# Cheat Sheet for comprehensive LESS

## Variables

- **Defining Variables**

```
@color: #ff0000;
@font-size: 14px;
```

- **Using Variables**

```
body {
  color: @color;
  font-size: @font-size;
}
```

## Mixins

- **Defining Mixins**

```
.border-radius(@radius: 5px) {
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
  border-radius: @radius;
}
```

- **Using Mixins**

```
.button {
  .border-radius(10px);
}
```

## Nested Rules

- **Basic Nesting**

```
.container {
  width: 100%;
  .header {
    background: #ccc;
  }
}
```

- **Parent Selector (`&`)**

```
a {
  color: blue;
  &:hover {
    color: red;
  }
}
```

## Operations

- **Arithmetic Operations**

```
@width: 100px;
.box {
  width: @width * 2; // 200px
}
```

- **Color Operations**

```
@base-color: #333;
.highlight {
  color: @base-color + #111; // #444
}
```

## Functions

- **Color Functions**

```
@base: #f04615;
.class {
  color: saturate(@base, 5%);
  background-color: fadeout(@base, 50%);
}
```

- **Math Functions**

```
.box {
  width: ceil(4.2px); // 5px
  height: floor(4.9px); // 4px
}
```

## Importing

- **Importing LESS Files**

```
@import "variables.less";
@import "mixins.less";
```

- **Importing CSS Files**

```
@import (css) "reset.css";
```

## Escaping

- **Escaping Special Characters**

```
.class {
  filter: ~"ms:alwaysHasItsOwnSyntax.For.Stuff()";
}
```

## Namespaces and Accessors

- **Defining Namespaces**

```
#bundle() {
  .button {
    display: block;
  }
}
```

- **Using Namespaces**

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

## Scope

- **Local Scope**

```
@var: red;
#page {
```

```
  @var: white;
  #header {
    color: @var; // white
  }
}
```

- **Global Scope**

```
@var: red;
#page {
  #header {
    color: @var; // red
  }
}
```

### Comments

- **Single-line Comments**

```
// This is a single-line comment
```

- **Multi-line Comments**

```
/*
This is a
multi-line comment
*/
```

### Conditionals

- **Guards (if-like conditions)**

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

### Loops

- **Recursive Mixins**

```
.generate-columns(4);
.generate-columns(@n, @i: 1) when (@i =< @n) {
  .column-@{i} {
    width: (@i * 100% / @n);
  }
  .generate-columns(@n, (@i + 1));
}
```

### Extend

- **Extending Selectors**

```
.button {
  display: inline-block;
}
.cta {
  &:extend(.button);
  background: red;
}
```

### Merge

- **Merging Properties**

```
.mixin() {
  box-shadow+: inset 0 0 10px #555;
}
.myclass {
  .mixin();
  box-shadow+: 0 0 20px black;
}
```

### JavaScript Evaluation

- **Using JavaScript in LESS**

```
@var: `"hello".toUpperCase() + '!'`; // HELLO!
```

### Source Maps

- **Generating Source Maps**

```
lessc --source-map styles.less styles.css
```

## Best Practices

- **Modularize Code**

- Use separate files for variables, mixins, and styles.
- Import them into a main file.

- **Avoid Deep Nesting**

- Limit nesting to 3-4 levels to maintain readability.

- **Use Descriptive Variable Names**

- Make variable names meaningful for easier maintenance.

- **Leverage Mixins for Vendor Prefixes**

- Define mixins for common vendor prefixes to avoid repetition.

## Tools and Integrations

- **LESS Compilers**

  - **Node.js**: `lessc` command-line tool.

  - **Gulp**: Use `gulp-less` for task automation.

  - **Grunt**: Use `grunt-contrib-less` for task automation.

- **Browser Support**

- Use `less.js` for client-side compilation.
- Ensure compatibility with modern browsers.

## Debugging

- **Inspecting Compiled CSS**

- Use browser developer tools to inspect compiled CSS.
- Compare with original LESS for debugging.

- **Error Handling**

- LESS compilers provide detailed error messages.
- Use these messages to identify and fix issues.

## Performance Tips

- **Minimize Imports**

- Only import necessary files to reduce compilation time.

- **Optimize Mixins**

- Avoid overly complex mixins to maintain performance.

- **Use Source Maps**

- Source maps help in debugging by mapping compiled CSS to original LESS.

### Advanced Techniques

- **Custom Functions**

- Extend LESS with custom functions using JavaScript.

- **Dynamic Imports**

- Use conditionals to import files dynamically based on environment.

- **Post-processing**

- Use tools like `postcss` for additional processing after LESS compilation.

### Resources

- **Official Documentation**

- [LESS Official Documentation](http://lesscss.org/)

- **Community Forums**

- [Stack Overflow](https://stackoverflow.com/questions/tagged/less)
- [LESS GitHub Repository](https://github.com/less/less.js)

- **Tutorials and Guides**

- [LESS Tutorial on Tutorialspoint](https://www.tutorialspoint.com/less/)
- [LESS CSS Preprocessor on MDN](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Preprocessors)

This cheat sheet provides a comprehensive overview of LESS, covering essential features, shortcuts, tips, and tricks. Use it as a quick reference guide for your LESS projects.

By Ahmed Baheeg Khorshid

ver 1.0