# CSS 수퍼파워


로

# 디자인하라

**예제 소스 다운로드**

http://www.roadbook.co.kr/163

**질의 응답 사이트**

http://roadbook.zerois.net


CSS 수퍼파워 **Sass로 디자인하라**

**지은이** 양용석 **1판 1쇄 발행일** 2016년 8월 22일 **펴낸이** 임성춘 **펴낸곳** 로드북 **편집** 장미경
**디자인** 이호용(표지), 심용희(본문) **주소** 서울시 관악구 신림로 29길 8 101-901호
**출판 등록** 제 2011-21호(2011년 3월 22일) **전화** 02)874-7883 **팩스** 02)6280-6901
**정가** 23,000원 **ISBN** 978-89-97924-23-3 93000



**이메일** chief@roadbook.co.kr **블로그** www.roadbook.co.kr

# 서문

SASS(사쓰), 사실 한국에서 사쓰라고 하면 좋아할 사람이 아무도 없습니다. 발음이 중증 급성호흡기증후군을 의미하는 SARS(사스)와 비슷하고, 이 사스로 인해 전세계가 많은 피해를 봤기 때문입니다. 하지만 웹 개발 특히 프론트 엔드 분야에서는 이 SASS가 현재 대세로 떠오르고 있습니다. SASS는 Syntactically Awesome StyleSheets의 약자로 "문장 구성적으로 끝내주는 스타일시트"라고 표현하고 있는데, 실제 필자도 SASS에 익숙하면 익숙해질수록 이 표현이 아주 피부에 와닿습니다. 필자가 몇 년 전 웹 표준 관련 책을 처음 집필할 당시에 HTML5와 CSS3로 웹사이트가 구조와 표현으로 분리되는 큰 변화를 겪게 됩니다. 이때 CSS의 역할이 중요해지면서 CSS의 속성이 엄청나게 확장되었고 시간이 지남에 따라 CSS를 효과적으로 제어하는 그 무언가가 필요해지는 시점이 있었습니다. 이때 혜성처럼 등장한 기술이 SASS입니다. 특히 SASS는 이와 유사한 LESS라는 CSS pre-processor와 병행하면서 개발자 사이에서 사용되다 현재 거의 모든 웹 프레임워크들이 SASS를 적극적으로 밀고 있습니다.

필자가 가장 좋아하는 웹 개발 프레임워크인 부트스트랩도 3.X 버전에서는 SASS와 LESS를 동시에 지원하였는데, 4.0버전부터는 SASS만 지원한다고 합니다. 부트스트랩뿐만 아니라 다른 모든 프레임워크들 또한 SASS를 적극 밀고 있으며, github이나 기타 웹 관련 소스를 제공하는 개발자들은 CSS 소스뿐만 아니라 SASS 소스를 병행해서 제공하고 있습니다.

한국 또한 SASS를 이용해서 개발하는 개발자들이 늘고 있는 것으로 알고 있습니다. 여러 경로를 통해 소식을 듣고 있거나 현재 실무에 적용하는 분들도 있을 것입니다. 이 책은 필자가 쓰는 여섯 번째 책으로, 한국의 웹 개발자들이 보다 빠르고, 편리하며, 유지 보수에서도 매우 유용한 SASS가 많이 사용되기를 바라면서 집필하게 되었습니다.

SASS를 처음 사용할 때는 매우 생소한 느낌이었지만, 현재는 SASS를 제외하고 웹 개발을 할 수 없을 정도로 손에 매우 익어버려, SASS를 사용하지 않으면 CSS 구문을 제어할 수 없을 정도가 되어 버린 느낌입니다. 아무쪼록 이 책을 통해 국내의 많은 웹 프론트 엔드 개발자들이 SASS의 매력에 빠져들었으면 좋겠습니다.



마지막으로 이 책을 쓰면서 많은 일들이 필자의 주변에 일어 났었습니다. 서울에 있는 열정 가득한 젊은 친구들과, 비록 끝이 아쉽긴 하지만, 다시 한번 필자의 감각이 살아있다는 느낌이 들 수 있게끔 해준 스타트 업도 같이 해보고, 학교를 졸업한 지 오래되었지만, 인생의 황금기에 만난 최고의 친구들인 고등학교 동창들과 많은 만남을 가지면서 인생의 기쁨이란 것이 멀리 있지 않고 항상 내 주변에 있다는 것을 느끼게 되었습니다.

이 책을 쓰면서 항상 옆에서 용기를 주는 큰딸 유지, 둘째 연수, 막내 혁준 그리고 사랑하는 내 와이프에게 감사하며, 특히 저를 세상에 태어나게 해주신 부모님에겐 무한 사랑을 드립니다. 또한 이 책을 쓰는 중간중간 저에게 기쁨과 용기 그리고 우정을 보여준 제주제일고 31회 삼삼회 친구들에게 정말 고맙다는 말을 전하고 싶습니다. 끝으로 이 책을 출간하기까지 고생해 주신 로드북 관계자분들에게 감사의 말을 전합니다.

2016년 8월 제주에서
양용석

# 목차


## 1부 CSS 수퍼파워 – SASS

### 1장 SASS 사용을 위한 준비 단계_011

1.1 PC에 SASS 설치하기_012
1.2 SASS의 기초_014

### 2장 SASS 레퍼런스_039

2.1 개요_040
2.2 CSS 확장(CSS Extensions)_041
2.3 주석 처리_048
2.4 SassScript_048
2.5 변수: $ 기호를 사용함_049
2.6 데이터 타입_051
2.7 연산(Operations)_053
2.8 @ 규칙과 지시어(@–Rules and Directives)_058
2.9 제어문 및 표현식(Control Directives & Expressions)_068
2.10 Mixin(정말 중요함!!)_074

### 3장 SASS 프레임워크: 컴퍼스(Compass)_083

3.1 컴퍼스 설치_084
3.2 컴퍼스 기초_085
3.3 컴퍼스 실전 예제_100

### 4장 버본(Bourbon)_115

4.1 버본 개발 환경 설정_116
4.2 믹스인의 속성들_118

## 2부 SASS(SCSS)를 이용한 실전 웹사이트 개발

### 5장 기본 레이아웃 및 프론트 페이지 작업_147

5.1 웹사이트 디자인 개요_148
5.2 색상과 폰트 정의_150
5.3 믹스인 변수 처리_152
5.4 페이지 전체 배경 이미지 처리_154
5.5 푸터에 대한 처리_155
5.6 페이지 전체 배경 이미지 다시 한번 살펴보기_157
5.7 내비게이션 디자인_159
5.8 푸터 디자인_166
5.9 내부 콘텐츠 처리_170
5.10 article에 Sass 코드 추가하기_171

### 6장 서비스 소개 및 리스트 페이지 제작_175

6.1 헤더와 푸터의 분리_176
6.2 서비스 소개 페이지 개발_179
6.3 리스트 페이지 제작_187

### 7장 콘텐츠 디테일 페이지 제작_197

7.1 디테일 페이지와 Sass 파일 작성_198
7.2 지도가 들어가는 부분 작성_202
7.3 Sass 파일 설명_204

### 8장 추천 요청하기 페이지 제작_211

8.1 페이지 전체 배경 이미지 처리_212
8.2 Sass 파일 만들기_214

### 9장 반응형 사이트로 만들기_219

9.1 부트스트랩으로 만든 디자인의 부족함_220
9.2 완벽한 반응형 웹 디자인으로 만들기_222

찾아보기_229

Sass로 디자인하라

# 1부

# CSS 수퍼파워 - SASS

사스는 Syntactically Awesome Style Sheet라는 단어의 줄임말이며, 한글로 번역하자면 "문장 구성적으로 끝내주는 스타일 시트"라는 의미를 지닌다고 볼 수 있습니다.

여기서 "문장 구성적으로 끝내준다?"라는 의미는 사스를 학습하게 되면 알게 될 것입니다. 웹사이트를 디자인할 때 가장 필수 요소 중 하나가 스타일시트입니다. 그런데, 이 스타일시트는 아주 단순하지만 사이트가 커지게 되면, 마치 에둘러서 표현하면 '중노동' 느낌이 들며, 심지어 코드 길이가 1000줄이 넘어가면 뭐가 뭔지 알 수 없는 경우도 생겨 버립니다. 하지만 사스를 이용하면 스타일시트 작업이 마치 프로그래밍하는 것처럼 체계적이고, 구조적으로 끝내주게 개발할 수가 있게 되는 것입니다.

말로 설명을 아무리 해봐도 직접 사용해 보는 것보다 이해가 빠를 순 없습니다. 이 책의 1부에서는 사스에 대한 모든 설명을 사스 공식 사이트에서 제시한 방법을 기준으로 필자 나름대로 최대한 자세히 설명해 드리겠습니다.

1장

# SASS 사용을 위한 준비 단계

1장에서는 SASS를 사용하기 위한 준비 단계에 대해서 학습합니다. SASS는 일종의 컴퓨터 언어와 같은 성질을 지니고 있습니다. 무슨 말인가 하면, SASS 파일은 단독으로 웹사이트에서 사용할 수 없습니다. 예를 들어, 컴퓨터 언어인 C의 경우 프로그래머가 소스코드를 입력하고 '.c'라는 확장자로 저장합니다. .c 확장자는 컴파일이란 과정을 통해 '.exe' 란 파일로 변환되어야만 사용할 수 있습니다. 이처럼 SASS 파일도 '.sass' 또는 '.scss' 란 파일이 컴파일 과정을 통해서 '.css' 로 변환되어야 사용할 수 있습니다.

여기서 확장자가 '.sass' 또는 '.scss' 라고 언급을 했는데, SASS는 두 가지 형태로 사용됩니다. 이 부분은 뒤에서 설명하록 하겠습니다. SASS를 가장 편리하게 사용하는 방법은 어플리케이션을 이용하여 컴파일하는 방법입니다. 커맨드라인 방식으로 컴파일하는 방법 또한 존재하는데, 명령어에 익숙해야 한다는 불편함이 있습니다.

## 1.1 PC에 SASS 설치하기

SASS를 PC에 설치하는 방법은 앞서 설명한 것처럼 어플리케이션 방식과 커맨드라인 방식이 있다고 했습니다. 반복하지만, SASS는 컴퓨터 프로그래밍과 비슷하기 때문에, 컴파일러 없이는 작동하지 않습니다. 일반적인 HTML 파일과 CSS 파일은 메모장과 같은 가장 간단한 에디터를 사용하여 웹사이트를 개발할 수 있지만, SASS는 컴파일러 없이는 무용지물이기 때문에 반드시 어플리케이션 방식을 사용하든지, 커맨드라인 방식을 사용하여 컴파일 과정을 거쳐야 합니다.

먼저 SASS를 컴파일 할 수 있는 어플리케이션은 다음과 같습니다.

| 종류 | 유/무료 | OS별 버전 |
|---|---|---|
| CodeKit | 유료 | 맥 버전만 있음 |
| Compass.app | 유료 및 오픈소스 | 맥, 윈도우즈, 리눅스 버전 |
| Ghostlab | 유료 | 맥 및 윈도우즈 버전 |
| Hammer | 유료 | 맥 버전만 있음 |
| Koala * | 오픈 소스 | 맥, 윈도우즈, 리눅스 버전 |
| LiveReload | 유료 및 오픈 소스 | 맥 및 윈도우즈 버전 |
| Prepros | 유료 | 맥, 윈도우즈, 리눅스 버전 |
| Scout | 오픈 소스 | 맥 및 윈도우즈 버전 |

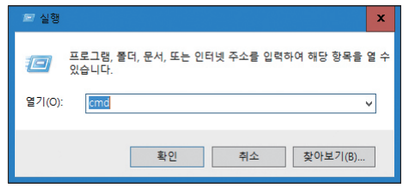
위의 어플리케이션에 대한 정보는 http://sass-lang.com/install에서 자세히 확인할 수 있습니다. 사용하는 OS에 맞춰서 사용하면 됩니다. 이 책에서는 Koala*라는 어플리케이션을 사용하여 모든 학습을 진행하도록 하겠습니다. Koala는 필자가 사용해 본 어플리케이션 중 가장 편리하고 간편하며 무료이고 모든 플랫폼에서 작동하기 때문입니다.

이런 어플리케이션을 사용하지 않고 커맨드라인을 통해서 SASS 파일을 CSS로 변환하는 방법 또한 존재합니다. 먼저 SASS 파일을 컴파일하기 위해선 사용하는 OS에 "Ruby(루비)"라는 프로그래밍 언어가 설치되어 있어야 합니다. 리눅스와 맥은 기본적으로 루비라는 언어가 설치되어 있어 바로 SASS 컴파일러를 설치하면 되는데, 윈도우즈는 별도로 http://rubyinstaller.org/ 여기로 가서 해당 파일을 다운로드 한 후 설치하면 됩니다(윈도우즈에서는 루비를 설치한 후 다시 sass 모듈을 설치해야 하기 때문에 어플리케이션 방식이 편리합니다).

* Koala 사이트의 주소는 http://koala-app.com/입니다.

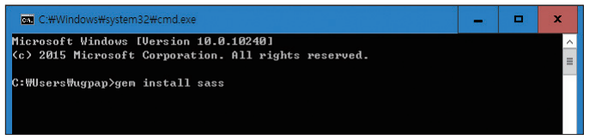


루비가 설치되어 있다면, 이제 윈도우에서는 **[윈도우 키 + R]**을 누른 후 실행 창에서 "cmd"라고 입력한 후 엔터키를 누릅니다.



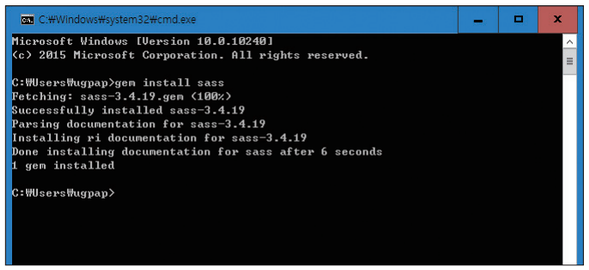

[그림 1-1] [윈도우 키 + R]을 누른 후 "cmd" 입력 화면

이제 커멘드 창에 `gem install sass`라고 입력하면 sass가 사용자의 PC에 설치됩니다. 맥과 리눅스에서는 기본으로 루비가 설치되어 있기 때문에 터미널만 열고 `gem install sass`라고 입력하고 엔터 키를 누르면 됩니다. 만약 맥이나 리눅스에서 이렇게 입력을 했는데, 설치가 되지 않거나, 에러가 발생하는 경우에는 관리자 계정으로 설치하면 됩니다. `sudo gem install sass` 이렇게 터미널에 입력하면 됩니다.




[그림 1-2] 커맨드 창에 gem install sass 입력 모습

[그림 1-2]에서와 같이 `gem install sass`를 입력한 후 엔터키를 누르면 [그림 1-3]과 같이 1개의 gem(보석 - 루비의 구성 요소 중 하나라는 의미)이 설치되었다는 메시지가 나타납니다.




[그림 1-3] sass 설치 모습

이제 커맨드 창에 sass -v라고 입력해보면 현재 설치된 버전이 표시될 것입니다.


[그림 1-4] sass 버전 확인

Sass 3.4.19 (Selective Steve)가 설치된 것을 확인하면 설치는 끝난 것입니다. SASS 버전은 SASS가 업그레이드 되면서 다를 수 있습니다. 이 책을 쓰는 시점에서는 3.4.19 버전이 가장 최신 버전입니다. 커맨드라인 방식을 간략하게 살펴보았는데, 필자가 추천하는 방식은 Koala라는 어플리케이션을 사용하는 방식입니다.

## 1.2 SASS의 기초

앞에서는 SASS를 작성하기 위한 아주 기초적인 컴파일러를 설치하였습니다.

이제 직접 SASS를 이용하여 CSS를 작성하는 방법을 간단하게 학습해보도록 하겠습니다.

우선 SASS는 두 종류의 파일이 존재합니다. 확장자가 .sass인 SASS 파일과 확장자가 .scss인 SCSS 파일이 있습니다. "어~~ 잠시만요!! SASS가 두 종류가 있다는 말이 뭐죠?"라는 질문이 나올 수도 있는데요. SASS와 SCSS는 동일한 파일입니다. 하지만 내부적으로 약간의 차이가 있습니다. 먼저 SASS는 Syntactically Awesome Style Sheet의 준말이며, SCSS는 Sassy Cascading Style Sheet의 준말입니다. 어떻게 보면 말 장난 같은 센스를 발휘하고 있는데, SCSS는 "멋진 CSS"라고 해석할 수 있겠습니다.

의미는 그렇다고 하지만, 내부적으로 다르다? 이게 또 무슨 말일까요? 우선 두 파일의 차이점은 중괄호({ })와 세미콜론(;)의 유무라고 할 수 있습니다. SASS 파일은 중괄호와 세미콜론이 존재하지 않고, SCSS는 존재합니다. 컴퓨터 프로그래밍에 관심이 있는 독자라면 알 수 있을 텐데요. 파이썬이라는 언어와 C 언어의 차이라고 할 수 있습니다. 파이썬이라는 언어는 중괄호({ })와 세미콜론(;)이 없지만, C 언어는 중괄호({ })와 세미콜론(;)이 있습니다.

SASS 공식 웹사이트에 나온 예제를 통해서 살펴 보겠습니다. 내용은 일단 건너뛰고 두 파일의 차이점만 보기 바랍니다.



우선 SCSS의 문법 구조입니다.

[코드 1-1] SCSS의 문법 구조

```
$font-stack: Helvetica, sans-serif;
$primary-color: #333;
body {
  font: 100% $font-stack;
  color: $primary-color;
}
```

이제 SASS의 문법 구조입니다.

[코드 1-2] SASS의 문법 구조

```
$font-stack: Helvetica, sans-serif
$primary-color: #333
body
  font: 100% $font-stack
  color: $primary-color
```

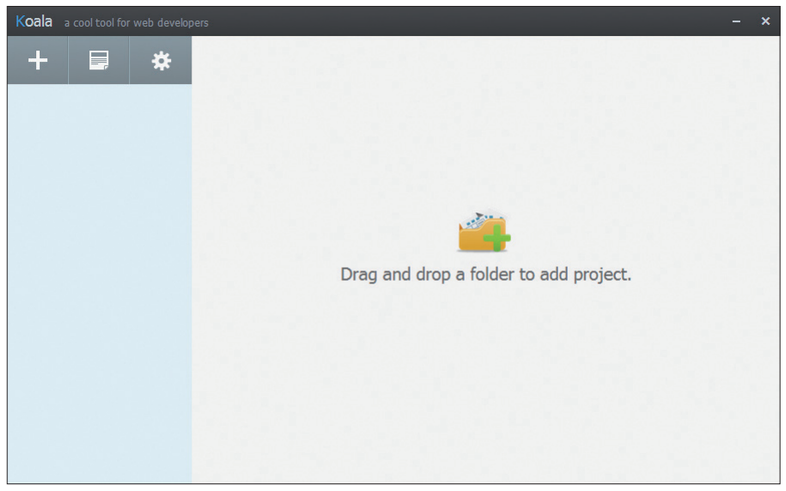
[코드 1-1]과 [코드 1-2]를 비교해 보면 중괄호와 세미콜론의 유무 이외에는 차이점이 없는 것을 볼 수 있습니다. 이렇게 SASS를 이용하여 CSS를 개발할 때는 두 가지 방식 중 편리한 방식을 사용하면 됩니다. 하지만 두 가지 방법을 혼용할 수는 없습니다. 따라서 하나의 방법을 선택하는 경우 반드시 한 가지 방법만 사용하기 바랍니다. 필자는 scss를 선호하기 때문에 이 책에서의 모든 코드는 scss 기반으로 작성할 것입니다. 두 가지 파일이 차이점은 확장자와 중괄호, 세미콜론의 사용 여부만 있고 결과물인 css 파일은 동일합니다.

SASS는 프로그래밍 언어와 유사하게 변수Variables를 사용할 수 있으며, 중첩Nesting 방식으로 코딩할 수 있습니다. 또한 파일을 파편화Partials해서 다른 파일에 불러올Import 수 있습니다. 또한 믹스인Mixins 방식으로 CSS3 속성 중 브라우저별로 속성값을 지정하는 경우(예를 들어 border-radius와 같은 CSS3의 속성 등) 간단하게 처리할 수 있습니다. 또한 확장extend과 상속Inheritance을 이용하여 동일한 속성값을 반복적으로 사용할 경우 편리하게 처리할 수 있습니다.

이제 앞서 설명한 Variables, Nesting, Partials/Import, Mixins, Extend/Inheritance에 대하여 예제를 이용하여 알아보겠습니다.

먼저 SASS 파일을 CSS 파일로 컴파일하기 위해 앞서 언급한 Koala라는 어플리케이션을 이용하여, 예제 파일들을 직접 하나씩 실행하는 방법을 알아보겠습니다.

Koala 어플리케이션을 설치하고 실행을 하면 [그림 1-5]와 같은 화면이 보일 겁니다.



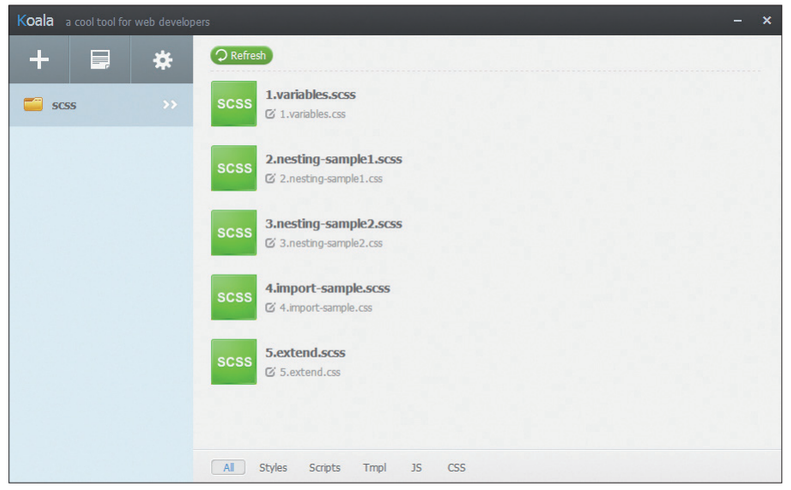

[그림 1-5] Koala 설치 후 실행 모습

이 책에서 제공하는 예제는 Sample이라는 폴더에 모든 파일이 저장되어 있습니다. 1장에서 사용되는 예제는 Sample/chapter1이라는 폴더에 있습니다. 이 폴더를 열어 보면 css, scss 그리고 imgs 이렇게 3개의 폴더가 있고 몇 개의 html 파일들이 있을 것입니다.

여기서 scss라는 폴더를 koala 어플리케이션의 왼쪽 파란 부분에 끌어 놓으면(Drag & Drop) [그림 1-6]과 같은 결과를 얻을 수 있습니다.

코알라는 해당 폴더를 끌어다 놓는 것만으로 자동으로 컴파일합니다. 따라서 현재 scss 폴더 내부에는 해당 scss 파일을 컴파일한 결과값들이 저장될 것입니다(제공된 예제 폴더에는 css 파일을 제거하였습니다).






[그림 1-6] Koala에 scss라는 폴더를 끌어 놓은 후 모습

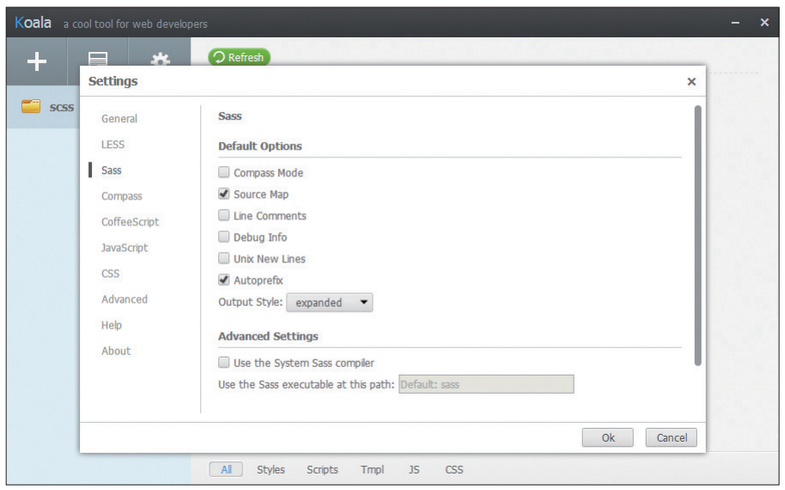
[그림 1-6] 왼쪽 부분을 보면 톱니바퀴 모양의 아이콘을 보실 수 있습니다. 이 부분이 환경설정 하는 부분인데, 그 부분을 클릭하여 Settings에서 Sass 관련 부분을 [그림 1-7]과 같이 해주기 바랍니다. 여기서 Output Style 부분은 실전 예제 작업을 하면서 좀 더 설명하도록 하고 현재는 그림과 같이 설정해주면 됩니다. 나머지 부분은 기본 설정 그대로 놔두면 됩니다.

**여기서 잠깐**

Sass 파일 컴파일시 에러가 나는 경우가 몇가지 있는데 가장 대표적인 원인은 다음과 같습니다.

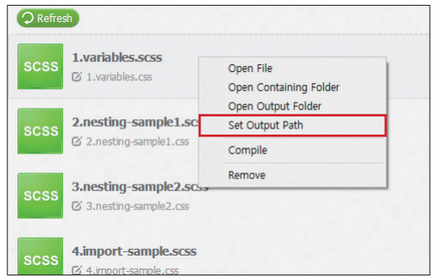
1. 주석에 한글이 들어가 있는 경우(일본어, 중국어 등도 동일)
2. 변수를 잘못 설정한 경우
3. Output 경로를 잘못 설정한 경우

이외에도 에러가 나는 경우는 다양합니다만, 컴파일시 에러가 발생하게 되면 해당 원인에 대한 에러 코드를 보여주기 때문에 그 원인을 제거하면 에러가 발생하지 않을 것입니다.



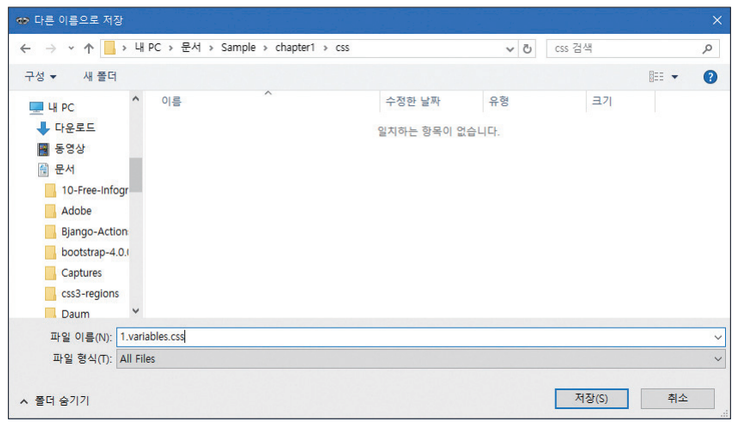

[그림 1-7] Koala 설정 모습

이제 마지막으로 SCSS로 된 파일의 output 경로를 설정해 주면 됩니다. SCSS 파일과 CSS 파일의 경로가 동일하더라도 상관은 없겠지만, 추후 관리를 위해서라면, 별도의 폴더에 분리해 주는 것이 좋습니다.




[그림 1-8.1] CSS 파일의 경로 설정 및 css 폴더에 파일 이름 설정






[그림 1-8.2] CSS 파일의 경로 설정 및 css 폴더에 파일 이름 설정

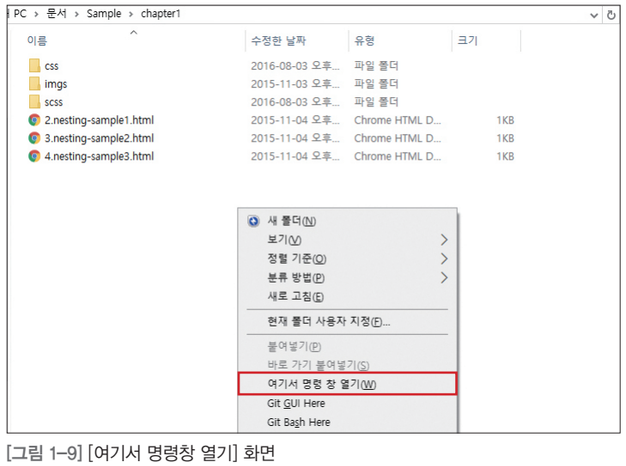
[그림 1-8.1]에서 set Output Path를 선택하면, 저장될 폴더와 파일명을 설정해줄 수 있습니다. 에제에서 scss 폴더에 있는 1.variables.scss 파일을 chapter1에 있는 css 폴더 내부에 1.variables.css라고 파일 이름과 경로를 지정하는 모습을 볼 수 있습니다. 만약 scss 파일이 여러 개가 있는 경우 각각의 파일에 대해서 경로를 설정해 주셔야 합니다.

**주의**

2부에서 실전 예제를 다룰 때는 실제로 컴파일하는 파일은 하나만 컴파일하게 되지만, 지금 여기서는 몇 가지 예제를 들기 위해서 여러 개의 파일을 만들어 본 것입니다. 특히 여기서 scss 파일 앞에 숫자가 들어가 있는 이유는 책의 순서에 맞게 예제를 정렬하기 위함입니다. 다른 이유는 없습니다.

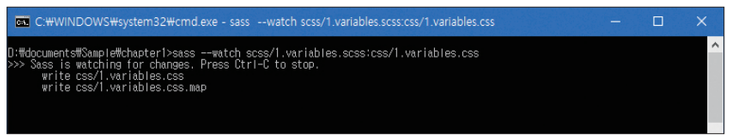
여기서 css 폴더 내부에 저장되는 파일 이름은 여러분들이 임의로 지정해 줄 수도 있고, scss 파일과 동일하게 지정해 줄 수도 있습니다. [그림 1-8.2]에서는 예제 파일과 동일하게 파일 이름을 지정하는 것을 알 수 있습니다. 이 책에서 제공하는 기본 예제에서는 css 파일들은 전부 제거하고 SCSS 파일만 제공하고 있습니다. 여러분이 직접 Koala를 이용해서 컴파일해 보기 바랍니다.

Koala는 모든 작업이 끝날 까지 절대 종료하면 안 됩니다. 백그라운드에서 SCSS 파일에서 변화가 일어나면 실시간으로 컴파일을 하기 때문입니다.




[그림 1-9] [여기서 명령창 열기] 화면

Koala라는 어플리케이션 말고 커맨드라인을 이용하여 컴파일을 할 경우에는 다음과 같이 "chapter1"이라는 해당 폴더에서 키보드의 Shift 키를 누르고 마우스 오른쪽 버튼을 누르면 [그림 1-9]와 같은 메뉴가 뜨며 **[여기서 명령 창 열기]**라는 메뉴를 클릭하면 됩니다. [그림 1-10]과 같이 sass --watch scss/파일명.scss:css/파일명.css라고 입력하고 엔터키를 누르면 컴파일이 시작되는데, 코알라와 동일하게 여기서도 절대 커맨드 창을 닫으면 안 되고, 작업이 진행되는 동안 계속 커맨드 창은 열려 있어야 합니다. Koala 어플리케이션과 같이 커맨드라인에서도 백그라운드로 항상 SCSS 파일의 변화를 읽고 실시간으로 컴파일하기 때문입니다.




[그림 1-10] 커맨드라인을 통한 SCSS 파일 실시간 컴파일 화면

이제 준비가 되었다면, 실제 SASS에 관한 기초지식을 학습해 보도록 하겠습니다.



## 변수(Variables)

프로그래밍 언어에서 가장 많이 사용되는 부분이 변수입니다. 변수를 이용하면 변수값만 변경함으로써 결과를 다르게 만들 수 있듯이, SASS에서도 반복적으로 많이 사용되는 부분, 예를 들어 font 모양이나 색상들을 변수로 지정해 놓고 이 변수값만 변경해 주면 CSS를 아주 간단하게 제어할 수 있습니다.

SASS에서는 변수를 정의할 때 $ 기호를 사용합니다. 변수의 이름은 사용자에 의해서 지정할 수 있지만, 속성은 CSS의 속성을 지정해 줘야 합니다. 예제를 한번 보도록 하겠습니다. 앞서 살펴 봤던 [코드 1-1]의 코드와 동일한 코드입니다.

[코드 1-3] 변수 설정 예제

```
$font-stack: Helvetica, sans-serif
$primary-color: #333
body
  font: 100% $font-stack
  color: $primary-color
```

Sample/chapter1/color.scss

[코드 1-3]을 보면 $font-stack이라는 변수와 $primary-color라는 변수가 있습니다. 각 변수들에는 CSS의 속성을 적용했습니다.

$font-stack에는 Helvetica라는 sans-serif 계열의 글꼴이, $primary-color에는 회색이 지정되어 있습니다.

[코드 1-3]이 컴파일되면 다음과 같은 결과를 얻을 수 있습니다.

[코드 1-4] [코드 1-3]의 컴파일 결과

```
body {
  font: 100% Helvetica, sans-serif;
  color: #333;
}
```

[코드 1-4]를 보면 body 태그 선택자 부분에 있는 font 속성에 Helvetica, sans-serif;가 적용되어 있고 color 속성에 #333이라는 값이 적용된 것을 보실 수 있습니다. 이 변수 부분만 알고 있어도, CSS를 코딩할 때 엄청 편리하게 사용할 수 있습니다. 예를 들어 font의 색상이나, div 또는 span이라는 선택자에 지정된 배경 색상, 그리고 다양한 선택자에 지정된 속성을 변수로 지정해 줌으로써, 변수에 있는 속성만 변경하면 전체 CSS의 속성을 한번에 변경해 줄 수 있기 때문입니다.

## 중첩(Nesting)

HTML 코딩을 하다 보면 중첩적인 표현을 위해 하위 선택자를 많이 사용할 수밖에 없습니다. 예를 들어 nav 태그를 이용해서 내비게이션을 만들 때 ul과 li라는 태그 선택자가 있는데, 이 두 선택자는 nav의 하위 선택자입니다.

다음의 코드 예를 한번 보기 바랍니다.

[코드 1-5] ul과 li 태그를 이용한 HTML 코드 구성

```
<nav>
  <ul>
    <li><a href="link1">Menu1</a></li>
    <li><a href="link2">Menu2</a></li>
    <li><a href="link3">Menu3</a></li>
  </ul>
</nav>
```

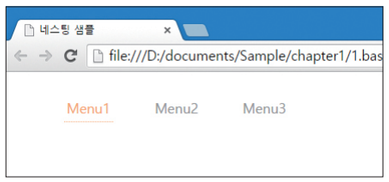
이제 [코드 1-5]의 HTML 태그 부분에 CSS 속성을 적용해 보도록 하겠습니다.

[코드 1-6] [코드 1-5]의 HTML에 CSS 속성 적용

```
nav ul {
    list-style: none;
}

nav ul li {
    display: inline-block;
}

nav ul li a {
    display: block;
    padding: 5px 3px;
    margin: 10px 20px;
    text-decoration: none;
    color: #999;
}

nav ul li a:hover {
    border-bottom: 1px dotted #999;
}
```

Sample/Chapter1/2.nesting-sample1.html

소스를 실행한 결과는 다음과 같습니다(여기서 파일 번호가 2부터 시작되는 이유는 scss 파일과 동일하게 처리하기 위함입니다).




[그림 1-11] 소스 실행 결과 화면 – 여기서 Menu1 부분이 마우스 오버인 상태를 나타냅니다.

결과를 확인했으니, 이제 중첩Nesting에 대해서 학습해 보도록 하겠습니다.

SASS의 중첩은 코드를 아주 간략하게 만들어 줍니다 [코드 1-6]에서와 같이 nav 밑에 ul이 있는 경우 nav {… ul { …}}와 같은 방법으로 중첩 처리해 주면 됩니다. 즉 중첩이라는 것은 알을 품는 형태를 의미하며, 하위 선택자를 안에 품는다는 의미를 지닙니다.

[코드 1-6]의 코드는 SASS에서는 다음과 같이 처리할 수 있습니다.

[코드 1-7] SASS의 중첩을 이용하여 [코드 1-6]과 동일한 결과를 얻을 수 있음

```
$default-color: #999;
$navhover-color: #ffb069;

nav {
    ul {
        list-style: none;

        li {
            display: inline-block;

            a {
                display: block;
```

```
            padding: 5px 3px;
            margin: 10px 20px;
            text-decoration: none;
            color: $default-color;

            &:hover {
                border-bottom: 1px dotted $navhover-color;
                color: $navhover-color;
            }
        }
    }
}
}
```

Sample/Chapter1/scss/2.nesting-sample1.scss

제공된 소스를 직접 컴파일할 경우 [코드 1-6]과 동일한 결과를 얻을 수 있습니다.

중첩의 장점은 코드의 길이를 많이 줄여줄 수 있다는 것과 HTML의 박스 모델에서 각 박스에 있는 개별 태그에 정확한 값을 지정해 줄 수 있습니다. 예를 들어, header라는 박스 모델 내부에 nav 태그가 있고, nav 태그 내에 li와 a 즉 [코드 1-5]와 유사한 코드로 구성되어 있고 하단 부분에 aside라는 박스 모델에 또한 동일한 태그 구성이 되어 있는 경우 header 부분에 있는 nav 태그와 aside에 있는 nav 태그에 별도의 속성을 확실하게 구분하여 처리할 수 있습니다.

실제 예제를 통해서 살펴보도록 하겠습니다.

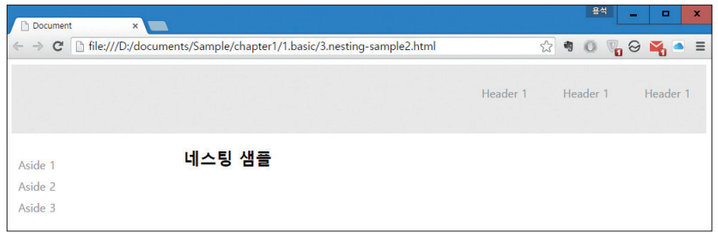
[코드 1-8] 중첩 설명을 위한 HTML 예제

```
…
<header>
  <nav>
    <ul>
      <li><a href=>Header 1</a></li>
      <li><a href="#">Header 1</a></li>
      <li><a href="#">Header 1</a></li>
    </ul>
  </nav>
</header>
<aside>
  <nav>
    <ul>
      <li><a href="#">Aside 1</a></li>
      <li><a href="#">Aside 2</a></li>
      <li><a href="#">Aside 3</a></li>
    </ul>
  </nav>
</aside>
<article>
  <h1> 네스팅 샘플 </h1>
</article>
…
```

Sample/chapter1/3.nesting-sample2.html

[코드 1-8]을 보면 header 태그와 aside 태그에 동일하게 nav 태그와 하위 태그인 ul, li 그리고 a 태그로 구성된 코드를 볼 수 있습니다. 여기서 header 태그 부분은 사이트의 헤더 부분이고, aside 부분은 서브 메뉴가 위치하는 곳입니다. 일반적으로 헤더 부분에는 메인 메뉴가 들어가는데, 메뉴는 보통 가로 배열을 합니다. 서브 메뉴 부분은 세로 배열을 하는 것이 일반적입니다.

우선 [코드 1-8]에 의한 결과물부터 확인하도록 하겠습니다.




[그림 1-12] [코드 1-8]과 별도의 CSS 파일에 의한 결과물

**주의**

예제로 제공된 html 파일을 실행하면 [그림 1-12]와 같은 결과가 나오질 않습니다. 반드시 제공된 scss 파일(3.nesting-sample2.scss)을 css 폴더에 경로를 설정한 후 nestingsample2.css라고 이름을 지정해서 컴파일해야만 [그림 1-12]와 같은 결과를 얻을 수 있습니다.

[그림 1-12]는 별도의 CSS 속성이 적용되어 있는 상태입니다. 그림에서 보듯이 header 부분과 aside 부분은 배열 자체도 다르며, 실제 예제를 실행해 보면 hover 효과까지 다른 것을 알 수 있을 것입니다.

이 부분을 앞서 학습한 변수를 이용하고 중첩 방식으로 조금 더 복잡하게 코드를 구성해 보도록 하겠습니다.

[코드 1-9] [그림 1-12]와 같은 결과물을 얻기 위한 SASS 코드

```
$default-color: #999;
$navhover-color: #ffb069;
$aside-hover: #ff0000;
$header-back: #eee;

header {
    background-color: $header-back;
    display: block;
    height: 100px;

    nav {
        ul {
            list-style: none;
            float: right;

            li {
                display: inline-block;

                a {
                    display: block;
                    padding: 5px 3px;
                    margin: 10px 20px;
                    text-decoration: none;
                    color: $default-color;

                    &:hover {
                        border-bottom: 1px dotted $navhover-color;
                        color: $navhover-color;
                    }
                }
            }
        }
    }
}

aside {
    width: 250px;
    float: left;

    nav {
        ul {
            list-style: none;
            padding: 0;
            margin-top: 30px;

            li {
                a {
                    display: block;
                    padding: 5px 10px;
                    text-decoration: none;
                    color: $default-color;
                    width: 100px;

                    &:hover {
                        border: 1px solid $aside-hover;
                        color: $aside-hover;
                    }
                }
            }
        }
    }
}
```

Sample/chapter1/3.nesting-sample2.scss

[코드 1-9]를 보면 header와 aside 태그 선택자 밑으로 하위 선택자들을 중첩하여, 개별적인 속성이 적용된 것을 확인할 수 있습니다. [코드 1-9]를 컴파일하면 어떤 결과가 나올까요?

[코드 1-10]을 확인해 보도록 하겠습니다.

[코드 1-10] [코드 1-9] SASS 파일을 컴파일한 CSS 결과 파일

```
header {
    background-color: #eee;
    display: block;
    height: 100px;
}

header nav ul {
    list-style: none;
    float: right;
```

```
}

header nav ul li {
    display: inline-block;
}

header nav ul li a {
    display: block;
    padding: 5px 3px;
    margin: 10px 20px;
    text-decoration: none;
    color: #999;
}

header nav ul li a:hover {
    border-bottom: 1px dotted #ffb069;
    color: #ffb069;
}

aside {
    width: 250px;
    float: left;
}

aside nav ul {
    list-style: none;
    padding: 0;
    margin-top: 30px;
}

aside nav ul li a {
    display: block;
    padding: 5px 10px;
    text-decoration: none;
    color: #999;
    width: 100px;
}

aside nav ul li a:hover {
    border: 1px solid #ff0000;
    color: #ff0000;
}
```



[코드 1-9]와 [코드 1-10]을 비교해 보면 얼핏 [코드 1-9]가 더 복잡하고, 코드의 양도 많게 보입니다. 하지만 [코드 1-9]에서는 선택자를 한 번만 지정했지만, [코드 1-10]에서는 태그 선택자와 하위 선택자에 CSS 속성을 부여하기 위해서 반복적으로 선택자와 하위 선택자 그리고 하위 선택자의 하위 선택자를 계속하여 코드를 입력해야 하지만, [코드 1-9]에서는 중첩을 이용해서 단순하게 처리된 것을 볼 수 있습니다. [코드 1-9]에서 `header` 부분에도 `nav ul li` 방식의 코드가 있고, `aside` 부분에도 `nav ul li` 방식으로 되어 있습니다. 또한 코드 몇 부분은 속성이 동일한 것도 있습니다. 이렇게 속성이 동일한 부분을 간단하게 처리할 수 있는 방법이 SASS에는 존재합니다. 이 부분은 조금 있다가 학습하도록 하겠습니다.

### 부분화(Partials) / 불러오기(Import)

'부분화'와 '불러오기'는 같이 다루도록 하겠습니다. SASS로 작업을 하다 보면  문장이 상당히 길어지고 복잡하게 되어 문장을 분리할 필요가 있습니다. 이때 문장을 분리하는 것을 '부분화'라고 합니다. 이렇게 부분적으로 쪼갠 SASS 문장을 하나의 문장에 불러오기Import를 한 후 컴파일 과정을 거치면, 완벽한 하나의 CSS 파일로 변환되는 것입니다.

[코드 1-9]를 보면 색상들을 별도의 변수로 처리한 부분이 있습니다. 이 부분을 부분화를 하여 별도의 파일로 만들고 불러오기를 해보겠습니다. 먼저 에디터에 새로운 파일을 생성하고 "변수로 처리한 부분"만 Copy & Paste를 합니다. 그리고 파일 이름은 _color.scss라고 합니다. **여기서 부분화를 할 때 제일 중요한 것이 파일 이름입니다. 부분화를 하는 파일 이름은 앞 부분에 반드시 "_(언더바)"를 추가해야 합니다.** 그래야 이 부분이 부분화 파일이구나 라고 알 수 있습니다. 언더바를 추가하지 않으면 하나의 CSS 파일로 컴파일되어 버립니다.

CSS 파일의 경우에도 보통 CSS 파일 내부에 불러오기를 하는 방법이 있습니다. 이 경우에도 import를 이용해서 처리하게 되는데, CSS의 경우에는 `@import url(color.css);` 라고 지정해 줍니다. SASS에서는 `@import 'color';`라고 처리해 줍니다. 확장자가 없습니다. 부분화를 할 때는 파일명 앞에 _(언더바)를 붙이지만, import를 할 때는 "_(언더바)"을 쓰지 않습니다. 이 부분이 CSS에서의 import와 다른 부분입니다.

"Sample/chapter1/scss/4.import-sample.scss"와 부속 파일인 "_color.scss"을 참조하기 바랍니다.

## 믹스인(Mixins)

믹스인은 CSS의 속성 중 특히 CSS3의 속성에서 사용되는 부분에 대해서 브라우저 제조사에 따라 별도의 접두사Vendor prefixes가 존재하는데, 믹스인은 브라우저별 접두사를 처리하거나 반복적인 속성을 손쉽게 처리할 수 있게 해주는 역할을 합니다. 예를 들면 `border-radius`라는 CSS3의 속성은 사각형의 모서리를 둥글게 만들어 주는 속성입니다. 이 속성은 구글과 사파리 브라우저에서는 `-webkit-`이란 접두사가 붙고, 파이어폭스는 `-moz-`라는 접두사가, IE 계열은 `-ms-`라는 접두사가 붙게 됩니다. 물론 최신의 브라우저에서는 해당 접두사가 없어도 브라우저에서 CSS3의 속성을 지원해 주기 때문에 별 필요는 없을 수 있지만, 하위 버전의 브라우저에서는 제대로 CSS3의 속성이 표시되지 않을 대비책으로 접두사를 지정해 주는 것이 가장 좋습니다. 이러한 접두사는 가장 먼저 크롬, 사파리 〉 파이어폭스 〉 IE계열 〉 CSS3 속성의 순으로 배열해 주는 것이 가장 좋습니다. 예제를 살펴 보겠습니다.

[코드 1-11] 믹스인을 이용한 브라우저별 접두사 처리하기

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

.box {
    @include border-radius(10px);
}
```

[코드 1-11]을 보면 `border-radius`라는 속성에 별도의 접두사를 처리하는 부분에 변수로 `$radius`를 지정했으며, `.box` 선택자 부분에 변수값을 `10px`로 지정했습니다. 이렇게 하면 결과값은 다음과 같이 얻을 수 있습니다.

[코드 1-12] [코드 1-11]을 컴파일 하면 얻을 수 있는 결과값

```
.box {
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  -ms-border-radius: 10px;
  border-radius: 10px;
}
```



[코드 1-12]의 결과값을 보면 CSS를 일반적으로 코딩할 경우 `.box`라는 선택자에는 `10px`의 `border-radius`를, `.box1`이라는 선택자에는 `15px`의 `border-radius` 값을 적용하려면, 각 브라우저별 접두사를 전부 입력해야 하지만, 믹스인을 사용하면 `.box`라는 선택자에는 [코드 1-11]에서와 같이 `.box { @include border-radius(10px); }`라고 해주고 `.box1`이라는 선택자에는 `.box {@include border-radius(15px);}`라고 지정하면 브라우저별 접두사는 자동적으로 생성됩니다. 사실 SASS의 가장 핵심적인 기능 중 하나가 "믹스인"이라고 할 수 있습니다.

SASS의 가장 큰 장점은 **반복되는 작업을 최대한 줄여주고, 생산성을 높이며, 작업의 효율성을 극대화**하는 데 있습니다.

간단한 예제를 통한 기초 과정을 마치게 되면 본격적인 SASS에 대해서 학습할 텐데, SASS 문법에는 프로그래밍 언어에서 사용되는 "if문" 그리고 "while문"과 같은 프로그래밍 기법들이 있습니다. 이러한 프로그래밍적 특성을 잘 활용하면, 아주 적은 코드를 이용해서 CSS의 결과값을 극대화할 수 있게 되는 것입니다.

## 확장(Extend) / 상속(Inheritance)

SASS에서 지금까지 변수와 중첩, 부분화와 불러오기 그리고 믹스인까지 학습했습니다. 이번에는 확장과 상속에 대해서 다뤄보겠습니다. 앞에서 중첩이 있는 [코드 1-9]에 대해서 설명할 때 중복되는 부분이 있다고 이야기했습니다. `nav` 태그 선택자의 하위 선택자로서, `ul`과 `li`의 속성이 비슷한 것들이 있는데, "이 부분을 묶어서 사용하면 얼마나 편리할까"라는 생각을 할 수도 있습니다. SASS에서는 확장과 상속을 통해 편리하게 비슷한 코드를 통합해 줄 수 있습니다. CSS의 속성은 위에서 아래로 적용됩니다. 다음과 같은 CSS 코드가 있을 경우 실제 사이트에 적용되는 결과값은 어떻게 될지 생각해 보세요.

[코드 1-13] CSS의 기본 속성 이해

```
.box {
    padding: 20px 10px;
    margin: 10px;
    color: #ccc;
}

.box {
    margin: 15px;
}
```

답은 다음과 같습니다.

[코드 1-14] CSS의 기본 속성 이해 – top down 방식에 의한 결과값의 변화

```
.box {
    padding: 20px 10px;
    margin: 15px;
    color: #ccc;
}
```

동일한 선택자의 속성값을 여러 번 변경하더라도, 결과값은 가장 마지막에 있는 값의 영향을 받습니다. [코드 1-13]에서 보면 padding 값과 color 값은 하단에 없기 때문에 그대로 반영되지만, margin 값은 하단에 15px로 변경되어 최종 결과는 [코드 1-14]와 같은 속성이 반영되는 것입니다. 이러한 설명을 하는 이유는 확장과 상속을 이해하기 위해서 필요하기 때문입니다. SASS에서 확장은 @extend라는 코드를 이용합니다.

다음의 예제를 보겠습니다.

[코드 1-15] 확장과 상속 예제

```
.message {
    border: 1px solid #ccc;
    padding: 10px;
    color: #333;
}

.success {
    @extend .message;
    border-color: green;
}

.error {
    @extend .message;
    border-color: red;
}

.warning {
    @extend .message;
    border-color: yellow;
}
```



[코드 1-15]를 보면 .message 선택자를 이용하여 작은 박스를 만들 수 있는 CSS 코드를 볼 수 있을 겁니다. 이게 가장 기본이라고 볼 때 3개의 동일한 작은 박스를 만들기 위해 .success, .error 그리고 .warning이라는 선택자를 추가했는데, 여기에 @extend .message라고 해서 .message 선택자의 속성을 그대로 가져 왔는데, 각 박스에는 별도의 테두리 색상을 지정해 줬습니다.

결과 값은 다음과 같습니다.

```
.message, .success, .error, .warning {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
  border-color: yellow;
}
```

다시 [코드 1-9]를 보면 동일한 속성의 경우 확장과 상속을 이용해서 처리할 수 있을 것입니다. 일단 SCSS 코드부터 알려드리겠습니다.

[코드 1-16] 확장과 상속을 이용해서 코드를 단순하게 처리할 수 있다.

```
…
nav {
    ul {
        list-style: none;
        float: right;

        li {
            display: inline-block;
```

```
            a {
                display: block;
                padding: 5px 3px;
                margin: 10px 20px;
                text-decoration: none;
                color: $default-color;

                &:hover {
                    border-bottom: 1px dotted $navhover-color;
                    color: $navhover-color;
                }
            }
        }
    }
}

header {
    background-color: $header-back;
    display: block;
    height: 100px;

    @extend nav;
}

aside {
    width: 250px;
    float: left;

    @extend nav;

    ul {
        padding: 0;
        margin-top: 30px;
        float: left;

        li {
            display: block;

            a {
                padding: 5px 10px;
                margin: 0;
                width: 100px;
                &:hover {
                    border: 1px solid $aside-hover;
                    color: $aside-hover;
                }
            }
        }
    }
}
```

Sample/chapter1/scss/5.extend.scss

[코드 1-16]을 보면 header라는 태그 선택자 밑에 있는 nav 태그 선택자의 속성을 별도로 분리해서 header 선택자 부분에 @extend nav;라고 처리했으며, aside 부분에도 @extend nav;라고 처리했습니다. 여기서 header nav에 있는 속성과 aside nav에 있는 속성들이 동일한 부분도 있지만, 동일하지 않는 부분은 aside 부분에서 별도의 속성을 다시 지정했습니다. 이렇게 하면 header 부분과 aside에 있는 nav 태그 선택자 속성에서 동일하게 적용되는 값을 변경할 때 nav 태그 선택자 부분에서 코드 한줄을 추가하거나 값을 변경하면, header와 aside에 있는 nav 부분도 동일하게 값이 변경됩니다. 이것은 사이트가 커지면 커질수록 CSS 관리가 편리해진다는 것을 의미합니다.

이제 마지막으로 기초 과정의 마지막인 계산에 대한 학습을 하겠습니다.

## 연산자(Operators)

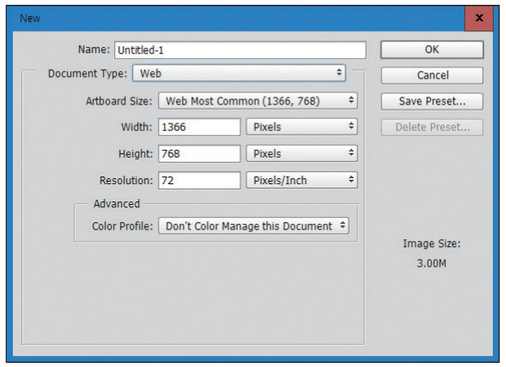
SASS는 프로그래밍적 요소가 있기 때문에 간단한 사칙연산이 가능합니다. 즉 더하기(+), 빼기(-), 곱하기(*), 나누기(/) 그리고 퍼센트(%) 처리가 가능합니다. 이 연산 기능은 CSS를 이용하여 레이아웃을 작업할 때 아주 편리하게 사용할 수 있으며, 기타 여러 용도로 사용 가능합니다. 예를 들어 색상을 섞는 기능도 가능합니다. white(흰색)와 black(검정색)을 혼합하면 회색이 나오는데, 이런 용도로도 사용할 수 있습니다. 또한 폰트 사이즈를 조정하거나 기타 길이 또는 높이를 조정할 때도 사용 가능합니다. 일단 간단하게 예제를 한번 보도록 하겠습니다.

[코드 1-17] 연산 기능 예제

```
.container {
    width: 100%;
}
```

```
article[role="main"] {
    float: left;
    width: 600px / 960px * 100%;
}

aside[role="complimentary"] {
    float: right;
    width: 300px / 960px * 100%;
}
```

[코드 1-17]은 아주 흥미로운 예제입니다. 보통 웹사이트를 디자인할 때는 포토샵을 많이 이용하게 됩니다. 포토샵을 이용하면 일반적으로 "artboard"의 크기를 정하고 작업을 하게 됩니다. [그림 1-13]를 참조하세요.




[그림 1-13] 포토샵을 이용한 웹사이트 작업시 초기 크기 설정 화면

그런데 웹사이트는 고정폭으로 사이트를 만드는 경우도 있지만 폭의 크기를 유동적으로 변경하는 경우도 있습니다. [코드 1-17]과 같이 container의 크기 즉 웹사이트의 크기를 100%로 정하는 경우 포토샵을 이용하여 레이아웃을 정하면 콘텐츠가 들어가는 부분과 사이드 바의 크기 또한 유동적으로 변경해 줄 필요가 있습니다. 왜냐하면, 웹 브라우저의 크기가 커지거나 작아질 경우 콘텐츠 부분과 사이드 바의 크기 또한 유동적으로 변하게 처리해야 하기 때문입니다. 이 경우 연산 기능을 이용하면, 해당 콘텐츠의 크기를 쉽게 처리할 수 있습니다. 사실 필자도 이런 경우 옆에 계산기를 두고 유동 크기 즉 크기가 몇 퍼센트인지 계산을 하곤 했는데, SASS를 학습하고 나선 계산기를 사용하지 않고 연산자를 이용해서 처리합니다.



[코드 1-17]의 결과값은 직접 소스 코드를 작성해서 컴파일해 보기 바랍니다.

여기까지 SASS의 가장 기본적인 기능을 살펴봤습니다. 사실 여기까지 학습한 내용만 잘 배워둔다면 CSS 개발이 엄청 쉬워집니다. 필자도 지금까지 학습한 내용을 가장 많이 사용합니다. 물론, SASS을 보다 자세하게 학습하게 되면 더 편리하게 CSS 개발을 할 수 있습니다.

지금까지 학습한 내용 중에 필자가 언급하지 않는 부분도 몇가지 있습니다. 특히 `&:hover`의 경우 왜 여기서 `&`라는 값을 이용했는지에 대해서는 설명을 하지 않았는데, 이 부분은 본격적인 SASS 레퍼런스에 들어가면 자세하게 설명하도록 하겠습니다.

2장

# SASS 레퍼런스

2장은 SASS 공식 웹사이트의 문서(documentation) 내용을 기반으로 설명하겠습니다. 이 문서가 가장 정확한 SASS 문법입니다. 이 책에서는 공식 웹사이트에 나온 것과는 약간 다르게 설명할 수도 있습니다. 그 이유는 조금이라도 여러분이 이해를 쉽게 할 수 있도록 풀어서 설명하기 위함입니다. 예제도 공식 사이트(레퍼런스)에는 없는 다양한 예제를 만들어 설명해 보겠습니다.