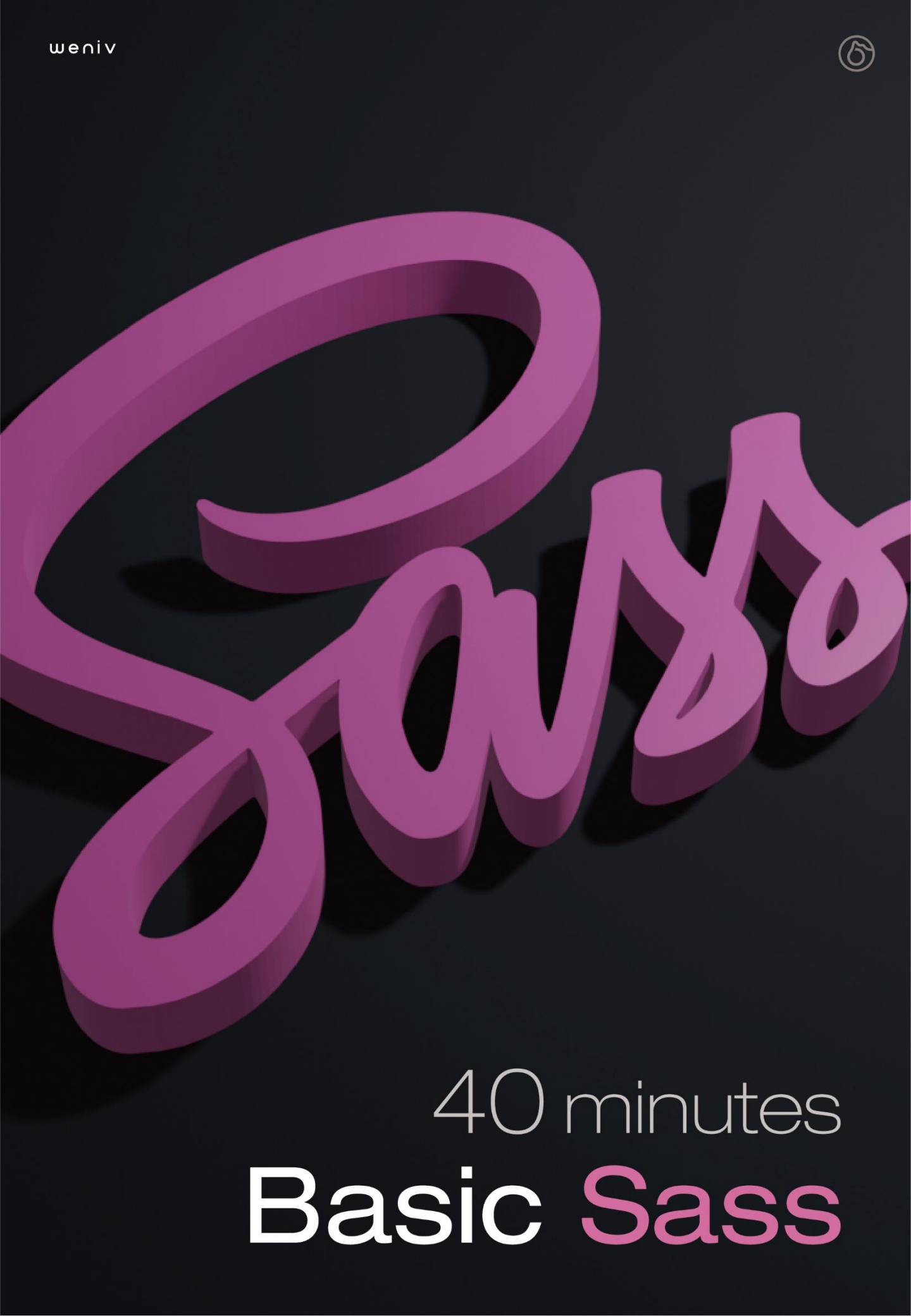
weniv



40 minutes

Basic Sass


Sass

Chapter 1. 강의소개 및 환경설정

1. 강의 특징
2. Sass를 쓰는 이유
3. Sass
    3-1 Sass 란
    3-2 Sass 기술방식 2가지
    3-3 환경 세팅
    3-4 번외

Chapter 2. 파일분리와 Nesting

1. 파일 분리 및 주석
    1-1 파일분리
    1-2 파일명 앞에 언더바를 붙이는 이유
    1-3 주석
2. 중첩(Nesting)
    2-1 중첩
    2-2 속성 Nesting
    2-3 앰퍼샌드
    2-4 @at-root
3. 전체 실습코드

Chapter 3. 변수(Variable)

1. CSS에서 변수라니?
2. 변수 생성하기
3. 변수 type
4. Lists, Maps
5. 변수의 Scope(변수의 유효범위)
6. Operator
7. 전체 실습 코드

Chapter 4. Mixin

1. Mixin이란
2. Mixin 사용하기
3. Arguments
4. Content
5. 전달인자 없는 믹스인
6. 예시
7. 전체 실습코드

Chapter 5. Extend

1. Extend
2. Extend 하는 2가지 방법
   2-1 class이름 가져오기
   2-2 %placeholder 사용하기
3. 예시 : 모달
4. 예시 : 포토 링크 박스
5. 전체 실습 코드

Chapter 6. 조건문과 반복문, 함수

1. 조건문
   1-1 @if
   1-2 @else
   1-3 @else if
2. 반복문
   2-1 @for
   2-2 @each
   2-3 @while
3. function
   3-1 function
   3-2 내장함수

Ch1. 강의 소개 및 환경 설정

1. 강의 특징

1. Notion 링크 제공
2. PDF로도 제공
3. 30분만에 훑어볼 수 있도록 속도감 있게 구성
4. 보다 상세한 내용은 부록으로 별도 제공

2. Sass나 SCSS 쓰는 이유

- 스타일시트가 점점 더 커지고 복잡해지고 유지관리가 어려워지고 있습니다.
- Sass안에 있는 변수, 네이스팅, 믹스인, 가져오기, 상속, 내장기능 등 css에는 없는 편의 기능들이 있어 시간을 절약할 수 있습니다.
- 코드 재사용이 가능합니다.

3. Sass

3-1. Sass 란

Sass는 CSS로 컴파일 되는 스타일 시트 확장 언어이며 CSS 전처리기의 하나입니다. 브라우저가 Sass를 직접 로딩하는 것이 아니라 **개발은 Sass를 기반으로 한 후**, CSS로 익스포트하는 과정을 거칩니다.

- 브라우저가 Sass파일을 직접 읽지 못하기 때문에 일반 CSS로 컴파일해야 합니다.
- 웹업계에서 실제 많이 사용하는 전처리기입니다.

3-2. Sass 기술방식 2가지

Sass를 작성하는데에는 기본적으로 두가지 방법이 있습니다.

- .sass 기술방식과 .scss 방식 → 다른 파일 확장자를 사용합니다.
- Sass와 Scss가 있는데 그 중에서 일반적으로 CSS와 좀 더 유사한 SCSS를 사용합니다. 왜냐하면 SCSS는 CSS와 동일하게 중괄호를 사용하는 방식이기 때문입니다.

- Sass와 Scss가 있는데 그 중에서 일반적으로 CSS와 좀 더 유사한 SCSS를 사용합니다. 왜냐하면 SCSS는 CSS와 동일하게 중괄호를 사용하는 방식이기 때문입니다.

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

body {
  font: 100% $font-stack
  color: $primary-color
}
```

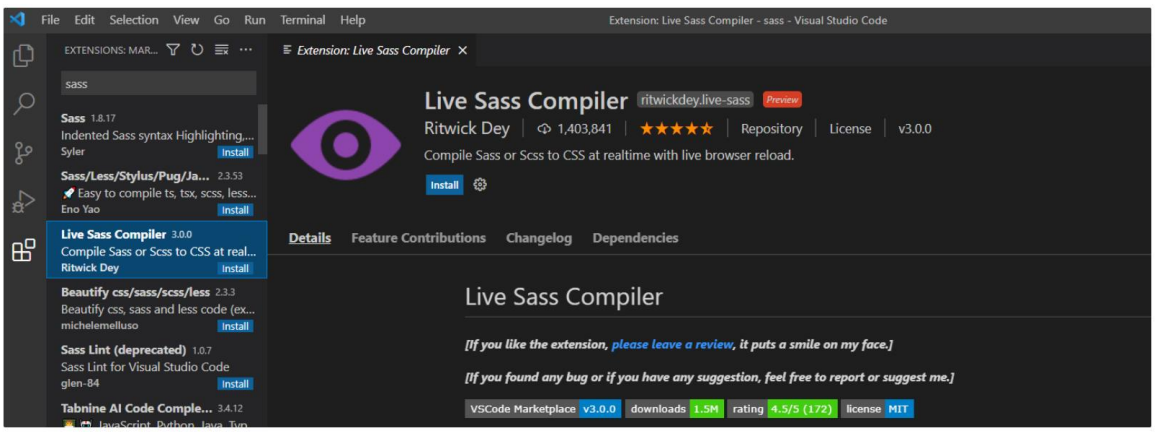
```sass
//Sass
$font-stack: Helvetica, sans-serif
$primary-color : #333

body
  font: 100% $font-stack
  color: $primary-color
```

3-3. 환경 세팅

VSC의 Extension을 사용합니다. VSC의 사용법 영상은 제주코딩베이스캠프 유튜브 채널 아래 영상을 참고해주세요.

install 해주시고, VSC를 재부팅 하시면 됩니다.




[VSCode 사용법] VSCode 와 유용한 extension 설치하기!
나중에 시...
공유
#1
비주얼 스튜디오 코드
#vscode설치 #extension설치
다음에서 보기: YouTube


[VSCode 사용법] VSCode와 Window ubuntu 연결하는 방...
나중에 시...
공유
#2
비주얼 스튜디오 코드
#VSCode와 Window ubuntu 연결
#Python 설치
다음에서 보기: YouTube

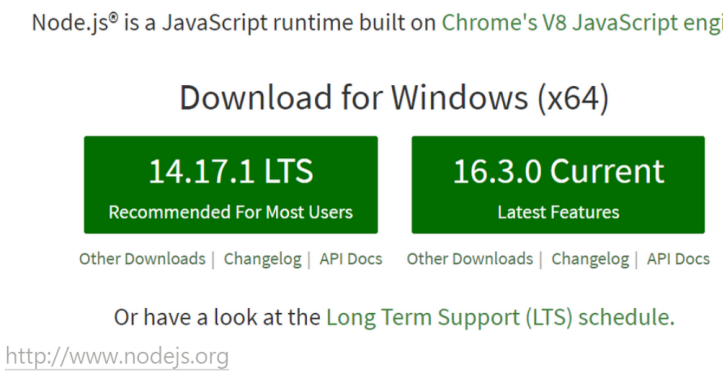
3-4. 번외 Sass → CSS 컴파일 방법

🏕️ 전처리기 처리 과정(Sass to CSS) :

1. Sass파일을 가져와서 웹 사이트에서 사용할 수 있는 일반 CSS 파일로 저장합니다.
2. 아래 설명한 설치 방법대로 Sass가 설치되면 터미널에서 Sass명령어를 사용하여 .SCSS파일을 Sass 컴파일러를 통해 컴파일 합니다.
3. .scss파일이 .css로 바뀝니다.

📌이번에는 VSC Extension이 아닌 **터미널을 사용** 방법을 소개합니다.

1. npm을 설치하도록 하겠습니다. node.js를 설치하면 자동적으로 npm도 설치됩니다. 홈페이지 (http://www.nodejs.org)에서 LTS 버전으로 다운로드 받아주세요.



2. VSC에서 터미널(`Ctrl + `)을 열어주세요.

1) npm버전이 출력되면 설치되어 있음을 의미합니다.

```
npm -v
```

```
PS C:\Users\paullabhome\Desktop\study> npm -v
6.14.13
```

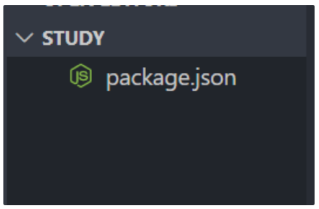
2) -y를 써서 질문 없이 기본세팅합니다. / 폴더명 한글은 애러 발생하므로 주의합니다.

```
npm init -y
```

```
PS C:\Users\paullabhome\Desktop\study> npm init -y
Wrote to C:\Users\paullabhome\Desktop\study\package.json:

{
  "name": "study",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

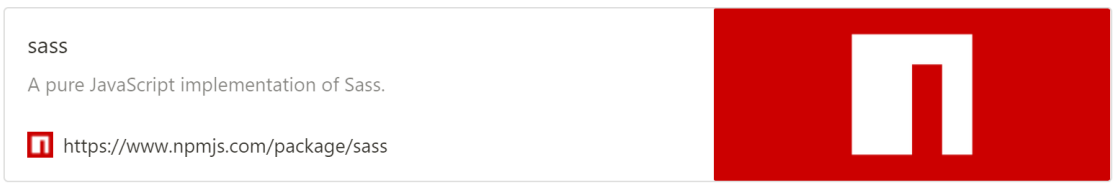
3) 위 명령어를 터미널에서 입력하면 아래 파일 package.json이 생성됩니다.



3. node-sass 설치해야 합니다.

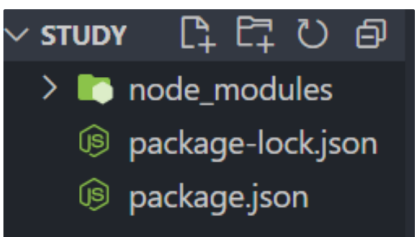
```
npm i node-sass
```

위 명령의 자세한 내용은 아래 사이트(https://www.npmjs.com/package/sass)를 참고해주세요.
node-sass는 Sass의 C 버전인 LibSass에 제공하는 라이브러리입니다.



위 명령어를 입력하게 되면 node_modules와 package-lock.json이 생성됩니다.

※ git commit 할 경우, node_modules와 package-lock.json 파일은 .gitignore파일안에 작성하여 git commit에서 제외시킵니다.



4. package.json안 "scripts"부분에 밑에 내용을 작성하여 컴파일할 파일과 컴파일될 파일명을 작성합니다.

scripts안에 sass를 실행할 명령어를 정의합니다.

https://www.npmjs.com/package/node-sass

"command line interface 검색"

Command Line Interface

The interface for command-line usage is fairly simplistic at this stage, as seen in the following usage section.

Output will be sent to stdout if the `--output` flag is omitted.

Usage

`node-sass [options] <input> [output]` Or: `cat <input> | node-sass > output`

Example:

```
node-sass src/style.scss dest/style.css
```

Sass에게 빌드할 파일과 CSS를 출력할 위치를 지정해야 합니다.

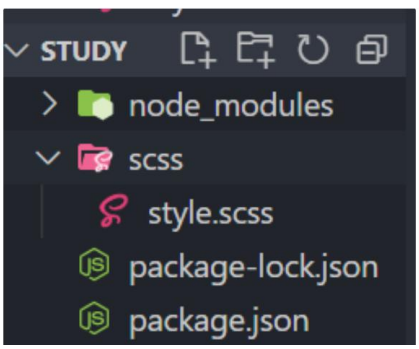
터미널에서 실행하면 단일 sass 파일인 input.scss가 발생하고 해당 파일이 output.css로 컴파일 됩니다.

```
"sass": "node-sass -w -r scss/input.scss src/output.css "
```

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "sass": "node-sass scss/style.scss src/style.css "
},
```

→ scss파일안에 있는 style.scss에서 src/style.css로 컴파일 할 수 있도록 설정합니다.(변경 가능합니다.)

scss폴더에 style.scss파일을 만듭니다.

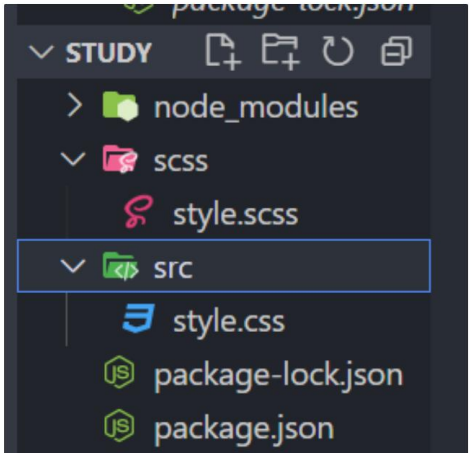


5. sass 실행하는 소스입니다.

```
npm run sass
```

```
PS C:\Users\paullabhome\Desktop\study> npm run sass
```

src/style.css 파일이 생성되는 것을 확인할 수 있습니다.



6. sass 업데이트 하기 - options 사용합니다.

https://www.npmjs.com/package/node-sass

[옵션] 부분에 -w과 -r을 추가합니다.

```
-w, --watch               Watch a directory or file
-r, --recursive           Recursively watch directories or files
```

-w

옵션이 없을 때는 sass파일을 수정할 때마다 sass를 실행합니다. 하지만 -w를 옵션으로 추가하게 되면 sass가 꺼지지 않고 계속적으로 sass 파일의 변경사항을 감시하면서 저장할 때마다 자동으로 컴파일을 해줍니다.

-r

-r은 -w와 같이 감시를 하는데 차이점은 -w만 추가했을 경우에 메인 파일만 감시하고 그 외에 파일들은 감시하지 않아서 변경을 하지 않습니다. 하지만 -r을 추가할 경우 메인파일에 import한 다른 파일도 함께 감시합니다.

Ch2. 파일 분리와 Nesting

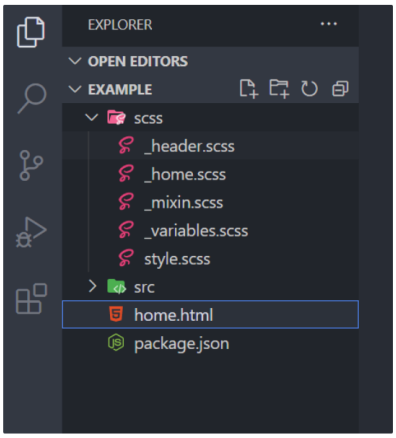
파일을 다운로드 받아 살펴봐주세요.

example.zip 3.8KB

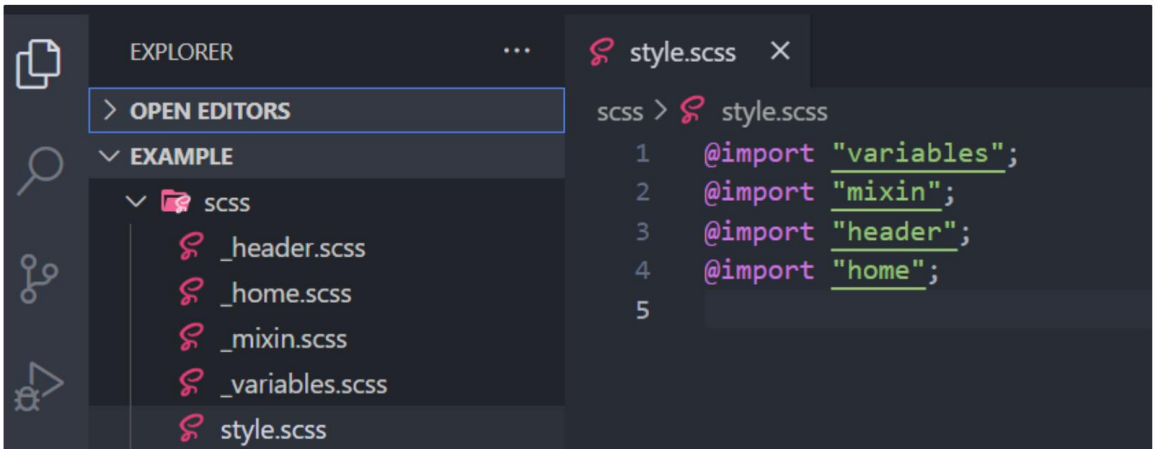
1. 파일 분리 및 주석

1-1. 파일 분리

본격적으로 코딩에 들어가기 전에 프로젝트를 진행할 때 파일들을 어떻게 저장하여 코드들을 관리하는지 보도록 하겠습니다. 아래 이미지를 보면 각 프레임 별(_header.scss, _home.scss) css를 분리하고, variable과 mixin 파일도 따로 분리하였습니다. 그리고 메인 파일인 style.scss에 분리했던 파일들을 import하였습니다. 이렇게 파일을 스타일 별로 기능별로 분리하여 사용할 수 있으며 파일을 기능별, 레이아웃 별로 분할해서 사용하기 때문에 코드를 관리하기 편리해집니다.(아래 파일을 참고해주세요)



위 파일에 `style.scss` 파일을 보면 나머지 파일들을 아래처럼 `import` 하고 있는 것을 볼 수 있습니다. 이때, style.scss 파일에는 @import와 주석 외에는 다른 코드들을 작성하지 않습니다.



1-2. 파일명 앞에 언더바 _를 붙이는 이유

- Partial : '_'(언더스코어)를 붙이지 않는다면 분할 된 파일들 모두가 컴파일되기 때문에 여러개의 .css파일이 나눠서 저장됩니다. 그러나 scss파일의 이름 앞에 '_'(언더스코어)를 붙여서 파일명을 정한다면, Sass에게 이 파일이 main파일의 일부분임을 알려줘서 해당 파일은 css파일로 컴파일 하지 않고 내부에서 @import 형태로 작동하게 됩니다.

※ css는 import할 때 파일 URL을 적어줘야 하지만, Sass에서 import할 때는 확장명을 제외하고 파일명만을 사용할 수 있습니다.

1-3. 주석

주석을 한 줄 작성할 때는 //을 사용하고, 작성한 주석 내용은 sass 내에서만 볼 수 있습니다. 여러 줄을 작성할 경우 /*을 사용하고, scss파일이 컴파일 될 때 주석 내용이 css 파일에 나타납니다.

```
/* 여러줄 주석은 볼 수 있습니다. */
// 한 줄 주석은 볼 수 없습니다.
```

2. 중첩(Nesting)

2-1. 중첩(Nesting)

Nesting(중첩)을 사용하면, html의 시각적 계층 방식과 동일하게 CSS를 중첩하여 작성할 수 있습니다. CSS코드가 구조화 되어 가독성이 높아지며 유지 보수하기 편리해집니다.

```
<!--HTML-->

<nav> <!--nav안에 ul이 중첩되어 있고-->
    <ul> <!--ul안에 세가지 li가 중첩되어 있다.-->
        <li>one</li>
        <li>two</li>
        <li>three</li>
        <li>four</li>
    </ul>
</nav>
```

```scss
//Scss
//Scss에서도 HTML처럼 계층구조로 스타일을 적용할 수 있다.
nav{
  background : #C39BD3;
  padding : 10px;
  height: 50px;
  ul{
    display: flex;
    list-style : none;
    justify-content: flex-end;
    li{
      color: white;
      margin-right: 10px;
    }
  }
}
```

Why. 중첩을 사용하는 이유는?

기존 CSS는 부모에게 상속된 자식 요소에게 스타일을 적용하려고 할 때마다 최상위 선택자를 반복 선언해야 합니다. 하지만 중첩을 사용하면 최상위 선택자를 한번만 선언하여도 자식 선택자에게 스타일을 적용할 수 있게 되어 코드 반복을 줄이게 됩니다.

```scss
/*CSS*/

info-list div {
  display: flex;
  font-size: 14px;
  color: #4f4f4f;
}

info-list div dt {
  font-weight: 700;
  margin-right: 7px;
}
/*기존 CSS : info-list에 있는 자식과 자손에게 스타일을 적용하려고 할때마다
부모 선택자를 적어준다*/
```

```scss
//Scss

info-list {
  div {
    display: flex;
    font-size: 14px;
    color: #4f4f4f;
    dt {
      font-weight: 700;
      margin-right: 7px;
    }
  }
}
// 중첩을 사용하여 부모선택자를 한번만 사용한다.
// 그리고 코드를 봤을 때 info-list div tag안에 dt가 들어있음을 한눈에 알아볼 수 있다
```

⚠️하지만 지나친 중첩된 코드는 삼가 해주세요. 깊이 중첩되면 코드를 보는 게 불편하고 컴파일 했을 경우 불필요한 선택자를 사용하게 됩니다.

2-2. 속성 Nesting

중첩은 선택자뿐만 아니라 스타일 속성들도 가능합니다. 아래 예시를 보면 .add-icon이라는 클래스 선택자에게 background 스타일을 주려고 합니다. 이때, background 이름을 가진 속성(background-image, background-position 등)들을 background안에 중첩 시켜서 스타일 코드를 작성할 수 있습니다. 속성을 중첩 할 때는 콜론`:`을 사용합니다. Sass는 속성이 중첩되었음을 인지하고 css 속성들로 변환합니다.

```scss
//Scss
.add-icon {
  background : {
    image: url("./assets/arrow-right-solid.svg");
    position: center center;
    repeat: no-repeat;
    size: 14px 14px;
  }
}
```

```css
/*CSS*/
.add-icon {
  background-image: url("./assets/arrow-right-solid.svg");
  background-position: center center;
  background-repeat: no-repeat;
  background-size: 14px 14px;
}
```

2-3. ampersand 앰퍼샌드

"&"는 상위에 있는 부모선택자를 가리킵니다.

1) `&`을 이용하여 after, hover 등의 가상요소, 가상클래스, class나 id 셀렉터 등을 참조할 수 있습니다.

```scss
//Scss
.box {
  &:focus{} // 가상선택자
  &:hover{}
  &:active{}
  &:first-child{}
  &:nth-child(2){}
  &::after{} // 가상요소
  &::before{}
}
```

```css
/*CSS*/
.box:focus{} /* 가상선택자 */
.box:hover{}
.box:active{}
.box:frist-child{}
.box:nth-child(2){}
.box::after{} /* 가상요소 */
.box::before{}
```

Example

```scss
//Scss
ul {
  li {
    &:hover {
      background: white;
      cursor: pointer;
      // 가상요소
    }
    &:last-child {
      border-bottom: 2px solid black;
      // 가상클래스
    }
  }
}
//li 태그의 가상요소와 가상클래스에게 스타일을 적용한 예시
//&을 사용하여 li와 :hover, :last-child를 연결
```

```css
/*CSS*/
ul li:hover {
  background-color: white;
  cursor: pointer;
}

ul li:last-child {
  border-bottom: 2px solid black;
}
```

2) `&` 를 응용하면 아래 예시와 같이 공통 클래스 명을 가진 선택자들을 중첩시킬 수 있습니다.

```scss
//Scss
.box {
  &-yellow {
    background: #ff6347;
  }
  &-red {
    background: #ffd700;
  }
  &-green {
    background: #9acd32;
  }
}
//.box라는 이름이 같기 때문에 &를 사용해 중첩구조로 만들 수 있다
```

```
/*CSS*/
.box-yellow {
  background: #ff6347;
}
.box-red {
  background: #ffd700;
}
.box-green {
  background: #9acd32;
}
```

3) `&` 은 자신의 부모 선택자를 참조하지만 중첩이 깊어지면, 자신의 직계 부모가 아닌 최상위 부모 선택자로부터 참조됩니다.

```
//Scss
.nav {
  height: 60px;
  font-size: 18px;
  .nav-list {
    background: #3e68ff;
    span {
      padding: 10px 13px;
      a {
        color: white;
        .one {
          & .two {
            color: skyblue;
          }
        }
      }
    }
  }
}
```

```
.nav {
  height: 60px;
  font-size: 18px;
}
.nav .nav-list {
  background: #3e68ff;
}
.nav .nav-list span {
  padding: 10px 13px;
}
.nav .nav-list span a {
  color: white;
}
.nav .nav-list span a .one .two {
  color: skyblue;
}
```

⚠️깊은 중첩을 하게 되면서 불필요한 선택자들이 사용되었습니다. 중첩을 과하게 사용하지 않도록 주의해주세요.

2-4. @at-root

`@at-root` 키워드를 사용하면 중첩에서 벗어날 수 있습니다. 중첩에서 벗어나고 싶은 선택자 앞에 `@at-root` 를 작성합니다. 컴파일된 css 코드에서 `@at-root` 를 사용한 선택자가 중첩에서 벗어났음을 확인할 수 있습니다. 중첩된 선택자에게만 사용할 수 있습니다.

```
//Scss
.article {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 10px;
  .article-content {
    font-size: 14px;
    opacity: 0.7;
    @at-root i {
      opacity: 0.5;
    }
  }
}
```

```css
/*CSS*/
.article {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 10px;
}
.article .article-content {
  font-size: 14px;
  opacity: 0.7;
}
/*중첩을 빠져나온 것을 확인할 수 있다.*/
i {
  opacity: 0.5;
}
```

3. 전체 실습 코드

001. 파일 분리와 Nesting.zip 945.7KB

Ch3. 변수(Variable)

1. css에서 변수라니?

변수를 선언한다는 것은 값을 일일이 고치지 않아도 된다는 의미입니다. 유지보수를 매우 쉽게 만들어준다는 것이죠.

하지만 무분별하게 CSS를 작성해서는 안됩니다. 충돌할 수도 있기 때문입니다. 타당한 이유가 있는 변수들만 먼저 생성하시고, 다음 변수는 팀의 적절한 논의를 거쳐 생성하도록 하세요.

※ 변수를 사용하는 기준

- 값이 두 번 이상 반복된다면 미리 변수로 만듭니다. 값을 기억하지 않고 변수명만을 가지고 스타일을 할 수 있습니다.
- 기존의 값을 다른 값으로 변경해야할 경우, 변수의 값만 변경하면 되기 때문에 값이 수정될 가능성이 있다면 변수 생성을 고려합니다.

보통 타이포그래피, 폰트색상, 폰트사이즈, 글자 간격 등의 값을 변수로 정의해서 사용합니다.

2. 변수 생성하기

변수를 만들 때는 `$` 기호를 사용하셔서 스타일을 적용할 값(색상, 폰트 사이즈, 이미지url)을 저장합니다.

```
//$변수 : 값
$bgColor : #FFF
```

css 전체에 걸쳐 반복되는 값들을 정의하면 편하게 스타일링을 할 수 있습니다.

```
//색상
$red: #ee4444;
$black: #222;
$bg-color: #3e5e9e;
$link-color : #red;
$p-color : #282A36;

//폰트사이즈
$font-p : 13px;
$font-h1 : 28px;

//폰트
$base-font : 'Noto Sans KR', sans-serif;

body{
    background-color : $bg-color;
    font-size : $font-p;
    font-family : $base-font;
}

h1{
   font-size : $font-h1;
   color : $black;
}

p{
  font-size : $font-p;
  color : $black;
}
```

3. 변수 type

numbers, strings, color, booleans, lists, null이 있습니다. 아래와 같은 형태로 입력 가능합니다.

- numbers : 1, .82, 20px, 2em 등
- strings : "./images/a.png", bold, left, uppercase 등
- colors : green, #FFF, rgba(255,0,0,.5) 등
- booleans : true, false
- null

- lists :

```
//sass 공식문서
$font-size : 10px 12px 16px; //폰트사이즈 리스트
$image-file : photo_01, photo_02, photo_03 //이미지 파일명 리스트

//아래와 같은 형태로 사용(순회도 가능)
list.nth(10px 12px 16px, 2); // 12px
list.nth([line1, line2, line3], -1); // line3
```

- maps :

```
//sass 공식문서
$font-weights: ("regular": 400, "medium": 500, "bold": 700); //글자 굵기 리스트

//아래와 같은 형태로 사용
map.get($font-weights, "medium"); // 500
map.get($font-weights, "extra-bold"); // null
```

4. Lists, Maps

4-1. Lists

리스트는 순서가 있는 값으로, 값마다 인덱스를 가지고 있습니다. lists를 만들려면 lists의 요소들을 쉼표 `,` 나 공백 또는 일관성이 있는 `/` 로 구분하여 생성하고, 다른 타입의 변수들과 다르게 특수 괄호를 사용하지 않아도 lists로 인식합니다. 빈 lists를 만들 때나 lists에 들어있는 값이 하나 인 경우 `[ ]` 나 `( )` 를 사용하여 생성합니다.

lists에 들어있는 값의 index는 0부터 시작하지 않고 1부터 시작합니다.

```
$sizes: 40px, 50px, 80px;
$valid-sides: top, bottom, left, right;
```

lists 관련 내장함수

`append(list,value,[separator])` , `index(list,value)` , `nth(list, n)` 등이 있습니다.

- `append(list,value,[separator])` : lists의 값을 추가하는 함수
- `index(list,value)` : lists의 값에 대한 인덱스를 리턴하는 함수
- `nth(list, n)` : lists의 인덱스에 해당하는 값을 리턴하는 함수

예시

`nth()`를 사용해서 lists의 인덱스에 해당하는 값을 불러옵니다.

```scss
// Scss
$valid-sides: left, center, right;

.screen-box{
  text-align : nth($valid-sides,1);
}
```

```css
/*CSS*/
.screen-box {
  text-align: left;
}
```

4-2. Maps

maps는 `( )` 괄호 안에 키:값의 형태로 저장하여 사용합니다. 키와 값을 정의할 때, 키는 고유해야 하지만 키에 해당하는 값은 중복이 가능합니다. 변수를 각각 선언하는 대신, 관련 있는 변수들을 한번에 모아 maps로 만들어서 사용할 수 있습니다.

map관련 내장함수

`map-get(map,key)`, `map-keys(map)`, `map-values(map)` 등이 있습니다.

- `map-get(map,key)` : 키에 해당하는 값을 값을 리턴하는 함수
- `map-keys(map)` : map에 들어있는 키(key) 전부를 리턴하는 함수
- `map-values(map)` : map에 들어있는 값(value) 전부를 리턴하는 함수

예시

`map-get`을 사용하여 map안에 있는 키에 해당되는 값을 불러옵니다.

```
// Scss
$font-sizes: ("h1": 45px, "h2": 19px, "p": 16px);

section {
  h2 {
    font-size : map-get($font-sizes, "h1");// 500
  }
}
map-get($font-size, "h3");// null
```

```
/*CSS*/
section h2 {
  font-size : 19px;
}
```

※ lists와 maps 뿐만 아니라 string이나 number같은 타입들도 function을 가지고 있습니다. 아래 사이트를 참고하시기 바랍니다.

Sass String Functions

Function Description & Example str-index(string, substring) Returns the index of the first occurrence of the substring within string. Example: str-index("Hello

https://www.w3schools.com/sass/sass_functions_string.php

5. 변수의 Scope(변수의 유효범위)

변수는 전역변수와 지역변수로 두가지 종류가 있습니다.

5-1. 지역변수

지역변수는 선언한 자기자신을 감싸고 있는 중괄호 { } 안에서 사용됩니다. 하위 단계에 있는 중괄호에서도 사용할 수 있습니다.

```
.info{
  $line-normal : 1.34; // 지역변수
  font-size : 15px;
  line-height : $line-normal;
  text-align : right;
  span{
    line-height : $line-normal;
  }
}
```

5-2. 전역변수

전역변수는 가장 윗부분에 정의하면 파일 내에 어디서든지 사용가능합니다.

전역변수를 파일로 만들어서 사용하는 경우, 메인 scss파일(파일분할부분에서 설명한 style.scss파일)에서 전역변수파일을 다른 파일들보다 윗부분에 위치시킵니다.

```
//Scss
$font-p : 15px; // 전역변수

.main-box{
  p {
    font-size : $font-p;
  }
  a {
    font-size : $font-p;
    color : blue;
    text-decoration : none;
  }
}
```

```
/*CSS*/
.main-box p {
  font-size: 15px;
}

.main-box a {
  font-size: 15px;
  color: blue;
  text-decoration: none;
}
```

`!global`을 사용하여 local 변수를 global 변수로 만들어 사용할 수도 있습니다.

```
$mycolor: #ffffff !global;
```

Variables

Sass variables are simple: you assign a value to a name that begins with $, and then you can refer to that name instead of the value itself. But despite their simplicity, they're one of the most useful tools Sass brings to the table.

https://sass-lang.com/documentation/variables

6. Operator

6-1. 비교연산자 (숫자형)

1 < , <=, >, >=

- `a < b` : a의 값이 b보다 값이 작은지 확인합니다.
- `a <= b` : a가 b보다 값이 작거나 같은지 확인합니다.
- `a > b` : a의 값이 b보다 값이 큰지 확인합니다.
- `a >= b` : a가 b보다 값이 크거나 같은지 확인합니다.

```
// Sass 공식문서
@debug 100 > 50; // true
@debug 10px < 17px; // true
@debug 96px >= 1in; // true
@debug 1000ms <= 1s; // true
```

⚠️ERROR : 비교하거나 연산하는 값의 단위가 일치하지 않으면 에러가 발생합니다. 그러나, 단위가 없는 숫자와 일반숫자와 비교하는 경우에는 에러가 발생하지 않습니다.

```
//Sass 공식문서

@debug 100px > 10s;
// Error: Incompatible units px and s

@debug 100 > 50px; // true
@debug 10px < 17; // true
// Not Error
```

2️⃣ ==, != (모든 타입)

- `a == b` : a가 b와 값이 같은지 확인합니다.
- `a != b` : a가 b와 값이 다른지 확인합니다.

변수의 타입에 따라 `true`, `false` 결과가 약간씩 다릅니다.(`==` 의 연산자의 경우를 말합니다. `!=` 연산자는 반대의 값을 반환합니다.)

- 색상, 숫자, 문자열은 값과 단위가 동일한 경우나, 값의 단위를 서로 변환했을 때 일치하는 경우 `true`를 반환합니다.
- 맵은 키와 값이 모두 동일할 때, 리스트는 들어있는 값들이 모두 동일해야만 `true`를 반환합니다.
- boolean의 경우, `true == true`, `false == false`, `null == null` 처럼 각자 자신과 비교할 때만 true를 반환합니다.

```
//Sass 공식문서

// 숫자
@debug 1px == 1px; // true
@debug 1px != 1em; // true
@debug 1 != 1px; // true
@debug 96px == 1in; // true

// 문자
@debug "Poppins" == Poppins; // true
@debug "Open Sans" != "Roboto"; // true

// 색상
@debug rgba(53, 187, 169, 1) == #35bba9; // true
@debug rgba(179, 115, 153, 0.5) != rgba(179, 115, 153, 0.8); // true
```

```
// 리스트
@debug (5px 7px 10px) != (5px, 7px, 10px); // true
@debug (5px 7px 10px) != [5px 7px 10px]; // true
@debug (5px 7px 10px) == (5px 7px 10px); // true
```

6-2. 산술연산자 (숫자나 색)

- `a + b` : a 와 b의 값을 더합니다.
- `a - b` : a 에서 b의 값을 뺍니다.
- `a * b` : a와 b의 값을 곱합니다.
- `a / b` : a를 b로 나눕니다.
- `a % b` : a 에서 b를 나눈 나머지 값을 구합니다.

```
//Sass 공식문서
@debug 10s + 15s; // 25s
@debug 1in - 10px; // 0.8958333333in
@debug 5px * 3px; // 15px*px
@debug 1in % 9px; // 0.0625in (1in == 96px)
```

※ 나누기를 할 때 사용하는 슬래시 `/` 는 리스트에서도 사용하기 때문에 혼동을 줄 수 있습니다. 그래서 괄호를 사용하거나, 변수와 함께 사용하거나, 덧셈을 할 때 함께 써서 Scss에게 `/` 를 나누기 연산자임을 알려줘야 합니다.

⚠️ERROR : 비교하거나 연산하는 값의 단위가 동일하지 않으면 에러가 발생합니다.

```
//Sass 공식문서
@debug 100px + 10s;
// Error: Incompatible units px and s.
```

6-3. String의 `a + b`

앞서 말했던 `+` 연산자에서 a와 b가 모두 문자열이라면 문자열 `a`, `b` 를 합쳐서 새로운 문자열로 반환합니다. 만약 a나 b중 하나만 문자열이라 하더라도 문자열이 아닌 값은 문자열로 변환하여 두 값을 합친 문자열로 반환합니다.

```
//Sass 공식문서
@debug "Helvetica" + " Neue"; // "Helvetica Neue"
@debug sans- + serif; // sans-serif
@debug sans - serif; // sans-serif


@debug "Elapsed time: " + 10s; // "Elapsed time: 10s";
@debug true + " is a boolean value"; // "true is a boolean value";
```

```
// Scss
.status-bar {
  font-family : "Noto Sans KR", sans- + serif;
}

td {
  &::after{
    content : "Heades" + " up!"; // 문자열 더하기
  }
}
```

```
/*CSS*/
.status-bar {
  font-family: "Noto Sans KR", sans-serif;
}

td::after{
  content : "Heades up!";
}
```

6-4. 논리연산자 (불리언 타입)

- `not` : `true`이면 `false`를, `false`이면 `true`를 반환합니다.
- `and` : 두개 다 `true`일때 `true`를 반환하고, 하나라도 `false`면 `false`를 반환합니다.
- `or` : 두개 다 `false`면 `false`를 반환하고, 하나라도 `true`라면 `true`를 반환합니다.

```
// Sass 공식문서
@debug not true; // false
@debug not false; // true

@debug true and true; // true
@debug true and false; // false

@debug true or false; // true
@debug false or false; // false
```

7. 전체 실습 코드

002. 변수(Variable).zip 1.2KB

Ch4. Mixin

1. Mixin이란

Mixin은 코드를 재사용하기 위해 만들어진 기능입니다. 선택자들 사이에서 반복되고 있는 코드들은 mixin을 사용하여 코드 반복을 줄입니다. 중복되는 코드는 mixin으로 만들어 놓고 원하는 선택자 블럭에 mixin을 include하면 됩니다.

2. Mixin 사용하기

```
@mixin 이름(매개변수) //생성
@include 이름(인수)  //사용
```

- 앞에 `@Mixin`을 쓰고 이름을 정해준 후, 중괄호 `{ }` 안에 중복되는 코드를 넣어줍니다.
- `@include`를 사용하여 스타일 하고자 하는 요소에 포함 시키면 됩니다.
- mixin은 파일을 만들어서 import하여 사용하거나, mixin을 사용할 파일 내에서 선언 후 사용할 수 있습니다. 이때, 여러 개의 mixin을 만들어 사용한다면, `_mixins.scss` 파일을 만들어서 관리합니다.

CSS

```
.card{
  display : flex;
  justify-content : center;
  align-items : center;
  background : gray;
}

.aside {
  display : flex;
  justify-content : center;
  align-items : center;
  background : white;
}
/*.card와 .aside 클래스 선택자의 적용한 스타일이 겹침*/
```

```scss
// scss를 사용

@mixin center-xy{
  display: flex;
  justify-content : center;
  align-items : center;
}

.card{
  @include center-xy; // 정의한 center-xy mixin을 사용하여 코드 한줄이면 끝!
}

.aside{
  @include center-xy;
}
```

⚠️ 반복하는 모든 코드를 하나의 mixin에 몰아넣어서 사용하는 건 바른 mixin 사용법이 아닙니다. 위에 코드처럼 스타일별로 나누어서 mixin을 만듭니다. 서로 연관 있는 스타일 속성끼리 묶어서 만들어야 좀 더 사용성이 높은 mixin을 만들 수 있습니다.

3. Arguments

1) 인수(Arguments)

mixin 이름 뒤에 인수를 넣어서 사용할 수 있으며, 일반 언어와 마찬가지로 매개변수와 인수의 개수가 같아야 합니다. mixin에 매개변수를 사용하면, 능동적으로 스타일을 적용할 수 있습니다. mixin의 매개변수에 들어가는 인자들의 값에 따라 형태는 같지만 스타일이 조금씩 달라지기 때문입니다.

```scss
@mixin image-style($ul, $size, $repeat, $positionX, $positionY) {
  background-image: url($u);
  background-size: $size;
  background-repeat: $repeat;
  background-position: $positionX $positionY;
}
//background관련 스타일 코드가 들어있다.
//mixin의 인수에 따라 조금씩 다른 스타일링이 가능하다.

.profile {
  @include image-style("./assets/user.jpg", cover, no-repeat, center, center);
}

.box-one {
  @include image-style(url("/images/poster1.svg"), cover, no-repeat, 40%, 50%);
}
```

```
.profile {
  background-image: url("./assets/user.jpg");
  background-size: cover;
  background-repeat: no-repeat;
  background-position: center center;
}

.box-one {
  background-image: url(url("/images/poster1.svg"));
  background-size: cover;
  background-repeat: no-repeat;
  background-position: 40% 50%;
}
```

2) 기본값 설정

기본값을 설정하여 매개변수에 값이 들어오지 않을 때 기본으로 설정한 값을 사용할 수 있도록 해줍니다.

```scss
// 위에 코드를 가져와서 기본값을 설정해주었다.
@mixin image-style($ul, $size, $repeat, $positionX : 50%, $positionY : 50%) {
  background-image: url($ul);
  background-size: $size;
  background-repeat: $repeat;
  background-position: $positionX $positionY;
}

.profile {
  @include image-style("./assets/user.jpg", cover, no-repeat);
}
// profile과 .box-one은 서로 관계가 없지만, 코드가 중복되기때문에 mixin을 만들어서
// 각 요소에서 사용합니다.
```

```
.profile {
  background-image: url("./assets/user.jpg");
  background-size: cover;
  background-repeat: no-repeat;
  background-position: 50% 50%;
}
```

4. Content

@content를 사용하면 원하는 부분에 스타일을 추가하여 전달할 수 있습니다.

```
// 정의하는 곳에서
@mixin sample{
  display: flex;
  justify-content : center;
  align-items : center;
  @content
}
```

```
// 사용하는 곳에서 (추가로 스타일링을 할 수 있음)
@include sample{
    color:white;
};
```

5. 전달인자 없는 믹스인

믹스인은 매개변수를 가지지 않더라도 만들 수 있습니다. 전달인자가 없는 믹스인에서는 `@ inlcude` 키워드에다가 믹스인 이름만 넣어주면 됩니다. 따로 괄호를 추가하지 않습니다.

```
a {
  @include text-style;
}
```

6. Example

예시를 보면서 다시 한번 확인해봅시다.

```scss
// Scss
// box의 사이즈를 정해주는 mixin
@mixin size($width, $height, $padding){
  width : $width;
  height : $height;
  padding : $padding;
}

article{
  @include size(100%, 65px, 10px 20px);
}
```

CSS

```css
/*CSS*/

article {
  width: 100%;
  height: 65px;
  padding: 10px 20px;
}
```

7. 전체 실습 코드

003. Mixin.zip 1.4KB

Ch5. Extend

1. Extend

Extend는 연관 있는 요소들끼리 스타일 코드가 중복된 경우에 사용됩니다. 이미 스타일이 작성된 선택자의 클래스를 extend하거나 , `%` 를 사용해서 따로 스타일을 정의한 후 extend하여 원하는 선택자에게 스타일을 적용해 줄 수 있습니다.

예를 들어 aside안에 존재하는 리스트에 요소들을 클릭할 때마다 보여주는 드롭메뉴와 프로필을 클릭했을 때, 프로필과 관련된 메뉴를 보여주는 드롭메뉴가 있다고 합시다. 안에 들어있는 내용들은 다르지만 드롭메뉴라는 연관성을 가진 경우입니다.

Sass를 사용하지 않고 스타일을 적용한다면 새로운 클래스를 정해서 중복되는 스타일을 넣어주거나, 각각 동일한 코드를 적어서 요소에 스타일을 적용할 것 입니다. 하지만 Sass의 Extend를 사용하면 중복된 코드를 `@extend` 를 사용하여 더 단순한 코드를 작성할 수 있습니다.

- mixin는 (관계 없는) 선택자에서 조금 다른 스타일을 적용할 때 사용
- extend는 관계 있는 선택자들의 동일한 소스코드 적용시 사용

2. extend 하는 2가지 방법

2-1. class이름 가져오기

기존에 작성한 클래스 내에 코드를 가져올 수 있습니다. `@extend` 에 클래스 명을 함께 적으면, 클래스에 있는 코드 전체가 extend 됩니다.

```scss
// Scss
.profile-user {
  background-image: url("../assets/user.jpg");
  background-size: cover;
  background-position : 50% 50%;
  border-radius: 50%;
  width : 50px;
  height : 50px;
}

.comment-user {
  @extend .profile-user;
}
```

```css
/*CSS*/
.profile-user,
.comment-user {
  background-image: url("../assets/user.jpg");
  background-size: cover;
  background-position : 50% 50%;
  border-radius: 50%;
  width : 50px;
  height : 50px;
}
```

`.profile-user` 클래스를 가진 로그인 프로필 user 이미지 박스가 `.comment-user`가 댓글을 작성할 때 user임을 나타내주는 이미지 박스와 중복되는 경우입니다. `.profile-user` 코드 전체를 `.comment-user` 에게 extend 해줍니다.

⚠️ class명을 extend 하는 경우 다중 선택자 클래스를 사용할 수 없습니다. (ex. `.box .container` , `.box1 + .box2`)

2-2. %placeholder

`%` 로 선택자를 만듭니다. `@extend`를 사용해서 앞서 `%placeholder` 스타일 블럭을 불러오면 됩니다. 그리고 %선택자는 CSS로 컴파일되지 않습니다.

```scss
// Scss
%base-button {
  width: 133px;
  height: 44px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 14px;
  border-radius: 10px;
}
```

```
.follow-button {
  @extend %base-button;
  background-color: #ffffff;
  color: #ff375f;
  border: 3px solid #ff375f;
}

.message-button {
  @extend %base-button;
  background-color: #ff375f;
  color: white;
}
```

CSS

```
/*CSS*/
.follow-button,
.message-button {
  width: 133px;
  height: 44px;
  display: flex;
  justify-content: center;
  align-items: center;
  font-size: 14px;
  border-radius: 10px;
}

.follow-button {
  background-color: #ffffff;
  color: #ff375f;
  border: 3px solid #ff375f;
}

.message-button {
  background-color: #ff375f;
  color: white;
}
```

⚠️Extend를 사용할 때, 클래스보다 `%`를 사용하기를 권장합니다.

3. 예시 : 모달

웹페이지 안에서 생성되는 여러가지 모달에 대한 코드입니다.

`%`를 사용하여 스타일 블럭을 만들고, 각각의 모달에 스타일을 적용해줍니다.

Scss

```scss
// Scss
%modal {
  position: relative;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: #fff;
  border-radius: 6px;
}

.login-modal {
  @extend %modal;
  width: 272px;
  height: 405px;
  padding: 10px 20px;
}

.event-modal {
  @extend %modal;
  width: 340px;
  height: 160px;
  padding: 18px;
}
```

```css
/*CSS*/
.login-modal,
.event-modal {
  position: relative;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: #fff;
  border-radius: 6px;
}

.login-modal {
  width: 272px;
  height: 405px;
  padding: 10px 20px;
}
```

```
.event-modal {
  width: 340px;
  height: 160px;
  padding: 18px;
}
```

4. 예시 : 포토 링크 박스

두 개의 div는 포스팅 링크로 이동하는 기능은 같지만, 조금 다른 형태의 스타일이 적용되어 있습니다. 공통된 코드를 box-frame으로 묶어주었습니다.


```
// Scss
%box-frame {
  border: 2px solid #bb6bd9;
  border-radius: 15px;
  width: 574px;
  height: 310px;
}

.phoster-span {
  @extend %box-frame;
  span {
    display: inline-block;
    border-top: 2px solid #bb6bd9;
    padding: 16px 0 17px;
    text-align: center;
  }
}
```

```
.phoster-none {
  @extend %box-frame;
}
```

```
/*CSS*/
.phoster-span,
.phoster-none {
  border: 2px solid #bb6bd9;
  border-radius: 15px;
  width: 574px;
  height: 310px;
}

.phoster-span span {
  display: inline-block;
  border-top: 2px solid #bb6bd9;
  padding: 16px 0 17px;
  text-align: center;
}
```

5. 전체 실습 코드

004. Extend.zip 1.6KB

Ch6. 조건문과 반복문, 함수

1. 조건문

조건에 따라 스타일을 주고자 할 때, if와 else문을 사용하는데요. if문 하나만 사용하는 경우도 있으며, 뒤에 나오는 else문, else if문과도 함께 사용하는 경우도 있습니다.

1-1. @if

@if에 괄호 없이 true나 false를 반환할 수 있는 조건문을 작성하시면 됩니다. (조건을 작성할 때 괄호를 사용할 수 있지만 일반적으로 괄호는 생략합니다) 조건에는 논리연산자 and, or, not을 사용하고, if문의 조건이 true일 때만 `{ }` 괄호 안에 있는 코드가 실행됩니다.

```
@if (조건) {
  // 조건이 참일 때 실행될 구문
}
```

Sass Copy Caption

```sass
// if문 예시 작성
// Sass 공식문서
// circle이 false면 사각형을, true이면 원형으로 스타일함
@mixin avatar($size, $circle: false) {
  width: $size;
  height: $size;

  @if $circle {
    border-radius: $size / 2;
  }
}
.square-av {
  @include avatar(100px, $circle: false);
}
.circle-av {
  @include avatar(100px, $circle: true);
}
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

```
/*CSS*/
.square-av {
  width: 100px;
  height: 100px;
}

.circle-av {
  width: 100px;
  height: 100px;
  border-radius: 50px;
}
```

1-2. @else

@else문은 if 문처럼 조건문이 필요하지는 않으며 if문에서 걸었던 조건이 false가 나오면 else문의 코드가 실행됩니다

```
@if (조건) {
  // 조건이 참일 때 실행될 구문
} @else{
  // if문의 조건이 거짓일 때 실행될 구문
}
```

```
// Scss - else문
// Sass 공식문서
// ture이면 밝은 색을, false면 어두운 색을 스타일함
$light-background: #f2ece4;
$light-text: #036;
$dark-background: #6b717f;
$dark-text: #d2e1dd;

@mixin theme-colors($light-theme: true) {
  @if $light-theme {
    background-color: $light-background;
    color: $light-text;
  } @else {
    background-color: $dark-background;
    color: $dark-text;
  }
}
```

```
.banner {
  @include theme-colors($light-theme: true);
  body.dark & {
    @include theme-colors($light-theme: false);
  }
}
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

```
/*CSS*/
.sidebar {
  float: left;
  margin-left: 64px;
}
```

1-3. @else if

@else if문은 if문으로 부족할 때, 즉 여러 개의 조건문을 사용해야 할 때 사용합니다. @else if문에도 true나 false를 반환하는 조건문을 작성합니다. if문의 조건에서 false가 나왔을 때, `else if` 문으로 넘어가서 조건에 대해 true인지 false인지 판단합니다. true인 경우 `else if` 문 내의 코드를 실행하고, false 라면 그 다음 조건문(`else` or `else if`)로 넘어가게 됩니다.

```
if (조건){
  // 조건이 참일 때 실행될 구문
} @else if(조건){
  // else if 조건이 참일 때 실행될 구문
} @else{
  // 위에 모든 조건이 거짓일 때 실행될 구문
}
```

Sass

```sass
// Scss - if, else if, else문
// Sass 공식문서
// 조건에 해당하는 방향에 맞춰서 border-bottom 컬러를 스타일함
@mixin triangle($size, $color, $direction) {
  height: 0;
  width: 0;

  border-color: transparent;
  border-style: solid;
  border-width: ($size/2);


  @if $direction == up {
    border-bottom-color: $color;
  } @else if $direction == right {
    border-left-color: $color;
  } @else if $direction == down {
    border-top-color: $color;
  } @else if $direction == left {
    border-right-color: $color;
  } @else {
    @error "Unknown direction #{$direction}.";
  }
}

.next {
  @include triangle(5px, black, right);
}
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

CSS

```css
/*CSS*/
.next {
  height: 0;
  width: 0;
  border-color: transparent;
  border-style: solid;
  border-width: 2.5px;
  border-left-color: black;
}
```

2. 반복문

2-1. @for

```
for ($변수) from (시작) through(끝){
  // 반복할 내용
}
```

@for은 정의한 횟수만큼 코드 실행을 반복합니다. @for문에 from(시작 : 이상) - through(끝 : 이하)를 사용하여 어디서 시작해서 어디서 끝날 지를 알려줍니다. `nth-` 선택자를 사용하는 경우 유용하게 사용할 수 있습니다.

Scss

```scss
// Scss - for문
// for문을 이용해 nth-선택자에게 각각의 image를 배경에 넣어준다.
@for $i from 1 through 5 {
  .photo-box:nth-child(#{$i}) {
    background-image: url("../assets/phoster#{$i}.png");
  }
}
// 범위 1이상 5이하
// for문에서 1부터 5까지 반복하라는 의미입니다. 총 5번 반복되면서 코드가 실행된다.
// 만약 from 3 throught 8 이라면 3에서 8까지 6번 실행된다.
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

```css
/*CSS*/
.photo-box:nth-child(1) {
  background-image: url("../assets/phoster1.png");
}

.photo-box:nth-child(2) {
  background-image: url("../assets/phoster2.png");
}

.photo-box:nth-child(3) {
  background-image: url("../assets/phoster3.png");
}
```

```
.photo-box:nth-child(3) {
  background-image: url("../assets/phoster3.png");
}

.photo-box:nth-child(4) {
  background-image: url("../assets/phoster4.png");
}

.photo-box:nth-child(5) {
  background-image: url("../assets/phoster5.png");
}
```

2-2. @each

each문은 lists나 맵의 각각의 요소마다 코드를 실행해서 스타일을 적용할 수 있게 합니다.

```
@each ($변수) in (리스트나 맵){
  // 반복할 내용
}
```

Sass

```
// Sass - each문
// color-palette 리스트에 들어있는 색상을 each문을 사용하여 background에 색상값을 넣어준다.
$color-palette: #dad5d2 #3a3532 #375945 #5b8767 #a6c198 #dbdfc8;

@each $color in $color-palette {
  $i: index($color-palette, $color); //index는 list의 내장함수
  .color-circle:nth-child(#{$i}) {
    background: $color;
    width: 20px;
    height: 20px;
    border-radius: 50%;
  }
}
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

CSS

```css
/*CSS*/
.color-circle:nth-child(1) {
  background: #dad5d2;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}

.color-circle:nth-child(2) {
  background: #3a3532;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}

.color-circle:nth-child(3) {
  background: #375945;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}

.color-circle:nth-child(4) {
  background: #5b8767;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}

.color-circle:nth-child(5) {
  background: #a6c198;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}
.color-circle:nth-child(6) {
  background: #dbdfc8;
  width: 20px;
  height: 20px;
  border-radius: 50%;
}
```

2-3. @while

```
@while 조건 {
  // 반복할 내용
}
```

특정 조건에 충족될 때까지 코드를 무한 반복하며, 조건을 만날 때 while문을 빠져나오게 됩니다. 참고로, while문은 빠져나오는 조건을 만드는 경우가 거의 없어서 잘 사용하지 않습니다.

```
// Scss - while문
// Sass 공식문서
// value값이 base보다 작을 때까지 일정한 값으로 계속 나눠준다.
@function scale-below($value, $base, $ratio: 1.618) {
  @while $value > $base {
    $value: ($value/$ratio);
  }
  @return $value;
}

$normal-font-size: 16px;
.sup {
  font-size: scale-below(20px, 16px);
}
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

```
CSS
/*CSS*/
.sup {
  font-size: 12.36094px;
}
```

3. function

3-1. function

`@function` 키워드를 사용하여 함수를 생성하고 **함수이름( )** 형태로 함수를 호출하고 실행합니다. 함수 안에서는 `@return` 이용해 값을 반환합니다. 함수는 Mixin과 비슷하지만 mixin은 스타일 코드를 반환하고 function은 `@return` 키워드를 사용해서 값 자체를 반환한다는 차이가 있습니다.

```
@function 함수이름($매개변수) {
  // 실행 코드
  @return 값
}
```

Sass

```
// Scss - function
// Sass 공식문서
// 거듭제곱을 구하는 함수
@function pow($base, $exponent) {
  $result: 1;
  @for $_ from 1 through $exponent {
    $result: $result * $base;
  }
  @return $result;
}

.sidebar {
  float: left;
  margin-left: pow(4, 3) * 1px;
}
```

※ Scss 파일 코드를 작성하여 실행한 후, css와 비교해 보세요!

```
/*CSS*/
.sidebar {
  float: left;
  margin-left: 64px;
}
```

3-2. 내장함수

Sass에는 기본적으로 내장되어 있는 함수가 있습니다. 앞에서 lists와 maps를 설명하면서 관련 내장 함수를 잠깐 확인했는데요. 그 외에 내장함수 중 몇 개만 더 살펴보겠습니다.

1) 색상 함수

- `lighten(color, amount)` : 기존 색상의 밝기를 높입니다.(0%-100% 사이의 값)
- `darken(color, amount)` : 기존 색상의 밝기를 낮춥니다.(0%-100% 사이의 값)
- `mix(color1, color2, weight)` : 2개의 색상을 섞어서 새로운 색상을 만듭니다.

2) 숫자 함수

- `max(number, ..)` : 괄호에 넣은 값 중에 가장 큰 수를 반환합니다.
- `min(number, ..)` : 괄호에 넣은 값 중에 가장 작은 수를 반환합니다.
- `parcentage(number)` : 퍼센트로 숫자를 바꿔줍니다.
- `comparable(num1,num2)` : 숫자1과 숫자2가 비교 가능한지 확인 후 true,false 값을 반환합니다.

3) 문자 함수

- `srt-insert(string,insert,index)` : 문자열에 원하는 위치(index)에 문자를 넣은 후(insert), 새로운 문자열을 반환합니다.
- `str-index(string,substring)` : 문자열에서 해당 문자의 index 값을 반환합니다.
- `to-upper-case(string)` : 문자열 전부를 대문자로 바꿔줍니다.
- `to-lower-case(string)` : 문자열 전부를 소문자로 바꿔줍니다.

4) 확인 함수

- `unit(number)` : 숫자의 단위를 반환해 줍니다.
- `unitless(number)` : 단위를 가지고 있는지 판단하여 true,false 값을 반환합니다.
- `variable-exists(name)` : 변수가 현재 범위에 존재하는지 판단하여 true,false 값을 반환합니다. 이 함수의 인수는 `$` 없이 사용합니다.

40 minutes

Basic Sass