Hanbit RealTime 131

처음 시작하는 Sass

CSS의 한계를 뛰어넘는 Sass를 만나다

김유리, 방지은, 양주희, 정대영, 홍보라 지음

한빛미디어 Hanbit Media, Inc.

처음 시작하는 Sass

CSS의 한계를 뛰어넘는 Sass를 만나다

김유리, 방지은, 양주희, 정대영, 홍보라 지음

한빛미디어 Hanbit Media, Inc.

표지 사진 **최현수**

이 책의 표지는 최현수님이 보내 주신 풍경사진을 담았습니다.
리얼타임은 독자의 시선을 담은 풍경사진을 책 표지로 보여주고자 합니다.

사진 보내기 **ebookwriter@hanbit.co.kr**

처음 시작하는 Sass CSS의 한계를 뛰어넘는 Sass를 만나다

초판발행 2016년 6월 16일

지은이 김유리, 방지은, 양주희, 정대영, 홍보라 / **펴낸이** 김태헌
펴낸곳 한빛미디어(주) / **주소** 서울시 마포구 양화로 7길 83 한빛미디어(주) IT출판부
전화 02-325-5544 / **팩스** 02-336-7124
등록 1999년 6월 24일 제10-1779호
ISBN 978-89-6848-823-8 95000 / **정가** 12,000원

총괄 전태호 / **책임편집** 김창수 / **기획·편집** 김상민
디자인 표지/내지 여동일, 조판 금미향
마케팅 박상용, 송경석, 변지영 / **영업** 김형진, 김진불, 조유미

이 책에 대한 의견이나 오탈자 및 잘못된 내용에 대한 수정 정보는 한빛미디어(주)의 홈페이지나 아래 이메일로 알려주십시오.
한빛미디어 홈페이지 www.hanbit.co.kr / **이메일** ask@hanbit.co.kr

Published by HANBIT Media, Inc. Printed in Korea.

지금 하지 않으면 할 수 없는 일이 있습니다.
책으로 펴내고 싶은 아이디어나 원고를 메일(ebookwriter@hanbit.co.kr)로 보내주세요.
한빛미디어(주)는 여러분의 소중한 경험과 지식을 기다리고 있습니다.

지은이_ **김유리**

평범한 일상 속 낭만을 꿈꾸는 프론트엔드 개발자입니다. 현재 NHN Technology Services에서 근무하고, 사내 교육 강사로도 활동하면서 누군가에게 도움이 된다는 것에 굉장히 보람을 느끼고 있습니다. 사용자에게 친절한 개발자가 되는 것을 목표로 언제나 긍정적인 자세로 새로운 UI를 구현하기 위해 삽질을 즐기며 연구하고 있습니다.

지은이_ **방지은**

현재 NHN Technology Services UIT 개발실에서 프론트엔드 개발자로 일하고 있습니다. 카페라떼를 사랑합니다.

지은이_ **양주희**

현재 NHN Technology Services UIT 개발실에서 프론트엔드 개발자로 일하고 있습니다. 『웹킷 CSS 바이블: 실무 편』, 『웹킷 CSS 바이블: 레퍼런스 편』(한빛미디어, 2015) 집필에 참여하였으며, 새로운 기술을 사용하는 데 큰 즐거움을 느낍니다.

지은이_ **정대영**

프론트엔드 개발자로 〈네이버 선거 특집 페이지〉, 〈런던 올림픽〉, 〈오늘의 신문〉, 〈네이버 톡〉 등 다수의 프로젝트를 진행했습니다. 교육, 세미나, 컨벤션 작업 등 다양한 방법으로 동료들에게 도움이 될 수 있는 활동에 보람을 느끼며 일하고 있습니다.

항상 힘이 되어주는 와이프에게 감사합니다.

지은이_ **홍보라**

새롭고 신기한 코드를 갈구하는 개발자입니다. NHN Technology Services에서 프론트엔드 개발자로 일하고 있습니다. 모든 커피를 사랑합니다.

문서를 꾸미는 보조 역할을 하던 CSS는 시간이 지날수록 웹 개발에서 중요한 역할을 차지하게 됐다. 특히 CSS2 이후 오랜 공백을 깨고 CSS3가 등장하면서 CSS를 통해 더욱 많은 표현이 가능하게 된 것이 큰 계기가 되었다. 배경 이미지를 자르고 구조를 나누어 표현했던 둥근 테두리는 속성 하나로 손쉽게 적용할 수 있게 되었으며, JavaScript나 서버 단에 의지했던 수많은 선택자도 CSS3 이후에는 CSS만으로 처리할 수 있게 되었다.

하지만 여전히 스타일을 정의하고 속성을 적용하는 방식은 매우 단순한 수준에 머무르고 있어, 반복되는 코드의 입력을 줄이려는 여타 개발 언어들과는 사뭇 다른 아쉬움을 남기고 있다. 특히 과거와 다르게 CSS의 활용 범위가 커지고 웹 사이트의 규모가 거대해지면서 1,000라인쯤은 쉽게 넘기는 코드가 작성되고 있으며, 이에 따라 이 코드들을 더욱 효율적으로 관리해야 할 필요성이 강조되고 있다.

다행히 이러한 안타까운 현실에서 웹 개발자를 구원해줄 한 줄기 빛이 생겨났는데, Sass를 필두로 좀 더 편리하고 효율적으로 CSS를 작성할 수 있도록 도움을 주는 전처리기들이 바로 그 구원자들이다. 그 가운데서도 Sass는 가장 큰 인기를 얻고 있으며 많은 웹 개발자의 찬사를 받고 있다. 물론 새로운 개념을 받아드리기는 언제나 쉽지 않지만, 새로움을 위한 도전에 이 책이 반드시 도움되리라 생각한다.

필자는 동료들 가운데 비교적 먼저 Sass를 사용했고, 많은 입문자를 가이드해주고 지켜보았다. 대부분 처음에는 두려움으로 시작하지만, 어느 정도 숙련도가 높아지면 공통으로 하는 이야기가 있다. Sass를 줄곧 활용하다 가끔 기존 CSS로 작업하는 업무를 진행하면 너무나 불편하고 코드를 작성하는 게 비효율적이라는 것이다. 이 책의 독자들도 지금은 낯설고 두려울 수 있지만, 이 책을 완독한 뒤에는 Sass가 없는 개발 환경은 생각하기조차 싫을 것이라고 확신한다.

_집필진 대표 정대영

차례

chapter 1

이해하기

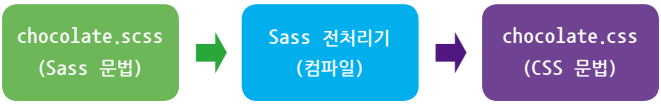
1.1 CSS 전처리기

반복되는 코드의 복잡함에서 우리를 구원해 줄 Sass가 어떤 방식으로 우리에게 도움을 주는지 알아보자.

CSS 전처리기[preprocessor]는 변수, 함수, 상속 등의 일반적인 프로그래밍 개념을 사용하여 코드를 작성한 뒤 기존 CSS 문법으로 컴파일 해 주는 방식을 취한다. 그리고 이러한 컴파일 방식 때문에 별도의 언어가 아닌 전처리기라고 부르게 됐다.

예를 들어 Sass는 Sass 문법에 맞추어 chocolate.scss 파일을 작성한 뒤 컴파일 명령어를 통해 브라우저가 이해할 수 있는 chocolate.css가 컴파일 되는 방식이다.

[그림 1-1] 컴파일 과정



이렇게 나타난 전처리기로는 Sass, Less, Stylus가 있으며, 지금부터 전처리기 각각의 특징을 비교해서 살펴보자.

1.1.1 Sass

- 가장 역사가 오래된 전처리기다.
- Ruby 언어를 기반으로 구동된다.
- Ruby 언어가 지닌 한계로 파일이 길어지면 컴파일이 다소 느리다.
  - C로 포팅된 Libass는 상당히 빠른 컴파일 속도를 제공하나 Sass의 업데이트 반영이 느리다는 단점이 있다.
- 상대적으로 다양한 기능을 제공하며 계속해서 새로운 기능들이 추가되고 있다.
- CSS3, ImageSprite 처리 등의 부가적인 기능들을 제공하는 Compass를 사용할 수 있다.
- IDE(에디터)를 잘 지원하고 있다.

[코드 1-1] Sass 예제 코드

Sass	CSS

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

1.1.2 Less

- Twitter의 Bootstrap에서 사용되어 널리 알려졌다.
- JavaScript(Node.js)를 기반으로 구동된다.
- 클라이언트 단(브라우저)에서 자바스크립트 파서parser를 통해 실행된다.
  - AJAX 콜을 사용하기 때문에 로컬 환경에서도 웹 서버 설정이 필요하다.
  - Internet Explorer 9 이상에서만 적용되며 반응 속도가 다소 느린 단점이 있다.
- 다른 전처리기와 같이 Simpless, Less Parser와 같은 외부 컴파일러(Node.js 기반)를 통해 컴파일 된 CSS를 컴파일 해 사용할 수도 있다.
- 초창기에는 Sass에 비해 컴파일이 빠른 것이 장점으로 꼽혔으나 Sass가 캐쉬를 적용한 뒤부터 큰 차이가 없어졌다.

- Sass와 Stylus에 비해 다소 기능이 부족하다는 평가가 있다.
- IDE도 플러그 인 등을 통해 잘 지원하고 있다.

[코드 1-2] Less 예제 코드

Less	CSS

`@nice-blue: #5B83AD;`
`@light-blue: @nice-blue + #111;`
`#header { color: @light-blue; }`
	`#header { color: #6c94be; }`

1.1.3 Stylus

- 가장 나중에 발표된 전처리기다.
- JavaScript(Node.js) 기반으로 구동된다.
- 기본적인 CSS 문법에 많은 변화를 주었다.
  - .styl 파일을 사용하면 표준 CSS 문법도 사용할 수 있으며, 콜론, 세미콜론, 중괄호 등을 입력하지 않아도 문제없이 컴파일 된다.
  - 지나치게 관대한 문법을 적용하고 있어 어떤 기호도 사용하지 않고 작성된 소스 코드를 접하게 되면 소스 코드 파악이 어려운 경우가 있다.
- 가장 늦게 발표된 전처리기지만, 많은 기능이 구현되어 있어 Sass와 기능상으로는 큰 차이가 없다는 평가를 받고 있다. 하지만 완성도가 낮아 여러 작은 버그가 존재한다.

[코드 1-3] Stylus 예제 코드

Stylus

`border-radius()`
`  -webkit-border-radius arguments`
`  -moz-border-radius arguments`
`  border-radius arguments`
`body`
`  font 12px Helvetica, Arial, sans-serif`
`a.button`
`  border-radius(5px)`

1.2 Sass 살펴보기

앞서 전처리기들을 비교하면서 Sass에 대해서도 간단히 살펴보았지만, 여기서 앞으로 공부하게 될 Sass의 특징들에 대해 좀 더 자세히 살펴보자.

1.2.1 Ruby 기반의 Sass

Sass의 특징 가운데 가장 독특한 부분은 Ruby 언어를 기반으로 동작한다는 것이다.

이 때문에 생기는 두 가지 단점이 있는데, 그 중 첫 번째는 사용자에게 설치 과정부터 부담을 준다는 것이다. 하지만 여기서는 해당 설치 과정을 윈도와 맥 두 가지 환경 모두 단계별로 학습할 수 있도록 안내하고 있으니 걱정하지 않아도 된다.

두 번째는 속도 문제다. Ruby 언어가 가지는 한계로 파일이 다소 길어지면 컴파일 속도가 조금씩 지연된다는 점인데, 1~2초 정도의 시간이지만 수시로 수정사항을 적용하면서 작업을 진행하하기 때문에 누적되면 다소 부담이 되기도 한다. 이러한 단점을 극복하기 위해 C로 이식된 Sass인 Libsass를 사용하기도 하지만, 최신 Sass의 업데이트 내용이 Libsass에 반영되기까지 시간이 걸리며, Compass를 사용할 수 없다는 또 다른 단점이 발생한다.

1.2.2 기존 문법과의 호환성

Sass가 인기를 얻게 된 가장 큰 이유는 강력한 기능 제공과 기존 문법과의 호환성 때문이다.

몇몇 전처리기는 CSS를 단순히 보강한 수준의 기능만을 제공하지만, Sass는 좀 더 완성도 높은 프로그래밍 언어처럼 동작하게 한다. 선택자를 손쉽게 입력할 수 있게 해주는 기본적인 기능부터 Sass 스크립트라고 불리는 함수 및 제어문까지 제공한다.

[코드 1-4] SCSS 코드

SCSS

```
@mixin rounded ($radius: 10px) {
    border-radius: $radius;
    -moz-border-radius: $radius;
    -webkit-border-radius: $radius;
}
@for $i from 1 through 3 {
  .item-#{$i} {
    width: 2em * $i;
  }
}
```

1.2.3 Sass와 SCSS

Sass는 Sass와 SCSS(Sassy CSS) 두 가지 문법을 제공한다. 처음 입문하는 독자에게 이 두 가지 문법이 혼란을 일으킬 수도 있는데 지금부터 각각의 차이를 찬찬히 살펴보자.

Sass도 초기에는 Haml[1]의 영향으로 더 편리한 개발을 위해 중괄호와 세미콜론 등을 사용하지 않는 Ruby 언어 스타일의 간략한 문법을 제공했다. 하지만 기존 CSS 문법과 지나치게 달라서 3.0 버전부터 기존 CSS의 문법을 확장해 주는 개념의 SCSS를 메인 문법으로 제공하게 된다.

기존 CSS 문법을 그대로 포함하기 때문에 CSS로 작성된 프로젝트를 SCSS로 전환하는 것에서도 문제가 없고, 코드를 접했을 때 Sass보다 좀 더 쉽게 구조를 파악할 수 있다는 점 때문에 SCSS의 인기가 계속해서 늘어나고 있다.

두 문법의 적용은 각 문법의 이름을 딴 확장자 명으로 구분하며 여기서는 SCSS만을 설명할 것이다.

1 Haml(HTML abstraction markup language): HTML을 더 쉽게 입력하려고 개발된 Ruby 언어 기반의 전처리기.

[코드 1-5] Sass와 SCSS 문법

Sass	SCSS

.class_name
  color= !primary-color
  width= 100%
  overflow= hidden
	.my-element {
	  color: $primary-color;
	  width: 100%;
	  overflow: hidden;
	}

1.2.4 프레임워크

Sass 기능을 확장하는 프레임워크로 Compass[2], Bourbon[3]이 있다. 유용한 mixin 등을 제공하며, 스프라이트 이미지도 손쉽게 관리할 수 있도록 도와준다. Compass에 비해 Bourbon은 다소 생소할 수도 있는데, 플러그 인 형태로 확장 기능을 제공한다는 점은 동일하다. 대체로 Compass는 강력하고 다양한 기능을 제공하는 반면 다소 무겁고, 반대로 Bourbon은 기능은 부족하지만 잘 만들어진 mixin 라이브러리 정도의 느낌으로 가볍다.

1.3 Sass에 대한 오해

Sass를 처음 입문하거나 아직 접해보지 않은 사람들에게 반복적으로 받는 질문이 있는데, 여기서는 Sass에 대한 몇 가지 오해를 짚고 넘어가려고 한다.

Q. 저는 제어문, 함수 등의 개념이 없어 사용하기 어려울 것 같아요.

Sass 입문을 망설이게 하는 가장 큰 이유는 평소 프로그래밍 로직에 대한 개념이 부족한 사람들이 가지는 프로그래밍에 대한 막연한 두려움이다. CSS에는 없었던 다양한 기능을 제공하기에 기능들을 이해하기 벅차지 않을까 하는 두려움이 생기

2 http://compass-style.org/

3 http://bourbon.io/

는 것이 사실이다.

하지만 SCSS의 경우 기존에 우리가 익숙한 CSS 문법을 온전히 포함하고 있기 때문에 기존 CSS 문법을 베이스로 중첩nesting, 변수, `mixin`, `@import` 등의 비교적 쉬운 기능을 시작으로 점차 고급 기능까지 단계적으로 추가하며 사용한다면 자연스럽게 마스터할 수 있으니 걱정하지 않아도 좋다.

Q. 기존에 이미 CSS로 작업한 프로젝트라서 적용하기가 어려워요.

앞선 오해와 마찬가지로 기존 CSS 문법을 온전히 포함하고 있기에 기존 CSS 코드를 SCSS로 변환하는 것은 전혀 문제가 없다. 기존에 작성된 CSS 코드를 그대로 복사해 붙여 넣은 후 SCSS 문법을 추가로 사용하면 된다.

Q. Sass로 생성한 CSS를 봤더니 불필요한 코드가 너무 많아요.

Sass를 포함한 전처리기들은 CSS를 좀 더 편리하게 작성하기 위한 도구일 뿐이다. 결국 전처리기들은 약속된 문법을 통해 CSS 코드를 만들어 내게 되는데, 본인이 작성한 Sass 문법이 CSS 코드를 더욱 완성도 있게 만들어 낸다는 개발자 관점의 사고가 개발 과정에 조금 포함된다면, 전처리기를 사용한다고 해서 불필요한 코드가 추가되는 것이 아님을 쉽게 파악할 수 있다.

Q. 추후 Sass를 모르는 담당자로 바뀌면 어떡하죠?

개인 프로젝트가 아닌 이상 본인이 진행한 프로젝트를 다른 사람이 유지 보수하는 경우가 생기게 된다. 이를 염두에 두고 추후 Sass를 모르는 후임자가 유지 보수를 하지 못하는 경우를 걱정해 Sass 도입을 주저하는 경우를 많이 볼 수 있다. 하지만 작성된 Sass 코드가 아까울 순 있지만, Sass가 컴파일 한 CSS 코드가 존재하기 때문에 후임자는 컴파일 된 CSS를 가지고 유지 보수를 이어갈 수 있어 문제가 되지 않는다.

Q. 설치가 어려워요.

앞서 Sass의 특징에 대해 설명할 때도 언급했지만 Ruby 언어 환경에 익숙하지 않은 사람들은 다소 설치 과정이 복잡하게 느껴질 수도 있다. 하지만 초기 설치 과정만 잘 넘긴다면 사용에는 큰 어려움이 없으며, 본 도서에서 설치 과정에 대해 자세히 소개할 예정이니 걱정하지 말자.

주의

Sass를 사용할 때 가장 주의해야 할 부분은 바로 남용이다. 손쉽게 선택자 중첩을 늘릴 수 있어 과도하게 깊은 중첩 상태의 선택자를 만들어 내기도 한다. 특히 특별한 목적 없이 `mixin`이나 변수 처리를 남발하게 되면 오히려 가독성이 떨어지고 유지 보수를 힘들게 하는 코드를 만들어 내기도 쉽다. 모든 도구는 적절하게 사용될 때 가치가 높아진다는 것을 명심하자.