Introducción JavaScript

Introduction JavaScript

Autor 1: Federico Pérez Ramírez

*Departamento o Escuela, Universidad, Ciudad, País*

Correo-e: ejemplo@org.es

***Resumen— En este tema haremos un resumen de lo visto en el manual de JavaScript. El resumen puede servirnos de recordatorio o de consulta para nuestros trabajos.***

***Aunque no sigamos exactamente el mismo orden que en el manual, pondremos todo lo más importante concerniente al código JavaScript.***

***Palabras clave—* Términos-Acerca del índice de cuatro palabras o frases clave en orden alfabético, separadas por comas. Para obtener una lista de palabras claves sugeridas, envíe un correo electrónico en blanco a keywords@ieee.org o visite http://www.ieee.org/organizations/pubs/ani\_prod/keywrd98.txt.**

***Abstract*— *In this topic we will summarize what was seen in the JavaScript manual. The summary can serve as a reminder or consultation for our work.***

***Although we do not follow exactly the same order as in the manual, we will put everything the most important concerning the JavaScript code.***

***Key Word* —About four key words or phrases in alphabetical order, separated by commas. For a list of suggested keywords, send a blank e-mail to** [**keywords@ieee.org**](mailto:keywords@ieee.org) **or visit the IEEE web site at** <http://www.ieee.org/web/developers/webthes/index.htm>**.**

1. INTRODUCCIÓN

 Un lenguaje de programación que te permite realizar actividades complejas en una página web —  cada vez más una página web hace más cosas que sólo mostrar información estática — como mostrar actualizaciones de contenido en el momento, interactuar con mapas, animaciones gráficas 2D/3D etc. — puedes estar seguro que JavaScript está involucrado. Es la tercera capa del pastel de los estándares en las tecnologías para la web, dos de las cuales son ([HTML](https://developer.mozilla.org/es/docs/Learn/HTML) y [CSS](https://developer.mozilla.org/es/docs/Learn/CSS)), hablaremos de ellas más adelante con más detalle en otra parte de nuestra Área de Aprendizaje.

HTML: es un lenguaje de marcado que usa la estructura para dar un sentido al contenido web, por ejemplo define párrafos, cabeceras, tablas, imágenes y vídeos en la página.

CSS: es un lenguaje de reglas en cascada que usamos para aplicar un estilo a nuestro contenido en HTML, por ejemplo colocando colores de fondo, fuentes y marginando nuestro contenido en múltiples columnas.

JavaScript: Es un lenguaje de programación que te permite crear contenido nuevo y dinámico, controlar archivos de multimedia, crear imágenes animadas y muchas otras cosas más.

1. CONTENIDO

El núcleo de JavaScript consiste en características comunes de programación que te permiten hacer cosas como:

* Almacenar valores útiles dentro de variables. En el ejercicio anterior por ejemplo, preguntamos por un nuevo nombre a introducir para luego almacenarlo en una variable llamada name.
* Las operaciones escritas en formato de texto (Conocidas como "**Strings**" en lenguaje de programación). En el ejemplo anterior tomamos el conjunto "Player 1:" y lo adjuntamos a la variable name para terminar de crear el código, ejemplo: "Player 1: Chris".
* Para hacer funcionar el código en respuesta a algunos eventos que están ocurriendo en la página web. Usamos un evento en nuestro ejemplo anterior llamado clic para detectar cuando el botón es cliqueado para luego hacer correr el código que actualice la etiqueta de texto.

Además de lo anterior se pueden realizar muchas más cosas.

Hay algo incluso más emocionante, sin embargo es la funcionalidad construida por encima del núcleo del lenguaje de JavaScript. Llamada **Interfaz de programación en aplicaciones - Application Programming Interfaces** (**API's** ) proveyéndote súper-poderes extra para usar en tú código JavaScript.

Los APIS son inserciones de líneas, incluso bloques gigantes de código listos para usar que permiten a un desarrollador implementarlo a programas que de cualquier otra forma podría ser difícil o incluso imposible de terminar. Así como las herramientas para construir una casa, es lo mismo para las cosas de programación – Es mucho más fácil tomar los paneles que ya estén cortados y atornillarlos para hacer un estante de libros, ya que es más trabajoso diseñarlo por ti mismo, ir y encontrar la madera correcta, cortarla del tamaño correcto y lijarla, buscar los tornillos del tamaño correcto y ensamblarla para hacer un estante de libros.

Generalmente los API's se dividen en dos categorías:

**Los Navegadores APIs (Browser APIs)**

 Son construidos dentro de tu buscador web, y son capaces de exponer información desde la cercanía en base al ambiente de tu ordenador, o hacer cosas complejas. Por ejemplo:

* El [DOM (Modelo de Objeto de Documento) API](https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model) te permite manipular, crear, remover y cambiar códigos escritos en lenguaje HTML y CSS, incluso aplicando dinámicamente nuevos estilos a tu página web, etc. Cada vez que aparezca un aviso (popup) en una página web, o nuevo contenido a mostrarse en ella, (Así como vimos en el ejemplo anterior de nuestro sencillo demo) es tan sólo un ejemplo, de lo que se puede hacer con la acción DOM.
* La [Localización-Geo API](https://developer.mozilla.org/en-US/docs/Web/API/Geolocation) restablece la información geográfica, Así es como [Google Maps](https://www.google.com/maps)  permite encontrar tu dirección e imprimirla en un mapa.
* Las [Canvas](https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API) y el [WebGL](https://developer.mozilla.org/en-US/docs/Web/API/WebGL_API) API te permiten crear animaciones y gráficos en 2D y 3D. La gente está haciendo cosas increíbles usando estas tecnologías web – puedes ver los [Chrome Experiments](https://www.chromeexperiments.com/webgl) y [webglsamples](http://webglsamples.org/).
* [Video y Sonido APIs](https://developer.mozilla.org/en-US/Apps/Fundamentals/Audio_and_video_delivery), como [HTMLMediaElement](https://developer.mozilla.org/es/docs/Web/API/HTMLMediaElement" \o "La interfaz HTMLMediaElement añade a HTMLElement las propiedades y métodos necesarios para soportar las capacidades básicas relacionadas con los medios de comunicación que son comunes en audio y vídeo. Los elementos HTMLVideoElement} y HTMLAudioElement} heredan esta interfaz.) y [WebRTC](https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API)te permite crear cosas realmente interesantes, tanto como poner música y vídeo en una página web o grabar un vídeo desde tu cámara web y que otra persona pueda verla desde su computadora (Prueba nuestro ejercicio [Snapshot demo](http://chrisdavidmills.github.io/snapshot/) para tener una idea).

**APIs de Terceras personas,**por determinado no son construidos dentro del navegador, y generalmente tienes que conseguir su código e información de alguna parte de la Web, por ejemplo:

* El [Twitter API](https://dev.twitter.com/overview/documentation) permite que hagas cosas como mostrar los últimos tweets de alguien en tu sitio web.
* El [Google Maps API](https://developers.google.com/maps/) permite incrustar mapas personalizados en tu sitio web y otro tipo de funcionalidades.

**¿Qué es lo que JavaScript hace en tu página web?**

En este apartado, actualmente comenzaremos a ver unas cuantas líneas de código, y mientras hacemos eso vamos explorando lo que actualmente está pasando cuando haces funcionar tu código JavaScript en tu página Web.

Hagamos un breve recordatorio del historial de lo que sucede cuando cargas una página web en tu navegador (primero hablamos acerca de [cómo funciona CSS](https://developer.mozilla.org/es/docs/Learn/CSS/Introduction_to_CSS/Como_funciona_CSS) en nuestro artículo). Cuando cargas una página Web en tu navegador, tu código (HTML, CSS y JavaScript) es leído dentro de un ambiente de ejecución (pestaña del navegador). Esto es como una fábrica que coge la materia prima (Las líneas de código) y lo presenta como el producto final (vea la fig.1 para entender mejor)

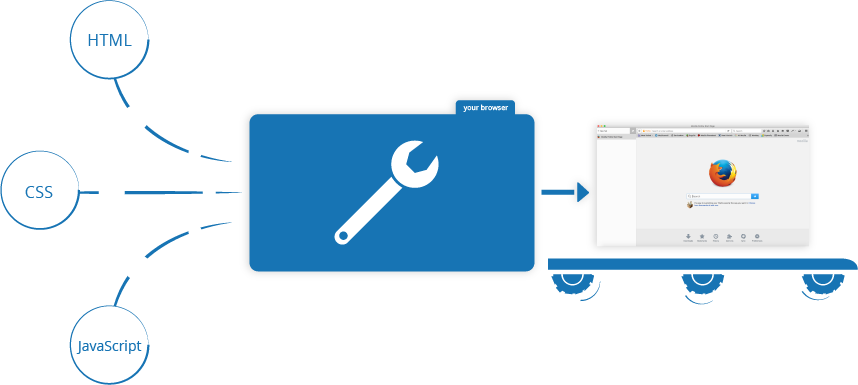


Figura 1

El lenguaje JavaScript es ejecutado por el motor del navegador de JavaScript, luego que el código HTML y CSS han sido juntados y congregados dentro de la página Web. Esto asegura que el estilo y la estructura de la página están en su lugar en el momento en que JavaScript comienza a ejecutarse.

Esto es algo muy bueno, algo muy común en el uso de JavaScript para modificar dinámicamente el código HTML y CSS, para que la interfaz de usuario sea actualizada, usando **DOM** o el modelo de objeto de documento (como se mencionó anteriormente). Si al cargar JavaScript e intentar hacerlo funcionar antes de que sea leído el código HTML y CSS, se verá afectado, ocurriendo errores de programación.

**Seguridad de navegador**

Cada pestaña del navegador se considera como una cubeta en un compartimento separado para hacer funcionar el código (en términos técnicos las cubetas son llamadas “ambientes de ejecución”) – significa que en la mayoría de los casos los códigos en cada pestaña funcionan completamente separados, y el código en una pestaña no puede afectar directamente el código de otra pestaña, o en otro navegador. Esta es una buena medida de seguridad – si este no fuera el caso, entonces los piratas informáticos podrían estar modificando tu código para robar información de otro sitio web, y otras tantas cosas malas.

**JavaScript Maneja un orden:**

Cuando el navegador encuentra un bloque de JavaScript, generalmente lo corre en orden, de arriba hacia abajo. Esto significa que tienes que tener cuidado en qué orden pones las cosas.

Ya que Si usted cambia el orden de las primeras dos líneas de código, no funcionaría – de hecho, conseguirás un error de retorno en la consola del desarrollador del navegador – “TypeError:” para” is undefined”. Esto significa que el objeto para no existe todavía, entonces no podremos añadir un evento espía.(Vea el ejemplo en fig.2)

****

Figura 2

**El interpretador entre el código compilado**

Quizás debiste escuchar sobre los términos**interpretador**y el **compilador**en contexto de programación, JavaScript es un lenguaje de interpretación – el código se hace funcionar de arriba hacia abajo y el resultado de leerlo de esta manera hace que inmediatamente responda. No tienes que transformar tu código en algo diferente antes de que el navegador lo haga por ti.

En otra mano los lenguajes compilados  son transformados (compilado) antes de que sean leídos por la computadora. Por ejemplo C/C++ son lenguajes congregados para compilarlos para luego ser leídos por la computadora.

Ambos accesos tienen diferentes ventajas, de los cuales no discutiremos en este punto.

**Estrategias de cargas de scripts**

Hay una serie de problemas relacionados con conseguir que los scripts se carguen en el momento adecuado. Nada es tan simple como parece! Un problema común es que todo el HTML de una página se carga en el orden en que aparece. Si utiliza JavaScript para manipular elementos de la página (o más exactamente, el [Document Object Model](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/Client-side_web_APIs/Manipulating_documents#The_document_object_model)), su código no funcionará si el JavaScript es cargado y analizado antes que el  código HTML al que está intentando manipular.

Una solución anticuada a este problema solía ser poner su elemento de script justo en la parte inferior del cuerpo (por ejemplo, justo antes de la etiqueta </body>), de modo que se cargara después de que todo el HTML haya sido analizado. El problema con esta solución (y la solución DOMContentLoaded vista anteriormente) es que la carga/análisis del script está completamente bloqueada hasta que se haya cargado el DOM HTML. En sitios más grandes con mucho JavaScript, esto puede causar un gran problema de rendimiento, ralentizando su sitio. Esta es la razón por la que se agregó async a los navegadores!

#### **async y defer**

En realidad hay dos maneras de evitar el problema del script de bloqueo — async y defer. Veamos la diferencia entre estos dos. (vea a continuación las figuras 3 y 4)

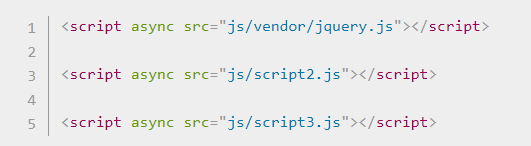


Figura 3

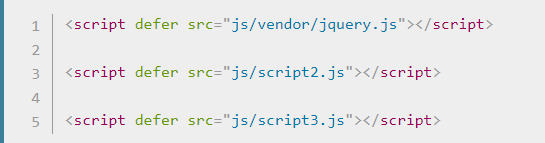
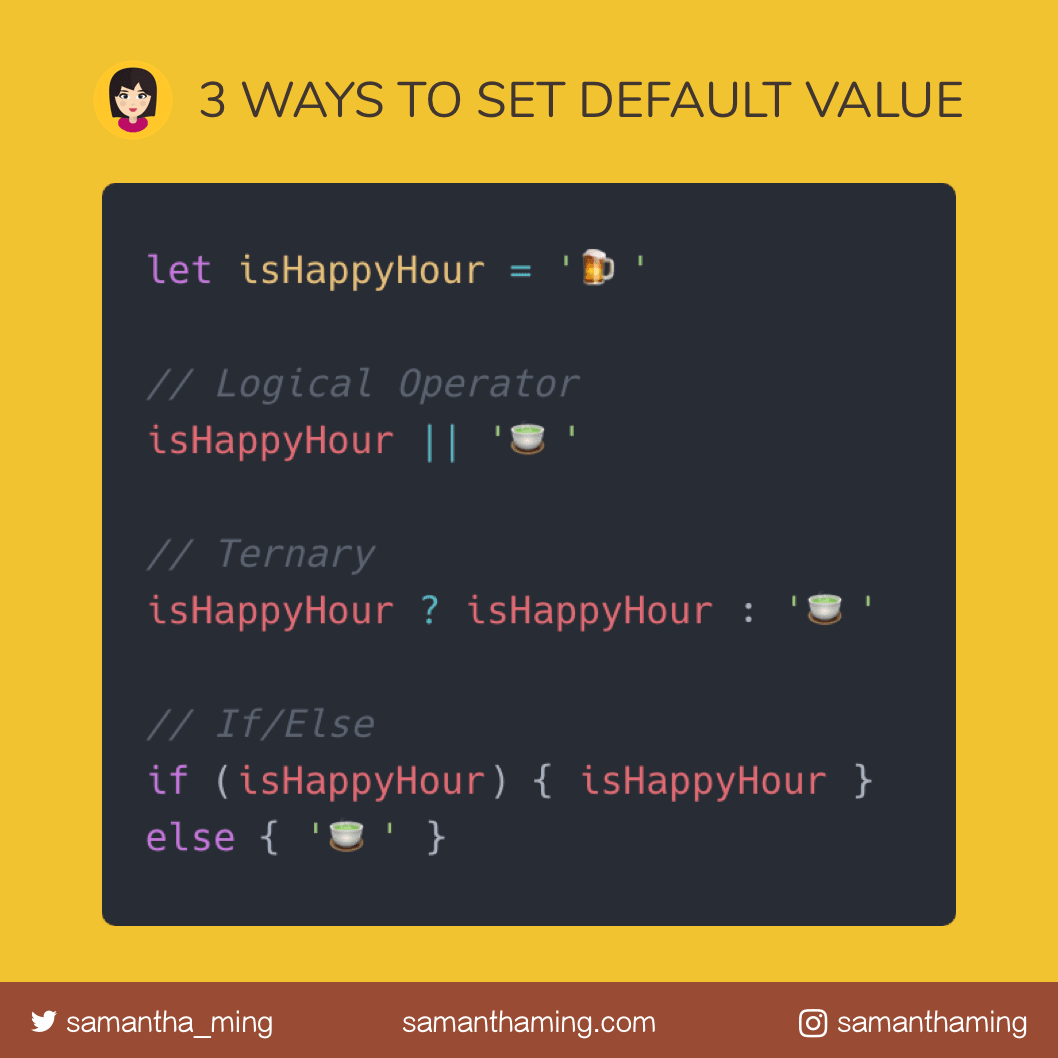


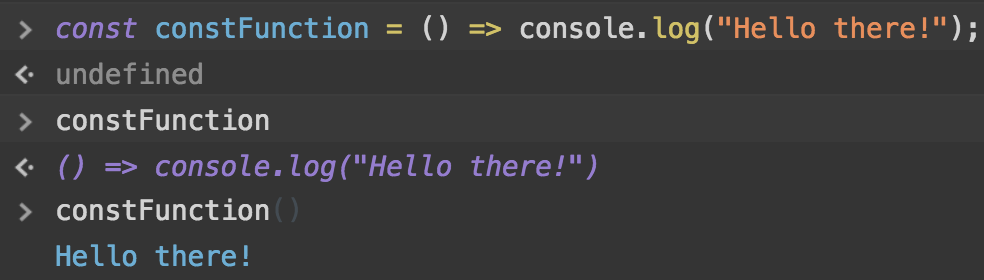
Figura 4

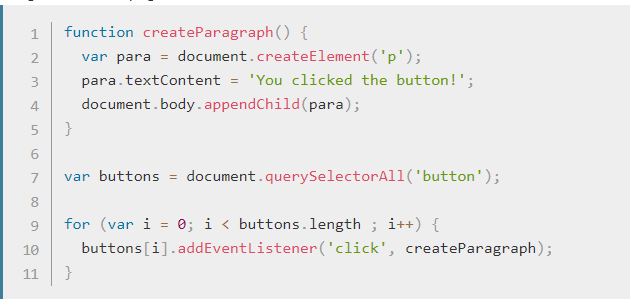
Los scripts async descargarán el script sin bloquear la renderización de la página y lo ejecutarán tan pronto como el script termine de descargarse. No se garantiza que los scripts se ejecutarán en un orden específico, sólo que no impedirán que se muestre el resto de la página. Es mejor usar async cuando los scripts de la página se ejecutan independientemente uno del otro y no dependen de ningún otro script de la página.

Imágen Que le pueden ayudar



Lineas de Codigo





1. CONCLUSIONES

.JavaScript es una ayuda indispensable para realizar páginas en las que queramos una alta interacción con el usuario, este nos proporciona una serie de eventos que reconoce, unas librerías de funciones y métodos y todo lo que queramos hacer con un lenguaje, todo con la ventaja que lo ejecuta el cliente y no tiene que estar llamando al servidor, y por lo tanto hace una navegación mucho mas rápida.​​

REFERENCIAS

<https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/Qu%C3%A9_es_JavaScript>

<https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics>

<https://aprende-web.net/javascript/js15_1.php>

<https://es.slideshare.net/guest044583b/javascript-1562899>