UNIDAD DE TRABAJO 2

INTRODUCCIÓN A JAVASCRIPT. INTEGRACIÓN CON HTML.

Índice

En esta unidad de trabajo se introduce al alumno al lenguaje JavaScript utilizado en la actualidad principalmente en los navegadores. Se define JavaScript, su historia, el modelo cliente/servidor, centrándose en el cliente, donde se ejecuta JavaScript, las herramientas y entornos de desarrollo que se van a utilizar en el desarrollo del módulo, inclusión de JavaScript en ficheros HTML, la sintaxis general del lenguaje, y por último un conjunto de ejercicios.

1.- Definición.

JavaScript es un lenguaje de programación interpretado, es decir no se genera un fichero en el código binario de la máquina donde se va a ejecutar, por ejemplo C, C++, sino que es ejecutado por un intérprete, en caso de JavaScript se encuentra dentro del navegador, aunque puede encontrarse en otros programas, entre ellos Photoshop para automatizar y ampliar la funcionalidad del mismo.

Otras de las características del lenguaje es su tipado. El tipado de un lenguaje se define en función de si es más o menos fuerte o débil, y más o menos dinámico o estático.

Un lenguaje es de tipado fuerte si no se permiten violaciones de los tipos de datos, es decir, dado el valor de una variable de un tipo concreto, no se puede usar como si fuera de otro tipo distinto a menos que se haga una conversión.

```
Char c='d';
float r;
float subrutina(float Parametro)
{
   return Parametro + 1.5;
}
.r=subrutina(c);
```

Error de compilación en C, Parámetro de tipo float y 1.5

Solución: Conversión implícita de tipos (responsabilidad programador).

```
r=subrutina((float)c);
```

En JavaScritpt

```
var c='d';
```

```
var r;
function subrutina( Parametro)
{
   return Parametro + 1.5;
}
r=subrutina(c);
```

El resultado obtenido a ser tipado débil depende del lenguaje.

Un lenguaje de tipado estático indica que durante la vida de la variable, su tipo permanece constante, en caso de tipado dinámico el tipo de dato de la variable puede cambiar durante la vida de la variable.

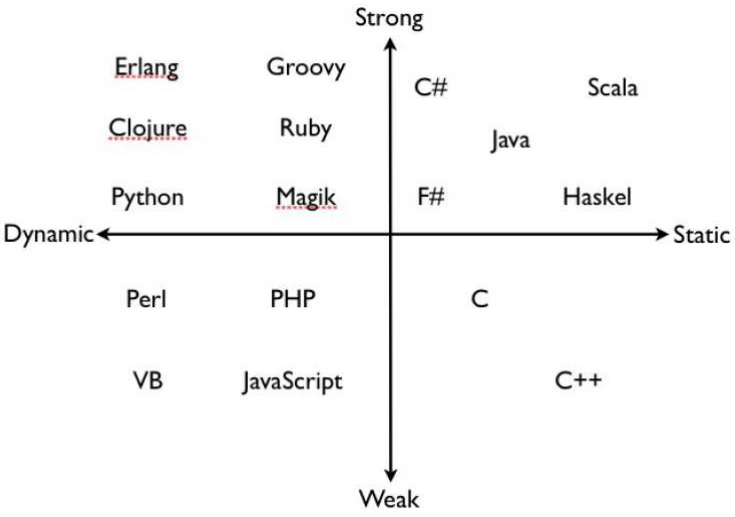
En C:

```
int d=5;
d=4.5;
```

Error de compilación, asignación incorrecta.

En Javascript:

```
var i=45454;
i="hola";
i=ventana.widht;
```



1.1.- Características.

Sus características generales son:

- La sintaxis es muy similar a C++ o Java.
- Aunque es orientado a objetos, no existe el concepto de clase, sino una colección (estructura de datos) de métodos y propiedades.

Se utiliza para:

- Interactuar con el navegador: Abrir ventanas, redireccionar...
- Interactuar con la página HTML: Verificar formularios, añadir y eliminar contenido, realizar animaciones...
- Abrir conexiones con servidores (AJAX).
- Trabajar con base de datos en el cliente (en las últimas versiones de HTML 5, es posible interactuar con base de datos en el cliente).

2.- Historia.

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.

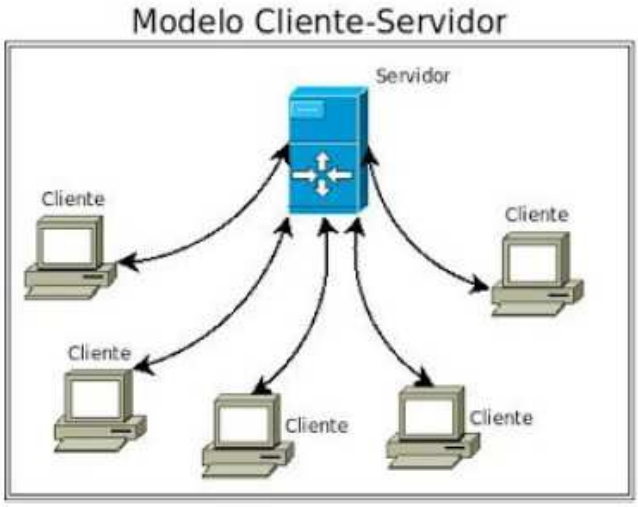
Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

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.

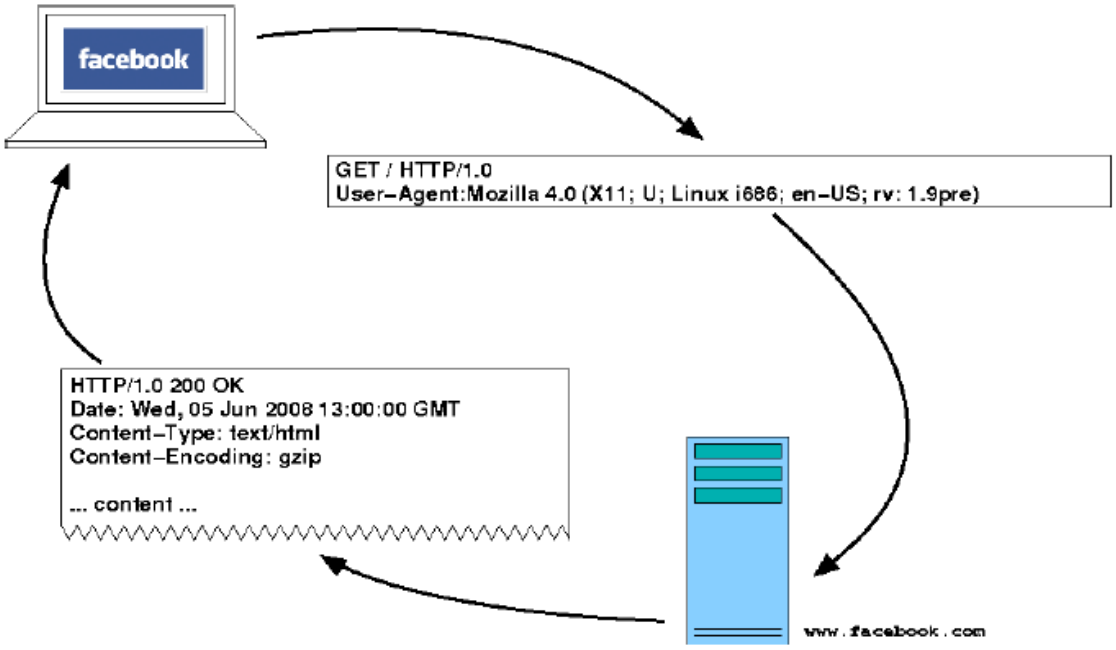
3.- Arquitectura cliente/servidor.

Existen múltiples arquitecturas para la implementación de soluciones informáticas, cada una de ellas con sus ventajas e inconvenientes.

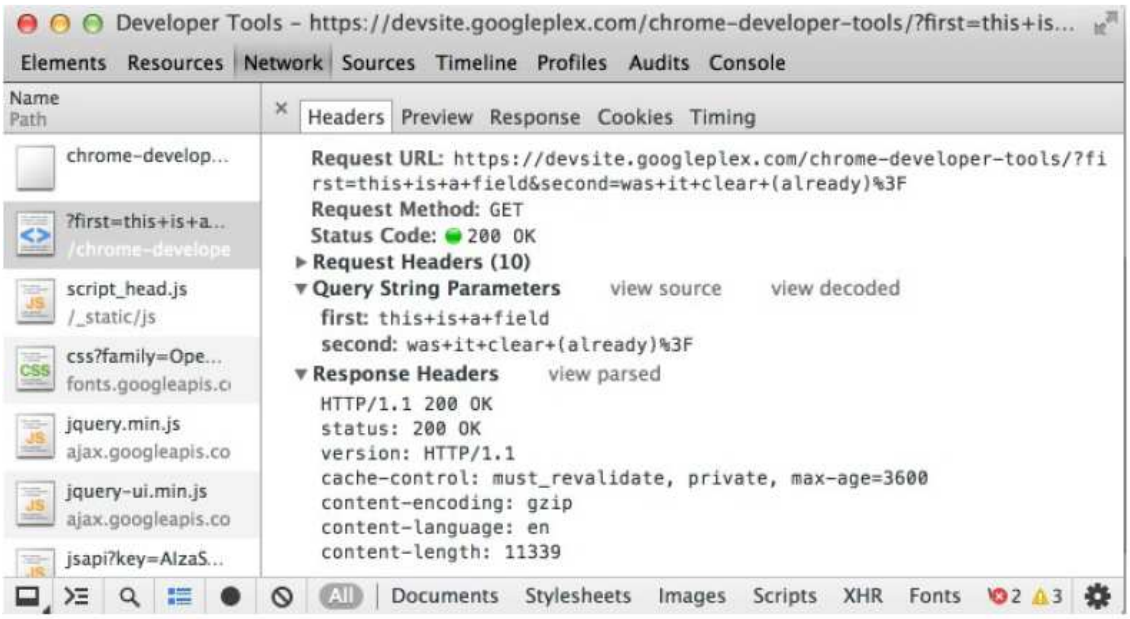
Una de las más utilizada es la de cliente/servidor, que como su nombre indica tiene dos elementos, por un lado el cliente, que se encarga de enviar información y recibir los datos del servidor, y por otra el servidor, que está a la espera de peticiones realizadas por los clientes, realizando las acciones oportunas para enviar a los clientes lo que han solicitado.



En el caso de la web, la arquitectura cliente/servidor se basa en la utilización del protocolo HTTP, cuya característica principal es que no posee estado, es decir cuando se necesita enviar datos al servidor se establece la conexión, se envían los datos (REQUEST, donde se especifica la acción a realizar por el servidor), el servidor procesa los datos y envía la respuesta al cliente(RESPONSE), para una vez recibidos por el cliente la conexión finaliza, teniéndose que volver a realizar la conexión si se necesita enviar nuevos datos.



Utilizando los navegadores actuales es posible ver la información que se envía y recibe, por ejemplo en Chrome, antes de realizar la petición abriendo las herramientas para desarrollador, se pueden ver los paquetes enviados al servidor y recibidos por el cliente, en la pestaña Network.



Una vez la información solicitada llega al cliente se hace necesario el tratamiento de dicha información, normalmente HTML, aquí es donde aparecen las tecnologías de cliente, en concreto Javascript, permitiendo modificar el HTML recibido, añadir efectos visuales, validar datos antes de realizar la petición y un largo etcétera.

4.- Incluir JavaScript en páginas HTML.

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

4.1.- Incluir JavaScript en el mismo documento XHTML

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>***):

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type="text/javascript">
  alert("Un mensaje de prueba");
</script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

Para que la página XHTML 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.

4.2- Definir JavaScript en un archivo externo

Las instrucciones JavaScript se pueden 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 necesarios y cada documento XHTML puede enlazar tantos archivos JavaScript como necesite.

Ejemplo:

Archivo codigo.js

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

```
Documento XHTML
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type="text/javascript" src="/js/codigo.js"></script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

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, VS Code, 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.

4.3- Incluir JavaScript en los elementos XHTML

Este último método es el menos utilizado, ya que consiste en incluir trozos de JavaScript dentro del código XHTML de la página:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
</head>

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

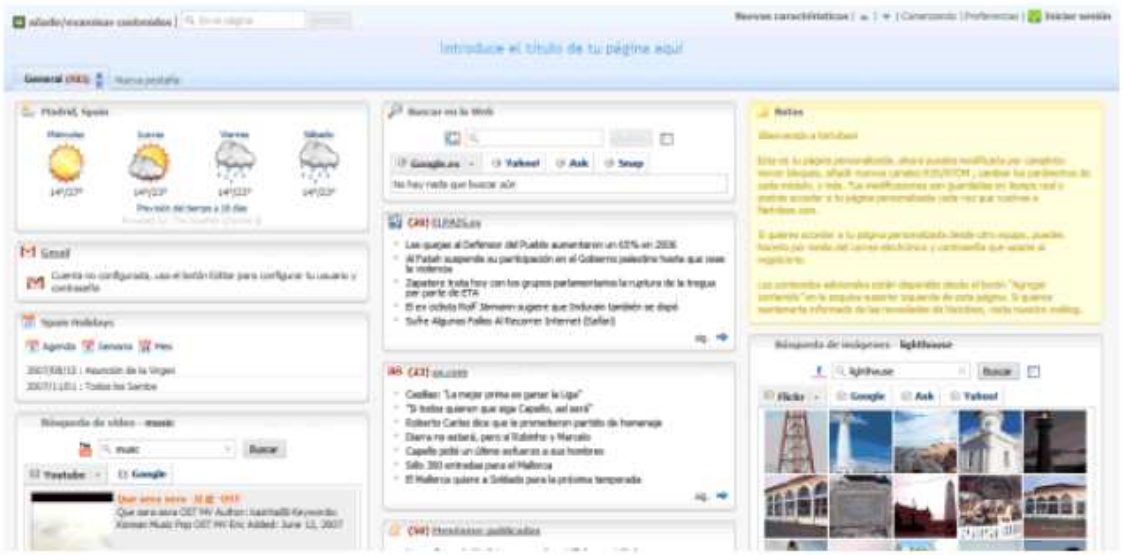
El mayor inconveniente de este método es que ensucia innecesariamente el código XHTML de la página y complica el mantenimiento del código JavaScript. En general, este método sólo se utiliza para definir algunos eventos y en algunos otros casos especiales, como se verá más adelante.

4.4- Noscript.

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.



El lenguaje 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>`:

```
<head> ... </head>
<body>
<noscript>
  <p>Bienvenido a Mi Sitio</p>
   <p>La página que estás viendo requiere para su funcionamiento el uso de
JavaScript.
Si lo has deshabilitado intencionadamente, por favor vuelve a
activarlo.</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.

5.- Sintaxis de lenguaje.

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 para 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 sobrante, 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, sí 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 */.

Las normas completas de sintaxis y de cualquier otro aspecto relacionado con JavaScript se pueden consultar en el estándar oficial del lenguaje que está disponible en:

http://www.ecma-international.org/publications/standards/Ecma-262.htm

6. Ejercicios.

1. Como crees que afecta el tipado de un lenguaje de programación a los siguientes aspectos del mismo:

- Velocidad de ejecución.
- Depuración y control de errores.
- Entornos de desarrollo (autocompletar).

2. Indicar además de en la web, otros lugares y situaciones donde se emplee el modelo cliente/servidor, por ejemplo servidor DHCP.

3. Piensa posibles problemas que pueden suceder para que una arquitectura cliente servidor deje de funcionar (que sucede si alguno de los elementos de la arquitectura falla), e indicar posibles soluciones.

4. Buscar y listar algunas de las arquitecturas de software más utilizadas, ¿alguna de ellas se han nombrado en algún módulo del ciclo formativo?

5. En la petición del protocolo HTTP, se pueden enviar diferentes métodos al servidor, buscar dichos métodos y su significado. ¿Crees que todos están disponibles para realizar una petición al cualquier servidor?

6. Otro de los campos importantes del protocolo HTTP es el campo status, ¿para qué sirve? Listar sus posibles valores (los más importantes).

7. Investiga si existen o han existido además de JavaScript, otras tecnologías en el lado del cliente.

8. Crea la siguiente página web:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>Ejemplo de código JavaScript en el propio
documento</title>
<script type="text/javascript">
  alert("Un mensaje de prueba");
</script>
</head>

<body>
<p>Un párrafo de texto.</p>
</body>
</html>
```

A continuación incluye en la web el siguiente fichero externo.js, que contiene el código.

```
alert("Un mensaje de prueba desde fichero externo");
```