Guía de Estilo para JavaScript

Primero que nada

Usa **SIEMPRE** JSHint en tu código.

> **Nota**
>
> Hay algunas excepciones para las cuales JSHint se queja de cosas en node que puedes ignorar, como por ejemplo no saber qué es 'const' y 'require'. Para estos casos puedes añadir palabras claves a ignorar en un archivo *.jshintrc*.

Formato de Nombres de Variables

```
// Clases: PalabrasCapitalizadas
var MyClass = ...

// Variables y funciones: camelCase
var myVariable = ...

// Constantes: MAYUSCULAS_CON_GUIONES_BAJOS
// Backend
const MY_CONST = ...

// Del lado del cliente
var MY_CONST = ...
```

Indentación

Se debe indentar con 4 espacios (no tabulaciones).

Para nuestros proyectos, siempre debes asignar variables en una nueva línea, no separado por comas:

```
// Malo
var a = 1,
    b = 2,
    c = 3;

// Bueno
var a = 1;
var b = 2;
var c = 3;
```

Usa `[]` para asignar un nuevo arreglo, no `new Array()`.

Igualmente, usa `{}` para crear nuevos objetos.

A continuación se presentan dos escenarios para el uso de `[]`, uno puede estar en una sola línea, el otro no tanto:

```
// Bien en una sola línea
var stuff = [1, 2, 3];

// Nunca en una sola línea
var longerStuff = [
    'some longer stuff',
    'other longer stuff'
];
```

Nunca asignes multiples variables en la misma línea

Malo, malo:

```
var a = 1, b = 'foo', c = 'wtf';
```

NO alinees los nombres de variable

Malo:

```
var wut    = true;
var boohoo = false;
```

Puntos y comas

Úsalos.

No porque la inserción automática de puntos y comas (ASI) sea magia negra, hazlo por consistencia.

Condicionales y bucles

```
// Malo
if (something) doStuff()

// Bueno
if (something) {
    doStuff();
}
```

Espacios después de una palabra clave y antes del corchete

```
// Malo
if(bad){

}

// Bueno
if (something) {

}
```

Funciones

Funciones con nombre

Siempre usa funciones con nombre, inclusive si la estás asignando a otra variable o propiedad. Esto mejora las pilas de error cuando se hace depuración.

No coloques espacios entre el nombre de la función y el paréntesis inicial, pero si entre el paréntesis de cierre y el corchete inicial:

```
var method = function doSomething(argOne, argTwo) {

}
```

Funciones anónimas

Lo estás haciendo mal, lee sobre las funciones con nombre más arriba.

Operadores

Siempre debes usar `===` con la única excepción de comparaciones con `null` y `undefined`.

Ejemplo:

```
if (value != null) {

}
```

Comillas

Siempre usa comillas simples: `'no dobles'`

La única excepción: `"no escapes comillas simples en las cadenas así: \'. Usa comillas dobles."`

Comentarios

Para funciones en `node`, provee siempre un comentario claro con el siguiente formato:

```
/* Explicación breve de lo que se hace
 * Expects: cualquier parámetro aceptado
 * Returns: cualquier cosa retornada
 */
```

Si los comentarios son realmente largos, utiliza el formato `/* ... */`. De lo contrario, usa comentarios cortos como:

```
// Este es un comentario corto y termina con un punto.
```

Ternarios

Trata de no utilizarlos.

Si un ternario usa varias líneas, no lo uses:

```
// Malo
var foo = (user.lastLogin > new Date().getTime() - 16000) ? user.lastLogin - 24000 : 'wut';

// Bueno
return user.isLoggedIn ? 'yay' : 'boo';
```

Buenas prácticas generales

Si te das cuenta que estás repitiendo algo que puede ser una constante, usa una sola definición de constante al comienzo del archivo.

Define las expresiones regulares como constantes siempre.

Siempre debes probar la certidumbre:

```
// malo
if (blah !== false) { ...

// Bueno
if (blah) { ...
```

Si el código es demasiado largo, trata de romperlo en varias líneas o refactoriza. Trata de mantenerte dentro del límite de 80 columnas por línea, pero si te pasas un poco no es un gran problema. Cuando rompas una línea, indenta las subsiguientes un nivel (2 espacios)

Si el código luce demasiado inteligente, probablemente lo sea, así que solo manténlo sencillo.