Cheat Sheet for comprensive Sass

Comprehensive Sass Cheat Sheet

1. Introduction to Sass

Sass (Syntactically Awesome Style Sheets) is a CSS preprocessor that extends CSS with features like variables, nested rules, mixins, and more. It helps in writing more maintainable and scalable CSS.

---

2. Installation and Setup

Using npm

```
npm install -g sass
```

Using Homebrew (macOS)

```
brew install sass/sass/sass
```

Using Dart Sass

```
dart pub global activate sass
```

Compiling Sass to CSS

```
sass input.scss output.css
```

---

3. Basic Syntax and Features

Variables

- **Definition**: Store reusable values.
- **Syntax**: `$variable-name: value;`

```
$primary-color: #3498db;
$font-stack: Helvetica, sans-serif;

body {
```

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

Nesting

- **Definition**: Nest selectors inside other selectors.

- **Syntax**:

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

  li { display: inline-block; }

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

Partials and Imports

- **Partials**: Smaller Sass files that can be imported into other Sass files.

- **Syntax**: `_filename.scss` (underscore indicates it's a partial)

- **Import**: `@import 'filename';`

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

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

// main.scss
@import 'base';

.container {
  width: 100%;
```

```
  margin: 0 auto;
}
```

Mixins

- **Definition**: Reusable blocks of styles.
- **Syntax**: `@mixin mixin-name { ... }`
- **Usage**: `@include mixin-name;`

```
@mixin border-radius($radius) {
  -webkit-border-radius: $radius;
     -moz-border-radius: $radius;
      -ms-border-radius: $radius;
          border-radius: $radius;
}

.box { @include border-radius(10px); }
```

Extend/Inheritance

- **Definition**: Share a set of CSS properties from one selector to another.
- **Syntax**: `@extend selector;`

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

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

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

Operators

- **Definition**: Perform basic math operations.
- **Syntax**: `+`, `-`, `*`, `/`, `%`

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

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

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

---

4. Advanced Features

Control Directives

- **Definition**: Control the flow of your styles.
- **Syntax**: `@if`, `@for`, `@each`, `@while`

```
$type: monster;
p {
  @if $type == ocean {
    color: blue;
  } @else if $type == matador {
    color: red;
  } @else if $type == monster {
    color: green;
  } @else {
    color: black;
  }
}
```

Function Directives

- **Definition**: Create custom functions.
- **Syntax**: `@function function-name($parameters) { ... }`

```
@function pow($base, $exponent) {
  $result: 1;
  @for $_ from 1 through $exponent {
    $result: $result * $base;
  }
  @return $result;
}
```

```
.sidebar {
  float: left;
  margin-left: pow(4, 3) * 1px;
}
```

Interpolation

- **Definition**: Inject SassScript into CSS.

- **Syntax**: `#{$variable}`

```
$properties: (margin, padding);
@mixin set-value($side, $value) {
  @each $prop in $properties {
    #{$prop}-#{$side}: $value;
  }
}
.login-box {
  @include set-value(top, 14px);
}
```

Placeholder Selectors

- **Definition**: Create reusable selectors without adding them to the CSS output.

- **Syntax**: `%placeholder-name`

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

.message {
  @extend %message-shared;
}

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

---

5. Best Practices

Organization

- **Partials**: Use partials to break down your styles into smaller, manageable files.
- **7-1 Pattern**: Organize your Sass files into 7 folders and 1 main file.

Performance Tips

- **Avoid Over-nesting**: Deeply nested selectors can lead to overly specific CSS.
- **Use `@extend` Sparingly**: Overuse can lead to bloated CSS.

Debugging

- **Use `@debug`**: Print the value of a variable or expression.
- **Source Maps**: Enable source maps to debug Sass in the browser.

---

6. Common Shortcuts and Tips

Shortcuts

- **Compile Sass**: `sass input.scss output.css`
- **Watch Mode**: `sass --watch input.scss:output.css`
- **Minify CSS**: `sass --style=compressed input.scss output.css`

Tips

- **Use Variables for Colors**: Makes it easier to update colors globally.
- **Leverage Mixins for Vendor Prefixes**: Keep your code DRY.
- **Comment Your Code**: Use `//` for Sass-specific comments that won't appear in the CSS output.

This cheat sheet provides a comprehensive overview of Sass, covering its essential features, advanced functionalities, best practices, and useful tips. Use it as a quick reference to enhance your Sass skills and write more efficient and maintainable CSS.

By Ahmed Baheeg Khorshid

ver 1.0