

JavaScript

CONTENIDO:

Breve historia	2
Incluir JavaScript en HTML	3
Sintaxis	5
Posibilidades y limitaciones	6
El primer script	7



Unidad 1 - Introducción a Javascript

JavaScript (abreviado comúnmente "JS") es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos,³ basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas⁴ aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la

web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM).

1.- Breve historia

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario.

A principios de los años 90, la mayoría de usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y por tanto, las páginas web comenzaban a incluir formularios complejos.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

Brendan Eich, un programador que trabajaba en Netscape, pensó que podría solucionar este problema adaptando otras tecnologías existentes (como ScriptEase) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript.

Posteriormente, Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por marketing, ya que Java

era la palabra de moda en el mundo informático y de Internet de la época.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA (European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, dando lugar al estándar ISO/IEC-16262.

2.- Cómo incluir JavaScript en documentos HTML

La integración de JavaScript y HTML es muy flexible, ya que existen al menos tres formas para incluir código JavaScript en las páginas web.

El código JavaScript se encierra entre etiquetas `<script>` y se incluye en cualquier parte del documento. Aunque es correcto incluir cualquier bloque de código en cualquier zona de la página, se recomienda definir el código JavaScript dentro de la cabecera del documento (dentro de la etiqueta `<head>`):

```
<head>
<script type="text/javascript">
  alert("Un mensaje de prueba");
</script>
```

Para que la página HTML resultante sea válida, es necesario añadir el atributo `type` a la etiqueta `<script>`. Los valores que se incluyen en el atributo `type` están estandarizados y para el caso de JavaScript, el valor correcto es `text/javascript`.

Este método se emplea cuando se define un bloque pequeño de código o cuando se quieren incluir instrucciones específicas en un determinado documento HTML que completen las instrucciones y funciones que se incluyen por defecto en todos los documentos del sitio web.

El principal inconveniente es que si se quiere hacer una modificación en el bloque de código, es necesario modificar todas las páginas que incluyen ese mismo bloque de código JavaScript.

Las instrucciones JavaScript se pueden también incluir en un archivo externo de tipo JavaScript que los documentos XHTML enlazan mediante la etiqueta `<script>`. Se pueden crear todos los archivos JavaScript que sean ne-

cesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

Ejemplo:

Archivo `codigo.js`

```
alert("Un mensaje de prueba");
```

Documento XHTML

```
<head>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>
```

Además del atributo `type`, este método requiere definir el atributo `src`, que es el que indica la URL correspondiente al archivo JavaScript que se quiere enlazar. Cada etiqueta `<script>` solamente puede enlazar un único archivo, pero en una misma página se pueden incluir tantas etiquetas `<script>` como sean necesarias.

Los archivos de tipo JavaScript son documentos normales de texto con la extensión `.js`, que se pueden crear con cualquier editor de texto como Notepad, Wordpad, EmEditor, UltraEdit, Vi, etc.

La principal ventaja de enlazar un archivo JavaScript externo es que se simplifica el código XHTML de la página, que se puede reutilizar el mismo código JavaScript en todas las páginas del sitio web y que cualquier modificación realizada en el archivo JavaScript se ve reflejada inmediatamente en todas las páginas XHTML que lo enlazan.

También podemos incluir código javascript en cualquier elemento del `body`. Este último método es el menos

Algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura

utilizado, ya que consiste en incluir trozos de JavaScript dentro del código XHTML de la página:

```
<body>
<p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
</body>
</html>
```

El mayor inconveniente de este método es que ensucia innecesariamente el código XHTML de la página y complica el mantenimiento del código JavaScript. En general, este método sólo se utiliza para definir algunos eventos y en algunos otros casos especiales, como se verá más adelante.

Algunos navegadores no disponen de soporte completo de JavaScript, otros navegadores permiten bloquearlo parcialmente e incluso algunos usuarios bloquean completamente el uso de JavaScript porque creen que así navegan de forma más segura.

En estos casos, es habitual que si la página web requiere JavaScript para su correcto funcionamiento, se incluya un mensaje de aviso al usuario indicándole que debería activar JavaScript para disfrutar completamente de la página. El siguiente ejemplo muestra una página web basada en JavaScript cuando se accede con JavaScript activado y cuando se accede con JavaScript completamente desactivado.

HTML define la etiqueta `<noscript>` para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript. El siguiente código muestra un ejemplo del uso de la etiqueta `<noscript>`:

```
<body>
<noscript>
  <p>La página que estás viendo requiere para su funcionamiento el uso de JavaScript..</p>
</noscript>
</body>
```

La etiqueta `<noscript>` se debe incluir en el interior de la etiqueta `<body>` (normalmente se incluye al principio de `<body>`). El mensaje que muestra `<noscript>` puede incluir cualquier elemento o etiqueta XHTML.

JavaScript, como cualquier lenguaje de programación, contiene una serie de palabras reservadas: que son las palabras (en inglés) que se utilizan para construir las sentencias de JavaScript y que por tanto no pueden ser utilizadas libremente.

3.— Sintaxis

La sintaxis de un lenguaje de programación se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para considerarse como correctos para ese lenguaje de programación.

La sintaxis de JavaScript es muy similar a la de otros lenguajes de programación como Java y C. Las normas básicas que definen la sintaxis de JavaScript son las siguientes:

- No se tienen en cuenta los espacios en blanco y las nuevas líneas: como sucede con XHTML, el intérprete de JavaScript ignora cualquier espacio en blanco sobran te, por lo que el código se puede ordenar de forma adecuada para entenderlo mejor (tabulando las líneas, añadiendo espacios, creando nuevas líneas, etc.)
- Se distinguen las mayúsculas y minúsculas: al igual que sucede con la sintaxis de las etiquetas y elementos XHTML. Sin embargo, si en una página XHTML se utilizan indistintamente mayúsculas y minúsculas, la página se visualiza correctamente, siendo el único problema la no validación de la página. En cambio, si en JavaScript se intercambian mayúsculas y minúsculas el script no funciona.
- No se define el tipo de las variables: al crear una variable, no es necesario indicar el tipo de dato que almacenará. De esta forma, una misma variable puede almacenar diferentes tipos de datos durante la ejecución del script.
- No es necesario terminar cada sentencia con el carácter de punto y coma (;): en la mayoría de lenguajes de programación, es obligatorio terminar cada sentencia con el carácter ;. Aunque JavaScript no obliga a hacerlo, es conveniente seguir la tradición de terminar cada sentencia con el carácter del punto y coma (;).

- Se pueden incluir comentarios: los comentarios se utilizan para añadir información en el código fuente del programa. Aunque el contenido de los comentarios no se visualiza por pantalla, si que se envía al navegador del usuario junto con el resto del script, por lo que es necesario extremar las precauciones sobre la información incluida en los comentarios.

JavaScript define dos tipos de comentarios: los de una sola línea y los que ocupan varias líneas.

Ejemplo de comentario de una sola línea:

```
// a continuación se muestra un mensaje  
alert("mensaje de prueba");
```

Los comentarios de una sola línea se definen añadiendo dos barras oblicuas (//) al principio de la línea.

Ejemplo de comentario de varias líneas:

```
/* Los comentarios de varias líneas son muy útiles  
cuando se necesita incluir bastante información  
en los comentarios */  
alert("mensaje de prueba");
```

Los comentarios multilínea se definen encerrando el texto del comentario entre los símbolos /* y */.

4.— Posibilidades y limitaciones

Desde su aparición, JavaScript siempre fue utilizado de forma masiva por la mayoría de sitios de Internet. La aparición de Flash disminuyó su popularidad, ya que Flash permitía realizar algunas acciones imposibles de llevar a cabo mediante JavaScript.

Sin embargo, la aparición de las aplicaciones AJAX programadas con JavaScript le ha devuelto una popularidad sin igual dentro de los lenguajes de programación web.

En cuanto a las limitaciones, JavaScript fue diseñado de forma que se ejecutara en un entorno muy limitado que permitiera a los usuarios confiar en la ejecución de los scripts.

De esta forma, los scripts de JavaScript no pueden comunicarse con recursos que no pertenezcan al mismo dominio desde el que se descargó el script. Los scripts tampoco pueden cerrar ventanas que no hayan abierto esos mismos scripts. Las ventanas que se crean no pueden ser demasiado pequeñas ni demasiado grandes ni colocarse fuera de la vista del usuario (aunque los detalles concretos dependen de cada navegador).

Además, los scripts no pueden acceder a los archivos del ordenador del usuario (ni en modo lectura ni en modo escritura) y tampoco pueden leer o modificar las preferencias del navegador.

Por último, si la ejecución de un script dura demasiado tiempo (por ejemplo por un error de programación) el navegador informa al usuario de que un script está consumiendo demasiados recursos y le da la posibilidad de detener su ejecución.

A pesar de todo, existen alternativas para poder saltarse algunas de las limitaciones anteriores. La alternativa más utilizada y conocida consiste en firmar digitalmente el script y solicitar al usuario el permiso para realizar esas acciones.

5.— El primer script

A continuación, se muestra un primer script sencillo pero completo:

```
<html>
<head>
<script type="text/javascript">
  alert("Hola Mundo!");
</script>
</head>
<body>
<p>Esta página contiene el primer script</p>
</body>
</html>
```

En este ejemplo, el script se incluye como un bloque de código dentro de una página XHTML. Por tanto, en primer lugar se debe crear una página XHTML correcta que incluya la declaración del DOCTYPE, el atributo xmlns, las secciones <head> y <body>, la etiqueta <title>, etc.

Aunque el código del script se puede incluir en cualquier parte de la página, se recomienda incluirlo en la cabecera del documento, es decir, dentro de la etiqueta <head>.

A continuación, el código JavaScript se debe incluir entre las etiquetas <script>...</script>. Además, para que la página sea válida, es necesario definir el atributo type de la etiqueta <script>. Técnicamente, el atributo type se corresponde con "el tipo MIME", que es un estándar para identificar los diferentes tipos de contenidos. El "tipo MIME" correcto para JavaScript es text/javascript.

Una vez definida la zona en la que se incluirá el script, se escriben todas las sentencias que forman la aplicación. Este primer ejemplo es tan sencillo que solamente incluye una sentencia: alert("Hola Mundo!");.

La instrucción alert() es una de las utilidades que incluye JavaScript y permite mostrar un mensaje en la pantalla del usuario. Si se visualiza la página web de este primer script en cualquier navegador, automáticamente se mostrará una ventana con el mensaje "Hola Mundo!".

A continuación se muestra el resultado de ejecutar el script :



Ejercicio 1

Modificar el primer script para que:

Todo el código JavaScript se encuentre en un archivo externo llamado codigo.js y el script siga funcionando de la misma manera.

Después del primer mensaje, se debe mostrar otro mensaje que diga "Soy el primer script"

Añadir algunos comentarios que expliquen el funcionamiento del código

Añadir en la página XHTML un mensaje de aviso para los navegadores que no tengan activado el soporte de JavaScript



Este material forma parte del currículum del módulo **Implantación de Aplicaciones WEB** correspondiente al ciclo formativo de grado superior “**Administración de sistemas informáticos en red**”.

La información contenida en el magazine ha sido extraída de la **Web** y, en parte, de la experiencia profesional acumulada durante mis años de docencia.

Por tanto, es de uso público para tareas de docencia y aprendizaje.