DUY TÂN UNIVERSITY
International School

Capstone Project 1

CMU-SE 450

Code Standard

Version 1.0
Date: 10/11/2021

Green Big5 Information System

Submitted by

Chinh, Thai Huu
Chung, Hoang Bao
Hau, Bui Phuc
Loc, Nguyen Tien

Approved by Nguyen Thanh Binh

Proposal Review Panel Representative:

Name	Signature	Date
Binh, Thanh Nguyen		31 - Nov- 2021

PROJECT INFORMATION

Project acronym	GB5		
Project Title	GreenBig5		
Start Date	19 Aug 2021	**Start Date**	19 Aug 2021
Lead Institution	International School, Duy Tan University		
Project Mentor	Doctor. Habil. Binh, Thanh Nguyen		
Scrum master / Project Leader & contact details	Chinh, Huu Thai Email: huuchinhdev@gmail.com Tel: 0962545506 Student ID: 24211207534		
Team members	Student ID	**Team members**	Student ID
1	24211207051	1	24211207051
2	24211206857	2	24211206857
3	24211202217	3	24211202217

REVISION HISTORY

Version	Date	Comments	Author	Approval
1.0	15/11/2021	Initial Release	All members	

1. Introduction

1.1 Purpose

- This Coding Standard requires certain practices for developing programs in the JavaScript language. The objective of this coding standard is to have a positive effect on
    - Avoidance of errors/bugs, especially the hard-to-find ones.
    - Maintainability, by promoting some proven design principles

1.2 Scope

- This standard pertains to the use of the JavaScript language.

2. Code Standands

2.1 Variables

- Using **camelCase** for identifier names (variables and functions).
- All names start with a **letter**.
- Constants (like PI) written in **UPPERCASE**
- No unused variables.
- For var declarations, write each declaration in its own statement.
- Avoid modifying variables of class declarations.
- Avoid modifying variables declared using const.
- No re-declaring variables.
- Avoid assigning a variable to itself.
- Avoid comparing a variable to itself.
- Restricted names should not be shadowed.

2.2 Spaces Around Operators

- Always put spaces around operators (= + - * /), and after commas.

2.3 Statement Rules

- Put the opening bracket at the end of the first line.
- Use one space before the opening bracket
- Put the closing bracket on a new line, without leading spaces.
- Keep else statements on the same line as their curly braces.

2.4 Object Rules

- Place the opening bracket on the same line as the object name.
- Use colon plus one space between each property and its value
- Do not add a comma after the last property-value pair.
- Place the closing bracket on a new line, without leading spaces.

- Maintain consistency of newlines between object properties.
- Always end an object definition with a semicolon.

2.5 Line Length

- For readability, avoid lines longer than 80 characters

2.6 Spaces

- Use 2 spaces for indentation.
- Add a space after keywords.
- Add a space before a function declaration's parentheses
- Commas should have a space after them.
- Add spaces inside single line blocks.
- No space between function identifiers and their invocations.
- Add space between colon and value in key value pairs.

2.6 Quotes

- Use single quotes for strings except to avoid escaping.

2.7 Comparative math

- Always use === instead of ==.

  Exception: obj == null is allowed to check for null || undefined.

2.8 Dot location

- Dot should be on the same line as property.

2.9 Array

- Use array literals instead of array constructors

2.10 Modules

- Use a single import statement per module.
- Renaming import, export, and destructuring assignments to the same name are not allowed.

2.11 Functions

- Avoid unnecessary function binding.
- No unnecessary parentheses around function expressions.

- No function declarations in nested blocks.

2.12 String

- Regular strings must not contain template literal placeholders.
- No octal escape sequences in string literals.
- No multiline strings.
- No spacing in template strings.

2.13 Error catching

- Only throw an Error object.

2.14 Files

- Files must end with a newline.

2.15 Others

- Semicolons must have a space after and no space before.
- Must have a space before blocks.
- Use isNaN() when checking for NaN
- Function typeof must be compared to a valid string.
- Never start a line with (, [, `, or a handful of other unlikely possibilities.