

IES Gran Capitán
Módulo: Desarrollo Web en entorno
cliente

Ciclo Formativo de Grado Superior “Desarrollo de aplicaciones Web”



Etiqueta script. Añadiendo comportamiento al html

Fecha entrega: 26-09-2018

Autor: Guillermo Boquizo Sánchez

Etiqueta script. Añadiendo comportamiento al html

Realiza las siguientes lecturas y contesta a las siguientes cuestiones:

[async vs defer attributes](#)

[Where should I put <script> tags in HTML markup?](#)

[Etiqueta <NOSCRIPT>, Curso práctico HTML5](#)

1. Ventajas de incorporar el script desde un fichero externo frente a hacerlo en línea (código javascript directamente en el fichero HTML).

Las ventajas que podemos enumerar a la hora de incorporar código JavaScript desde un fichero externo, frente a hacerlo de manera embebida en el propio código HTML son las siguientes:

1. Mejor legibilidad del código, al encontrarse separado del HTML
2. Mayor control del código desarrollado, al poder localizar rápidamente lo que se quiera buscar sin perderse entre etiquetas HTML.
3. Permite una programación por capas, separando presentación de comportamiento.
4. Permite la modularización de aplicaciones.
5. Permite la reutilización de funciones de uso común.
6. Facilita el mantenimiento de las mismas.
7. Obliga al desarrollador a ser ordenado.
8. Permite la vinculación a un mismo fichero desde diferentes documentos HTML.

2. Atributos aplicables a la etiqueta script:

Fuente: https://www.w3schools.com/tags/tag_script.asp

La etiqueta script soporta los siguientes atributos, algunos predeterminados, otros denominados booleanos:

- type
- src
- charset
- defer
- async
- xml:space (no soportado en HTML5)
- crossorigin
- importance
- integrity
- nomodule
- nonce
- text
- language

Además, también soporta atributos globales en HTML como se recoge [aquí](#) y [aquí](#)

3. Atributos por defecto.

Los atributos por defecto de la etiqueta script son type, src y charset

```
<script type="text/javascript" src="ruta/script.js"  
charset="charset"></script>
```

type: Indica el tipo de archivo.

src: Indica la ruta del archivo js externo.

charset: Especifica la codificación del archivo js externo.

4. Atributos booleanos. Qué implican.

Los atributos booleanos `async` y `defer` indican que, al estar presentes, especifican que el script debe ser ejecutado de forma asíncrona en cuanto éste se encuentre disponible.

Atributo **async**:

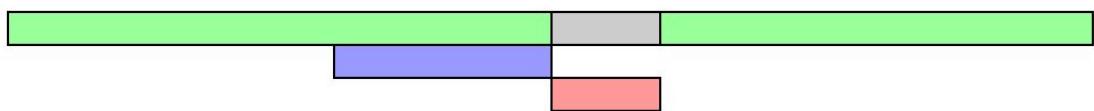
Fuente: https://www.w3schools.com/tags/att_script_async.asp

```
<script type="text/javascript" src="ruta/script.js" async></script>
```

Este atributo permite la descarga asíncrona (paralela) de los archivos que contienen los scripts mientras se parsea el HTML, parando dicho parseo para su ejecución tras la descarga, y continuando tras dicha ejecución el parseo. Evitamos de esta manera que el código HTML se vea bloqueado mientras el script se está ejecutando.

Legend

- █ HTML parsing
- █ HTML parsing paused
- █ Script download
- █ Script execution



Atributo **defer**:

Fuente: https://www.w3schools.com/tags/att_script_defer.asp

```
<script type="text/javascript" src="ruta/script.js" defer></script>
```

Este atributo permite la descarga asíncrona (paralela) de los archivos que contienen los scripts mientras se parsea el HTML, evitando que el navegador quede bloqueado mientras los descarga. A diferencia del atributo ASYNC, la ejecución del código javascript descargado se da tras el parseo del HTML completo, por lo que se garantiza la totalidad de carga del árbol DOM.

Legend

- █ HTML parsing
- █ HTML parsing paused
- █ Script download
- █ Script execution



Uso combinado:

```
<script type="text/javascript" src="ruta/script.js" async defer></script>
```

Este modo ejecuta defer si async no es compatible con el navegador.

Diferencias entre asyn y defer

El atributo defer (desarrollado en Internet Explorer 4 y posteriormente añadido dentro de las especificaciones de HTML 4) lleva utilizándose más tiempo y nos permite indicar que el script se ejecutará una vez se haya cargado el resto de la página.

Sin embargo, async se ejecutará sin bloquear el resto de descargas que pudieran existir en paralelo, aunque el DOM no haya terminado de

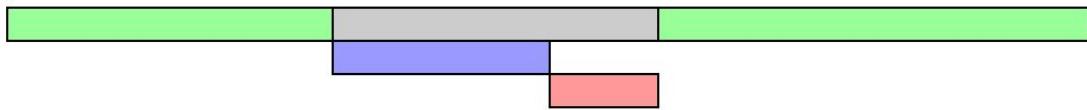
cargarse por completo, lo cual significa que el propio script podría estar disponible antes que la página, cosa que en ocasiones no debe suponer un inconveniente.

Resumiendo:

Script **sin atributos**: se ejecutará inmediatamente, bloqueando el parseo del resto del documento.

Legend

- HTML parsing
- HTML parsing paused
- Script download
- Script execution



Script y **defer**: se ejecutará una vez que el resto del documento haya sido parseado.

Script y **async**: se ejecutará mientras el resto del documento termina de parsearse, de manera asíncrona.

5. Sitio recomendado para colocar la etiqueta script.

Esto es lo que ocurre cuando un navegador carga una web con una etiqueta <script>:

1. Toma la página HTML (v.g: index.html)
2. Comienza a parsear el HTML
3. El parser encuentra una etiqueta <script> referenciando un archivo con script externo.
4. El navegador busca el archivo. Mientras tanto, el parseador se bloquea y deja de parsear el resto de HTML de la página.
5. Después de un tiempo el script es descargado y ejecutado.
6. El parseador continúa parseando el resto de la página.

El punto 4 genera una mala experiencia de usuario, al detenerse la carga de la web hasta que no se hayan descargado todos los scripts.

La antigua recomendación era colocar la etiqueta <script> al final de <body>, puesto que así se garantiza que el parseador no es bloqueado hasta que no se haya descargado el árbol DOM en su totalidad.

No obstante, en sitios grandes, con hojas de estilos grandes y scripts grandes, descargar cuanto antes los scripts es importante para la puesta a punto.

Como hemos visto, el uso de atributos como `async` y `defer` permiten una descarga asíncrona, permitiendo una descarga temprana de los scripts y sin bloquear el navegador.

La mayoría de los navegadores (un 94,59 % según caniuse.com) soportan este tipo de atributos.

Por todo ello, la recomendación actual es colocar el script en la etiqueta <head> y usar los atributos `async` y `defer`, permitiendo tanto la descarga rápida como evitando el bloqueo del navegador.

6. Etiqueta noscript: utilidad, atributos y dónde colocar

Fuente: https://www.w3schools.com/tags/tag_noscript.asp

La etiqueta <noscript> es una etiqueta que complementa a <script>, puesto que se utiliza para definir un contenido alternativo en el documento web cuando el navegador no soporta la ejecución de scripts o cuando éstos se encuentran desactivados en el navegador que recibe el documento.

La etiqueta se puede utilizar dentro del encabezado del documento o bien en el cuerpo del mismo, dependiendo de su posición se suele utilizar de forma distinta.

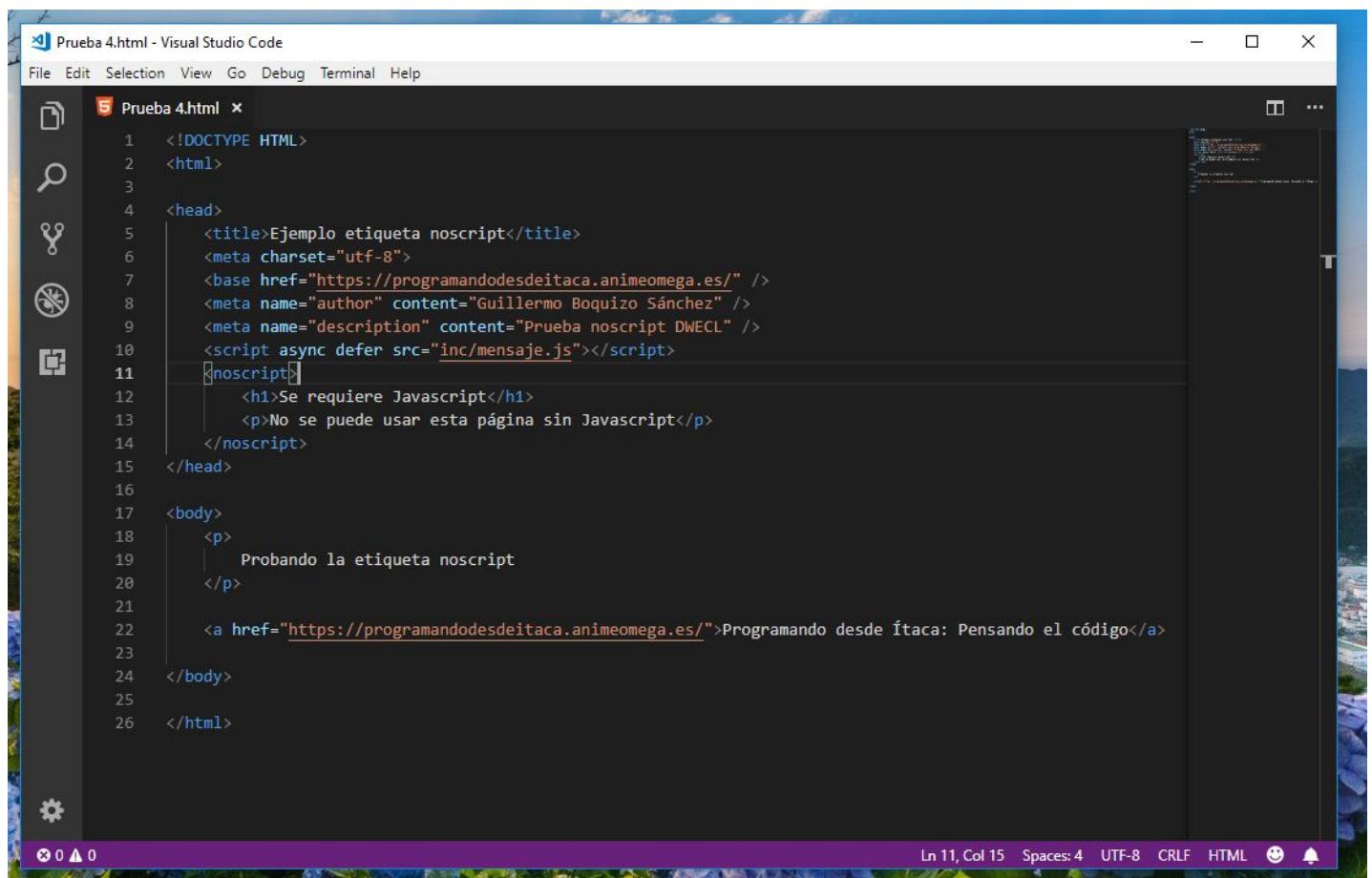
Normalmente si se utiliza en el encabezado del documento, es decir dentro de la etiqueta <head>, suele afectar a etiquetas como <link>, <style> y <meta>, de forma que permite realizar la carga alternativa de estos elementos en caso de que el navegador no permite la ejecución de scripts.

Por otra parte se suele utilizar dentro del cuerpo del documento para indicar que el navegador no permite la ejecución del script que se intenta ejecutar.

La etiqueta es soportada en la mayoría de navegadores, siendo su sintaxis la siguiente:

<noscript> Contenido a visualizar o cargar por el navegador </noscript>

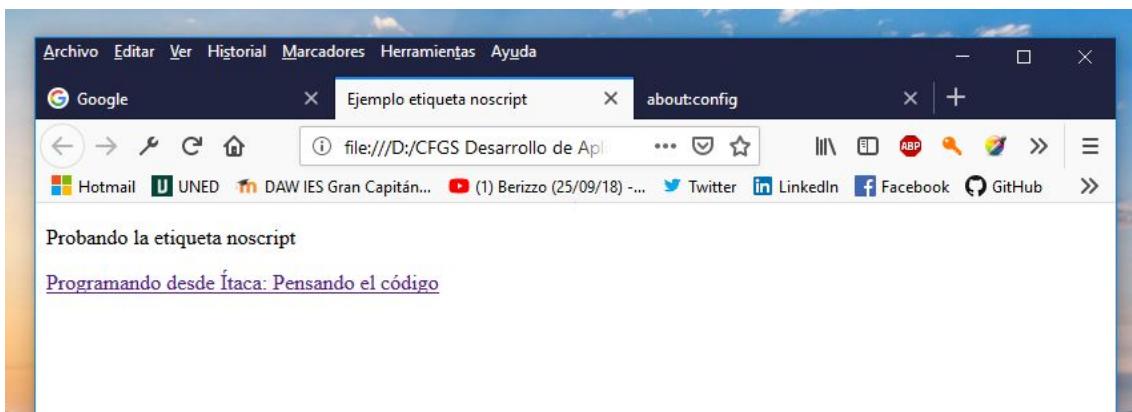
Así, si probamos este código de ejemplo:



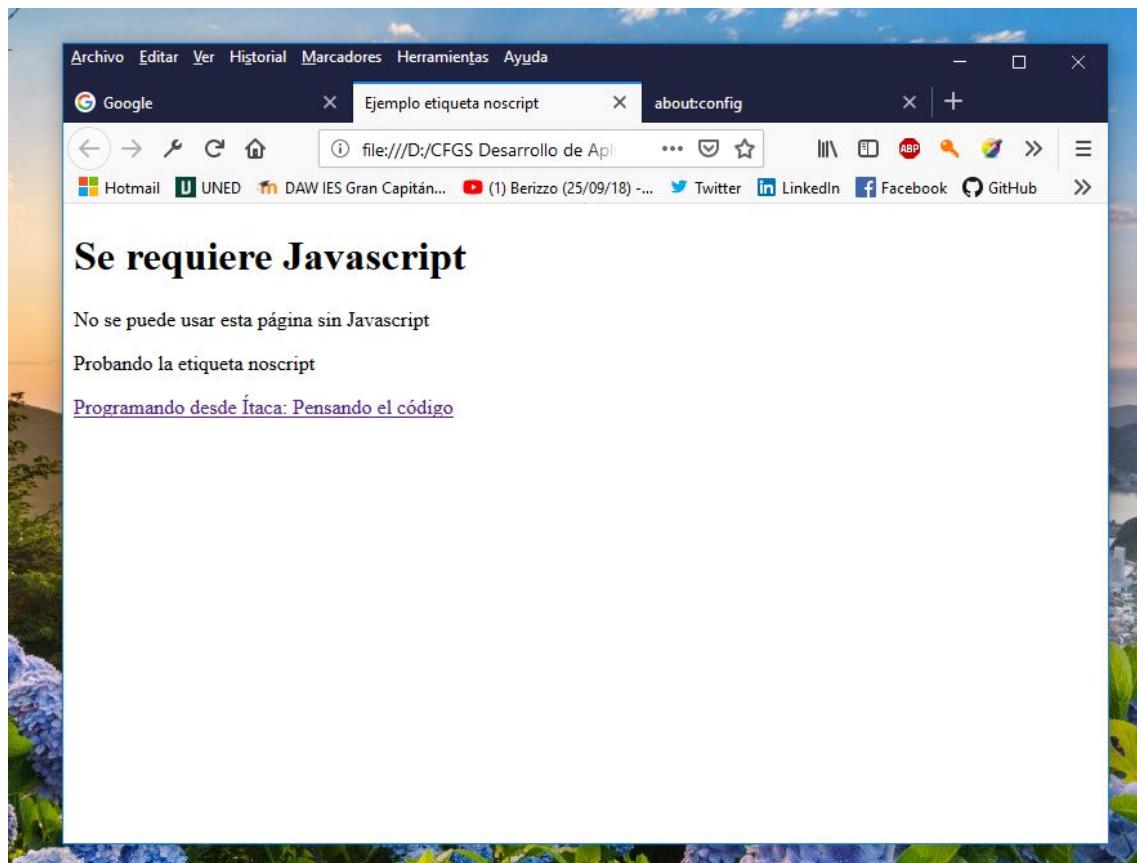
The screenshot shows the Visual Studio Code interface with the following details:

- Title Bar:** Prueba 4.html - Visual Studio Code
- Menu Bar:** File, Edit, Selection, View, Go, Debug, Terminal, Help
- Code Editor:** The code is written in HTML. It includes a `<head>` section with a title, meta tags for charset, href, author, and description, and a `<script>` tag. The `<head>` section is closed with a `</head>`. The `<body>` section contains a `<p>` tag with the text "Probando la etiqueta noscript" and an `` tag with the text "Programando desde Ítaca: Pensando el código".
- Status Bar:** Shows "Ln 11, Col 15" and other settings like "Spaces: 4", "UTF-8", "CRLF", "HTML", and icons for file status and notifications.

Nos encontramos que en un contexto donde sí se carga el archivo, en este caso Firefox Developer, visualizamos el mensaje incluido en `p`, pero no veremos los mensajes que se encuentran dentro de `<noscript>`:



Sin embargo, al probar con javascript desactivado:



Observamos en este caso la acción de la etiqueta, al no poder cargar el script.

Los atributos de esta etiqueta son:

atributo	descripción	valor
Genéricos		
<code>title</code>	Texto informativo o título del elemento. Suele mostrarse a modo de "tool tip".	Texto legible por personas. Sensible a <u>M/m.</u> . Por defecto: Lo fija el navegador.
<code>id</code>	Le da un nombre al elemento que lo diferencia de todos los demás del documento.	Un <code>nombre único</code> . Sensible a <u>M/m.</u> . Por defecto: Lo fija el navegador.
<code>class</code>	Asigna nombres de clases al elemento. Por defecto, clases CSS.	Lista de clases separadas por espacio en blanco. Sensible a <u>M/m.</u> . Por defecto: Lo fija el navegador.
<code>style</code>	Permite especificar <code>información de estilo</code> . Por defecto, estilos CSS.	<code>Declaraciones de estilo</code> . Por defecto: Lo fija el navegador.
<code>lang</code>	Información sobre el <code>idioma del contenido</code> del elemento y del valor de sus atributos.	Un <code>código de idioma</code> . Por defecto: "desconocido". Lo fija el navegador.
<code>dir</code>	Indica la <code>dirección de texto</code> y tablas.	Uno de los siguientes: 'ltr' o 'rtl'. Por defecto: En castellano 'ltr'. Lo fija el navegador.

Fuente: <https://developer.mozilla.org/es/docs/Web/HTML/Elemento/noscript>

