APRENDERAPROGRAMAR.COM

GUÍA DE ESTILO JAVASCRIPT PARA COMENTARIOS DE PROYECTOS. JSDOC. @CONSTRUCTOR, @DEPRECATED, ETC. EJEMPLOS (CU01192E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

Resumen: Entrega nº92 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

GUÍA DE ESTILO PARA COMENTARIOS

Los programadores JavaScript suelen atenerse a unas reglas de estilo a la hora de escribir código. Esto no sólo es válido para el código en sí, sino también para los comentarios. En proyectos de gran extensión pueden existir miles de líneas de código JavaScript. Este código debe estar bien comentado para facilitar su comprensión y mantenimiento.

MANUALES DE ESTILO

En una empresa en la que trabajan 100 ó 1000 programadores cada programador no puede "hacer las cosas a su manera". En estas empresas es frecuente que cada programador desarrolle un fragmento de una aplicación y luego esos fragmentos hay que unirlos. Para que esa unión sea sólida y no haya problemas de integración y mantenimiento posterior, todos los programadores deben atenerse a unas mismas normas.

Lo ideal sería que en una aplicación donde han trabajado 20 programadores, un supervisor no fuera capaz de detectar qué parte del código ha hecho cada programador. Esto es difícil, porque cada programador deja su impronta, pero siguiendo unas reglas predefinidas se puede conseguir como resultado un código bien estructurado y bastante homogéneo.

Ya hemos hablado de reglas de estilo a la hora de crear código en relación formas de uso preferentes del lenguaje y en relación a cuestiones organizativas relacionadas con el código.

Vamos a centarnos ahora en los comentarios dentro del código JavaScript. No hay unas reglas obligatorias que cumplir en relación a los comentarios. Cada empresa puede usar diferentes estilos a la hora de comentar el código, si bien es cierto que suele hacerse de forma similar cuando hablamos de programación profesional.

Nosotros vamos a presentar aquí unas reglas de estilo para comentarios que se corresponden con las que utilizan grandes empresas, pero ten en cuenta que ni son obligatorias ni todas las empresas siguen los mismos criterios.

REGLAS DE ESTILO PARA COMENTARIOS

A continuación presentamos una lista de reglas de estilo que suelen recomendarse para la programación JavaScript. Como hemos indicado, son opcionales, pero un programador que no sigue unas reglas de estilo no suele considerarse un buen programador. Más que seguir estrictamente estas reglas que presentamos a continuación, lo importante es que cuando crees código sigas unas reglas de estilo (que pueden ser estas o similares a estas) y que no las modifiques. De esa manera crearás código homogéneo.

También ten en cuenta que las reglas aquí presentadas son una forma resumida de reglas respecto a lo que sería un manual de estilo propio de una empresa, que sería mucho más extenso. El objetivo de presentar aquí estas reglas es didáctico y para familiarizarnos con ellas, no consideres que son obligatorias ni que aquí tienes todas las reglas necesarias.

Las reglas aquí presentadas siguen el estándar JSDoc. Seguir un estándar tiene ventajas como poder generar de forma automática documentación de proyectos usando el software adecuado.

Regla 1. El mejor comentario es el implícito o auto-documentador

Considera que estás revisando un código y que en una función encuentras esto:

```
function f33 (a, b) { ... }
```

¿Te haces una idea de lo que hace esta función? ¿Te haces una idea de qué tipo de parámetros recibe la función? Difícilmente nos podremos hacer una idea si los nombres de variables, funciones, parámetros, etc. son cortos y no descriptivos.

Sería mucho mejor esto:

```
function getFullName (name, surname) { ... }
```

Fíjate que estamos usando nombres descriptivos. En este caso hemos usado el inglés para que tengas en cuenta que el inglés es el lenguaje más universalmente utilizado en programación. No solo lo utilizan quienes hablan inglés, sino que lo utilizan muchas empresas aunque sean de habla hispana. Si prefieres utilizar el español puedes escribirlo así:

```
function obtenerNombreCompleto (nombre, apellidos) { ... }
```

Este código, aunque no hemos comentado nada, podemos decir que está autodocumentado, porque se entiende (aproximadamente) lo que hace simplemente leyéndolo. Esto es un buen estilo de programación.

Regla 2. Usa un estilo de comentarios y no lo cambies, pero respeta el existente

Si estamos trabajando en un nuevo proyecto usaremos un estilo para comentarios que debemos mantener en todo el proyecto. Sin embargo, si estamos revisando el código desarrollado por otras personas y observamos que siguen un estilo de comentarios diferente al que nosotros estemos habituados a usar, deberemos respetarlo (para no afectar a la homogeneidad del código).

El estilo de comentarios que vamos a proponer aquí es un estilo similar al que se usa en otros lenguajes de programación como C++ y Java.

Regla 3. Usa comentarios multilínea en cabeceras de funciones y en línea en otros casos

Para comentar lo que hace una función usaremos comentarios de este tipo:

```
/**
 * El comentario comienza con una barra y dos asteriscos.
* Cada nueva línea lleva un asterisco al comienzo.
 * @param {string} nombre indica que una función recibe un parámetro de tipo string y que
      * el nombre del parámetro es nombre.
 * @descriptor Cada descriptor que añadamos irá en una línea independiente.
 */
```

Si una línea es demasiado larga la dividimos en varias líneas. La primera estará sin indentar y el resto indentada (desplazada hacia dentro) respecto de la primera.

Los comentarios puntuales en línea se harán con //. Por ejemplo:

var tmpPers; //Variable temporal que almacenará un objeto Persona

Regla 4. Usa un comentario al principio de un archivo js para describir su contenido

En un proyecto podemos tener decenas de archivos js correspondientes a código JavaScript. Al principio de cada archivo deberíamos incluir un comentario descriptivo de qué contiene dicho archivo que comenzará con la etiqueta @fileoverview. Por ejemplo:

```
/**
 * @fileoverview Menú aprMenu, desplegable con efecto expansión suavizado
 *
 * @version          2.2
 *
 * @author           César Krall <cesarkrall@aprenderaprogramar.com>
 * @copyright        aprenderaprogramar.com
 *
 * History
 * v2.2 – Se mejoró el efecto de expansión de los submenús dándole efecto aceleración
 * v2.0 – Se evitó que quedaran supersupuestos textos de submenús
 * v1.1 – Se mejoró la compatibilidad con navegadores Opera
 * ----
 * La primera versión de aprMenu fue escrita por Karl Monitrix
 */
```

Regla 5. Documenta las funciones incluyendo los parámetros que usa y lo que devuelve

Un comentario típico para describir una función incluirá una descripción de lo que hace la función, la descripción de sus parámetros, y la descripción de lo que devuelve la función.

La descripción de un parámetro se hace comenzando con @param

La descripción de lo que devuelve la función se hace comenzando con @return. Si la función no devuelve nada se omitirá esta etiqueta.

Ejemplo:

```
/**
 * Verifies if the string is in a valid email format
 * @param  {string}
 * @return  {boolean}
 */
```

Regla 6. Usa comentarios para constructores y para documentar las propiedades

Si estamos definiendo un tipo de objeto usaremos una declaración @constructor para indicar que estamos definiendo el constructor para un tipo de objeto. Además, las propiedades serán comentadas para reflejar su significado. La etiqueta @type se usará para describir qué tipo de dato corresponde a la propiedad.

Ejemplo:

```
/** @constructor */
project.MiDefinicionDeTipoDeObjeto = function() {
  /**
   * Propiedad que indica el número máximo de colores permitidos.
   * @type {number}
   */
  this.maxNumeroColores = 4;
}
```

Regla 7. Usa etiquetas normalizadas

Cuando queramos incluir una descripción de un elemento usaremos etiquetas normalizadas. Por ejemplo una etiqueta normalizada para describir al autor de un código es @author. En cambio no es normalizada <<@autor>> ni <<@Author>> ni <<@authors>>.

A continuación indicamos la lista de etiquetas normalizadas recomendadas:

Etiqueta	Descripción	Ejemplo
@author	Indica el nombre del autor del código	@author César Krall
@const {type}	Indica que una propiedad o variable serán constantes y su tipo	/** * El nombre de la empresa. * @const {string} */
@constructor	Indica que se define un tipo de objeto	/** * Representa un círculo. * @constructor */ function Circulo() { ... }
@deprecated	Indica que una función, método o propiedad no deberían usarse por ser obsoletas. Debe indicarse cuál es la función, método o propiedad que debe ser usada en su lugar.	@deprecated Usar getNombre().

@enum	Indica que se trata de una enumeración	/**
		* Enum para tres estados posibles.
		* @enum {number}
		*/
		project.TresEstado = {
		SI: 1,
		NO: -1,
		DESCONOCIDO: 0
		};

@extends	Usado con @constructor para indicar que un tipo de objeto hereda de otro tipo de objeto	/** * Una lista de objetos persona. * @constructor * @extends apr.bc.BasicList */
@fileoverview	Comentario para describir los contenidos de un fichero	Ver el ejemplo de la regla 4 expuesto anteriormente.
@interface	Usado para indicar que una función define una interface	/** * Una forma. * @interface */ function Forma() {}; Forma.prototype.dibujar = function() {};

@implements	Usado con @constructor para indicar que un tipo de objeto implementa una interface	/**
		* @constructor
		* @implements {Forma}
		*/
		function Square() {};
		Square.prototype.dibujar = function() {
		...
		};

Etiqueta	Descripción	Ejemplo
@license	Indica los términos de licencia	/* Copyright aprenderaprogramar.com, 2002-2005 * This script is developed by aprenderaprogramar. Visit us at www.aprenderaprogramar.com. * This script is distributed under the GNU Lesser General Public License. * Read the entire license text here: http://www.gnu.org/licenses/lgpl.html */
@override	Indica que un método o propiedad de una subclase sobreescribe el método de la superclase. Si no se indica otra cosa, los comentarios y documentación serán los mismos que los de la superclase.	/** * @return {number} Sueldo. * @override */
@param {type}	Indica un parámetro para un método, función o constructor y el tipo de dato que es.	* @param {string}
@return {type}	Indica qué devuelve un método o función y su tipo.	* @return {boolean}
@see	Indica una referencia a documentación más amplia	@see http://aprenderaprogramar.com
@supported	Indica navegadores para los que se conoce que aceptan el código	@supported Testado en IE10+ y en FF26+
@type {Type}	Identifica el tipo de una variable, propiedad o expresión.	@type {number}
Otros	Existen más etiquetas que no vamos a estudiar ahora	Por ejemplo @code, @define, @dict, @export, @expose, @externs, @inheritDoc, @lends, @preserve, @noalias, @nocompile, @nosideeffects, @private, @protected, @public, @struct, @suppress, @template, @this, @typedef y otros)

GENERADORES DE DOCUMENTACIÓN AUTOMÁTICOS

Existen generadores automáticos de documentación de proyectos JavaScript. Estos generadores lo que hacen básicamente es extraer los comentarios en el código y transformarlos en un documento (por

ejemplo un documento HTML) organizado donde se pueden leer todos los detalles del código con una presentación adecuada.

En este curso no vamos a entrar a estudiar esta posibilidad (que sí es habitual usar en empresas), aunque si lo deseas puedes hacer pruebas por tu cuenta.

Cada empresa o equipo de desarrolladores puede seguir distintos estándares o usar distintos generadores de documentación, por ello deberás atenerte en cada empresa a la herramienta que se esté usando para la documentación.

Las reglas de documentación que hemos expuesto están adaptadas al estándar JSDoc (http://usejsdoc.org/). Puedes hacer pruebas usando este estándar si lo deseas.

EJERCICIO

Un programador ha creado este código y nos han pedido que a) Lo comentemos de forma adecuada. b) Lo indentemos y organicemos de forma adecuada. Revísalo, coméntalo y organízalo conforme a los estándares que hemos explicado, incluyendo comentarios de cabecera de función y comentarios en línea para describir los aspectos más relevantes:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Cursos aprende a programar</title><meta charset="utf-8">
<meta name="description" content="Aprende a programar con cursos reconocidos por su calidad didáctica: HTML, CSS,
JavaScript, PHP, Java, Visual Basic, Joomla, pseudocodigo, diagramas de flujo, algoritmia y más.">
<meta name="keywords" content="HTML, CSS, JavaScript, Java, Visual Basic, Joomla, PHP, pseudocodigo, diagramas de
flujo, cursos, tutoriales">
<style type="text/css"> *{font-family: verdana, sans-serif;} #principal{text-align:center;width:95%; margin: 0 auto;}
.tituloCurso {color: white; float:left; padding: 36px 44px; font-size: 2.65em; font-style:bold; text-decoration:none;}
a:hover{color:orange !important;}</style>
<script type="text/javascript">
function ejemplo(){
var nodosTituloCurso = document.querySelectorAll(".tituloCurso");
var contador = 0;
var nodosCambiados = new Array();
var tocaCambiar = true;
setTimeout(function() {ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar)}, 1500);
}
function ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar) {
        if (contador<nodosTituloCurso.length){
        var indice = Math.floor((Math.random() * (nodosTituloCurso.length)));
        if (nodosCambiados.length!=0) {
                for (var i=0; i<nodosCambiados.length; i++) {
                        if(nodosCambiados[i]==indice) {tocaCambiar = false;}
                }
        }
        if (tocaCambiar==true) {
        cambiarNodo(nodosTituloCurso[indice]);
        nodosCambiados.push(indice);
        contador = contador+1;
        setTimeout(function() {ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar)}, 1500);
        } else {tocaCambiar=true; ejemploAccion(nodosTituloCurso, nodosCambiados, contador, tocaCambiar);}
        }
```

```
		else { //Caso base fin de la recursión
			document.body.style.backgroundColor='black';
			document.getElementById('principal').style.color='white';
			for (var i=0; i<nodosTituloCurso.length; i++) {
				nodosTituloCurso[i].style.color='yellow';
			}
		}
}

function cambiarNodo(elNodo){elNodo.style.backgroundColor = 'black';}
</script>
</head>
<body onload="ejemplo()">
<div id="principal"><h1>Cursos de programación</h1>
<h3>Reconocidos por su calidad didáctica</h3>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=28&Itemid=59"> Fundamentos</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=68&Itemid=188"> Java</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=69&Itemid=192"> HTML</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=75&Itemid=203"> CSS</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206"> JavaScript</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=70&Itemid=193"> PHP</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=38&Itemid=152"> Joomla</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=37&Itemid=61"> Visual Basic</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=28&Itemid=59"> Pseudocódigo</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=section&layout=blog&id=7&Itemid=26">
Libros/ebooks</a></div>
<div ><a class="tituloCurso"
href="http://aprenderaprogramar.com/index.php?option=com_content&view=article&id=64&Itemid=87">Cursos tutorizados</a></div>
</div>
</body>
</html>
```

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU01193E

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206