An Informal Coding Convention

Table of Contents

To ensure uniformity in the project, all group members must enforce the prescribed coding standards stipulated in this document. All pull requests will be immediately rejected or responded with “Request changes” before approval for merging is granted.

If you feel strongly for the rules created, or if you wish to modify it to better suit our purposes, you are welcome to discuss it with the team and if your concern is reasonable enough, a revision to this document shall be made.

Variable Scope

A strict adherence to recommended use of keywords **var** and **let** is to be expected. The former shall only be used when creating global variables outside of the scope of functions, while the latter shall exist inside functions and blocks of code, and shall remain accessible only in the scope which they are declared.

Example:

```
var GLOBAL_FOO_FLAG = false; (global variable)

function foo() {

  let name = 'John'; (local variable)

  if(name.length > 0) {
    let age = 16; (local variable, block scope)

    alert(`Hi! My name is ${name} and I'm currently ${age} years old.`); (string literal)
  }

  return name;
}
```

Single and Double Quotes

Single quotes shall be used extensively in cases wherein the enclosed string or text does not contain single quotes as well – if such case happens, double quotes shall be used. Aside from that scenario, only single quotes will be used all throughout the source code.

Example:

```
function foo() {

  let name = 'John'; (single quotes)
  let message = "Don't worry, "  + name + ". You'll reach your dreams!"; (double quotes)

  alert(message);
}
```

Bracket Styling

For all intents and purposes, the opening bracket to any code statement shall exist on the same line as the statement itself, separated by a single space. For control statements such as if-else statements, the next control statement **must** be placed on a new line, not immediately after a closing bracket. Single line code blocks must also be enclosed in brackets, regardless of the brevity of code present inside.

Example:

```
/* Meets standards */
function foo() { (a space immediately after the statement, then the opening bracket)

  if(false === false) {
    alert(‘Congrats, false is false.”);
  } (the next control statement is on a new line)
  else {
    alert(‘Wowie!’); (one liners are still enclosed in brackets)
  }
}

/* Violates standards */
function foo(){

  if(false === false){ (no spacing)
    alert(‘Congrats, false is false’);
  }else{ (this is just horrible)
    alert(‘Wowie!’);
  }
}
```

Spacing

Allow your code to breathe. A generous application of spacing between code helps improve readability tremendously. The following scenarios are where spaces (a new line) between block/s of code are recommended:

- Immediately after the function header
- Immediately after variable/s declaration
- Immediately after a new code block irrelevant to current code block
- Code statements that modify similarly named objects
- Argument list exceeds maximum character length limit of a line
- Multiple arguments (each argument is on new line)

Note: It is expected that variables are to be declared and/or initialized immediately preceding the new line after the function header.

Example:

```
function foo() {

  let foovar = false; (variable declaration and/or initialization right after new line)

  if(foovar === false) {
    alert('Yey!');
  }

  return foovar; (new line separates code of block above)
}

/* Maximum character length, bad example */

let container = document.querySelector('#myContainer');

container.classList.add('border', 'border-1', 'border-dark', 'rounded', 'rounded-pill', 'fw-bold', 'fs-3',
'text-center');

/* Maximum character length, multiple arguments good example */
container.classList.add(
  'border',
  'border-1',
  'border-dark',
  'rounded',
  'rounded-pill',
  'fw-bold',
  'fs-3',
  'text-center'
);
```

Operators

Firstly, when checking the value of a variable, always use the **strict equality** (===) operator if possible. This will ensure that our code has fewer bugs in the long run.

Second, see to it that there are spaces between operators. Again, let our code have breathing space.

Example:

```
/* Good example */
function foo() {

  let fname = 'John';
  let lname = 'Doe';
```

```
  alert(“Hi “ + fname + lname + “! Welcome to the site.”);   (spaces before and after operators)
}

/* Bad example */
function foo() {

  let fname = ‘John’;
  let lname = ‘Doe’;

  alert(“Hi “+fname+lname+ “! Welcome to the site.”);   (no spaces, horrible)
```