Coding Standards - Team 8

JavaScript Coding Conventions

- Naming and declaration rules for variables and functions.
- Rules for the use of white space, indentation, and comments.
- Programming practices and principles

Variable Names

All names start with a **letter**.

```
firstName = "John";
lastName = "Doe";

price = 19.90;
tax = 0.20;

fullPrice = price + (price * tax);
```

Spaces Around Operators

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

Examples:

```
var x = y + z;
var values = ["Volvo", "Saab", "Fiat"];
```

Code Indentation

Always use 2 spaces for indentation of code blocks:

Functions:

```
function toCelsius(fahrenheit) {
  return (5 / 9) * (fahrenheit - 32);
}
```

Do not use tabs (tabulators) for indentation. Different editors interpret tabs differently.

Statement Rules

General rules for simple statements:

- Always end a simple statement with a semicolon.

Examples:

```
var values = ["Volvo", "Saab", "Fiat"];

var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

General rules for complex (compound) statements:

- 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.
- Do not end a complex statement with a semicolon.

Functions:

```
function toCelsius(fahrenheit) {
  return (5 / 9) * (fahrenheit - 32);
}
```

Loops:

```
for (i = 0; i < 5; i++) {
  x += i;
}
```

Conditionals:

```
if (time < 20) {
  greeting = "Good day";
} else {
  greeting = "Good evening";
}
```

Object Rules

General rules for object definitions:

- Place the opening bracket on the same line as the object name.
- Use colon plus one space between each property and its value.
- Use quotes around string values, not around numeric values.
- Do not add a comma after the last property-value pair.
- Place the closing bracket on a new line, without leading spaces.
- Always end an object definition with a semicolon.

Example

```
var person = {
  firstName: "John",
  lastName: "Doe",
  age: 50,
  eyeColor: "blue"
};
```

Short objects can be written compressed, on one line, using spaces only between properties, like this:

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
```

Line Length < 80

For readability, avoid lines longer than 80 characters.

If a JavaScript statement does not fit on one line, the best place to break it, is after an operator or a comma.

Example

```
document.getElementById("demo").innerHTML =
"Hello World.";
```

Naming Conventions

Always use the same naming convention for all your code. For example:

- Variable and function names written as **camelCase**
- Global variables written in **UPPERCASE** (We don't, but it's quite common)
- Constants (like PI) written in **UPPERCASE**

camelCase:

camelCase is used by JavaScript itself, by jQuery, and other JavaScript libraries.

Do not start names with a $ sign. It will put you in conflict with many JavaScript library names.

Loading JavaScript in HTML

Use simple syntax for loading external scripts (the type attribute is not necessary):

```
<script src="myscript.js"></script>
```

Accessing HTML Elements

A consequence of using "untidy" HTML styles, might result in JavaScript errors.

These two JavaScript statements will produce different results:

```
var obj = getElementById("Demo")
```

```
var obj = getElementById("demo")
```

If possible, use the same naming convention (as JavaScript) in HTML.

File Extensions

HTML files should have a **.html** extension (**.htm** is allowed).

CSS files should have a **.css** extension.

JavaScript files should have a **.js** extension.

Use Lower Case File Names

If you move from a case insensitive, to a case sensitive server, even small errors can break your web site. To avoid these problems, we always try to use lower case file names (if possible).