rayleighko add file

1 contributor

299 lines (216 sloc) | 9.9 KB

Sass

Sass는 Syntactically Awesome StyleSheets의 약자로 CSS의 한계와 단점을 보완하기 위한 CSS 전처리기(pre-processor)입니다. Sass 전처리기는 웹에서 CSS가 동작하기 전에 작동하며 기존 CSS보다 가독성이 높고 코드의 재사용에 유리한 CSS를 생성하기 위해 사용할 수 있도록 도와줍니다. 따라서 Sass는 CSS의 확장기능(extension)이라고 할 수 있습니다.

CSS는 문법이 간결하고 배우기 쉽기에 작은 규모의 프로젝트나 프로젝트 초기에는 탁월한 선택일 수 있습니다. 하지만, 대규모 프로젝트나 중규모 이상으로 커져가는 프로젝트에서는 수정이 빈번하게 발생하고, 그에 따라 기능의 추가 및 UI의 개선이 필수불가결한 상황에서는 탁월하지 않을 수 있습니다.

더불어 Agile 개발 방법론이 확산됨에 따라 Agile적인 조직에서는 CSS만으로 프로젝트를 운영하기에 어려움이 있습니다. 이를 위해 Sass는 다음과 같은 기능을 제공해 CSS를 보다 효율적으로 사용할 수 있도록 도와줍니다.

- Variables
- Nesting
- Partials
- Modules
- Mixins
- Extend/Inheritance
- Operators

SCSS

각 기능을 설명하기 전에 SCSS를 알아야 합니다. SCSS는 Sassy CSS의 약자로, Sass 버전 3부터 사용할 수 있습니다. Sass만으로는 CSS와의 호환이 어렵고 CSS와 같이 사용할 경우 혼란이 생길 수 있기 때문에 SCSS를 통해 Sass의 문법과 css의 문법을 합친 CSS의 Superset입니다. `.scss` 확장자를 사용해 선언합니다(Sass는 `.sass` 확장자를 사용).

> 이 문서에서는 Sass와 SCSS를 같은 의미로 보며, SCSS의 문법과 `.scss` 확장자를 사용합니다.

Sass 공식 문서에 기록된 내용을 바탕으로 각 기능에 대해 설명하면 다음과 같습니다.

Variables

변수(Variables)는 스타일 시트 전체에서 재사용하려는 저장하는 탁월한 방법입니다. 색상, 글꼴 또는 재사용하려는 CSS 요소와 같은 것을 저장할 수 있습니다. Sass는 $ 기호를 사용하여 변수를 만듭니다. CSS와 비교하면 다음과 같이 사용할 수 있습니다.

- SCSS

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

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

- CSS

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

Sass가 처리 될 때 $font-stack 및 $primary-color에 정의한 변수를 가져 와서 CSS에 배치된 변수 값과 함께 일반 CSS를 출력합니다. 변수는 브랜드 색상을 작업하거나 사이트 전체에서 사용되는 일관된 디자인을 사용할 때 매우 유용합니다.

Nesting

평소에 HTML을 작성할 때 분명 중첩과 시각적인 계층 구조를 갖고 있다는 것을 알고 있습니다. 반면에 CSS는 그런 중첩과 계층 구조를 명시적으로 나타내지 않습니다.

Sass를 사용하면 HTML과 동일한 시각적 계층 구조를 따르는 방식으로 CSS Selector를 중첩시킬 수 있습니다. 지나치게 중첩된 규칙(CSS의 문법)은 유지보수가 어렵고 일반적으로 나쁜 관행으로 간주됩니다. CSS와 비교하면 다음과 같이 사용할 수 있습니다.

- SCSS

```scss
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

- CSS

```css
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

위의 예에서 ul, li 및 Selector가 nav Selector 안에 중첩되어 있음을 알 수 있습니다. 이것은 CSS를 구성하고 더 읽기 쉽게 만드는 좋은 방법입니다.

Partials

Partials은 다른 Sass 파일에 포함될 수 있는 적은 CSS의 코드 조각을 Sass 파일로 만들 수 있습니다. 이는 CSS를 모듈화하고 보다 쉽게 유지 관리할 수 있는 방법입니다.

_partial.scss와 같은 형태의 파일을 만들어 사용합니다. 밑줄은 파일이 부분 파일일 뿐이며 CSS 파일로 생성되어서는 안된다는 것을 Sass에 알려줍니다. Sass Partials는 @use 규칙과 함께 사용됩니다.

Modules

모든 Sass를 하나의 파일에 작성할 필요는 없습니다. @use 키워드를 사용해 모듈화할 수 있습니다. @use는 다른 Sass 파일을 하나의 모듈로 가져올 수 있습니다. 파일 이름을 기반으로 하는 네임 스페이스를 사용해 다른 Sass 파일의 변수, 믹스 인 및 함수를 참조할 수 있습니다. CSS와 비교하면 다음과 같이 사용할 수 있습니다.

- SCSS

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

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

```scss
// styles.scss
@use 'base';

.inverse {
  background-color: base.$primary-color;
  color: white;
}
```

- CSS

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

.inverse {
  background-color: #333;
  color: white;
}
```

style ㄴ.scss에서 @use 'base'를 사용하고 있는데, base.scss가 아닌 base를 사용한 이유는 파일 확장자를 포함시킬 필요가 없디 때문입니다.

Mixins

CSS를 사용할 때 여러 브라우저를 지원하기 위해서는 각 공급자에 맞는 접두사를 사용해야 합니다. Sass를 사용하면 믹스 인을 사용해 서비스 전체에서 재사용할 CSS 선언 그룹(groups of CSS declarations)를 만들어 관리할 수 있습니다. 믹스 인을 보다 유연하게 만들기 위해 값을 전달해 하나의 함수처럼 사용할 수도 있습니다. CSS와 비교하면 다음과 같이 사용할 수 있습니다.

- SCSS

```scss
@mixin transform($property) {
  -webkit-transform: $property;
  -ms-transform: $property;
  transform: $property;
}

.box {
  @include transform(rotate(30deg));
}
```

- CSS

```css
.box {
  -webkit-transform: rotate(30deg);
  -ms-transform: rotate(30deg);
  transform: rotate(30deg);
}
```

믹스 인을 만들기 위해서는 @mixin 문법을 사용하고 믹스 인의 이름을 지정해야 합니다. 또한 괄호 안에 변수 $property를 사용하여 함수의 매개변수처럼 사용했습니다. 믹스 인을 만든 후에는 @include로 시작하여 믹스 인 이름을 사용하는 CSS 문법처럼 사용할 수 있습니다.

Extend/Inheritance

확장과 상속은 Sass의 가장 유용한 기능 중 하나입니다. @extend를 사용하면 한 Selector에서 다른 Selector로 원하는 CSS 속성을 공유 할 수 있습니다. 아래 예제에서는 'extend', 'placeholder 클래스'와 함께 사용되는 다른 기능을 사용하여 오류, 경고 및 성공에 대한 간단한 일련의 메시지를 작성합니다. placeholder 클래스는 확장시에만 인쇄되는 특수한 유형의 클래스이며 컴파일된 CSS를 깔끔하게 유지하는 데 도움이됩니다.

- SCSS

```
/* This CSS will print because %message-shared is extended. */
%message-shared {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

// This CSS won't print because %equal-heights is never extended.
%equal-heights {
  display: flex;
  flex-wrap: wrap;
}

.message {
  @extend %message-shared;
}

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

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

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

- CSS

```
/* This CSS will print because %message-shared is extended. */
.message, .success, .error, .warning {
  border: 1px solid #ccc;
  padding: 10px;
  color: #333;
}

.success {
  border-color: green;
}

.error {
  border-color: red;
}

.warning {
```

```
  border-color: yellow;
}
```

위의 코드는 .message, .success, .error, .warning Selector가 %message-shared처럼 동작하도록 만듭니다. 즉, %message-shared가 표시되는 모든 위치인 .message, .success, .error, .warning도 나타납니다. 이런 마법은 생성 된 CSS에서 발생하며,이 클래스들 각각은 %message-shared와 동일한 CSS 속성을 갖습니다. 이렇게하면 HTML 요소에 여러 클래스 이름을 반복해서 쓰지 않아도 됩니다.

Sass의 placeholder 클래스뿐만 아니라 대부분의 간단한 CSS Selector를 확장 할 수 있지만 placeholder를 사용하는 것이 스타일의 다른 곳에 중첩 된 클래스를 확장하지 않도록 하는 가장 쉬운 방법입니다.

> %equal-heights는 확장되지 않으므로 %equal-heights의 CSS는 생성되지 않습니다.

Operators

Sass에는 +,-, *, / 및 %와 같은 소수의 표준 수학 연산자가 있습니다. 아래 예제에서는 aside와 article의 너비를 계산합니다.

- SCSS

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

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

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

- CSS

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

article[role="main"] {
  float: left;
  width: 62.5%;
}
```

```
aside[role="complementary"] {
  float: right;
  width: 31.25%;
}
```

우리는 960px를 기반으로 매우 간단한 유동 격자(fluid grid)를 만들었습니다. Sass에서의 작업을 통해 픽셀 값을 가져와 쉽게 백분율로 변환 할 수 있습니다.

이외에도 for(@for)문이나 While(@while)문 등을 포함해 여러 기능들을 사용할 수 있습니다.

더 많은 정보는 Sass 공식 문서를 참고합시다.

Less

Less는 Sass와 거의 같지만, Sass의 일부 기능이 빠져있습니다. 따라서 Sass를 이해하고 사용할 수 있다면, Less를 사용하는데 무리가 없을 것입니다. 따라서 이 문서에서는 Less에 대해 설명하지 않습니다.

더 많은 정보를 원한다면 Less 공식 홈페이지를 참고합시다.