댄 시더홈 Dan Cederholm

웹디자이너를 위한 SASS

SASS FOR WEB DESIGNERS
By A Book Apart

웹디자이너를 위한 SASS

초판 인쇄 2014년 7월 28일
초판 발행 2014년 8월 4일

저자 댄 시더홈
역자 윤원진

펴낸곳 웹액츄얼리코리아(주)
주소 서울시 강남구 논현로 707 3층
전화 (02)542-0411
팩스 (02)541-0414
웹사이트 books.webactually.com
페이스북 fb.com/webactually
트위터 @webactually

책임편집 천진아
윤문 임훈아
교정·교열 이현숙
편집디자인 s.t

가격 13,000원
출판 등록 제2014-000175호
ISBN 979-11-85885-02-5 13000

TABLE OF CONTENTS

13 CHAPTER 1 왜 SASS인가?

27 CHAPTER 2 SASS 워크플로

45 CHAPTER 3 SASS 사용하기

93 CHAPTER 4 SASS와 미디어 쿼리

119 참고 자료

124 참조

127 감사의 글

128 인덱스

출간에 앞서

독자 여러분, 안녕하세요? 아름다운 웹사이트 만들기 시리즈 열 번째 책, 《웹디자이너를 위한 SASS》를 소개합니다. 이 책을 선택하신 여러분은 CSS의 미래에 대해 관심을 가지고 있으며 시대에 앞선 기술을 터득하고자 노력하는 전문가이거나 앞으로 전문가가 될 분입니다. Sass가 바로 그런 기술이기 때문입니다.

1996년 첫 CSS1이 공식적으로 권고되었고 CSS3까지 발전했으며 이 순간에도 CSS3의 새 규칙이 소개되고 있습니다. 웹 개발에서 CSS 업무의 비중이 커지고 있습니다. 브라우저 호환성을 맞추고 반응형 디자인을 적용하는 작업 등 CSS의 몫이 큽니다. 이에 따라 스타일시트 문서는 길어지고 용량도 커집니다.

이렇게 된 주요 원인 중 하나는 코드의 반복입니다. 전 세계 웹 디자이너들에게 인기 있는 CSS-Tricks의 블로그 운영자인 크리스 코이어Chris Coyier는 "CSS에는 추상화 개념이 없고 코드의 반복만 있습니다. 이런 이유로 CSS가 널리 채택되고 사용되었지만 이제는 관리하기가 어려워졌습니다"라고 말합니다.

코드의 반복으로 인해 CSS 작업자는 파일에 산재해 있는 코드를 찾고 수정하는 데 많은 시간을 보냅니다. 코드를 관리하는 것은 어려워지고 그 효율성도 시간이 갈수록 떨어집니다. CSS를 작성하고 관리하는 분이라면 누구나 공감할 것입니다.

과연 이 문제를 해결하기 위한 좋은 방안은 없을까요? 물론 있지요! CSS의 전처리기가 바로 그것입니다. 전처리기는 위에 언급된 문제에 대해 해결안을 제시합니다. CSS 작업자에게 전처리기는 선택사항입니다. 그럼에도 전 세계 웹 관련 종사자들의 전처리기에 대한 관심은 높습니다. 그들은 여러 전처리기를 비교·분석하고 본인에게 맞는 전처리기를 찾는 데 열심입니다. 마치 이런 변화의 흐름과 거리가 먼 듯 국내에는 아직 그 불꽃이 일어나지 않은 상태입니다.

이 책의 저자인 댄 시더홈은 유명한 웹 디자이너이자 프론트엔드 개발자입니다. 글로벌 웹 디자인계에서 실력 있는 최고 전문가들이 참여해서 저술하는 A Book Apart의 두 번째 책인 《웹디자이너를 위한 CSS3》의 저자입니다. CSS를 10년 이상 직접 손으로 코드를 작성해온 댄은 본인만의 노하우가 담긴 CSS 작업 방식을 고집스럽게 지켜왔습니다. 그렇기에 그는 CSS 전처리기인 Sass에 대해 처음부터 호의적이지 않았다고 고백합니다. Sass를 사용하면서 그의 불신은 강한 믿음으로 완전히 바뀌었고, 이제는 열성적인 Sass 전도사가 되었습니다.

독자 여러분을 위해 댄은 Sass를 처음 접하는 독자의 눈높이에서 복잡하고 어려운 Sass의 용어와 문법을 쉽게 풀어서 설명해줍니다. 그가 제시하는 가상 웹사이트 프로젝트와 예제 코드는 글로 설명할 수 없는 내용을 여러분의 머릿 속에서 쉽게 구체화해줄 것입니다. 댄과 함께 재미있는 Sass 여행을 떠나보세요.

독자 여러분의 많은 관심과 격려 바랍니다.

웹액츄얼리 북스팀

저자 인사말

《웹디자이너를 위한 SASS》를 한국에서 출간하게 되어 기쁩니다. 솔직히 CSS는 어렵습니다. 지금 작성하고 있는 스타일시트는 예전보다 더 복잡하죠. 우리는 그 명세서를 작업에 필요한 한 많이 변경하고 있습니다. 댄 시더홈은 Sass로 전향하는 데 열성적이지 않았었지만 인기 있는 CSS 전처리기로 돌아온 이야기를 나누고(항상 해오던 방식대로 일하면서) 스타일시트를 잘 다룰 수 있는 명쾌한 방법을 제시합니다.

제프리 젤드먼, 댄 시더홈

We are pleased to present the publication of *Sass for Web Designers* in Korea. Let's face it: CSS is hard. Our stylesheets are more complex than they used to be, and we're bending the spec to do as much as it can. A reluctant convert to Sass, Dan Cederholm shares how he came around to the popular CSS pre-processor, and provides a clear-cut path to taking better control of your code (all the while working the way you always have).

Jeffrey Zeldman and Dan Cederholm

역자의 글

해외에서는 Sass, LESS, Stylus와 같은 CSS 전처리기를 많이 사용하지만, 국내에서는 그 사례가 드물고 사용경험을 가진 개발자를 찾아보기도 어려웠습니다. 작업방식의 변화에 대한 두려움과 CSS 작성을 도와줄 도구에 대한 필요성을 크게 느끼지 못한 점이 이유가 아닐까 싶습니다.

디자이너이자 개발자인 댄 시더홈은 이런 작업자의 정서를 잘 알고 있습니다. 작업자가 우려하는 부분을 정확히 짚어주어 안심시키면서 Sass를 사용해 손쉽게 누릴 수 있는 장점을 설명해줍니다. Sass의 많은 기능 중에서 의도적으로 Sass의 복잡한 수식과 함수를 배제해주어 Sass를 처음 접하는 사람도 마음 편히 그의 인도에 따라 Sass에 입문할 수 있습니다.

작업에 신기술을 적용하는 것은 개발자의 단조로운 일상에 새로운 활력을 더해주는 단비와도 같습니다. 그 신기술이 매일 반복하는 작업을 편리하게 해주는 것이라면 더할 나위 없겠지요. 여러분도 Sass를 통해 그런 기쁨을 누리시길 바랍니다.

윤원진

현재 NHN Technology Services에서 프론트엔드 개발자로 근무하고 있으며, 블로그와 트위터를 통해 해외의 CSS 소식을 국내에 전파해주는 활동을 하고 있습니다. 2011년 A List Apart에서 주최한 웹앱 공모전인 '10K Apart Contest'에서 최고 기술상(Best Technical Achievement)을 수상했습니다. e스포츠 커뮤니티 사이트인 pgr21.com의 운영진이기도 합니다.

블로그 http://tobyyun.com **트위터** @tobyyun

감수자 인사말

CSS는 배우기 쉽고 재미있는 언어이지만 구조가 단순하고 코드가 반복됩니다. CSS3의 등장과 웹사이트의 개발 범위가 모바일까지 확장되면서 CSS 코드는 점점 복잡해지고 관리도 어려워지고 있습니다. 지난 10여 년간 CSS를 능숙하게 다룬 댄 시더홈은 처음에는 Sass를 적극적으로 받아들이지 않았지만, 지금은 Sass를 프로젝트에 도입하라고 권장합니다. 도대체 무엇이 그의 완고한 마음을 움직여서 Sass를 사용하라고 외치게 만든 걸까요?

'CSS를 확장한 언어'인 Sass는 CSS의 부족한 부분을 채워주는 강력한 도구입니다. 재사용 가능한 모듈을 활용해서 생산성을 높이고 편리하게 유지보수할 수 있습니다. 압축된 스타일 형식으로 출력파일을 만들어서 웹사이트의 성능을 최적화할 수 있습니다. 독자 여러분도 저와 같이 'Sass의 매력'에 빠지리라 믿습니다. 이 책은 국내에서 'Sass의 대중화 바람'을 불러일으킬 '첫 날갯짓'이 될 것입니다. 여러분도 동참하시길 바랍니다.

야무(지훈)

강원대학교 산업디자인과(시각디자인 전공)를 졸업했고, 실무자를 대상으로 한 교육 현장의 최전선에서 진보적이고 혁신적인 테크닉과 노하우 전수에 공들이고 있습니다. 《만들면서 배우는 HTML5+CSS3+jQuery》, 《만들면서 배우는 모던 웹사이트 디자인》 등을 저술했습니다. 매년 세미나를 통해 웹 분야의 새로운 트렌드와 이슈를 발표하고 공유하는 것을 낙으로 삼고 있으며, 업무의 생산성과 효율성 등을 극대화하는 새로운 디자인과 개발 테크닉에 관심이 많습니다.

서문

컴퓨터 언어들의 진화 과정을 되돌아보면 대략 12년마다 새로운 추상화abstraction[1] 계층이 나오는 듯합니다. '1과 0'에서 어셈블리어로 상향 조정되면서 컴파일 언어로 한층 올라갔습니다. 이 컴파일 언어들은 진화했고, 우리는 이것으로 웹 브라우저를 만들었습니다. 웹 브라우저들은 HTML, CSS, 자바스크립트 같은 언어들을 해석합니다. 이제 한 층 더 올라갈 준비가 되었습니다.

HTML, CSS, 자바스크립트는 지금까지 대단한 성공을 거둔 언어들이며 전례 없는 방식으로 웹을 발전시켰습니다. 현재 우리는 더 크고 복잡한 웹사이트를 만들고 있습니다. 정말 멋지죠. 하지만 다시 새로운 발걸음을 내디뎌야 할 곳에 와 있습니다. 우리는 여기서 관리와 유지보수를 잘할 수 있는 것을 만들어야 합니다. 이것은 추상화로 가능합니다.

CSS가 가장 절실히 추상화를 필요로 합니다. 오늘날 HTML은 흔히 백엔드 코드와 템플릿으로 만들어지고, 거기에는 우리가 필요로 하는 추상화가 적용되어 있습니다. 프로그래밍 언어인 자바스크립트에는 이미 추상화 기능이 내장된 툴들이 있습니다. 하지만 CSS에는 추상화 개념이 없고 코드의 반복만 있습니다. 이런 이유로 CSS가 널리 채택되고 사용되었지만 이제는 관리하기가 어려워졌습니다. CSS가 한 단계 올라갈 차례네요!

1 추상화: 객체와 프로시저들의 공통 특질들을 골라내는 과정. 소프트웨어 공학에서 가장 중요한 기법 중 하나로 2개의 다른 기법인 캡슐화 및 데이터 은폐와 밀접한 관련이 있다. 이 세 가지 기법은 복잡성을 줄이기 위해 사용한다.

이 책에서 댄 시더홈이 가르쳐줄 Sass는 우리가 필요로 하는 추상화 툴들을 모두 가지고 있습니다. 반복되는 값들은 변수가 됩니다. 반복되는 스타일 구문들은 extend가 됩니다. 복잡한 규칙 집합들rulesets과 장황하게 쓰인 브라우저 개발사 접두어vendor-prefixed 들은 믹스인mixin이 됩니다. 그래서 CSS 코드의 규모가 커지더라도 수월하게 관리하고 유지보수할 수 있습니다.

Sass로 옮겨가는 과정이 쉽지 않은 분들이 있습니다. 댄도 그것을 너무나 잘 알고 있습니다. 제가 `div`를 알기 전부터 댄은 CSS를 구현하는 일을 해왔고, 이것을 전 세계에 널리 가르쳐왔습니다. 하지만 그는 CSS 전문가일 뿐만 아니라 웹 분야의 장인입니다. 능숙한 목공기사는 자신의 끌이 무뎌진 때를 압니다. 댄은 CSS를 직접 작성해서 일하는 요즘 마치 끌이 무뎌진 때와 같다는 것을 알고 있었습니다. 물론 여러분이 직접 CSS를 작성하고 싶다면 그렇게 해도 좋습니다. 다만 그 선택의 책임은 본인에게 있습니다.

이 책을 다 읽고 Sass를 첫 프로젝트에 실제로 적용한 뒤 완료할 즈음이면 여러분은 Sass의 가장 중요하고 진정한 가치가 담긴 부분의 95%를 통달하게 될 것입니다. 댄을 안내자로 삼으세요. Sass는 업무를 더 힘들게 하지 않습니다. 오히려 더 쉬워지는 것을 알게 될 것입니다.

— 크리스 코이어

1

왜 SASS인가?

저는 Sass의 열렬한 신자는 아니었습니다. 직접 손으로 스타일시트를 작성하면 되니까요! 도움이 필요 없다고요! 왜 새로운 작업을 넣어 일을 더 복잡하게 만드나요? 필요 없다니까요!

단지 그런 생각뿐이었어요. 하지만 Sass(와 다른 CSS 전처리기들 CSS pre-processors[2])는 든든한 친구가 될 수 있습니다. Sass는 스타일 작성자라면 누구나 작업 과정에 쉽게 넣을 수 있는 툴입니다. 생각을 바꾸는 데 시간이 걸리긴 했지만 이렇게 변화되어 정말 기쁩니다.

그래서 이 작은 책을 쓰고 싶었습니다. 지난 10년간 능숙해진 CSS 작업 과정을 유지하면서도 Sass를 이용해 어떻게 일을 더 효율적으로

2 CSS 전처리기: Sass, LESS, Stylus와 같은 전처리 언어로 작성된 코드를 일반적으로 사용하는 CSS 코드로 변환해주는 프로그램.

할 수 있었는지 여러분과 나누고자 합니다. Sass에 대해 오해를 많이 했기 때문에 처음에는 시도조차 하지 않았습니다. 스타일시트를 작성하고 관리해온 방식을 완전히 바꿔야 하나 싶어 걱정이 앞섰습니다. 가끔 CSS가 웹 브라우저에서 깨져 보일 수 있기 때문에 CSS 작성자들이 자신들의 작업 방식을 고수하려는 것도 이해가 갑니다. 믿습니까? 믿으면 '아멘' 하세요!

그래서 제가 여기 있습니다. Sass로 여러분의 작업과 워크플로가 혼란스럽지 않고 삶이 더 편해질 수 있다는 것을 보여드리겠습니다. Sass의 설치와 사용 방법, 프로젝트에서 어떻게 도움이 되었는지 등 Sass의 여러 측면을 설명해드리겠습니다. 잘하면 여러분도 Sass 신자로 만들지 모르겠네요.

SASS 엘리베이터 피치elevator pitch[3]

스타일시트에서 어떤 색상값을 바꿔야 한다고 가정해봅시다. 그 색상값이 여기저기 흩어져 있어 일일이 찾아서 바꿔야 한다면요? 아래와 같은 구문으로 CSS에서 처리했으면 하고 바라지 않을까요?

```
$brand-color: #fc3;

a {
  color: $brand-color;
}
nav {
  background-color: $brand-color;
}
```

3 엘리베이터 피치: 엘리베이터를 타고 가는 짧은 시간에 내용이나 아이디어를 간단하고 빠르게 전달한다는 의미.

한 곳만 값을 변경하고, 그 값이 모든 스타일시트에 반영된다면 어떨까요? Sass로 그렇게 할 수 있답니다!

스타일시트 여기저기에서 반복되는 스타일 블록들은 어떨까요?

```
p {
  margin-bottom: 20px;
  font-size: 14px;
  line-height: 1.5;
}
footer {
  margin-bottom: 20px;
  font-size: 14px;
  line-height: 1.5;
}
```

반복되는 이런 규칙들을 재사용할 수 있는 블록으로 감싸주면 멋지지 않을까요? 한 번 더 말하지만, 여러분은 규칙을 단 한 번만 정의하고 필요한 곳마다 넣으면 됩니다.

```
@mixin default-type {
  margin-bottom: 20px;
  font-size: 14px;
  line-height: 1.5;
}

p {
  @include default-type;
}
footer {
  @include default-type;
}
```

이 또한 Sass랍니다! 앞에서 보여드린 간단한 2개의 예제는 Sass가 스타일시트를 얼마나 쉽고 빠르고 융통성 있게 작성해주는지 살짝 맛본 것에 불과합니다. Sass는 웹 디자인 세계에서 환영받는 친구입니다. 웹사이트를 만들어본 사람이라면 누구나 알 거예요.

CSS는 어렵습니다

솔직히 CSS를 배우는 것은 쉽지 않습니다. 각 속성이 무엇을 하고, 캐스케이드cascade[4]는 어떻게 적용되고, 어떤 브라우저가 무엇을 지원하고, 선택자selector, 쿼크 모드the quirks 등을 이해하기는 쉽지 않습니다. 현재 우리는 CSS에 복잡한 인터페이스를 추가하고, 이에 맞추어 유지보수를 하고…. 잠깐, 왜 이것을 반복해야 하죠? 퍼즐 놀이네요. 물론 마지막 완성을 즐기는 이들도 있지만요.

CSS의 문제점 중 하나는 CSS가 오늘날 우리가 작업하고 있는 것들을 목표로 해서 처음부터 설계되지 않았다는 사실입니다. 브라우저의 혁신, CSS3와 그다음 버전의 구현이 빠르게 진행된 덕분에 CSS는 이에 발맞추어 빠르게 진보하고 있습니다. 하지만 사실상 핵hacks이라는 기술에 여전히 의존할 수밖에 없습니다. 한 예로, `float` 속성은 텍스트 블록 안에서 단지 이미지를 정렬하려고 설계되었습니다. 그뿐이죠. 그런데도 우리는 모든 인터페이스를 웹 페이지 안에 배치하는 데 `float` 속성을 사용하려고 용도를 변경해야 했습니다.

스타일시트가 지나치게 반복적인 것도 문제입니다. 색상, 서체, 자주 사용하는 속성 그룹 등. 전형적인 CSS 파일은 위에서 아래로 작성되는, 전적으로 선형적인 문서입니다. 객체 지향 프로그래밍을 하는 프로그래머가 자신의 머리를 쥐어뜯고 싶어 할 만하죠(저는 객체 지향 프로그래밍을 하지 않지만 머리카락은 거의 남지 않았네요. 해석은 여러분께 맡길게요).

인터페이스와 웹 애플리케이션이 점점 견고하고 복잡해지면서 우리는 CSS를 애초에 설계자들이 전혀 꿈꾸지 않았던 것으로 용도를 변경해 사용하고 있습니다. 그만큼 어찌나 솜씨가 좋은지요. 다행스럽게도 웹이 제기하고 있는 이런 문제점을 해결하기 위해 최근 웹 브라우저 제작사들은 더 효율적이고 강력한 속성과 선택자들로 새로운 CSS 기능들을 신속하게 탑재하고 있습니다. 레이아웃을 위한 새로운 CSS3 옵션들, `border-radius`, `box-shadow`, 고급 선택자advanced selectors[5], 트랜지션transition, 트랜스폼transform, 애니메이션과 같은 기능들입니다. 흥이 절로 납니다. 그럼에도 여전히 CSS 자체에서 많은 부분을 놓치고 있습니다. 채워져야 할 구멍이 많고, 작성자들의 삶도 편안해져야 합니다.

DRY 원칙

소프트웨어 공학의 세계를 자세히 들여다보면(저는 가볍게 훑어보고 만족하는 것보다 좀 더 유심히 살펴보는 편입니다), 복잡한 시스템을 만드는 이들은 구조화organization, 변수variables, 상수constraints 등이 습관처럼 몸에 배어 있고, 이것들이 얼마나 중요한 작업인지 금방 알 수 있습니다.

4 캐스케이드: CSS 선언의 충돌을 피하기 위해서 정해놓은 우선순위.

5 고급 선택자: 기본 선택자(엘리먼트, id, class)보다 더 구체적인 기준으로 엘리먼트를 선택할 수 있는 선택자들. 예: 전체 선택자(Universal Selectors), 속성 선택자(Attributes Selectors), 가상 클래스(Pseudo-classes) 등.

아마 여러분은 '반복되는 코드를 만들지 말라don't repeat yourself'(DRY)는 원칙을 들어본 적이 있을 것입니다. 이것은 앤디 헌트Andy Hunt와 데이브 토머스Dave Thomas가 쓴 《실용주의 프로그래머Pragmatic Programmer》(http://bkaprt.com/sass/1/)에서 처음 제시된 원칙입니다. 이 책에서는 DRY를 다음과 같이 명시하고 있습니다.

> 시스템 안의 모든 지식 조각은 권위 있고 명확한 단 하나의 표현만을 가져야 한다.

이는 중복되어 작성된 코드가 개발자들에게 혼란을 주고 작업 실패의 원인이 된다는 의미입니다(http://bkaprt.com/sass/2/). 일단 공통적으로 반복되는 패턴을 작성해놓고, 그것을 애플리케이션 전체에서 재사용해야 한다는 것 역시 잘 알려진 상식입니다. 이런 방식이 더 효율적이고 유지보수하기가 훨씬 쉽습니다.

CSS에는 DRY를 적용할 수 없습니다. CSS 문서는 반복되는 규칙rules, 선언declarations, 값values들로 채워집니다. 우리는 색상, 서체에 관한 동일한 코드와 자주 사용되는 스타일 패턴을 스타일시트 전체에서 계속 반복해서 작성하고 있습니다. DRY에 익숙한 소프트웨어 개발자는 제대로 작성된 CSS 파일을 한번 보고 처음에 어리둥절해하다가 곧 좌절감에 빠져 흐느낄 것입니다.

"이런! 답답한 코드를 어떻게 유지보수할 수 있죠?"라고 개발자가 물을 것입니다. 그러면 여러분은 "아, 내가 IE 버그에 대해서는 얘기를 했었나요?"라고 씁쓰레하며 대답을 하겠죠.

CSS로 작업하는 것은 왜 어려운가요?

CSS의 공동 창시자인 버트 보스Bert Bos의 글(http://bkaprt.com/sass/3/)을 보면 CSS가 왜 수년간 문법의 한계를 가지고 있을 수밖에 없었는지 이해할 수 있습니다.

> CSS는 프로그래머들이 매크로macros, 변수, 기호 상수symbolic constants, 조건문, 표현식과 같은 프로그래밍 언어에서 사용하는 효과적인 기능들을 받아들이지 않고 멈춰 있습니다. 그런 기능들이 제공된다면 고급 전문가들은 자신이 원하는 대로 작성할 수 있지만, 경험이 부족한 초보자들은 자신도 모르게 포기할 것입니다. 아니, 아예 겁을 먹고 시작도 하지 않을 것입니다. 일종의 균형잡기인 셈이지요. CSS에서의 균형은 다른 언어들과 다릅니다.

CSS의 초기 설계자들은 이런 기능을 도입하는 것에 대해 우려했습니다. 그들은 (공평하게) 가능한 한 많은 사람이 웹사이트를 만들기를 원했습니다. CSS가 웹 페이지에 스타일을 적용하면서 내용과 표현을 분리할 수 있을 정도로 강력하며, 동시에 쉽게 이해되고 사용되기를 바랐습니다. 물론 저는 그들의 생각을 존중합니다. 하지만 일은 더 많아지고 복잡해지며, 코드의 미묘한 차이로 다루기가 까다로워집니다. 또한 우리는 코드를 유지하고 미래의 변화에 대응해야 하는 도전을 받고 있습니다.

다행히 눈앞에 우리를 도와줄 방법들이 있습니다. 그중 하나가 Sass입니다.

SASS란 무엇인가?

Sass는 CSS 전처리기입니다. 여러분이 작성하는 스타일시트와 브라우저에서 해석할 `.css` 파일 중간에 위치하는 하나의 계층입니다. Sass(Syntactically Awesome Stylesheets)는 하나의 언어로서 CSS의 구멍들을 메워주고, DRY 방식을 적용해서 더 빠르고 효율적으로 코드를 작성하고 쉽게 유지보수할 수 있도록 해줍니다(그림1).

Sass 웹사이트(http://bkaprt.com/sass/4/)에서는 Sass에 대해 명료하게 서술하고 있습니다.

> Sass는 CSS 상위에 있는 메타언어meta-language[6]입니다. 단조로운 CSS에 강력한 힘을 주며, CSS 문서의 스타일을 깔끔하고 구조적으로 묘사하는 데 사용합니다. Sass는 보다 간결하고 격식을 갖춘 CSS 문법을 제공하며, 스타일시트를 쉽게 관리할 수 있는 다양한 기능을 구현합니다.

보통 CSS에서는 변수나 믹스인(재사용 가능한 스타일 블록), 그 외의 좋은 기능들을 사용할 수 없지만 Sass에서는 그것 모두, 아니 그보다 더 활용 가능한 문법들이 있습니다. 그 문법들은 표준 CSS에 '최상의 기능성'을 갖춰줍니다. 이후 명령행 프로그램이나 웹 프레임워크web framework 플러그인을 이용해서 Sass 문법을 흔히 사용하는 CSS 파일로 변환하거나 컴파일시킵니다.

조금 더 자세히 말하면 Sass는 CSS3가 확장된 것이며 잠시 후 살펴볼 SCSS('Sassy CSS') 문법은 CSS3의 상위 집합superset입니다. CSS3

6 메타언어: 상위언어라고도 하며, 대상언어(예:CSS)를 정의하고 설명하는 언어.

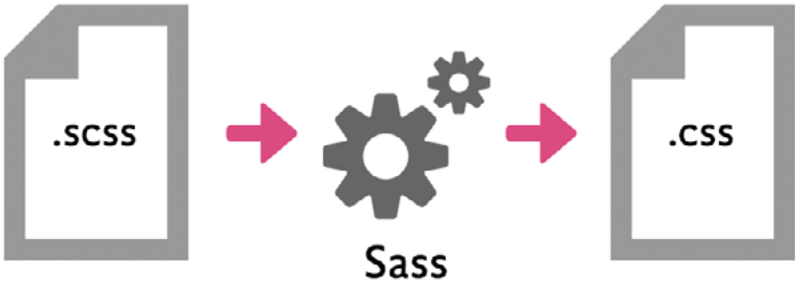

그림 1 Sass는 자신이 가지고 있는 '동적인 문법'들을 흔히 사용하는 CSS로 변환합니다.

문서가 유효하다면 이 또한 SCSS 문서에서도 유효하다는 의미입니다. 이것은 여러분이 쉽게 Sass와 친해지도록 해주는 필수 요소입니다. Sass 문법을 사용해서 무난히 작업을 시작할 수 있고, 적든 많든 원하는 만큼 적용할 수 있습니다. 여러분은 Sass의 기능들을 하나씩 익히고 적용해보면서 기존의 스타일시트 파일을 CSS에서 SCSS로 점차적으로 변환할 수 있습니다.

시간이 흐르고 여러분이 Sass를 능숙하게 작성할 수 있게 되면(그리 오래 걸리지 않을 겁니다) 자연스레 더해진 CSS 확장판처럼 느낄 것입니다. 마치 우리가 메우고 싶었던 CSS의 빈틈을 CSS 명세 자체로 메우는 것처럼. 그런 까닭에 저는 Sass를 사용하기 시작하면서 한 번도 어렵다거나 해야 할 일이 많아져서 번거롭다고 생각해본 적이 없습니다. 여러분도 일단 시작하면 앞으로 계속 Sass를 고수하게 될 것입니다.

더욱이 Sass는 CSS를 더 좋아지게 하고 있습니다. 현재 전처리기의 도움 없이는 불가능한 기능을 빠르게 적용함으로써 CSS 작성자들

은 실제로 그 기능을 구현해보고 테스트할 수 있습니다. 이것이 이 치에 맞다고 생각한다면 Sass의 기능은 미래의 CSS 명세를 잘 알려 주고 있는 것입니다.

SASS 문법

Sass에는 서로 다른 2개의 문법이 있습니다. 먼저 언급했던 SCSS가 최신 문법입니다. SCSS 파일은 `.scss`라는 파일 확장자를 사용합니다. 제가 사용하고 권장하는 문법으로, 그 이유는 다음과 같습니다.

- SCSS는 CSS3의 상위 집합으로 지난 10년간 CSS를 작성하던 방식으로 작성할 수 있으며, 아무런 문제가 없습니다.
- 작업된 스타일시트에 Sass 기능을 적용하여 코드를 점차적으로 변경할 수 있습니다.
- 코드 형식을 변경할 필요가 없습니다.

간단한 SCSS 예제

SCSS 문법이 적용되는 예입니다. 변수를 먼저 정의하고, 그 변수를 CSS 선언에서 사용하고 있습니다.

```
$pink: #ea4c89;

p {
  font-size: 12px;
  color: $pink;
}
p strong {
  text-transform: uppercase;
}
```


위의 구문은 다음과 같이 컴파일됩니다.

```
p {
  font-size: 12px;
  color: #ea4c89;
}
p strong {
  text-transform: uppercase;
}
```

`$pink` 변수를 제외하고 나면 나머지 코드가 매우 친숙하게 보입니다. 변수에 대해서는 이후에 자세히 살펴보도록 하겠습니다. SCSS의 작성 방식은 여러분이 잘 알고 있는 CSS의 작성 방식과 유사합니다. 그래서 저는 SCSS를 정말 좋아합니다.

'들여쓰기'를 하는 초기 Sass 문법

위와 달리 Sass의 초기 문법은 외관상 다릅니다. 몇몇 사람들은 가장 기본적인 것만 남겨두고 중괄호와 세미콜론을 없애며, 들여쓰기한 문법을 더 좋아하기도 합니다. 만약 여러분이 루비Ruby나 파이선Python 같은 간결한 프로그래밍 언어를 사용해본 적이 있다면 Sass 문법이 좀 더 친숙하고 편하게 느껴질 것입니다.

다음은 Sass 초기 문법의 간단한 예입니다. 아래의 코드는 전에 살펴보았던 SCSS 코드와 완전히 동일한 코드로 컴파일될 것입니다.

```
$pink: #ea4c89

p
  font-size: 12px
```

```
    color: $pink

p strong
  text-transform: uppercase
```

괄호와 세미콜론이 사라지고, 공백과 들여쓰기로 선언들의 구조를 알려줍니다. 비교적 깔끔하고 간단하기 때문에 이런 문법 작성 방식에 마음이 끌리는 분들이 있을 것입니다. 이 방식으로 초기에 코드를 빨리 작성할 수 있으며, 다른 지저분한 코드를 깔끔하게 정리할 수 있습니다. 그럼에도 저는 앞서 말했던 이유로 여전히 CSS 정렬 방식과 유사한 SCSS를 선호합니다.

이후 챕터에 나오는 예제들은 SCSS 문법을 사용합니다. 여러분이 간결한 Sass 문법을 더 좋아한다면 쉽게 바꿔 쓸 수 있습니다. 우리가 더 깊이 다루게 될 Sass의 모든 기능은 두 가지 문법으로 적용할 수 있습니다. 어느 것을 쓸 것인가는 개인 취향의 문제일 뿐입니다.

SASS에 대한 오해들

앞서 말했듯이 저는 Sass를 사용하는 것을 꺼렸습니다. 수많은 오해 때문이었죠. 루비나 고급 커맨드 라인의 허튼 소리들을 알아야 하나? 스타일시트를 작성해온 나만의 방식을 완전히 바꾸어야 하나? 최종 CSS 코드가 장황하게 늘어나거나 코드의 가독성이 떨어지는 것은 아닐까?

감사하게도 각 질문에 대한 대답은 당연히 "아니오"입니다. 하지만 인터넷의 여러 사이트에서 사람들이 Sass에 대해 언급할 때마다 이런 얘기들이 나옵니다. 이제 오해들을 풀어보도록 하죠.

명령행이 두려워요!

저는 절대로 명령행 전문가가 아닙니다. 다만 수년간 여기저기에서 조금씩 배웠습니다. 꽤 애를 먹긴 했지만요. 저는 명령행에서 파일 시스템의 여러 디렉터리로 이동하거나 Git 명령어 등을 사용하는 것을 두려워하지 않습니다.

물론 그런 작업을 원하지 않는 디자이너와 프론트엔드 개발자들의 마음을 이해할 수 있습니다. 그중 명령행 공포증이 있는 사람도 있을 것입니다. Sass에서 해야 할 명령행 작업은 거의 없습니다. 솔직히 명령어 단 한 줄이 여러분이 배워야 할 것입니다. 게다가 명령행을 필요 없게 하는 앱과 웹 프레임워크도 있습니다(다음 장에서 소개하겠습니다).

단지 명령행을 사용하고 싶지 않다는 이유만으로 Sass를 포기하지 마세요!

CSS 작성 방식을 바꾸고 싶지 않아요!

저도 겪었던 오해입니다. 저는 스타일시트를 새로 만들고 구조화하는 것에 까다로운 편입니다. 꽤 공을 들여 문서를 작성합니다. 하지만 기억하세요. SCSS 문법은 CSS3의 상위 집합이므로 평소 CSS를 작성하던 방식을 변경할 필요가 없습니다. `.scss` 파일에서 작업할 때는 주석을 달거나 들여쓰기를 하든 안 하든 여러분이 좋아하는 형식을 그대로 작성할 수 있습니다. 이것을 깨닫고 나니 저는 안심

하고 더 깊이 몰두할 수 있었습니다.

Sass 때문에 디자인 방식을 바꾸고 싶지 않아요!

한편 Sass는 여러분이 가지고 있는 모든 문제를 해결해주거나 여러분의 좋지 않은 습관을 고쳐주지는 않습니다. 비효율적이고 장황하게 늘어진 스타일시트는 Sass를 사용하더라도 여전히 비효율적이고 장황하게 늘어져 있을 수 있습니다. Sass에도 잘 짜인 구조와 현명한 생각이 필요합니다. 솔직히 Sass로 인해 오히려 안 좋은 습관을 키우는 사례도 있습니다. 그것에 대해서도 조금 짚고 넘어가겠습니다. 하지만 올바르고 똑똑하게 사용한다면 Sass는 웹사이트를 제작하는 데 굉장히 좋은 조력자가 될 수 있습니다.

그럼 더 이상의 오해들은 떨쳐버리고 재미있게 놀아봅시다. Sass가 할 수 있는 것을 보면 깜짝 놀랄 것입니다. 다음 장에서 Sass가 여러분의 작업 과정과 얼마나 잘 맞는지, 명령행이나 앱을 얼마나 쉽게 이용할 수 있는지 살펴볼 수 있는 워크플로를 만들어보겠습니다. 여러분, Sass 하러 가실까요?

2 SASS 워크플로

Sass란 무엇인가에 대해 감을 잡았으므로 이제 환경을 설정하고 Sass를 사용해봅시다. 여러분이 해야 할 첫 번째 임무는 컴퓨터에 Sass를 설치하는 것입니다. Sass는 루비Ruby 언어[7]로 쓰인 프로그램이고, Sass의 기본 문법을 CSS로 변환해준다고 1장에서 언급했습니다. 그러므로 Sass는 사용하기 전에 설치부터 해야 합니다.

맥에서 SASS 설치하기

맥 컴퓨터를 사용하고 있다면(만세! 여러분은 정말 운이 좋네요) Sass를 설치하는 것이 이보다 더 간편할 수는 없을 것입니다. Mac OS X에는 루비가 이미 설치되어 있고, Sass는 루비의 'gem'[8]으로 패키지

7 루비 언어: 마츠모토 유키히로가 개발한 순수 객체지향 스크립트 프로그래밍 언어.
8 gem: 루비에서 지원하는 패키지 시스템. Gem 명령어를 사용해 인터넷에서 자동으로 필요한 프로그램을 다운로드해서 설치 및 관리합니다.

인덱스

@content 101

@extend 83

@import 규칙 77

ㄱ

그림자 믹스인 box-shadow mixin 71

ㄴ

네임스페이스 속성 중첩 nesting name-spaced properties 49

ㄷ

다중 @extend multiple@extend function 87

데이브 토머스 Thomas, Dave 18

둥근 모서리 믹스인 border-radius mixin 69

들여쓰기한 문법 indented syntax 23

ㄹ

레티나 이미지 HiDPI images 107

레티나 화면 retina screens 106

ㅁ

맥에서 Sass 설치하기 installing Sass on a Mac 27

미디어 쿼리 media queries 93

미디어 쿼리 중첩 nested media queries 94

믹스인 mixins 60

믹스인 라이브러리 mixin library 77

믹스인 안의 믹스인 mixins inside mixins 115

믹스인 인자 mixin with arguments 63

ㅂ

반복되는 코드를 만들지 말라 원칙 DRY (don't repeat yourself) principle 18

버번 Bourbon 82

버트 보스 Bos, Bert 19

변수 variables 55

변수 정의 defining variables 55

부모 선택자 참조 parent selectors, referencing 50

ㅅ

소트봇 thoughtbot 82

스콧 젤 Jehl, Scott 107

스타일 가이드 style guides 56

스타일 덮어쓰기 overrides 52

실용주의 프로그래머 *Pragmatic Programmer, The* 18

ㅇ

압축된 스타일 compressed style 41

Sass용 애플리케이션 apps for Sass 32

앤디 헌트 Hunt, Andy 18

윈도우에서 Sass 설치하기 installing Sass on Windows 29

ㅈ

주석 달기 commenting 54

중첩 스타일 nested style 39

지나 볼턴 Bolton, Jina 56

ㅊ

축약된 스타일 compact style 40

출력 형식 output formatting 38

ㅋ

컬러 기능 color functions 57

콤파스 프레임워크 Compass framework 81

크리스 엡슈타인 Eppstein, Chris 81

ㅍ

파일 정리 file organization 31

플레이스홀더 선택자 placeholder selectors 88

픽처필 Picturefill 107

ㅎ

해상도 분기점 breakpoints, setting 96

확장된 스타일 expanded style 39

CSS 규칙 안에서 중첩 nesting CSS rules 46

CSS3 그레이디언트 CSS3 gradients 72

CSS 변수 variables in CSS 59

retinize 믹스인 retinize mixin 110

Sass 문법 Sass syntax 22

Sass 엘리베이터 피치 elevator pitch for Sass 14

Sass란 무엇인가 definition of Sass 20

watch 명령어 watch command 32

@content 101

@extend function 83

multiple 87

@import rule 77

A

apps for Sass 32

B

breakpoints, setting 96

Bolton, Jina 56

border-radius mixin 69

Bos, Bert 19

Bourbon 82

box-shadow mixin 71

C

color functions 57

commenting 54

compact style 40

Compass framework 81

compressed style 41

concatenation 112

CSS3 gradients 72

D

DRY (don't repeat yourself) principle 18

E

elevator pitch for Sass 14

Eppstein, Chris 81

expanded style 39

F

file organization 31

H

HiDPI images 107

Hunt, Andy 18

I

indented syntax 23

installing Sass

- on a Mac 27
- on Windows 29

J

Jehl, Scott 107

M

media queries 93

mixins 60

- library of 77
- inside mixins 115
- with arguments 63

N

nested style 39

nesting

- CSS rules 46
- media queries 94
- namespaced properties 49

O

output formatting 38

overrides, inserting 52

P

parent selectors, referencing 50

Picturefill 107

placeholder selectors 88

Pragmatic Programmer, The 18

R

retina screens 106

retinize mixin 110

S

Sass

- definition of 20
- syntax 22

SCSS ("Sassy CSS") 20

style guides, creating 56

T

Thomas, Dave 18

thoughtbot 82

V

variables 55

- defining 55
- in CSS 59

viewports, multiple 101

W

watch command 32