Cheat Sheet for comprensive LESS

Comprehensive LESS Cheat Sheet

1. Introduction to LESS

LESS is a dynamic stylesheet language that extends CSS with dynamic behavior such as variables, mixins, operations, and functions. It compiles to standard CSS.

---

2. Variables

Variables allow you to store values that can be reused throughout your stylesheet.

Syntax

```
@variable-name: value;
```

Example

```
@primary-color: #4D926F;

#header {
  color: @primary-color;
}

h2 {
  color: @primary-color;
}
```

---

3. Mixins

Mixins allow you to embed all the properties of a class into another class by including the class name as one of its properties.

Syntax

```
.mixin-name {
  property: value;
}

selector {
  .mixin-name;
}
```

Example

```
.bordered {
  border-top: dotted 1px black;
  border-bottom: solid 2px black;
}

#menu a {
  color: #111;
  .bordered;
}

.post a {
  color: red;
  .bordered;
}
```

---

4. Nesting

Nesting allows you to nest selectors inside other selectors, which helps in organizing your code.

Example

```
#header {
  color: black;
  .navigation {
    font-size: 12px;
  }
  .logo {
    width: 300px;
  }
}
```

Compiled CSS

```
#header {
  color: black;
}
#header .navigation {
  font-size: 12px;
}
#header .logo {
  width: 300px;
}
```

---

5. Operations

Operations allow you to perform arithmetic operations on colors, numbers, and variables.

Example

```
@base: 5%;
@filler: @base * 2;
@other: @base + @filler;

color: #888 / 4;
background-color: @base-color + #111;
height: 100% / 2 + @filler;
```

---

6. Functions

LESS provides a variety of functions to manipulate colors, strings, and other values.

Example

```
@base: #f04615;
@width: 0.5;

.class {
  width: percentage(@width); // returns `50%`
  color: saturate(@base, 5%);
  background-color: spin(lighten(@base, 25%), 8);
}
```

---

7. Namespaces and Accessors

Namespaces help in organizing your mixins and variables.

Example

```
#bundle {
  .button {
    display: block;
    border: 1px solid black;
    background-color: grey;
    &:hover {
      background-color: white
    }
```

```
  }
}

#header a {
  color: orange;
  #bundle > .button;
}
```

---

8. Importing

You can import other LESS files into your main LESS file.

Syntax

```
@import "file-name";
```

Example

```
@import "library"; // library.less
@import "typo.css";
```

---

9. Escaping

Escaping allows you to use any arbitrary string as a property or variable value.

Example

```
@min768: ~"(min-width: 768px)";
.element {
  @media @min768 {
    font-size: 1.2rem;
  }
}
```

---

10. Comments

LESS supports both single-line and multi-line comments.

Example

```
/* This is a multi-line comment
that will appear in the compiled CSS */
```

```
// This is a single-line comment
// that will not appear in the compiled CSS
```

---

11. Guards

Guards allow you to apply styles conditionally.

Example

```
.mixin (@a) when (lightness(@a) >= 50%) {
  background-color: black;
}
.mixin (@a) when (lightness(@a) < 50%) {
  background-color: white;
}
.mixin (@a) {
  color: @a;
}

.class1 { .mixin(#ddd) }
.class2 { .mixin(#555) }
```

---

12. Loops

LESS supports looping through mixins to create repetitive styles.

Example

```
.generate-columns(4);

.generate-columns(@n, @i: 1) when (@i =< @n) {
  .column-@{i} {
    width: (@i * 100% / @n);
  }
  .generate-columns(@n, (@i + 1));
}
```

---

13. Color Functions

LESS provides a variety of color functions to manipulate colors.

Example

```
@base: #f04615;

.class {
  color: saturate(@base, 5%);
  background-color: fadeout(@base, 25%);
}
```

Common Color Functions

- `lighten(@color, 10%)`
- `darken(@color, 10%)`
- `saturate(@color, 10%)`
- `desaturate(@color, 10%)`
- `fadein(@color, 10%)`
- `fadeout(@color, 10%)`
- `spin(@color, 10)`

---

14. Media Queries

LESS allows you to nest media queries within selectors.

Example

```
.component {
  width: 300px;
  @media (min-width: 768px) {
    width: 600px;
  }
  @media (min-width: 1280px) {
    width: 800px;
  }
}
```

---

15. Tips and Tricks

- **Use Variables for Common Values**: Store common values like colors, font sizes, and spacing in variables.

- **Organize with Namespaces**: Use namespaces to organize your mixins and variables.

- **Leverage Nesting**: Nest selectors to keep your code organized and maintainable.

- **Use Functions for Color Manipulation**: LESS provides powerful color functions to manipulate colors easily.

- **Minimize Repetition with Mixins**: Use mixins to avoid repeating the same styles.

This cheat sheet provides a comprehensive overview of LESS, covering its essential features, syntax, and best practices. Use this as a quick reference to enhance your LESS workflow.

By Ahmed Baheeg Khorshid

ver 1.0