C.F.G.S.: DESARROLLO DE APLICACIONES WEB
Módulo: DESARROLLO WEB EN ENTORNO CLIENTE

01 INTRODUCCIÓN A JAVASCRIPT

Información general	
Extensiones comunes	.js
Paradigma	Multiparadigma, programación funcional,[1] programación basada en prototipos, imperativo, interpretado (*scripting*)
Apareció en	4 de diciembre de 1995
Diseñado por	Netscape Communications, Fundación Mozilla
Última versión estable	ECMAScript 2016; (17 de junio de 2016 (5 años, 2 meses y 21 días))
Sistema de tipos	Débil, dinámico, *duck*
Implementaciones	SpiderMonkey, Rhino, KJS, JavaScriptCore, V8, Chakra.
Dialectos	ECMAScript
Influido por	Java, Perl, Self, Python, C, Scheme
Ha influido a	ObjectiveJ, [[]], JScript .NET, TIScript

[editar datos en Wikidata]

https://es.wikipedia.org/wiki/JavaScript

https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript
http://www.w3schools.com/js/default.asp

JavaScript es un lenguaje de programación interpretado que se utiliza fundamentalmente para dotar de **comportamiento dinámico** a las páginas web. Fue diseñado para que su código se incrustase en documentos HTML y fuera interpretado por el navegador web.

Mediante JavaScript nos encargaremos de realizar acciones en el cliente, como pueden ser:

- ✓ pedir datos
- ✓ confirmaciones
- ✓ validación de datos en formularios
- ✓ incorporar efectos como texto que aparece y desaparece
- ✓ animaciones
- ✓ acciones que se activan al pulsar botones
- ✓ manejar ventanas con mensajes de aviso al usuario….

Cualquier **navegador** web actual incorpora un intérprete para código JavaScript.

Historia

https://www.youtube.com/watch?v=i18gWXhv5aA

El primer lenguaje de scripting para la Web fue **LiveScript**, desarrollado por la compañía **Netscape**, para ofrecer una alternativa "ligera" a Java a la hora de proporcionar comportamiento dinámico tanto en el lado del servidor como en el lado del cliente.

En **1995**, con **Brendan Eich** a la cabeza, Netscape y Sun aprovecharon el lanzamiento de la versión 2 de Navigator, el navegador web de Netscape, para presentar JavaScript.

El objetivo era proporcionar a los diseñadores web (cuyo perfil era por entonces más artístico/plástico que técnico) un lenguaje sencillo que les permitiese mejorar la interactividad en el lado del cliente de las hasta entonces estáticas (como páginas de un libro) páginas web.

> Inicialmente la profesión de diseñador web se entendió como una especialidad de los oficios de diseñador/editor, y la mayoría de los que creaban páginas web eran profesionales de este sector con interés por explorar las posibilidades que les podían brindar las nuevas tecnologías. Actualmente el diseño web no puede entenderse como competencia de un único profesional, sino como la colaboración necesaria de especialistas en diversos sectores (comerciales, diseñadores, publicistas, programadores, ingenieros de sistemas, …), más aún si tenemos en cuenta que el diseño web en sí ha pasado a ser una subdisciplina del diseño de aplicaciones web.

Sus comienzos no fueron fáciles. En su nacimiento fue considerado, con motivo, un lenguaje de baja calidad, lento, y propenso a provocar problemas de seguridad. Además tuvo que competir con **VBScript**, otro lenguaje de scripting para la Web basado en BASIC y afrontar el acoso de variantes desarrolladas por otras empresas como el **JScript** de Microsoft que se integraba en sus navegadores a partir de Internet Explorer 3

A pesar de que las diferencias entre JavaScript y Jscript no eran muy grandes, el hecho de que cada navegador soportase su propio lenguaje creaba confusión entre los desarrolladores e incompatibilidades entre navegadores.

> A menudo, los desarrolladores se veían obligados a programar dos veces la misma funcionalidad (una con cada lenguaje) e introducir en sus páginas web sentencias condicionales para distinguir en qué navegador se estaba ejecutando la página e invocar la ejecución del código escrito con el lenguaje soportado por el navegador en cuestión.

Estos problemas empezaron a resolverse cuando, en 1996, Netscape propuso **JavaScript** a la ECMA International (organización de estandarización responsable, entre otras, de la especificación de C++) para convertirlo en un estándar industrial, y fue capaz de superar este trámite alcanzando acuerdos de compromiso con Microsoft. El nombre del lenguaje estandarizado resultante de este proceso fue **ECMAScript o ECMA-262**, y JavaScript pasó a ser un dialecto suyo, es decir una versión extendida de ese lenguaje, como también los son **JScript y ActionScript** (el lenguaje de programación de las aplicaciones Flash), todos ellos conformes a dicho estándar.

Hoy en día el término JavaScript es el que se utiliza para referirse al propio lenguaje y al estándar.

JavaScript ha sido elegido por la W3C (World Wide Web Consortium), consorcio responsable de la especificación del lenguaje HTML, como estándar para HTML5.

Actualmente la responsable del desarrollo de JavaScript es la Mozilla Foundation

Hitos en el desarrollo de JavaScript

- **Inicios (1996-1997)**:

  Inicialmente JavaScript se utilizaba básicamente para automatizar la creación de código HTML (por ejemplo, generar un calendario mensual mediante bucles en lugar de tener que escribir manualmente el código de todas las celdas de la tabla del calendario), y responder a eventos realizados por el usuario sobre elementos de formularios, enlaces (*links*), e imágenes de un modo rudimentario (por ejemplo, permitiendo realizar validación de formularios o botones *rollover* (botones que cambian su aspecto al colocar el puntero del ratón sobre ellos)).

- **DOM (*Document Object Model*) (1997-2004)**: Efectos de animación

  El desarrollo de la especificación del DOM (por parte del W3C) permitía manipular cualquier elemento de una página web mediante JavaScript, incluso una vez cargada y presentada al usuario. Esto dio lugar a una enorme mejora en las posibilidades de **interacción** con el usuario, y a una plétora de efectos de animación que definitivamente ponían de manifiesto que la web era algo más que una enorme biblioteca estática con enlaces entre informaciones relacionadas.

- **AJAX (*Asynchronous JavaScript and XML)* (1999-…)**.

  Esta tecnología permitía por primera vez intercambiar datos con el servidor sin tener que recargar una página web, y aunque fue aprovechada de un modo marginal inicialmente con aplicaciones muy rudimentarias, rápidamente despertó el interés de los desarrolladores que le encontraron usos cada vez más avanzados, que finalmente alumbrarían aplicaciones tan complejas y populares como Gmail, Google Maps o FaceBook.

- ***Bibliotecas* (2005-…)**.

  Un factor importante en el desarrollo de la web 2.0 fueron las bibliotecas JavaScript, que crean una capa de abstracción entre el programador y el intérprete JavaScript del navegador, permitiéndole despreocuparse de los detalles de más bajo nivel (como la gestión de incompatibilidades entre navegadores, la creación de interfaces de usuarios ricas (autocompletar, arrastrar y soltar, …), y la aplicación de efectos visuales atractivos (menús desplegables, *light box* (superposición de imágenes sobre la página web), …)) y mejorar la productividad. Algunas de las bibliotecas más populares son Prototype, Ext Core, MooTools, Dojo y **jQuery**).

- ***HTML5* (2008-…)**.

  HTML5 ha elegido JavaScript como lenguaje de programación estándar y esto supone que será un pilar esencial de algunas de sus revolucionarias novedades, como:

  - El canvas o lienzo: Capacidad para dibujar y crear animaciones en una página web.
  - Audio y vídeo: Capacidad para controlar distintos medios sin recurrir a plugins externos.

- Arrastrar y soltar: Esta funcionalidad se ha incluido de forma nativa en HTML5, de modo que ya no es necesario recurrir a técnicas complejas.
- Aplicaciones offline o en caché: Aplicaciones que pueden seguir funcionando aunque no dispongamos de conexión con el servidor.
- Almacenamiento web: Almacenamiento de datos en el lado del cliente, similar a las Cookies, pero con más capacidad.
- Geolocalización: La posición geográfica del usuario es un disparador de eventos más, como las operaciones del teclado o del ratón, de modo que podemos generar aplicaciones que "reaccionen" a los cambios de posición, como por ejemplo un entrenador de mountain bike que almacene perfiles de ruta, o aplicaciones que muestren sugerencias de restaurantes en función de la posición del usuario.
- Notificaciones: Permite a las aplicaciones HTML5 enviarnos mensajes de escritorio que se muestran de una forma similar a los avisos de la bandeja de iconos de Windows (la zona de iconos que hay a la izquierda del reloj). Por ejemplo, una aplicación se seguimiento de cotizaciones bursátiles podría avisarnos cuando se alcanzase el punto de *stop-loss* de alguno de los valores de nuestra cartera.

- ***Frameworks***

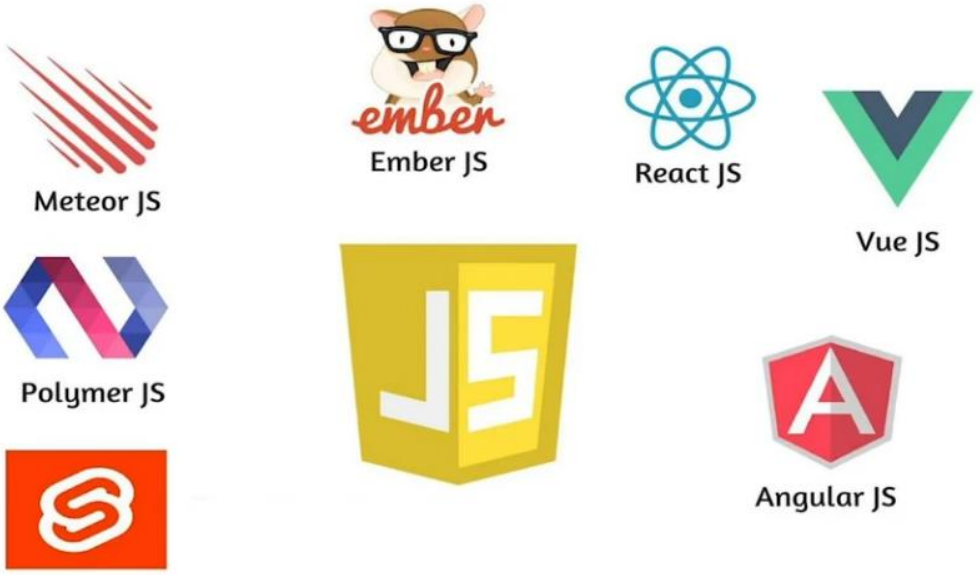
  Como ANGULAR:

  - ✓ Framework para aplicaciones web desarrollado en TypeScript
    - TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft.
    - TypeScript es un superconjunto de JavaScript, que esencialmente añade tipado estático y objetos basados en clases
  - ✓ De código abierto, mantenido por Google.
  - ✓ Se utiliza para crear y mantener aplicaciones web de una sola página (SPA Single Page Application) en las que la interacción entre las secciones es más rápida que si hubiera varias páginas recargándose.

- ✓ Como cualquier framework de desarrollo su objetivo es simplificar el desarrollo de una aplicación web de manera más sencilla y optimizada.

Otros:

https://ifgeekthen.everis.com/es/los-mejores-javascript-frameworks



Características del lenguaje

- Sintaxis semejante a Java:
  - De hecho JavaScript está basado en el concepto de objeto, y utilizan herencia basada en prototipos, diferente de la herencia basada en clases propia de los lenguajes orientados a objetos, como Java. En JavaScript los objetos heredan propiedades directamente de otros objetos sin necesidad de que sean instancias de una misma clase sino mediante la clonación de otros objetos o mediante la escritura de código por parte del programador.

https://es.wikipedia.org/wiki/Programaci%C3%B3n_basada_en_prototipos

- Lenguaje débilmente tipado, lo que permite que una variable pueda contener valores de distintos tipos a lo largo de la ejecución del programa.

- Por defecto todas las variables son globales, es decir, aquellas variables no definidas dentro de una función se ubican en un espacio de nombres común denominado *global object*.

Dónde y cómo incluir JavaScript

```
<html>
<head>
 <meta charset="UTF-8">
<title>
 Bienvenidos a DAW2
 </title>
<script>
 alert ("Hemos vuelto");
 </script>
```

```
<script src="saludo.js">
</script>
```

```
<html>
<head>
 <meta charset="UTF-8">
<script type="text/javascript">
 document.getElementById("demo2").style.display="none";
 </script>
 </head>
```

```
<input type="button" value="CODIGO ROJO"
 onclick="alert('Has tocado el rojo');
          document.getElementById('div1').style.color='red'">
<input type="button" value="CODIGO AZUL"
 onclick="alert('Has tocado el azul');
          document.getElementById('div1').style.color='blue'">
</body>
</html>
```

Existen dos modos de incluir lenguaje JavaScript en una página:

1. A través del elemento **<script>:**

   El código JavaScript se encierra entre etiquetas <script> </script> que marcan el principio y fin del bloque de código JavaScript de modo que el navegador sabe que lo contenido entre estas etiquetas no corresponde a código HTML, si no que se trata de código que debe ser procesado antes de mostrar el resultado en pantalla.

   El formato es el siguiente:

```
<script type="text/javascript">
        código
</script>
```

   Dado que hay distintos lenguajes de script, el atributo type indica al navegador a cual corresponde el texto que encontrará a continuación para que utilice el intérprete adecuado. Puede omitirse pues los navegadores más conocidos utilizan JavaScript como lenguaje de script por defecto, pero no está demás especificarlo y además es obligatorio según normas de la W3C encargada de la especificación del estándar.

   A su vez, el elemento <script> nos permite aplicar dos técnicas diferentes para insertar el código:

   1.1. Escribir el código JavaScript directamente en el documento HTML (esta técnica se denomina **incrustación**)

   Es correcto incluir cualquier bloque de código en cualquier zona de la página, suele definirse el código JavaScript dentro de la cabecera del documento dentro de la etiqueta <head>

```
<html>
<head>
<title> PROBANDO JAVASCRIPT </title>
<script type="text/javascript">
alert ("bienvenido");
</script>
<body>
…
</body>
</html>
```

1.2. Hacer referencia a un archivo independiente que contiene el código JavaScript (esta técnica se denomina **inclusión**).

```
<html>
<head>
<title> PROBANDO JAVASCRIPT </title>
<script  type="text/javascript" src="saludo.js"  > </script>
</head>
<body>
…
</body>
</html>
```

saludo.js contendrá el código:

```
alert ("bienvenido");
```

- ✓ El archivo tendrá extensión js
- ✓ Es la forma más adecuada de trabajar pues al localizar el código JavaScript en un fichero externo facilitamos la reutilización y modularización de dicho código y aprovechar las funcionalidades de caché de los navegadores (los navegadores almacenan una copia local de los archivos de código JavaScript y, si otra página visitada por el usuario requiere ese mismo archivo, utilizan la versión local en lugar de volver a descargarla de nuevo)

¿Código en el <head> o en el <body>?

Colocar scripts en la parte inferior del elemento <body> mejora la velocidad de visualización, porque la interpretación de los scripts ralentiza la visualización.

> *Tradicionalmente se colocaban todos los elementos <script> que hacían referencia a archivos js externos en la sección <head> de la página web para tenerlos agrupados y facilitar su localización. Al encontrarse con cada uno de estos elementos <script> el navegador tenía que cargar el archivo js correspondiente y, si estos archivos eran grandes (del orden de cientos de kilobytes), podían mantener al navegador ocupado en el head durante bastante tiempo, impidiendo que iniciase la interpretación del body y, consecuentemente, sin mostrar nada en pantalla. Actualmente este procedimiento se considera inadecuado porque genera confusión en el usuario, que no recibe información de lo que está pasando y simplemente espera mientras ve una página en blanco. Por este motivo ahora se recomienda colocar los elementos <script> lo más abajo posible dentro del body. En **HTML5** se ha añadido el atributo **async** al elemento script, que sirve para indicar al navegador que no es necesario que detenga la carga de la página hasta que se haya completado la descarga del archivo js, pues el código que contiene no afecta a la representación de la página; sin embargo, la compatibilidad de este atributo aún no está extendida ni unificada entre todos los navegadores.*

Otra razón para añadir al final el código JavaScript es que si se está haciendo en él referencias al DOM, hasta que no está cargada toda la página no podemos hacer estas referencias, por ejemplo, si añadimos el código al principio y referenciamos un id, nos dará error.

En nuestro caso, con funciones y códigos pequeños, los veremos en los ejemplos casi siempre en el <head> para facilitar su visión al primer golpe de vista.

2. Otra manera de incluir código JavaScript en **como respuesta a algún evento**:

```
<html>
<head>
</head>
<body>
<input type="submit"
  onclick="alert('Acabas de hacer click');return false;" value="Click">
</body>
</html>
```

Probar nuestro código

Para probar nuestros programas podemos utilizar los métodos:

window.alert ("mensaje"); o simplemente alert("mensaje");

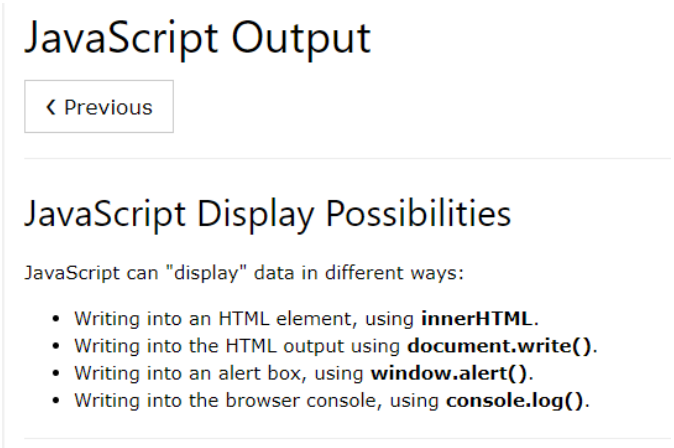
y aparecerá una ventana con el mensaje correspondiente. Pero este método tiene un inconveniente: detiene la ejecución del script hasta que el usuario acepte esta alerta.

console.log ("mensaje");

muestra el mensaje en el panel console del navegador

Ver ejemplos

http://www.w3schools.com/js/js_output.asp



Saber más: https://www.w3schools.com/js/js_intro.asp