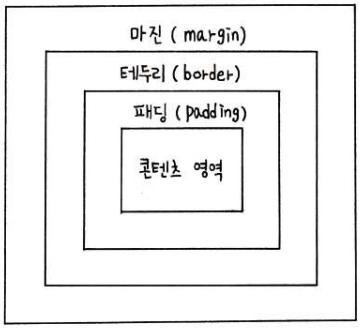
레이아웃

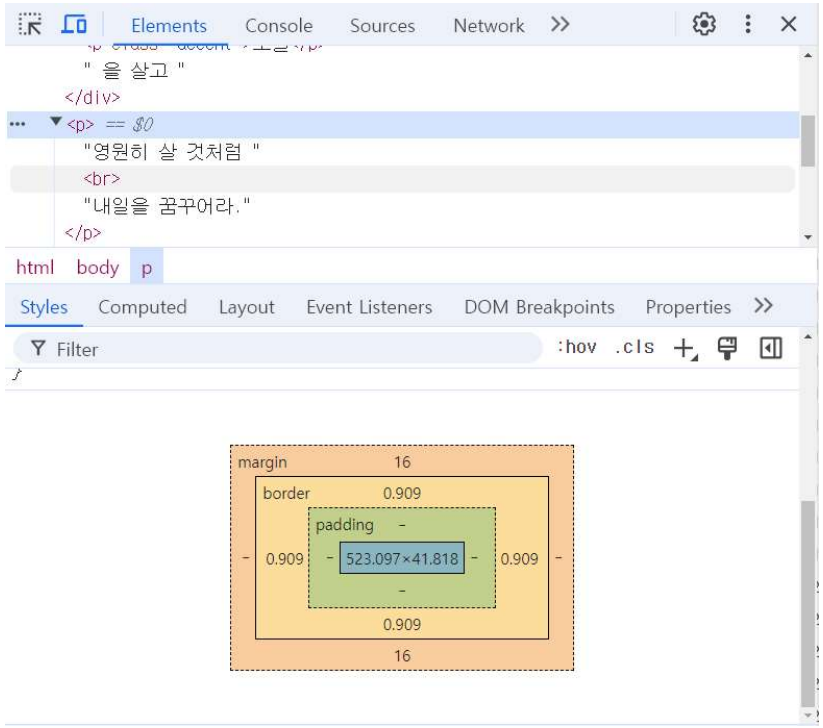
```
<body>
    <h1>Block...</h1>
    <div>
      내일 죽을 것처럼
      <p class="accent">오늘</p>
      을 살고
    </div>
    <p>영원히 살 것처럼 <br />내일을 꿈꾸어라.</p>
    <br/>
    <br/>

    <h1>Inline</h1>
    <div>모든 질적 변화는
        <span class="accent">양적</span>
        변화가 선행되어야 한다.
    </div>
    <p>과거를 평가하고 <br />오늘을 분석해서 <br/> 내일을 설계하라</p>
</body>
```

레이아웃

박스 모델

- 실제 콘텐츠 영역, 패딩(padding), 박스의 테두리(border), 그리고 마진(margin) 등의 요소로 구성됨.

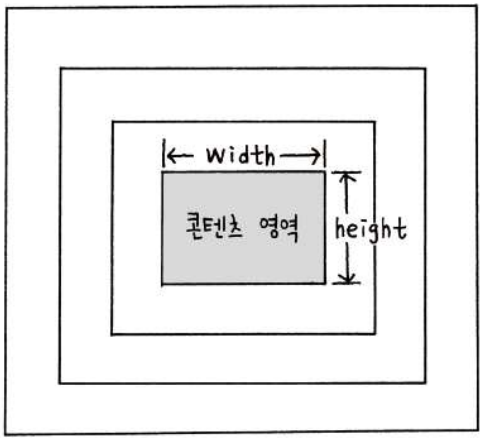




레이아웃

width, height 속성

- 실제 콘텐츠 영역의 크기 지정
- 인라인요소에는 적용되지 않는다. 블록요소에만 지정할 수 있다.
- 숫자(단위 px, em 등), 백분율(부모의 영역을 몇 %), auto 는 자동으로 계산
- 박스의 사이즈는 width + padding + border + margin 이 된다.



레이아웃

box-sizing 속성

- 실제 박스 모델의 너비를 계산할 때 어디까지 포함할지 결정하는 속성

종류	설명
border-box	테두리까지 포함해서 너빗값을 지정합니다.
content-box	콘텐츠 영역만 너빗값을 지정합니다. 기본값입니다.

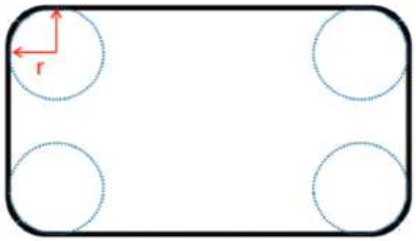
```
.a1 {
        box-sizing: border-box;
        width: 200px;
        height: 100px;
        padding: 20px;
        border: 10px sold #ccc;
}
```

- 이렇게 하면 content 의 사이즈는 140 이 된다. (200 – 40 – 20 = 140)

레이아웃

border-radius 속성

- 박스 모델의 테두리를 둥글게 처리
- 박스 모델의 꼭짓점 부분에 원(반지름 r)이 있다고 가정해서 둥글게 처리



- % 로 지정해도 된다.
- 현재 요소의 사이즈에 대한 % 이다.
- 50% 가 되면 원형이 된다.

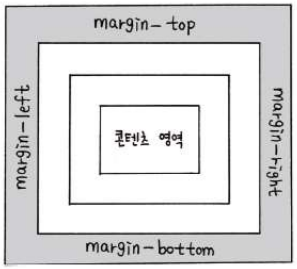
- border와 radius 사이에 위치를 나타내는 예약어를 사용하면 꼭짓점마다 다르게 처리 가능

```
#round1 {
  border: 2px solid blue;
  border-top-left-radius: 20px;
  border-top-right-radius: 20px;
}
```

레이아웃

margin 속성

- 현재 요소 주변의 여백
- 마진을 이용하면 요소와 요소 간의 간격 조절 가능



종류	설명	예시
<크기>	너빗값이나 높잇값을 px이나 em 같은 단위와 함께 수치로 지정합니다.	`margin: 50px;`
<백분율>	박스 모델을 포함한 부모 요소를 기준으로 너빗값이나 높잇값을 퍼센트(%)로 지정합니다.	`margin: 0.1%;`
auto	display 속성에서 지정한 값에 맞게 적절한 값을 자동으로 지정합니다.	

- margin-left와 margin-right의 속성값을 auto로 지정하면 가운데 정렬

레이아웃

display 속성

- display 는 값이 많다.
- 주로 블록->인라인 혹은 인라인->블록 으로 지정하기 위해서 사용된다.

종류	설명
block	인라인 레벨 요소를 블록 레벨 요소로 만듭니다.
inline	블록 레벨 요소를 인라인 레벨 요소로 만듭니다.
inline-block	인라인 레벨 요소와 블록 레벨 요소의 속성을 모두 가지고 있으며 마진과 패딩을 지정할 수 있습니다.
none	해당 요소를 화면에 표시하지 않습니다.

레이아웃

block

- block은 한 영역을 차지 하는 박스형태
- 기본적으로 block은 width값이 100%
- 라인이 새로 추가
- block은 height와 width 값을 지정 할 수 있다.
- block은 margin과 padding을 지정 할 수 있다.

inline

- inline은 주로 텍스트를 주입 할 때 사용 되는 형태
- width값이 100%가 아닌 컨텐츠 영역 만큼 자동으로 설정
- 라인이 새로 추가 되지 않는다.
- 높이 또한 폰트의 크기만큼 차지
- width와 height를 명시 할 수 없다.

레이아웃

inline-block

- inline-block 은 말그대로 inline의 특징과 block의 특징을 모두 가진 요소
- 줄바꿈이 이루어지지 않는다.
- block처럼 width와 height를 지정 할 수 있다.
- 만약 width와 height를 지정하지 않을 경우, inline과 같이 컨텐츠만큼 영역이 잡힌다.

- 주로 inline-block 은 inline 요소에 block 의 width, height 를 지정하기 위해서 사용된다.

레이아웃

float 속성

- 요소를 왼쪽이나 오른쪽에 떠 있게 만듦
- 떠 있다는 것은 다른 요소 영역 위에 배치된다는 것을 의미한다.

기본형 `float: left | right | none`

속성 값	설명
left	해당 요소를 문서의 왼쪽으로 배치합니다.
right	해당 요소를 문서의 오른쪽으로 배치합니다.
none	좌우 어느 쪽으로도 배치하지 않습니다.

레이아웃

clear 속성

- float 속성을 무효화 시키는 속성

종류	설명
left	float: left를 해제합니다.
right	float: right를 해제합니다.
both	float: left와 float: right를 해제합니다.

다양한 선택자

하위 선택자(descendant selector)

- 부모 요소에 포함된 모든 하위 요소에 스타일이 적용된다
- 자식 요소뿐만 아니라 손자 요소, 손자의 손자 요소 등 모든 하위 요소까지 적용

- A, B { } 이면 A 혹은 B 이다. 즉 A 와 B 에 모두이다.
- A B 이면 A 하위의 B 이다.

다양한 선택자

자식 선택자(child selector)

- 자식 요소에 스타일을 적용하는 선택자
- 두 요소 사이에 '>(부등호)'를 표시해 부모 요소와 자식 요소를 구분
- a1>a2>a3 처럼 지정도 가능하다.

다양한 선택자

인접 형제 선택자(adjacent selector)

- 같은 부모를 가진 형제 요소 중 첫 번째 동생 요소에만 스타일 적용
- 요소1과 요소2 사이에 '+' 기호 사용
- 요소1과 요소2는 같은 레벨이면서 요소1 이후 첫번째 요소2에 적용

- h1 + p, h1 과 p 가 형제 레벨이어야 한다.
- h1 다음에 가장 처음 나오는 p

다양한 선택자

[속성] 선택자

- 지정한 속성을 가진 요소를 찾아 스타일 적용
- [속성] - 속성 명을 자지고 있는 요소
- [속성=값] - 속성에 지정한 값이 할당된 요소
- [속성~=값] - 속성 값에 지정된 값이 포함된 요소(값이 여러개 지정되는 요소에 이 값이 포함되었다면), 단어 기준

div[class~="apple"] { background-color: red; }

- layer 선택함
- layer 선택함
- layer 선택하지 않음

다양한 선택자

[속성|=값] - 하이픈이 포함된 단어

div[class|="layer"] { background-color: red; }

- layer 선택함
- layer 선택함
- layer 선택함
- layer 선택하지 않음
- layer 선택하지 않음

[속성^=값] - 시작 값

div[class^="minions"] { background-color: red; }

- layer 선택함
- layer 선택함
- layer 선택함
- layer 선택함
- layer 선택하지 않음
- layer 선택하지 않음

다양한 선택자

속성$=값] - 끝 값
div[class$="end"] { background-color: red; }
end라는 class로 끝나는 요소를 선택한다.
.pdf등을 값(value)으로 지정해 특정 파일만 선택하는 것도 가능하다.

- layer 선택함
- layer 선택함
- layer 선택함
- layer 선택하지 않음

[속성*=값] - 위치 상관 없이
div[class*="wow"] { background-color: red; }

- layer 선택함
- layer 선택함
- layer 선택함
- layer 선택함

다양한 선택자

사용자 동작에 반응하는 가상 클래스 선택자

- 가상이라는 의미는 실제 태그로 존재한다는 것이 아니라
- 유저의 동작에 적용하기 위한 선택자이다.

- a 는 a 태그를 지칭하는 선택자이지만
- a:hover 는 a 태그에 마우스 커서가 올라오면을 지칭하는 선택자이다.

- 결국 어떤 조건을 명시하기 위한 선택자이다.

- 주로 유저 이벤트가 가능한 요소에 적용되며 대표적 사례가 링크에 적용

다양한 선택자

- **:link** - 방문하지 않은 링크에 스타일 적용
- **:visited** - 방문한 링크에 스타일 적용
- **:active** - 웹 요소를 활성화했을 때의 스타일 적용
- **:hover** - 웹 요소에 마우스 커서를 올려놓을 때의 스타일 적용
- **:focus** - 웹 요소에 초점이 맞추어졌을 때의 스타일 적용

다양한 선택자

요소 상태에 따른 가상 클래스 선택자

- **:target** – 앵커로 연결된 부분에 스타일 적용

- div id="intro"> 라고 되어 있는 부분을 #intro:target 이라고 선택자를 지정
- <a href="#intro"> 이 클릭되어 요소가 선택되었을 때 적용된다.

- :enabled, :disabled – 요소의 사용 여부에 따라 스타일 적용
- :checked – 라디오 버튼이나 체크 박스에 체크했을 때 스타일 적용

다양한 선택자

문서 구조에 따른 가상 클래스 선택자

- 웹 문서의 구조를 기준으로 특정 위치에 있는 요소를 찾아 스타일 적용

종류	설명
:only-child	부모 안에 자식 요소가 하나뿐일 때 자식 요소를 선택합니다.
A:only-type-of	부모 안에 A 요소가 하나뿐일 때 선택합니다.
:first-child	부모 안에 있는 모든 요소 중에서 첫 번째 자식 요소를 선택합니다
:last-child	부모 안에 있는 모든 요소 중에서 마지막 자식 요소를 선택합니다
A:first-of-type	부모 안에 있는 A 요소 중에서 첫 번째 요소를 선택합니다.
A:last-of-type	부모 안에 있는 A 요소 중에서 마지막 요소를 선택합니다.
:nth-child(*n*)	부모 안에 있는 모든 요소 중에서 n번째 자식 요소를 선택합니다.
:nth-last-child(*n*)	부모 안에 있는 모든 요소 중에서 끝에서 n번째 자식 요소를 선택합니다.
A:nth-of-type(*n*)	부모 안에 있는 A 요소 중에서 n번째 요소를 선택합니다.
A:nth-last-of-type(*n*)	부모 안에 있는 A 요소 중에서 끝에서 n번째 요소를 선택합니다.

다양한 선택자

가상 요소

- 화면 꾸미기용 요소를 웹 문서에 포함시키지 않기 위해 가상 요소 사용

- ::first-line : 특정 요소의 첫번째 줄에 스타일 적용
- ::first-letter : 특정 요소의 첫번째 글자에 스타일 적용
- ::before : 특정 요소의 앞에 지정한 콘텐츠 추가
- ::after : 특정 요소의 뒤에 지정한 콘텐츠 추가

```
li.new::after {
  content: "NEW!!";
  font-size: x-small;
  padding: 2px 4px;
  margin: 0 10px;
  border-radius: 2px;
  background: #f00;
  color: #fff;
}
```

반응형 웹 디자인

반응형 웹 디자인

- 웹 사이트의 내용을 그대로 유지하면서 다양한 화면 크기에 맞게 웹 사이트를 표시하는 방법
- 다양한 화면 크기의 모바일 기기들이 계속 쏟아져 나오는데 그 때마다 그 크기에 맞춘 사이트를 별도로 제작하는 것은 비효율적
- 화면 크기에 '반응'해 화면 요소들을 자동으로 바꾸어 사이트를 구현하는 것이 바로 반응형 웹 디자인

반응형 웹 디자인

뷰포트(viewport)

- 웹킷 기반의 브라우저(크롬, 사파리등)은 viewport 를 지정하지 않으면 width 값이 기본으로 980px 이 지정된다.
- 실제 사이즈가 320인 디바이스에서는 모든 컨텐츠가 너무 작게 나오게 된다.
- 실제 사이즈가 1920인 디바이스에서는 모든 컨텐츠가 너무 크게 나오게 된다.

- 웹킷에서 판단하는 브라우저의 논리적인 사이즈를 지정하는 것이다.

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

종류	설명	사용 가능한 값	기본값
width	뷰포트 너비	device-width 또는 크기	브라우저 기본값
height	뷰포트 높이	device-height 또는 크기	브라우저 기본값
user-scalable	확대·축소 가능 여부	yes 또는 no (yes는 1로, device-width와 device-height값은 10으로 간주)	yes
initial-scale	초기 확대·축소 값	1~10	1

반응형 웹 디자인

뷰포트 단위

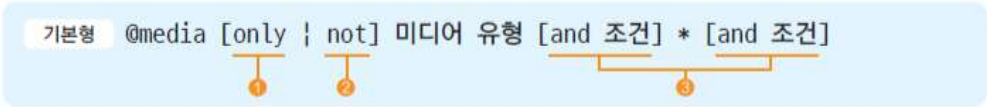
- 반응형 웹이 나오기 시작하면서 만들어진 단위이다.
- viewport 에서 지정한 사이즈에 상대적인 사이즈를 지정할때 이용된다.
- vm, vh 등을 이용해 디바이스에 따라 사이즈를 다르게 적용할 수 있게 된다.

- vw(viewport width): 1vw는 뷰포트 너비의 1%와 같습니다.
- vh(viewport height): 1vh는 뷰포트 높이의 1%와 같습니다.
- vmin(viewport minimum): 뷰포트의 너비와 높이 중에서 작은 값의 1%와 같습니다.
- vmax(viewport maximum): 뷰포트의 너비와 높이 중에서 큰 값의 1%와 같습니다.

반응형 웹 디자인

미디어 쿼리(media queries)

- 반응형 웹을 지원하기 위한 CSS 기법
- 원래는 js 에서 처리해야 하지만 디자인 적인 측면임으로 CSS 에서 동적으로 속성을 지정할 수 있는 기법이 추가
- 접속하는 장치(미디어)에 따라 특정한 CSS 스타일을 사용하는 방법



- @media 속성을 사용해 특정 미디어에서 어떤 CSS를 적용할 것인지 지정함
- only, not, and 는 생략이 가능
- only 는 미디어 쿼리를 지정하지 않는 브라우저에서 이를 무시하게 하기 위함인데 현재의 웹브라우저는 모두 미디어쿼리를 지원
- 여러 조건을 같이 나열하기 위해서 조건과 조건사이에 and 가 주로 사용

```
@media screen and (min-width: 768px) and (max-width: 1439px) {
 (... 생략 ...)
}
```

반응형 웹 디자인

미디어 유형의 종류

- print - 인쇄할 때만 적용할 css
- screen – 화면에 표시할 css
- 일반적으로 screen 이다.

종류	설명
all	모든 미디어 유형에서 사용할 CSS를 정의합니다.
print	인쇄 장치에서 사용할 CSS를 정의합니다.
screen	컴퓨터 스크린에서 사용할 CSS를 정의합니다. 스마트폰의 스크린도 포함됩니다.
tv	음성과 영상이 동시에 출력되는 TV에서 사용할 CSS를 정의합니다.
aural	음성 합성 장치(주로 화면을 읽어 소리로 출력해 주는 장치)에서 사용할 CSS를 정의합니다.
braille	점자 표시 장치에서 사용할 CSS를 정의합니다.
handheld	패드(pad)처럼 손에 들고 다니는 장치를 위한 CSS를 정의합니다.
projection	프로젝터를 위한 CSS를 정의합니다.
tty	디스플레이 기능이 제한된 장치에 맞는 CSS를 정의합니다. 이런 장치에서는 픽셀(px) 단위를 사용할 수 없습니다.
embossed	점자 프린터에서 사용할 CSS를 정의합니다.

반응형 웹 디자인

웹 문서의 가로 너비와 세로 높이(뷰포트)

- 실제 웹 문서 내용이 나타나는 영역의 너비와 높이를 조건으로 사용

종류	설명
width, height	웹 페이지의 가로 너비, 세로 높이
min-width, min-height	웹 페이지의 최소 너비, 최소 높이
max-width, max-height	웹 페이지의 최대 너비, 최대 높이

- 화면 너비가 1440px 이상일 때

```
@media screen and (min-width: 1440px) {  /* 너비가 최소 1440px인 화면용 스타일 */
  (... 생략 ...)
}
```

반응형 웹 디자인

단말기의 가로 너비와 세로 높이

- 브라우저 크기와 단말기 크기는 다르다.
- 그럼으로 일반적으로 width, height 을 주로 이용한다.
- 데스크탑에서 브라우저의 사이즈가 최대화가 되지 않았다면 브라우저 크기와 단말기 크기는 다르게 된다.

종류	설명
device-width, device-height	단말기의 가로 너비, 세로 높이
min-device-width, min-device-height	단말기의 최소 너비, 최소 높이
max-device-width, max-device-height	단말기의 최대 너비, 최대 높이

- https://yesviz.com/devices.php
- 다양한 디바이스가 출시될 때마다 그 디바이스의 사이즈가 업데이트 되는 사이트이다.
- 각 디바이스 클릭하면 그 디바이스를 위한 미디어쿼리 구문도 제시해 준다.

반응형 웹 디자인

화면 회전

- 스마트폰이나 태블릿에서 기기를 가로나 세로로 돌려보는지 확인

종류	설명
orientation: portrait	단말기의 세로 모드
orientation: landscape	단말기의 가로 모드

반응형 웹 디자인

미디어 쿼리 중단점

- **중단점(breakpoint)** : 서로 다른 CSS를 적용할 화면 크기
- 모든 기기를 반영할 수 없기 때문에 스마트폰과 태블릿, 데스크톱 정도로 구분
- 모바일 퍼스트(mobile first) : 모바일 기기 레이아웃을 기본으로 작성 → 태블릿 & PC 레이아웃 작성

- 미디어 쿼리 중단점은 개발자나 작업 조건에 따라 달라질 수 있다.

스마트폰: 모바일 페이지는 미디어 쿼리를 사용하지 않고 기본 CSS로 작성. 만일 스마트폰의 방향까지 고려해서 제작한다면 min-width를 각각 portrait 320px, landscape 480px로 지정.

• **태블릿**: 세로 크기가 768px 이상이면 태블릿으로 지정. 가로 크기는 데스크톱과 똑같이 1024px 이상으로 지정.

• **데스크톱**: 화면 크기가 1024px 이상이면 데스크톱으로 설정.

반응형 웹 디자인

외부 CSS 파일 연결

- 방법1) <link> 태그 사용하기

```
<link rel="stylesheet" media="print" href="css/print.css">
```

- 방법2) @import 구문 사용하기

```
@import url("css/tablet.css") only screen and (min-width: 321px) and (max-width:
768px);
```

반응형 웹 디자인

웹 문서에서 직접 정의하기

- 방법1) <style> 태그 안에서 media 속성 사용하기

```
<style media="screen and (max-width: 320px)">
  body {
    background-color: orange;
  }
</style>
```

- 방법2) @media문 사용하기

```
<style>
  @media screen and (max-width: 320px) {
    body {
      background-color: orange;
    }
  }
</style>
```

반응형 웹 디자인

그리드 레이아웃이란

- 반응형 웹을 적용해서 디자인하는 것은 쉽지 않다.
- 그래서 나온 레이아웃 개념이 그리드 레이아웃이다.

- 그리드 레이아웃은 출판물의 레이아웃 개념이다.
- 화면을 여러 컬럼으로 나눈후에
- 컬럼별로 레이아웃을 정의하는 것이다.

- 화면을 12개의 컬럼으로 나누고
- 제목을 12개 컬럼을 차지하고
- 왼쪽 그림을 6개의 컬럼, 왼쪽 글을 6개의 컬럼을 차지하게 하자는 것이다.

반응형 웹 디자인

그리드 레이아웃을 만드는 방법

- 두가지 방법이 있다.
- Css 그리드 레이아웃을 이용하는 방법과 플랙스 박스 레이아웃을 이용하는 방법

플렉스 박스 레이아웃(플렉서블 박스 레이아웃)

- 수평 방향이나 수직 방향 중 하나를 주축으로 정하고 박스를 배치
- 여유 공간이 생길 경우 너비나 높이를 적절하게 늘리거나 줄일 수 있음

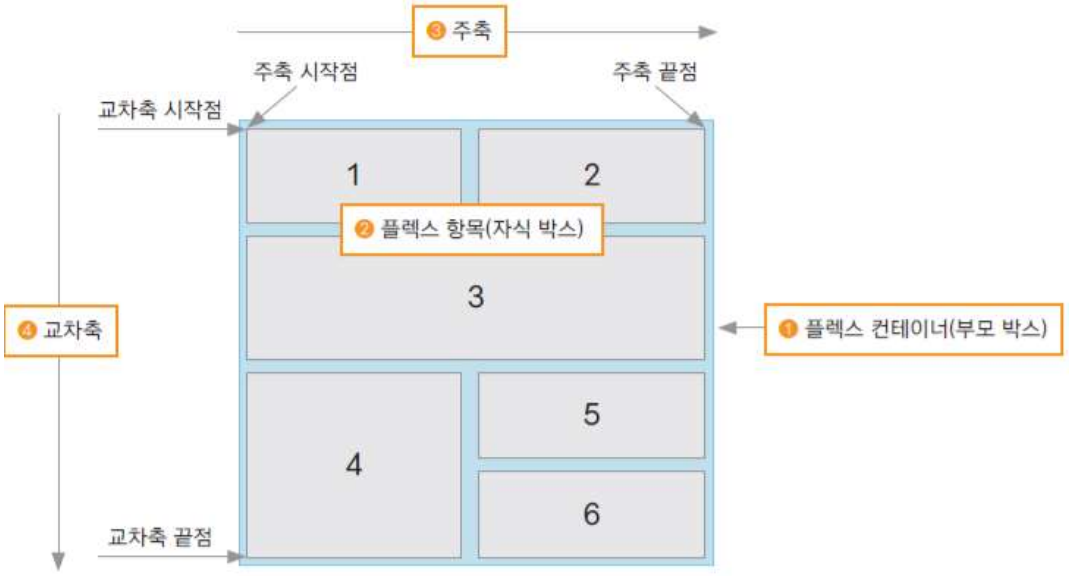
CSS 그리드 레이아웃

- 수평 방향이나 수직 방향 어디로든 배치 가능
- 마치 레고 블록을 끼워 맞추듯 요소를 배치할 수 있음

반응형 웹 디자인

플렉스 박스 레이아웃(flex box layout)

- 먼저 플렉스 박스 레이아웃을 적용할 전체를 플렉스 컨테이너로 지정
- 플레스 컨테이너에 배치되는 각각의 요소를 플렉스 항목
-
- 기준축(main axis) 와 교차축(cross axis) 를 지정하여 배치
- 대부분 가로 방향을 기준축으로 한다.



반응형 웹 디자인

플렉서블 박스 레이아웃 기본 속성

- display 속성
- 배치 요소들을 감싸는 부모 요소를 플렉스 컨테이너로 지정

종류	설명
flex	컨테이너 안의 플렉스 항목을 블록 레벨 요소로 배치합니다.
inline-flex	컨테이너 안의 플렉스 항목을 인라인 레벨 요소로 배치합니다.

- flex-direction 속성
- 플렉스 항목의 배치를 위해 주축과 방향 지정.
- 방향을 지정하지 않으면 주측의 기본 값은 row 이다.

종류	설명
row	주축을 가로로 지정하고 왼쪽에서 오른쪽으로 배치합니다. 기본값입니다.
row-reverse	주축을 가로로 지정하고 반대로 오른쪽에서 왼쪽으로 배치합니다.
column	주축을 세로로 지정하고 위쪽에서 아래쪽으로 배치합니다.
column-reverse	주축을 세로로 지정하고 아래쪽에서 위쪽으로 배치합니다.

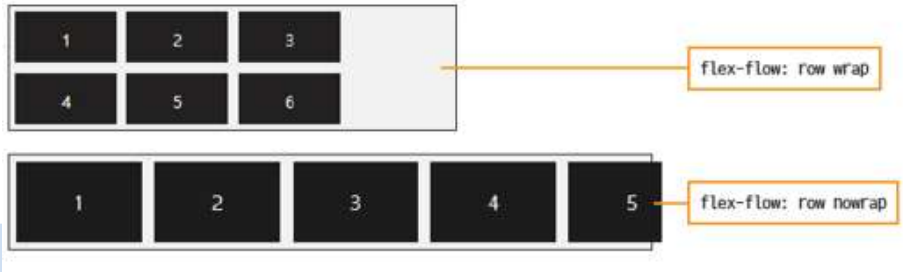
반응형 웹 디자인

플렉서블 박스 레이아웃 기본 속성

- flex-wrap 속성
- 방향이 지정되면 줄바꿈을 할 것인지, 아니면 한줄에 나열할 것인지를 결정

종류	설명
nowrap	플렉스 항목을 한 줄에 표시합니다. 기본값입니다.
wrap	플렉스 항목을 여러 줄에 표시합니다.
wrap-reverse	플렉스 항목을 여러 줄에 표시하되, 시작점과 끝점이 바뀝니다.

- flex-flow 속성
- flex-flow 는 방향과 줄바꿈을 같이 설정하기 위한 속성이다.
- nowrap 이면 컨테이너 범위를 벗어나더라도 줄바꿈이 되지 않는다.

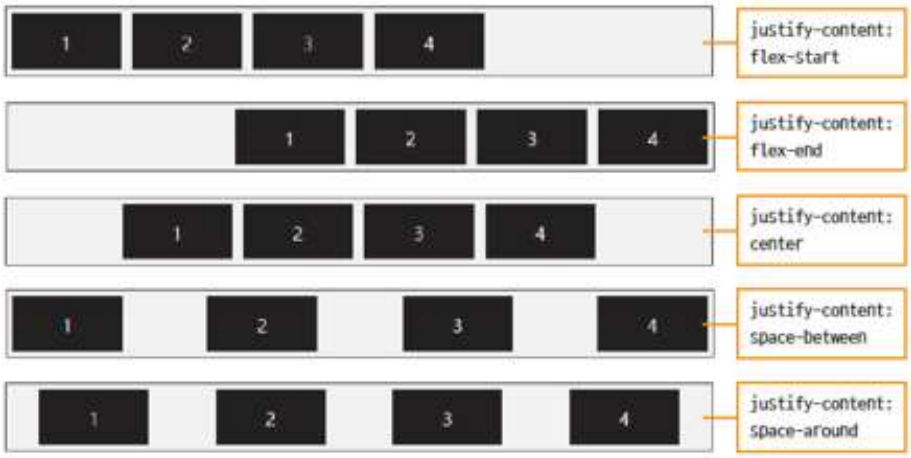


반응형 웹 디자인

플렉스 항목 배치를 위한 속성

- justify-content 속성
- 플렉스 항목을 주축 방향으로 배치할 때의 배치 기준 (align)
- 주측에 요소를 배치하다가 공간이 남게 되면 어떻게 배치할 것인지를 결정한다.

종류	설명
flex-start	주축의 시작점에 맞춰 배치합니다.
flex-end	주축의 끝점에 맞춰 배치합니다.
center	주축의 중앙에 맞춰 배치합니다.
space-between	첫 번째 항목과 끝 항목을 주축의 시작점과 끝점에 배치한 후 나머지 항목은 그 사이에 같은 간격으로 배치합니다.
space-around	모든 항목을 주축에 같은 간격으로 배치합니다.



반응형 웹 디자인

플렉스 항목 배치를 위한 속성

- align-items 속성, align-self 속성
- 교차축의 align

- 한줄로 배치되는 경우의 align-items 이다.
- 여러줄로 배치되는 경우에는 align-content 를 사용한다.

- align-items 는 컨테이너에 지정하여 컨테이너의 요소들이 교차축으로 어떻게 배치되는 지를 결정하는 것
- align-self 는 컨테이너에 추가되는 요소에 지정하는 속성으로 해당 요소가 교차축으로 어떻게 배치되어야 하는지를 결정한다.

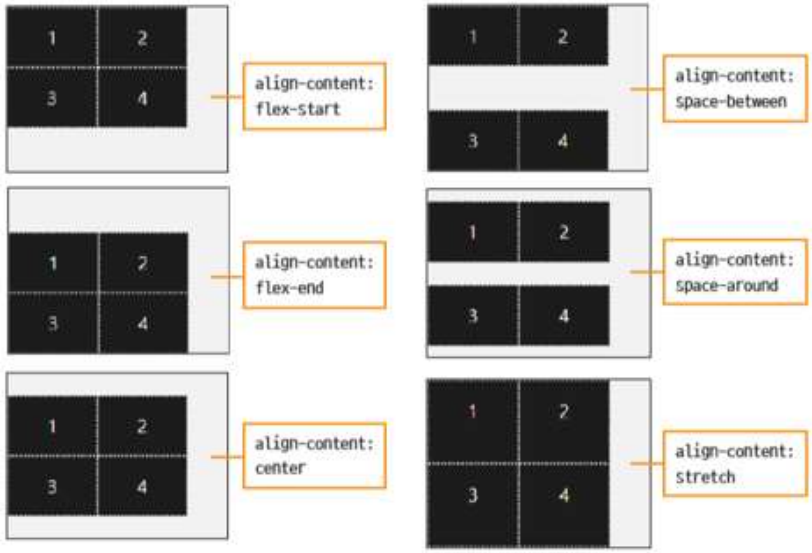
종류	설명
flex-start	교차축의 시작점에 맞춰 배치합니다.
flex-end	교차축의 끝점에 맞춰 배치합니다.
center	교차축의 중앙에 배치합니다.
baseline	교차축의 문자 기준선에 맞춰 배치합니다.
stretch	플렉스 항목을 늘려 교차축에 가득 차게 배치합니다.

반응형 웹 디자인

플렉스 항목 배치를 위한 속성

- align-content 속성
- 플렉스 항목이 여러 줄로 표시될 때 교차 축 기준의 배치 방법 지정

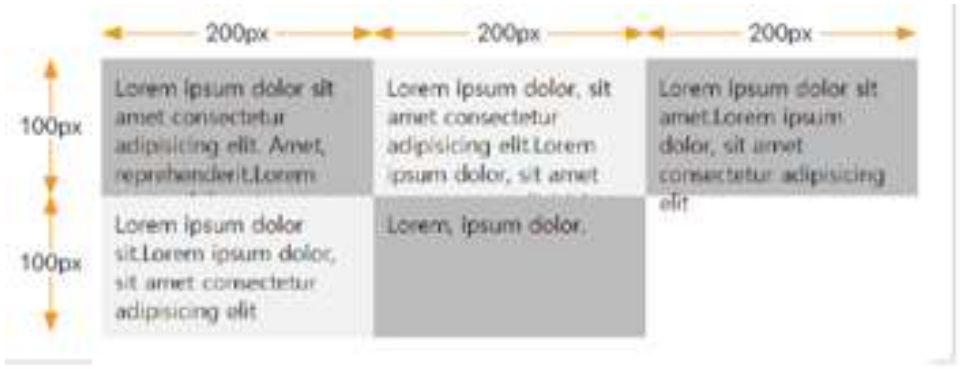
종류	설명
flex-start	교차축의 시작점에 맞춰 배치합니다.
flex-end	교차축의 끝점에 맞춰 배치합니다.
center	교차축의 중앙에 맞춰 배치합니다.
space-between	첫 번째 항목과 끝 항목을 교차축의 시작점과 끝점에 맞추고 나머지 항목은 그 사이에 같은 간격으로 배치합니다.
space-around	모든 항목을 교차축에 같은 간격으로 배치합니다.
stretch	플렉스 항목을 늘려서 교차축에 가득 차게 배치합니다.



반응형 웹 디자인

CSS 그리드 레이아웃

- flex box 레이아웃은 방향만 있고 컨테이너 사이즈를 넘어서면 줄바꿈으로 흐르는 구조
- grid 레이아웃에서는 row, column 구조
- 플렉스 그리드 레이아웃은 1차원, CSS 그리드 레이아웃은 2차원이라고도 함



반응형 웹 디자인

CSS 그리드 레이아웃 항목을 배치하는 속성

- display 속성
- 배치 요소들을 감싸는 부모 요소를 그리드 컨테이너로 지정

종류	설명
gird	컨테이너 안의 항목을 블록 레벨 요소로 배치합니다.
inline-grid	컨테이너 안의 항목을 인라인 레벨 요소로 배치합니다.

- grid-template-columns, grid-template-rows 속성
- 칼럼/줄의 크기와 개수 지정
- grid-template-columns : 그리드 컨테이너 안의 칼럼 개수와 너빗값
- grid-template-rows: 그리드 컨테이너 안의 줄 개수와 높이값

```
#wrapper2 {
        display: grid;
        grid-template-columns: 200px 200px 200px;
        grid-template-rows: 100px;
}
```

반응형 웹 디자인

상대적인 크기를 지정하는 fr 단위

- 칼럼/줄의 크기를 지정할 때 px 단위는 반응형 웹 디자인에 적합하지 않음
- 상대적인 크기를 지정하는 **fr**(fraction) 단위 사용
- fr 로 지정하게 되면 화면 사이즈에 따라 값이 자동으로 지정되면서 비율이 유지된다.

- 너비의 비율이 2:1:2인 칼럼 3개를 배치한다면

```
grid-template-columns: 2fr 1fr 2fr;
```

- 똑같은 값을 여러 번 반복한다면 내장 함수 repeat( ) 함수 사용

```
grid-template-columns: 1fr 1fr 1fr;
```

```
grid-template-columns: repeat(3, 1fr);
```

반응형 웹 디자인

자동으로 칼럼 개수를 조절하는 auto-fill, auto-fit

- 칼럼 너비와 함께 auto-fit이나 auto-fill을 지정하면 화면 너비에 따라 칼럼 개수를 조절할 수 있음
- auto-fit : 남는 공간 없이 꽉 채우기
- auto-fill : 칼럼의 최소 너비까지만 표시하고 남는 공간은 그대로 둠

- 사이즈가 자동으로 등분으로 나오게 할때는 fr 을 이용하면 되지만
- 컬럼의 개수도 사이즈에 따라 동적으로 나오게..
- 이때 사용하는 속성이 auto-fit, auto-fill 이다.
- repeat(auto-fit, 200px) 로 지정하게 되면 스크린 사이즈가 커지면 200px 의 개수가 더 많이 한줄이 나오게 된다.

반응형 웹 디자인

그리드 라인을 사용해 배치하기

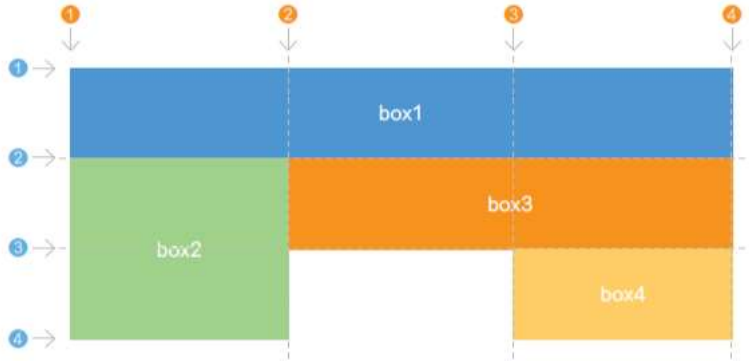
- 그리드 레이아웃에서 각 요소를 배치하는 방법은
- 각 방향의 라인 숫자를 이용해 배치하는 방법과 템플릿 영역을 이용해 배치하는 방법이다.

- css 그리드 레이아웃에는 눈에 보이지 않는 그리드 라인이 포함되어 있음
- 그리드 라인을 사용해 그리드 항목을 배치할 수 있음

종류	설명	예시
grid-column-start	칼럼 시작의 라인 번호를 지정합니다.	grid-column-start: 1
grid-column-end	칼럼 마지막의 라인 번호를 지정합니다.	grid-column-end: 1
grid-column	칼럼 시작 번호와 칼럼 끝 번호 사이에 슬래시(/)를 넣어 사용합니다.	grid-column: 1/4
grid-end-start	줄 시작의 라인 번호를 지정합니다.	grid-end-start: 2
grid-row-end	줄 마지막의 라인 번호입니다.	grid-row-end: 4
grid-row	줄 시작 번호와 줄 끝 번호 사이에 슬래시(/)를 넣어 사용합니다.	grid-row: 2/4

반응형 웹 디자인

그리드 라인을 사용해 배치하기




```
.box1 {
  background-color: #3689ff;
  grid-column: 1/4;
}
.box2 {
  background-color: #00cf12;
  grid-row: 2/4;
}
.box3 {
  background-color: #ff9019;

  grid-column: 2/4;
  grid-row-start: 2;
}
.box4 {
  background-color: #ffd000;
  grid-column-start: 3;
  grid-row-start: 3;
}
```

반응형 웹 디자인

템플릿 영역을 만들어 배치하기

- 레고블럭 넣는 것처럼 각 영역에 이름을 주고
- 그 이름의 요소를 추가하는 방법이다

- grid-area 속성을 사용해 템플릿 영역을 만든 후 배치


```
#wrapper {
  width: 700px;
  display: grid;
  grid-template-columns: repeat(3, 1fr);
  grid-template-rows: repeat(3, 100px);
  grid-template-areas:
    "box1 box1 box1"
    "box2 box3 box3"
    "box2 . box4";
}
```

한 줄에 들어갈 템플릿 영역을 큰따옴표(" ")로 묶습니다.

빈 영역은 마침표(.)를 넣습니다.

```
.box1 {
  background-color: #3689ff;
  grid-area: box1;
}
.box2 {
  background-color: #00cf12;
  grid-area: box2;
}
.box3 {
  background-color: #ff9019;
  grid-area: box3;
}
.box4 {
  background-color: #ffd000;
  grid-area: box4;
}
```

BootStrap

개요

- Twitter에서 만든 반응형 웹 개발을 위한 프론트엔드 프레임워크
- CSS + JS + 컴포넌트
- jQuery, React.js, Vue.js, Angular.js 와 연동 가능
- 제공기능
  - Grid, Layout
  - Typography
  - Table
  - List, List Group
  - Form
  - Button, Button Group, Image, Card
  - Nav, Nav Group, Tab
  - Badge, Spinner
  - Modal, DropDown, Tooltip, Carousel

BootStrap

Grid 시스템 - breakpoint

Breakpoint	Class infix	Dimensions
X-Small	*None*	<576px
Small	`sm`	≥576px
Medium	`md`	≥768px
Large	`lg`	≥992px
Extra large	`xl`	≥1200px
Extra extra large	`xxl`	≥1400px

BootStrap

Grid 시스템 – Container

- 가장 기본적인 레이아웃 요소
- 그리드 시스템을 작성할 때 반드시 필요
- 3가지 유형의 Container
  - .container : 각 breakpoint 에서 최대 폭이 지정되어 있음
  - .container-fluid : 모든 breakpoint에서 너비를 Viewport 폭의 100%를 사용함
  - .container-{breakpoint} : 지정된 중단점까지 폭이 100%로 지정됨

Classes Bootstrap Grid System	X-Small <576px	Small ≥576px	Medium ≥768px	Large ≥992px	X-Large ≥1200px	XX-Large ≥1400px
.container	100%	540px	720px	960px	1140px	1320px
.container-sm	100%	540px	720px	960px	1140px	1320px
.container-md	100%	100%	720px	960px	1140px	1320px
.container-lg	100%	100%	100%	960px	1140px	1320px
.container-xl	100%	100%	100%	100%	1140px	1320px
.container-xxl	100%	100%	100%	100%	100%	1320px
.container-fluid	100%	100%	100%	100%	100%	100%

BootStrap

Grid 시스템 – Container

```
<div class="container">
   <!-- 8: 4 비율로 -->
   <div class="row py-2">
      <div class="col-8 border bg-primary">col-8</div>
      <div class="col-4 border bg-primary">col-4</div>
   </div>
    <!-- 2번째 Cell을 6/12만큼 나머지를 균등 분할 3 : 6 : 3 -->
   <div class="row py-2">
      <div class="col border bg-info">col</div>
      <div class="col-6 border bg-info">col-6</div>
      <div class="col border bg-info">col</div>
   </div>
</div>
```

BootStrap

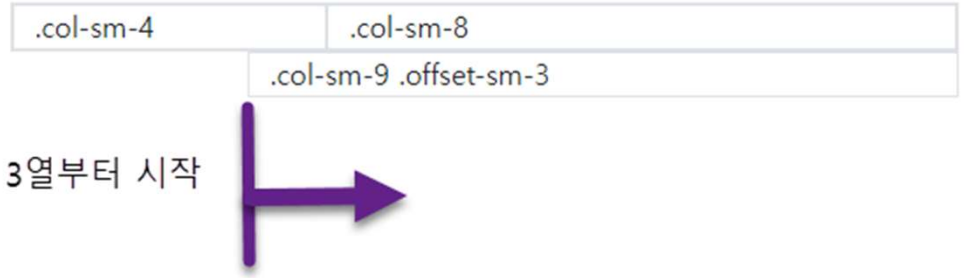
Grid 시스템 – 수평정렬

- 12개 열을 모두 채우지 않을 경우 정렬할 기준을 지정할 수 있음
- 정렬 방법
  - justify-content-start
  - justify-content-end
  - justify-content-center


```
    <div class="container">
      <div class="row justify-content-start bg-dark py-1">
        <div class="col-4 bg-light border">컬럼 1</div>
        <div class="col-4 bg-light border">컬럼 2</div>
      </div>
</div>
```

Grid 시스템 – offset

- Column이 offset으로 지정된 지점부터 나타남




```
<div class="container mt-3">
  <div class="row border">
    <div class="col-sm-4 border">.col-sm-4</div>
    <div class="col-sm-8 border">.col-sm-8</div>
  </div>
  <div class="row">
    <div class="col-sm-9 offset-sm-3 border">.col-sm-9 .offset-sm-3</div>
  </div>
</div>
```

BootStrap

Table

- HTML Table에 부트스트랩의 스타일을 적용해 손쉽게 정제된 스타일의 테이블을 생성
- class를 'table'로만 지정하면 됨.

```
<table class="table">
  <thead>
    <tr>
      <th>번호</th>
      <th>이름</th>
      <th>전화</th>
      <th>이메일</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>홍길동</td>
      <td>010-2222-2222</td>
      <td>gdhong@gmail.com</td>
    </tr>
```

번호	이름	전화	이메일
1	홍길동	010-2222-2222	gdhong@gmail.com
2	이몽룡	010-3333-3333	mrlee@gmail.com
3	성춘향	010-5555-555	chsung@gmail.com

BootStrap

Striped Table

- table 클래스로 .table-striped 지정
- 줄무늬

```
<table class="table table-striped">
  <thead>
    <tr class="table-dark text-white">
      <th>번호</th>
      <th>이름</th>
      <th>전화</th>
      <th>이메일</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>홍길동</td>
      <td>010-2222-2222</td>
      <td>gdhong@gmail.com</td>
    </tr>
    ......(생략)
```

번호	이름	전화	이메일
1	홍길동	010-2222-2222	gdhong@gmail.com
2	이몽룡	010-3333-3333	mrlee@gmail.com
3	성춘향	010-5555-5555	chsung@gmail.com
4	뺑덕어멈	010-7777-7777	bbduck@gmail.com
5	신사임당	010-8888-8888	sidshin@gmail.com
6	이율곡	010-9999-9999	yglee@gmail.com

BootStrap

Typography

- 제목, 단락 등의 텍스트 컨텐츠의 스타일 지정 기능
- <h1>~<h6>을 사용해도 되지만 .h1, .h5와 같은 클래스명을 사용해도 됨
- 더 눈에 띄는 큰 제목을 사용하려면 .display-1~6 클래스명을 사용

대소문자 자동 변환

- .text-lowercase, .text-uppercase, .text-capitalize
- capitalize는 영단어의 첫글자를 대문자로 변환

텍스트 색상 지정

- .text-primary와 같은 방법으로 지정
- primary, secondary, success, warning, info, danger 등 지정 가능

BootStrap

List Group

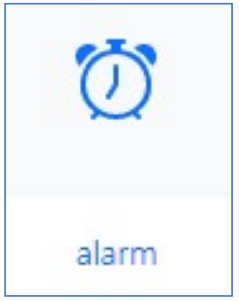
- .list-group
- .list-group-item
- .active : 활성화 표시

```
<div class="container px-2 py-2">
  <div class="h3 text-center">여행하고 싶은 국가</div>
  <hr />
  <div class="row justify-content-center">
    <div class="col-6">
      <ul class="list-group">
        <li class="list-group-item">몽골</li>
        <li class="list-group-item active">인도</li>
        <li class="list-group-item">튀니지</li>
        <li class="list-group-item">부탄</li>
      </ul>        </div>
  </div>
</div>
```

BootStrap

Bootstrap Icon

- bootstrap을 사용하지 않아도 icon만 별도로 이용할 수 있음
- https://icons.getbootstrap.com/
- 1600개 이상의 오픈소스 아이콘 라이브러리
- CSS 클래스 이름을 이용해 아이콘 사용
  - 아이콘 이름이 alarm 이면 → `<i class="bi-alarm"></i>`
- 설치 방법
  - CDN 이용
    - <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css">
    - @import url("https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css");
  - 직접 다운로드
    - https://github.com/twbs/icons/archive/refs/tags/v1.8.1.zip



BootStrap

다양한 버튼 스타일

- <button type="button" class="btn **btn-primary**">Primary</button>

OutLine 버튼

- <button type="button" class="btn **btn-outline-primary**">Primary</button>

버튼 크기

- btn-lg, btn-sm, btn-xl 등

버튼 비활성화

- <button type="button" class="btn btn-primary" **disabled**>Primary button</button>

스피너 버튼

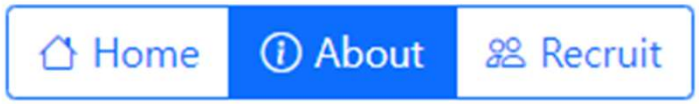
```
<button type="button" class="btn btn-primary" disabled>
    <span class="spinner-border spinner-border-sm"></span> 처리중
</button>
```

BootStrap

Button Group

- 버튼들의 그룹.
- 으로 버튼들을 묶어줌

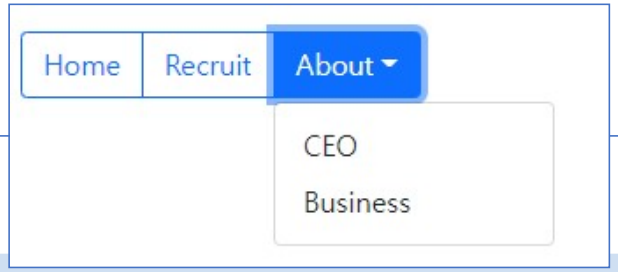
```
<div class="container px-2 py-2">
  <div class="py-2">
    <div class="btn-group">
      <button type="button" class="btn btn-outline-primary"><i class="bi-house"></i> Home</button>
      <button type="button" class="btn btn-outline-primary active"><i class="bi-info-circle"></i> About</button>
      <button type="button" class="btn btn-outline-primary"><i class="bi-people"></i> Recruit</button>
    </div>
  </div>
</div>
```



BootStrap

Button Group 다른 UI 조합

```
<div class="container px-2 py-2">
  <div class="py-2">
    <div class="btn-group">
      <a href="#" class="btn btn-outline-primary">Home</a>
      <a href="#" class="btn btn-outline-primary">Recruit</a>
      <button class="btn btn-outline-primary dropdown-toggle" data-bs-toggle="dropdown">About</button>
      <div class="dropdown-menu">
        <a class="dropdown-item" href="#">CEO</a>
        <a class="dropdown-item" href="#">Business</a>
      </div>
    </div>
  </div>
</div>
```



BootStrap

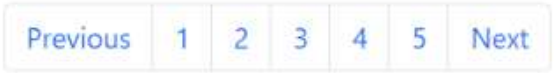
이미지 테두리 스타일링

- .rounded, .rounded-circle, .img-thumbnail

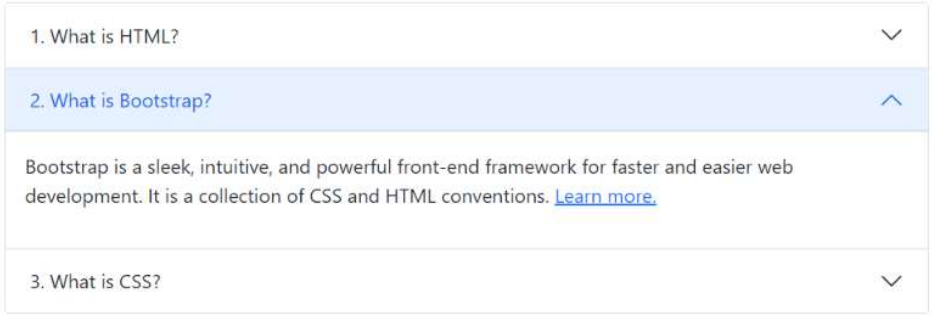
```
<div class="container px-2 py-2">
    <div class="py-2 row justify-content-center">
      <div class="col-2 border text-center">
        <img src="photos/clara.jpg" class="rounded" alt="Rounded Image" />
      </div>
      <div class="col-2 border text-center">
        <img src="photos/jane.jpg" class="rounded-circle" alt="Rounded Image" />
      </div>
      <div class="col-2 border text-center">
        <img src="photos/sean.jpg" class="img-thumbnail" alt="Rounded Image" />
      </div>
    </div>
</div>
```

BootStrap

- Pagination



- Accordion

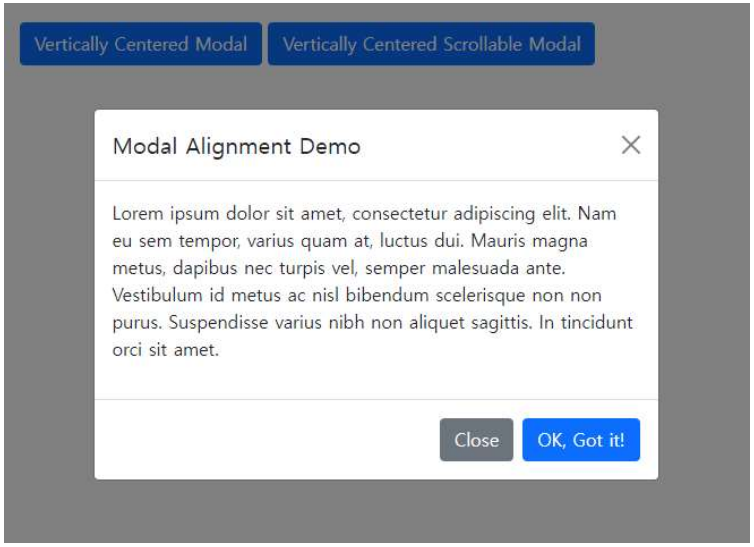


- badge

BootStrap

- Spinner

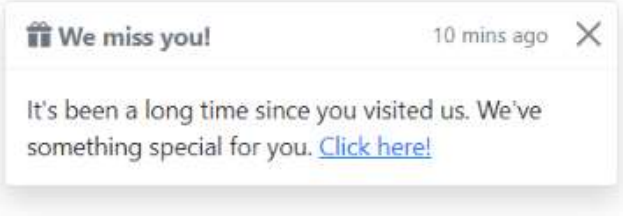
- Modal



BootStrap

- Carousel

- Toast
  - 가벼운 알림 기능



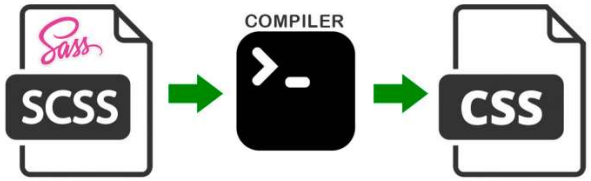
SCSS

CSS 의 한계

- 프로그래밍 언어가 아님으로 인한 개발 및 유지보수의 어려움
  - 선택자를 명시할 때 동일한 부모 요소 선택자가 매번 반복된다.
  - 색상, 사이즈등 값을 변경할 때 모든 곳의 내용을 일일이 변경해 주어야 한다.

SCSS(SASS) 란?

- CSS Preprocessor
- 컴파일 되어 CSS 로 만들어 져야 한다.



SCSS

SCSS(SASS) 의 이점

- 선택자의 중첩 표현을 통해 부모 요소의 반복을 현저하게 줄일 수 있다.
- 변수 사용을 통해 속성 값 관리를 쉽게 할 수 있다.
- 조건문, 반복문 등 프로그래밍 언어의 기법을 통해 동적인 CSS 양산이 가능하다.

SCSS vs SASS

- SCSS : Sassy Cascading Style Sheets
- SASS : Syntactically Awesome Style Sheet
-
- SASS 가 먼저 나왔고 그 이후 SCSS 나왔다.
- SASS 와 SCSS 는 완벽히 호환되며 최근에는 SCSS 를 주로 이용한다.
- SCSS와 SASS 는 미세한 구문 작성 법의 차이가 있으며 SCSS 는 CSS 문법을 따른다.

SCSS

중첩

```
.section {
  width: 100%;
  .list {
    padding: 20px;
    li {
      float: left;
    }
  }
}
```

```
.section {
  width: 100%;
}
.section .list {
  padding: 20px;
}
.section .list li {
  float: left;
}
```

SCSS

Ampersand(&) 중첩

```scss
.btn {
  position: absolute;
  &.active {
    color: red;
  }
}

.list {
  li {
    &:last-child {
      margin-right: 0;
    }
  }
}
```

```scss
.btn {
  position: absolute;
}
.btn.active {
  color: red;
}
.list li:last-child {
  margin-right: 0;
}
```

SCSS

변수

```
$color-primary: #e96900;
$url-images: "/assets/images/";
$w: 200px;
.box {
  width: $w;
  margin-left: $w;
  background: $color-primary url($url-images + "bg.jpg");
}
```

```
.box {
  width: 200px;
  margin-left: 200px;
  background: #e96900 url("/assets/images/bg.jpg");
}
```

감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare