



Cheatsheet is useful weapon that summarize what we should remember.
We summarize various topic into cheatsheet to be able to refer instantly.

Sass

Web CheatSheet for Engineer

Sass(Syntactically Awesome Stylesheets) is a CSS meta language for making CSS structure better. There are two notations, SASS and SCSS.
Readable and maintainable because you can use variables, four arithmetic operations, if and while statement, so Sass boost your development.
This is cheatsheet page about Sass.

Table Of Contents

- 1 How to write SCSS
- 2 variable
- 3 import
- 4 mixins
- 5 inheritance
- 6 operators



☐ nest

The most biggest Sass feature is nested structure.

```
ul {  
  list-style: none;  
  li {  
    a {  
      color: blue;  
      &:hover {  
        text-decoration: underline;  
      }  
    }  
  }  
}
```

```
ul {  
  list-style: none;  
}  
a {  
  color: blue;  
}  
a:hover {  
  text-decoration: underline;  
}
```

☐ Nested class name

Sass allows to nest class name. It is useful when incorporate css design(BEM, SMACCS, OOCSS). [◀ BEM](#)

```
.post-list {  
  list-style: none;  
  &__item {  
    /* some properties */  
  }  
}
```

```
.post-list {  
  list-style: none;  
}  
.post-list__item {  
  /* some properties */  
}
```

variable

☐ Sass variable

Sass allows to define and use variable. It is useful to provide a consistent UI, for example defining frequent color.

```
$red: #ed6b5f;
```

```
.alert {
```



```
.alert {  
  color: $red;  
}
```

```
color: #ed6b5f;  
}
```

import

☐ import

Sass allows to import splitted css files.

```
// _reset.scss  
html,  
body,  
ul,  
ol {  
  margin: 0;  
  padding: 0;  
}
```

```
// base.scss  
@import 'reset';  
  
body {  
  font: 100% Helvetica, sans-serif;  
  background-color: #efefef;  
}
```

```
// base.css  
html,  
body,  
ul,  
ol {  
  margin: 0;  
  padding: 0;  
}  
  
body {  
  font: 100% Helvetica, sans-serif;  
  background-color: #efefef;  
}
```

mixins

☐ mixins

A mixin lets you make groups of CSS declarations that you want to reuse. The good use of a mixin is vendor prefix.



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

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

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

inheritance

○ inheritance

Using @extend lets you share a set of CSS properties from one selector to another. It is likely to inherit in object-oriented programming. It is helpful to let CSS DRY.

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

.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;
}
```

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

.success {
  border-color: green;
}

.error {
  border-color: red;
}

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



○ four arithmetic operations

Doing math in your CSS is useful.

```
.container {  
  width: 100%;  
}  
  
article[role="main"] {  
  float: left;  
  width: 600px / 960px * 100%;  
}  
  
aside[role="complementary"] {  
  float: right;  
  width: 300px / 960px * 100%;  
}
```

```
.container {  
  width: 100%;  
}  
  
article[role="main"] {  
  float: left;  
  width: 62.5%;  
}  
  
aside[role="complementary"] {  
  float: right;  
  width: 31.25%;  
}
```

○ @for

```
@for $i from 1 through 3 {  
  .list-item-#{ $i } { left: 10px * $i;  
}
```

```
.list-item-1 { left: 10px; }  
.list-item-2 { left: 20px; }  
.list-item-3 { left: 30px; }
```

○ @each

```
ems: home about contact;  
  
item in nav-items {  
  #{ $item } {  
    kground: url('images/#{ $item }.png');
```

```
-home {  
  ckground: url('images/home.png');  
  
-about {  
  ckground: url('images/about.png');
```



```
-contact {  
  ckground: url('images/contact.png');
```

○ @while

```
$i: 10;  
@while $i <= 18 {  
  .font-size#{ $i } { font-size: #{ $i }px;  
  $i: $i + 1;  
}
```

```
.font-size14 { font-size: 14px; }  
.font-size15 { font-size: 15px; }  
.font-size16 { font-size: 16px; }  
.font-size17 { font-size: 17px; }  
.font-size18 { font-size: 18px; }
```

- [official - Sass](#)

YouTube



GraphicCodeプログラミングchannel



設計やデザインパターンなど、特定のプログラミング言語に依存しないトピックを【図解】付きで解説するYouTubeチャンネルです。

初級エンジニアが中・上級エンジニアにステップアップするお手伝いできれば嬉しいです。

Share Free



copy link

Related Cheatsheets

[CSS Flexbox](#)-Flexbox(Flexible Box Layout Module) is a CSS3 web layout model. The flex layout allows...

[CSS Selector](#)-[With illustration] This is cheatsheet page about useful css selector. There are adjacent...

