**Coding Standards (CSS)**  
**v1.0**  
**October 26, 2024**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Version Description

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 10/06/24 | RM and LA | Document Created |
| V1.0 | 10/26/24 | RM | CSS  Documentation Included |
| v2.0 |  |  |  |
| v3.0 |  |  |  |

# Contents of this Document

* General
* Naming Conventions
* Commenting Code
* Line Spacing
* Indentation
* Braces
* Error Handling
* CSS Specific Rules

# General

CSS will be used to style the front-end of web applications and should follow consistent conventions for maintainability, readability, and scalability. Use modern CSS techniques while avoiding deprecated practices.

**Goals for code guidelines:**  
1. All styles should be easy to read and understand.

2. The resulting stylesheet should be easy to maintain and extend.

3. Changes to the CSS file should be easy to manage without requiring full refactoring.

# Naming Conventions

**Class and ID Names:** Use kebab-case for class and ID names. Names should be descriptive, reflecting the purpose or content of the element.

**Example:**

.main-header {

background-color: #f4f4f4;

}

#sidebar-menu {

width: 250px;

}

**Variables (Custom Properties):** Use kebab-case for CSS variables (custom properties). Variables should be prefixed with -- and should be scoped to specific components where necessary.

**Example:**

:root {

--primary-color: #3498db;

--secondary-color: #2ecc71;

}

.button {

color: var(--primary-color);

}

# Commenting Code

**Single-line Comments:** Use comments to describe sections of the stylesheet or explain complex styles. Use /\* \*/ for single-line comments.

**Example:**

# /\* Main Layout \*/

# .container {

# max-width: 1200px;

# margin: 0 auto;

# }

**Multi-line Comments:** Use comments to describe more complex sections, such as media queries or large blocks of related styles.

/\*

Media queries for responsive layout

Adjusts the layout for screens smaller than 768px

\*/

@media (max-width: 768px) {

.container {

max-width: 100%;

}

}

**Line Spacing**

**Separate Rule Sets:** Add a blank line between different CSS rule sets for better readability.

**Example:**

**.header {**

**background-color: #fff;**

**padding: 10px;**

**}**

**.footer {**

**background-color: #f8f8f8;**

**padding: 20px;**

**}**

**Multiple Selectors:** When listing multiple selectors on a single line, ensure each selector is on a new line if they share the same style rule.

**Example:**

.header,

.footer,

.sidebar {

background-color: #333;

}

# Indentation

# Use 2 spaces for indentation: Indent nested rules, media queries, or child selectors by 2 spaces to ensure the CSS is easy to follow.

# Example:

# .nav {

# display: flex;

# justify-content: space-between;

# 

# .nav-item {

# margin-right: 10px;

# }

# }

**Braces**

Always use braces for grouping styles, even for single-line declarations, for consistency and readability.

**Example:**

.button {

background-color: #3498db;

color: #fff;

}

**Error Handling**

* Avoid browser-specific hacks and ensure that your CSS works across all major browsers by using standardized, well-tested properties.
* Use CSS validation tools to catch errors before deployment.

**CSS Specific Rules**

**Avoid Inline Styles:** All styles should be applied via external CSS files or <style> blocks in the head of the document. Inline styles reduce reusability and readability.

Example of **bad practice:**

<div style="color: red;">Hello</div>

**Use Shorthand Properties where applicable:** Use shorthand syntax for properties like margin, padding, font, background, etc., to reduce repetition and improve readability.

**Example:**

.box {

margin: 10px 15px 5px 0;

background: #fff url('image.jpg') no-repeat center;

}

**Media Queries:** Group media queries at the bottom of your CSS file or within specific sections for responsive design.

**Example:**

@media (max-width: 768px) {

.container {

max-width: 100%;

}

}

Use Flexbox or Grid for Layout: Whenever possible, use modern layout techniques like Flexbox or Grid instead of floats or positioning to create more responsive and maintainable layouts.

**Example:**

.container {

display: flex;

justify-content: space-between;

}