Javascript Coding Conventions

Purpose and Scope

This document covers the coding conventions for Persona Switchers Javascript files.

References

These conventions are based off of w3schools coding conventions for Javascript that can be found here: https://www.w3schools.com/js/js_conventions.asp

Variable Names

- Identifier Names use **camelCase** (variables and functions)
- All names start with a **letter**
- More naming conventions can be found in the naming conventions section of this document.

Examples:

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

price = 19.90;
tax = 0.20;

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

More on Naming Conventions

- Global variables written in **UPPERCASE**
- Constants (like PI) written in **UPPERCASE**

Hyphens can be mistaken as subtraction attempts and are not allowed in JavaScript names except in the following cases involving HTML and CSS:

- HTML5 attributes can start with data- (data-quantity, data-price).
- CSS uses hyphens in property-names (font-size).

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 4 spaces for indentation of code blocks:

Functions:

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

Statement 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"
};
```

For complex (compound) statements:

- Put the opening bracket on a new line below the first line.
- 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

Rules for object definitions

- Place the opening bracket on a new line below 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:

Example

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

Line length

- 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 Dolly.";
```

Loading JavaScript in HTML

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

Example

```
<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 (not **.htm**).
CSS files should have a **.css** extension.
JavaScript files should have a **.js** extension.

Use Lowercase File Names

Javascript files should be lowercase. For example ‘*Bootstrap.js’* is not acceptable and instead should be ‘*bootstrap.js*’.