

SASS

- CSS전처리기

CSS 전처리기는 CSS 코드를 작성할 때 보다 효과적이고 조직적으로 스타일을 관리하기 위한 스크립팅 언어이다. 이 전처리기는 특별한 문법을 사용하여 작성되며, 웹 사이트를 실행하기 전에 일반 CSS로 컴파일(vs code extension - compile hero)해서 배포합니다. 종류는 Sass와 SCSS가 있다.

- SASS 사용의 장점

- 코드 재사용 : 믹스인, 변수, 함수 등을 사용하여 코드 중복을 줄인다.
- 유지 및 관리 용의성 : 분할 및 구조화 된 코드로 관리한다.
- 고급 기능 사용 : 조건문, 반복문, 연산 등

- SCSS와 Sass의 차이

Sass	SCSS
<ul style="list-style-type: none">• 들여쓰기 기반 문법• 중괄호 {} 및 세미콜론 ; 없음• .sass 확장자 사용	<ul style="list-style-type: none">• CSS와 유사한 문법• 중괄호와 세미콜론 사용• .scss 확장자 사용
<div><h3>Sass</h3><pre>\$main-font: "Baskerville" =title(\$font) font-size: 30px font-family: \$font #header +title(\$main-font)</pre></div>	<div><h3>SCSS</h3><pre>\$main-font: "Baskerville"; @mixin title(\$font) { font-size: 30px; font-family: \$font; } #header { @include title(\$main-font); }</pre></div>

- 설치

브라우저는 Sass의 문법을 알지 못하기 때문에 Sass(.scss) 파일을 css 파일로 트랜스파일링(컴파일)하여야 한다. 따라서 Sass 환경의 설치가 필요하다.

npm] npm install -g sass

yarn] yarn add sass

- 컴파일

형식] sass <변환할 scss 파일명> <변환될 css 파일명>

ex) sass style.scss style.css

--watch : 자동 컴파일 옵션

ex) sass --watch style.scss style.css

--no-source-map : .map 파일이 생성되지 않는 옵션, 배포시에 사용

- 소스맵 파일(.css.map)

컴파일된 소스를 원본 소스로 맵핑해 주는 역할을 한다.

브라우저에서는 컴파일된 css를 이용해 렌더링 하므로 debugging할 때 어려움이 있으나

소스맵이 있으면 맵핑이 되기 때문에 CSS가 압축되어 있거나 모듈로 쪼개져 있어도 볼 수가 있다.

소스맵이 없는 경우	소스맵이 있는 경우
<pre>element.style { } body { font: 100% Helvetica, sans-serif; color: #333; }</pre>	<pre>element.style { } body { font: 100% Helvetica, sans-serif; color: #333; }</pre>

- SCSS 기본 문법

- 변수(Variables)

CSS의 값을 저장하고 재사용하는 방법

주로 중복해서 사용하는 font-family와 베이스 color 들을 변수로 지정한다.

선언 형식] \$변수명 : 값;

사용 형식] 속성: \$변수명;

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;

.submit-btn {
  background-color : $primary-color;
}
```

■ 중첩(Nesting)

계층적으로 상속되는 CSS를 더욱 계층적으로 보이게 만드는 기능

계층을 선택자로 표시하던 것을 그룹으로 묶을 수 있게 되어 더욱 깔끔하고 가독성 좋은 방식으로 코딩한다.

지나치게 깊은 depth로 사용하면 오히려 유지 보수성이 떨어지므로 중첩은 3 depth 이내로 하는 것이 권장된다.

기존 CSS

```
nav ul {
  margin: 0;
  padding: 0;
}
nav li {
  display: inline-block;
}
nav a {
  text-decoration: none;
}
```

SCSS

```
nav {
  ul {
    margin: 0;
    padding: 0;
  }
  li { display: inline-block; }
  a { text-decoration: none; }
}
```

■ 파일 분리와 임포트(Partials & Import)

.scss 파일을 여러 파일로 나누고 필요한 곳에서 임포트하는 기능

부분 파일 이름은 _로 시작하고, import 할 때는 생략한다.

_reset.scss

```
* {
  margin: 0;
  padding: 0;
}
```

main.scss

```
@import 'reset';
```

*** _(언더스코어)로 시작하는 파일 이름

독립적이지 않음 : stand alone으로 컴파일 되지 않는 것을 의미. 다른 SCSS 파일에 import되어 사용될 목적으로 만들어진다.

재사용성 : 부분 파일은 공통적으로 사용되는 변수, 믹스인, 함수 등을 정의하여 다른 SCSS 파일에서 재사용하기 위한 목적으로 만들어진다.

가독성 관리: 큰 스타일 시트를 여러 개의 작은 파일로 나누면 코드의 구조와 관리가 쉬워진다.

Ex)

_variables.scss : 색상 폰트 등의 변수를 저장한다.

_mixins.scss : 재사용 가능한 믹스인을 저장한다.

Main.scss : 부분 파일들을 import하여 전체 스타일을 구성한다.

■ Mixins

재사용 가능한 스타일의 그룹을 정의하는 사용.

선언 형식] @mixin 이름 { }

```
@mixin reset-text {  
  margin: 0;  
  padding: 0;  
  font-size: 16px;  
}
```

사용 형식] @include 이름; -- 해당 mixin이 상용구처럼 추가된다.

```
p {  
  @include reset-text;  
}
```

-함수 형식] 인자를 전달 받아 유연한 스타일링을 할 수 있다.

@mixin 이름(\$인자){

속성:\$인자

}

```
@mixin border-radius($radius) {  
  -webkit-border-radius: $radius;  
  -moz-border-radius: $radius;  
  -ms-border-radius: $radius;  
  border-radius: $radius;  
}
```

```
.button {  
  @include border-radius(5px);  
}
```

인자의 기본 값을 정의할 수 있다.

```
@mixin box-shadow($x: 0, $y: 0, $blur: 5px, $color: black) {
  box-shadow: $x $y $blur $color;
}
```



■ Extend

재사용 가능한 스타일 상속 받은 선택자들이 공유하는 방식

선택자 병합 방식이기 때문에 예기치 못한 사이드 이펙트가 발생할 수 있음

형식] 선택자 { @extend 선택자; }

SCSS 구문	CSS로 컴파일 후
<pre>main.scss > ... 1 .button-basic{ 2 border: none; 3 padding: 15px 30px; 4 text-align: center; 5 font-size: 16px; 6 cursor: pointer; 7 background-color: #pink; 8 } 9 10 .button-report { 11 @extend .button-basic; 12 background-color: #red; 13 } 14 15 .button-submit { 16 @extend .button-basic; 17 background-color: #green; 18 color: #white; 19 }</pre>	<pre>dist > # main.css > .button-report 1 .button-basic, .button-submit, .button-report { 2 border: none; 3 padding: 15px 30px; 4 text-align: center; 5 font-size: 16px; 6 cursor: pointer; 7 background-color: #pink; 8 } 9 10 .button-report { 11 background-color: #red; 12 } 13 14 .button-submit { 15 background-color: #green; 16 color: #white; 17 }</pre>



Cascading 이므로 override한 내용이 적용된다.

■ Placeholder

SCSS에서 **extend** 전용으로 쓰이는 가상의 선택자

형식] %placeholder{ }

사용형식] @extend %placeholder

■ Mixin과 extend 차이점

항목	@mixin	@extend
목적	재사용 가능한 스타일 블록 정의	기존 스타일을 상속하여 선택자 병합
파라미터 지원	가능	불가능
코드 출력 방식	스타일이 각 선택자에 복사됨	선택자들이 하나의 스타일 블록으로 병합됨
유연성	높음 (조건부 스타일, 동적 값 가능)	낮음
CSS 파일 크기	중복 스타일로 인해 커질 수 있음	스타일 중복 없이 작아질 수 있음
사용 시 주의점	불필요한 스타일 중복에 주의	선택자 병합으로 인한 예기치 못한 영향 가능
현업 사용 빈도	자주 사용됨	제한적으로 사용됨

■ & 연산자 (and)

Nesting 에서 자식이 아닌 부모(self)를 참조하는 역할

```
1  .button {
2      color: white;
3      background: blue;
4      font-size: 16px;
5
6      //1. &를 사용하여 중첩된 선택자가 부모 선택자를 참조하기
7      &:hover {
8          background: darkblue;
9      }
10
11     //2. 부모 선택자를 유지하면 미디어 쿼리 적용하기
12     @media (max-width: 600px) {
13         & {
14             font-size: 14px;
15         }
16     }
17 }
```

```

18  //3. 부모 선택자와 연결된 새로운 클래스명을 만들기
19  .card {
20      &-header {
21          font-size: 20px;
22      }
23      &-body {
24          padding: 15px;
25      }
26  }
27
28  //4. 부모 선택자가 포함된 구조를 만들기
29  .icon {
30      &.large {
31          width: 50px;
32          height: 50px;
33      }
34  }
35

```

■ 조건문(@if, @else if, @else)

주어진 조건에 따라 다른 스타일을 적용할 때 사용 – ex) theme

```

$theme: light;

.button {
    @if $theme == light {
        background-color: white;
        color: black;
    } @else {
        background-color: black;
        color: white;
    }
}

```

■ for

반복문을 활용해서 스타일을 자동 생성

형식]

@for 반복변수명 from 시작 through 끝{


#{반복변수}

}


#{변수명}: 변수 값을 클래스명이나 속성 값에 동적으로 적용

일반 web에서 사용.

VS code extension – compile here 사용



Sass/Less/Stylus/Pug/Jade/Typescript/Javascript Compile Hero ... 797K ★ 4

 Easy to compile ts, tsx, scss, less, stylus, jade, pug and es6+ on save without u...
Eno Yao

Install

파일명을 .scss로 작성 후 우클릭하여 compile 한다.

