

International School

**Capstone Project 2**

CMU-SE 451

**Code Standard**

**Version 2.0**

**Date: 21/11/2020**

**ViVu - The smart travel system supports with chatbot and short video social network**

**Submitted by**

**Sang, Nguyen Thanh Thuyen, Pham Van Tuan, Doan Kim**

**Loc, Phung Hoang Phu**

**Approved by Nguyen Duc Man**

**Proposal Review Panel Representative:**

Name Signature Date

**Capstone Project 2- Mentor:**

Name Signature Date

**PROJECT INFORMATION**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project acronym** | viBOTour | | |
| **Project Title** | Smart Chatbot for Tourist 4.0 | | |
| **Start Date** | 24 Aug 2020 | **End Date** | 30 May 2021 |
| **Lead Institution** | International School, Duy Tan University | | |
| **Project Mentor** | Man, Nguyen Duc, Ph.D | | |
| **Scrum master / Project Leader & contact details** | Sang, Nguyen Thanh  Email: thanhsang3111999.ntb@gmail.com  Tel: 0344684069  ID: 23211212363 | | |
| **Partner Organization** |  | | |
| **Project Web URL** |  | | |
| **Team members** | Name | Email | Tel |
| 2321121321 | Tuan, Doan Kim | doantuan21101999@gmail.com | 0913622928 |
| 2321124963 | Loc, Phung Hoang Phu | phunghoangphuloc@gmail.com | 0912698173 |
| 2321125061 | Thuyen, Pham Van | phamvanthuyen2802@gmail.com | 0795697332 |

REVISION HISTORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Comments** | **Author** | **Approval** |
| 1.0 | 21/11/2020 | 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.