SCSS Style Guide

일반적인 CSS Formatting 규칙 / 스타일 가이드를 사용합니다.

폴더 구조

```
scss/
  - base/
    - _reset.scss
    - _typography.scss
    ...
  - components/
    - _buttons.scss
    - _dropdown.scss
    ...
  - layout/
    - _navigation.scss
    - _header.scss
    - _footer.scss
    - _sidebar.scss
    ...
  - pages/
    - _home.scss
    - _contact.scss
    ...
  - utils/
    - _variables.scss
    - _functions.scss
    - _mixins.scss
    - _helpers.scss
    ...
  - vendors/
    - _bootstrap.scss
    - _jquery-ui.scss
    ...
- main.scss
```

루트 레벨에는 main.scss 하나만 있습니다. 이 파일은 `@import`와 주석 외에는 아무 것도 포함하지 않습니다.

1. vendors
2. utils
3. base
4. layout

5. components

6. pages

의 순서대로 파일들을 불러옵니다.

```
/**
* - 파일 확장자와 앞에 붙는 언더스코어는 생략합니다.
* - 가독성을 위해 한 폴더로부터 마지막 import 다음에는 간격을 줍니다.
*/
@import 'vendors/bootstrap';
@import 'jquery-ui';

@import 'utils/variables';
@import 'utils/mixins';
@import 'utils/functions';

@import 'base/reset';
@import 'base/typography';

@import '...';
```

구문 & 서식

문자열

숫자

색상

List

Javascript의 Array와 유사합니다.

```
// Sample
$font-list: ('NotoSansKR', 'Arial', sans-serif);

$font-list: (
  'NotoSansKR',
```

```
  'Arial',
  sans-serif,
);
```

Map

Javascript의 Object와 유사합니다.

```
// Sample
$breakpoint: (
  'small': 767px,
  'medium': 1023px,
  'large': 1200px,
);
```

Nesting

@at-root (중첩 벗어나기)

중첩에서 벗어나고 싶을 때 `@at-root` 키워드를 사용합니다.

중첩 안에서 생성하지만 중첩 밖에서 사용해야 하는 경우에 유용합니다.

```
// Sample
.list {
  $w: 100px;
  $h: 200px;
  li {
    width: $w;
    height: $h;
  }
  @at-root .box {
    width: $w;
    height: $h;
  }
}

// Error
.box {
  width: $w;
  height: $h;
}
```

중첩된 속성

`margin-`, `padding-` 등과 같이 동일한 네임 스페이스를 가지는 속성들을 다음과 같이 사용할 수 있습니다.

```
// Sample
.box {
  margin: {
    top: 10px;
    left: 20px;
  }
  padding: {
    right: 30px;
    bottom: 40px;
  }
}
```

Naming

주석

```
// 컴파일 되지 않는 주석

/*
컴파일 되는 여러 줄 주석
*/
```

변수

기본적으로 수정 될 일이 없거나 한 군데에서만 사용되는 변수를 선언할 필요가 없습니다.

- 값이 적어도 두 번 이상 반복된다.
- 값이 적어도 한 번은 수정될 가능성이 크다.
- 변수 이름 앞에는 항상 `$`를 붙입니다.
- 선언된 블록(`{ }`) 내에서만 유효범위를 가집니다.

`!global` 플래그를 사용하면 변수의 유효범위를 전역(Global)로 설정할 수 있습니다.

같은 이름의 변수가 있을 경우 값이 덮어져 사용될 수 있습니다.

`!default` 플래그는 할당되지 않은 변수의 초기값을 설정합니다.

할당되어있는 변수가 있다면 기존 할당 값을 사용합니다.

```
// SCSS
$color-primary: red;
```

```
.box {
  $color-primary: blue !default;
  background-color: $color-primary;
}
```

```
// CSS Compiled
.box {
  background-color: red;
}
```

`#{}`을 이용해서 코드의 어디든지 변수 값을 넣을 수 있습니다.

`unquote()` 는 Sass의 내장함수로 문자에서 따옴표를 제거해줍니다.

```
$family: unquote('Droid+Sans');
@import url('http://fonts.googleapis.com/css?family=#{$family}');
```

Mixin

반복되는 CSS속성들의 그룹들을 mixin으로 넣을 수 있습니다.

여러가지 로직들을 넣어서 여러가지 기능을 하는 믹스인을 만들고 싶어지더라도 간결성을 유지해주세요.

```
// float을 해제하는 mixin
@mixin clearfix {
  &::after {
    display: block;
    content: '';
    clear: both;
  }
}
```

인수(Arguments)

Mixin은 함수처럼 인수를 가질 수 있습니다. 자바스크립트와 비슷한 면이 많습니다.

```

```

인수 기본값 설정

인수에 기본값을 설정할 수 있습니다.

`@include`를 하면서 인수를 넘겨주지 않을 경우 기본값을 사용합니다.

```

```

키워드 인수

Mixin에 전달할 인수를 입력할 때 변수이름을 입력하여 넘겨줄 수 있습니다.

입력 순서와 상관없이 사용할 수 있어 편리합니다. 전달되는 인수가 없어도 적용될 수 있도록 기본값을 설정해주면 좋습니다.

```
@mixin position($p: absolute, $t: auto, $b: auto, $l: auto, $r: auto) {
  position: $p;
  top: $t;
  bottom: $b;
  left: $l;
  right: $r;
}

.absolute {
  @include position($t: 20px, $l:50px);
}

.fixed-center {
  @include position($p: fixed, $t: 20px, $l: 0);
}
```

가변 인수

넘겨줄 인수의 개수가 확실치 않을 경우 사용합니다.

가변인수는 파라미터 뒤에 `...`을 붙입니다.

```
@mixin font($family: sans-serif, $size: 16px, $weight: normal, $style: normal) {
  font: {
    family: $family;
    size: $size;
    weight: $weight;
    style: $style;
  }
}

.sample-01 {
```

```
  // 순서와 개수에 맞게 전달
  $font-value: 'NotoSansKr', 20px, bold, underline;
  @include font($font-value...);
}

.sample-02 {
  // 필요한 값만 키워드 인수로 변수에 담아 전달
  $font-value: (style: underline, size:20px);
  @include font($font-value...);
}

.sample-03 {
  // 필요한 값만 키워드 인수로 전달
  @include font((weight: bold, size: 20px)...);
}
```

@content

`@content`를 이용하여 기존 mixin이 가지고 있는 기능에 스타일블록을 전달하여 선택자나 속성 등을 추가할 수 있습니다.

```
@mixin icon($url) {
  &::after {
    content: $url;
    @content;
  }
}

.icon-sample-01 {
  // mixin 사용
  @include icon('/assets/images/icon/bullet.png');
}

.icon-sample-02 {
  // mixin 에 스타일 블록을 추가
  @include icon('/assets/images/icon/bullet.png') {
    position: absolute;
  }
}
```

@include

@extend

조건문

- 필요한 경우가 아니라면 괄호는 생략
- `@if` 앞에는 빈 새 줄 하나.
- 여는 중괄호(`{`) 뒤에는 줄 바꿈
- `@else` 문은 이전의 닫는 중괄호(`}`)와 같은 줄에.
- 다음 줄이 닫는 중괄호(`}`)가 아닌 마지막 닫는 중괄호(`}`) 뒤에는 빈 새 줄 하나.

```
@if $support-legacy {
  // ...
} @else {
  // ...
}
```

거짓 값을 테스트 할 때는 `false` 또는 `null` 대신 `not` 키워드를 사용합니다.

```
// Yes
@if not index($list, $item) {
  // ...
}

// No
@if index($list, $item) == null {
  // ...
}
```

어떤 조건에 따라 다른 결과를 반환하는 함수 안에서 조건문을 사용할 때는 조건문 블록 밖에서도 `@return`문을 갖도록 합니다.

```
// Yes
@function sample($flag) {
  @if $flag {
    @return true;
  }

  @return false;
}

// No
@function sample($flag) {
  @if $flag {
    @return true;
  } @else {
    @return false;
```

```
  }
}
```

반복문

Each

리스트나 맵을 이용하기에 수월합니다.

```
// Sample
@each $theme in $themes {
  .section-#{$theme} {
    background-color: map-get($colors, $theme);
  }
}
```

맵에서 반복할 때, 일관성을 갖기 위해 `$key`와 `$value`를 변수이름으로 사용합니다.

```
// Sample
@each $key, $value in map {
  .section-#{$key} {
    background-color: $value;
  }
}
```

For

`:nth-*` 가상 클래스를 사용할 때에 가장 유용합니다.

```
// Sample
@for $i from 1 through 10 {
  .foo:nth-child(#{$i}) {
    margin-left: 10px * $i;
  }
}
```

While

반복문을 중단 시킬 방법이 없어요.

`@each`문과 `@for`문을 사용합니다. 쓰지 마세요.

경고와 오류

`@debug`

`@warn`

`@error`

@extend

```
.foo {
  @extend %bar;
}
```

이 클래스가 다른 곳에 있는 규칙을 상속받았다는 것을 바로 알 수 있게 하는 것이 좋습니다.

`@extend`를 사용할 때에는 다음과 같은 문제를 고려해야 합니다.

- 내 현재 선택자가 어디에 첨부될 것인가?
- 원치 않는 부작용이 초래될 수도 있는가?
- 이 한 번의 확장으로 얼마나 큰 CSS가 생성되는가?

막강한 기능만큼 부작용이 일어날 수 있으니 mixin 기능을 활용합니다.

@include을 그 다음에 사용합니다.

```
.foo {
  @extend %bar;
  @include transition(all 0.3s ease-out);
  ...
}
```

참조를 위해 상단 근처에 있는 것이 좋지만 재정의를 허용합니다.

일반 스타일 목록을 다음에 작성합니다.

```
.foo {
  @extend %bar;
  @include transition(all 0.3s ease-out);
  background-color:#c0ffee;
  ...
}
```

@extends 및 @includes 다음에 일반 스타일을 추가하면 필요한 경우 해당 속성을 재정의 할 수 있습니다.

VW

```
$maxpc-max-wid: 1920;
$pc-max-wid: 1440;
$mo-max-wid: 720;
```

```
// vw conversion PC 1440
@mixin vw-pc($property, $pc-pixel) {
  @if type-of($pc-pixel) == number {
    #{$property}: $pc-pixel * 1px;
    @media screen and (max-width:$pc-max-wid*1px) {
      #{$property}: $pc-pixel * 100 / $pc-pixel * 1vw;
    }
  }
  @else if type-of($pc-pixel) == string {
    #{$property}: auto;
    @media screen and (max-width:$pc-max-wid*1px) {
      #{$property}: auto;
    }
  }
  @else if type-of($pc-pixel) == list {
    $px: ();
    $vw: ();
    @each $value in $pc-pixel {
      @if type-of($value) == number {
        $px: append($px, $value * 1px);
        $vw: append($vw, $value * 100 / $pc-max-wid * 1vw);
      }
      @else if type-of($value) == string {
        $px: append($px, auto);
        $vw: append($vw, auto);
      }
    }
    #{$property}: $px;

    $px: ();
    @each $value in $pc-pixel {
      @if type-of($value) == number {
        $px: append($px, $value * 1px);
      }
      @else if type-of($value) == string {
        $px: append($px, auto);
      }
    }
    @media screen and (max-width: $pc-max-wid * 1px) {
      #{$property}: $vw;
    }
```

```
  }
}
```

```
// vw conversion PC 1920
@mixin vw-max($property, $pc-pixel) {
  @if type-of($pc-pixel) == number {
    #{$proerty}: $pc-pixel * 1px;
    @media screen and (max-width: $maxpc-max-wid * 1px) {
      #{$property}: $pc-pixel * 100 / $maxpc-max-wid * 1vw;
    }
  }
  @else if type-of($pc-pixel) == string {
    #{$proerty}: auto;
    @media screen and (max-width: $maxpc-max-wid * 1px) {
      #{$property}: auto;
    }
  }
  @else if type-of($pc-pixel) == list {
    $px: ();
    $vw: ();
    @each $value in $pc-pixel {
      @if type-of($value) == number {
        $px: append($px, $value * 1px);
        $vw: append($vw, $value * 100 / $maxpc-max-wid * 1vw);
      }
      @else if type-of($value) == string {
        $px: append($px, auto);
        $vw: append($vw, auto);
      }
    }
    #{$property}: $px;

    $px: ();
    @each $value in $pc-pixel {
      @if type-of($value) == number {
        $px: append($px, $value * 1px);
      }
      @else if type-of($value) == string {
        $px: append($px, auto);
      }
    }
    @media screen and (max-width:$maxpc-max-wid * 1px) {
      #{$property}: $vw;
    }
  }
}
```

```
// vw conversion mobile
@mixin vw-mo($property, $mo-pixel) {
  @if type-of($mo-pixel) == number {
    #{$property}: $mo-pixel * 1px;
    @media screen and (max-width:$mo-max-wid*1px) {
      #{$property}: $mo-pixel * 100 / $mo-pixel * 1vw;
```

```
    }
  }
  @else if type-of($mo-pixel) == string {
    #{$property}: auto;
  }
  @else if type-of($mo-pixel) == list {
    $px: ();
    $vw: ();
    @each $value in $mo-pixel {
      @if type-of($value) == number {
        $px: append($px, $value * 1px);
        $vw: append($vw, $value * 100 / $mo-max-wid * 1vw);
      }
      @else if type-of($value) == string {
        $px: append($px, auto);
        $vw: append($vw, auto);
      }
    }
    #{$property}: $px;

    $px: ();
    $vw: ();
    @each $value in $mo-pixel {
      @if type-of($value) == number {
        $px: append($px, $value * 1px);
        $vw: append($vw, $value * 100 / $mo-max-wid * 1vw);
      }
      @else if type-of($value) == string {
        $px: append($px, auto);
        $vw: append($vw, auto);
      }
    }
  }
}
```

```
.class {
  @include vw-mo(padding, 20 30);
}
```