**Coding Standards (JavaScript)**  
**v1.0**  
**October 14, 2024**

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

# Version Description

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Who | Comment |
| v0.0 | 10/06/24 | RM and LA | Document Created |
| V1.0 | 10/14/24 | RM and LA | JavaScript Documentation Included |
| v2.0 |  |  |  |
| v3.0 |  |  |  |

# Contents of this Document

* General
* Naming Conventions
* Commenting Code
* Parenthesis
* Constants
* Line Spacing
* Braces
* Declarations

# General

JavaScript will be used as part of the project for front-end development and back-end scripting. JavaScript code should use the most updated and modern syntax to ensure maintainability and readability.   
  
Goals for code guidelines:  
1. All source code should be easily readable and understandable.

2. The resulting documentation should be easy to maintain.

3. Code changes should be relatively easy with no need for complete code refactoring.

# Naming Conventions

**Variables**: Use camelCase for variables names.

Example:

let userName = “John”;

**Function Names:** Functions names should also be written in camelCase.

Example:

function calculateTotal(price, tax){

return price + tax;

}  
  
**Class Names:** Concrete classes should use natural descriptive names, begin with a capital, and follow PascalCase.

Example:

class AnimalWithHair{}  
  
**Member Function Names:** Member function names should also follow camelCase.

# Commenting Code

Classes: Class attributes and member methods should include comments specifing types of variables and the purpose of functions.

Main: Same rules apply to main functions not in a class and variables outside of main. Use single-line comments with // for short explanations. Use block comments /\* \*/ for multi-line comments. For parameters use @param {type} name and for return statements include @return {type} followed by a short description.

NOTE: For methods that don’t return any value, don’t include a “Return” in the method comment. Single-line comments should use double forward slash (//). Multi-line should use

/\* \*/

**Example:**

/\*\*

\* Adds two numbers together.

\* @param {number} a - The first number.

\* @param (integer b) - The second number.

\* @return {number} - The sum of both numbers.

/\*

function add(a, b){

return a + b;

}

# Parenthesis

Parenthesis should always be used to ensure accurate conditionals and avoid syntax errors, especially in complex statements.

**Example:**  
if (x + y) {}

NOTE: This example is really easy to interpret, but parenthesis is still needed.

if (x\*\*2 \* (variable // value) % (17 - variableTwo)){}

NOTE: This example is more complex, which is why we need the use of parenthesis to make the order of statements more obvious.

# Constants

Constants should be defined using the “const” keyword and written in all caps to differentiate them from variables.

**Example:**

const DAYS = 7;

# Line Spacing

Proper indentation and line spacing are required for better readability. Use tabs with a default size of 4 spaces. Blank lines should be used between logical blocks, such as functions, loops, and conditionals, to ensure good readability.

**Example:**

function calculateArea(radius) {

const PI = Math.pi;

return (PI \* radius) \*\* 2;

}

let result = calculateArea(5);

# Braces

Always use braces {} for code blocks, even for single line if statements, for loops, or while loops to prevent future errors.

**Example:**

if (isValid) {

console.log(“Valid”);

}

# Declarations

Variables should always be declared either using the “let” or the “const” keywords instead of using “var.”

**Example:**

let itemCount = 0;

const MAX\_ITEMS = 50;

Variables should always be initialized as well to avoid errors with undefined values.

let userOption = null;

# Error Handling

Try-catch blocks should be used for handling errors. This should always be used for asynchronous functions if they are used.

Always use “console.error()” instead of “console.log()” to handle errors.

**Example:**

asnyc function tryExample(){

try{

//try code here

}

catch(error){

console.error(‘Error encountered!’, error);

}

}