Desarrollo Web en Entorno Cliente.

# Unidad 1.

# JavaScript

## Introducción al Lenguaje

# Introducción.

**JavaScript se utiliza para diversas tareas auxiliares, como son**:

– Realizar una validación primaria de la entrada de datos.
– Responder a ciertas acciones del ratón ó del teclado.
– Para visualizar Banners ó animaciones Flash.
– Para añadir los enlaces patrocinados de un sitio Web.

JavaScript facilita responder a las acciones de los usuarios sin necesidad de que el proceso se realice en el Servidor.
JavaScript nos permite lograr que una aplicación Web tenga una velocidad de respuesta que se parezca a una aplicación local.
JavaScript nos va a permitir crear Scripts que se integran en el código HTML de la página.
El usuario de la página Web podrá ver en cualquier momento el código fuente cliente del sitio visitado por lo que habrá que tenerlo en cuenta por motivos de seguridad.
(Por ello, no debemos incluir datos que puedan ser usados por hackers, como son incluir el nombre de la cadena de conexión a una base de datos ó contraseñas).

**¿Qué es JavaScript?**

JavaScript es un Lenguaje Interpretado, por lo que el código se analiza y se ejecuta en el navegador del cliente, en el mismo momento.
(En el servidor se ejecutaría por ejemplo: PHP, ASP.NET...)
Cuando el documento se carga, el interprete se encarga de convertir el código JavaScript, en código binario que dependerá de la máquina en la que se procesa la página.
Esto hace que el tiempo de respuesta sea muy bajo (prácticamente nulo).
El código JavaScript puede aparecer en cualquier parte del documento HTML, y puede haber, cero, uno ó más bloques de código.
El orden de análisis de los bloques de código es el que corresponde a su posición física dentro del documento y el código de funciones o gestores de eventos se analiza (o interpreta) sólo si se realiza una llamada a esas funciones.
Esto quiere decir que una página funcionaría aparentemente bien, y hasta que no se invoque a una función errónea no nos daríamos cuenta.

**Limitaciones de JavaScript:**

– El ámbito del Lenguaje es el navegador cliente.
– No puede escribir información en la base de datos del servidor.
– Puede colaborar en la creación de sitios Web dinámicos, pero no puede hacerlo sin la ayuda de un lenguaje de servidor y de un lenguaje para el acceso a las bases de datos.

**Mediante JavaScript:**

– No se puede acceder a los archivos almacenados en el cliente.
– No se puede averiguar qué software tiene instalado el ordenador (excepto el tipo de navegador y

del sistema operativo).
– No se puede acceder a la lista de enlaces favoritos del cliente.
– No se puede acceder a las contraseñas almacenadas.
– No se puede acceder a la libreta de direcciones del usuario.
No se puede acceder a datos personales, sólo a los introducidos en el formulario de la aplicación vigente.
– No puede modificar el contenido de un sitio externo.
– No puede realizar lecturas de páginas en dominios diferente al de su página.
– No puede autorizar la instalación automática de programas ejecutables.

**Amenazas:**

Actualmente nuestros equipos se ven amenazados por:
– Virus, Worms, Troyanos, Espía, Phishing, etc,
Esto se combate, mediante:
– Antivirus, Antispam, Firewall, etc
– Manteniendo actualizado: Sistema Operativo, Navegador, Cliente de Correo …

El Riesgo que va en alza es el **Phishing**: Robo de Información Personal, que se suele hacer a través de componentes ActiveX que se instalan en el ordenador de modo consentido por el usuario por el deseo de visualizar alguna página y a veces hasta sabiendo el riesgo que se corre.

**Compatibilidad:**

JavaScript es interpretado de modo diferente según sea el navegador utilizado.
Para estar seguros de que nuestro código JavaScript es correcto en todos los entornos debemos realizar pruebas completas en todos los navegadores principales, ó en el entorno en el que se va a instalar en el caso de una intranet.
Hay muchas funciones JavaScript que son exclusivas de un navegador determinado.
Actualmente Internet Explorer 8 y FireFox abarcan el mercado de los navegadores y en cuanto a sistemas el mercado es en el siguiente orden: Windows, Macintosh, Linux.
Por tanto, habría que realizar el menor número de pruebas según entornos.

**¿Cómo codificamos con JavaScript?**

El IDE de Visual Studio, Komodo, TextPad, EditPlus, Bloc de Notas.

**Complementos en los Navegadores:**

**Mozilla Firefox:**

Incluye extensiones, complementos que se añaden según la necesidad del usuario.
En https://addons.mozilla.org/es-ES/firefox/extensions.php encontramos extensiones de todo tipo:
– Para gestionar Cookies.
– Emular navegadores.
– Depurar código JavaScript.

– Ejecutar JavaScript en una consola auxiliar.

Una vez instalado el complemento, podemos verlo en:

– El Navegador Firefox → Herramientas / Complementos.

Una de las Extensiones mas usadas es: "Web Developer" y es una barra especial que se coloca debajo de la barra de direcciones, con diferentes funcionalidades, por ejemplo:

– Desactivar: Permite Activar/Desactivar JavaScript en el navegador. Activar/Desactivar Caché.
– Cookies: Da información sobre los archivos cookies y permite desactivarlas.
– CSS: Gestiona los estilos que se aplican a elementos de la página.
– Formularios: Visualizar elementos de formularios.
– Imágenes: Gestiona las imágenes del documento

***Internet Explorer:***

Incluye herramientas que nos ayudan en el desarrollo JavaScript
En el Menú Herramientas del Navegador / Herramientas de Desarrollo
Nos permite visualizar código: HTML, CSS, JavaScript y permite hacer depuración paso a paso, fijar puntos de interrupción, verificar el valor de las variables, entre otras funciones.

**Transferencia de Servidor Local a Remoto.**

Una vez realizadas las pruebas de la aplicación Web en un entorno local ó de producción (donde se hace la página web), se debe poner a disposición de todos los usuarios de internet.
El paso de transferencia de nuestra aplicación web probada en nuestra red local a un servidor en donde se aloja nuestro dominio depende de la organización de cada uno de los proveedores del alojamiento Web.
El Proveedor nos suministra: Servidor Web, Lenguaje de Programación, Espacio en Disco en el servidor para alojar las páginas, recursos, archivos generales y bases de datos, Sistema de gestión de Bases de Datos, Capacidad de Transferencia de archivos.
Accedemos al alojamiento utilizando FTP mediante usuario y contraseña.

**Importante:**

Cuando el Navegador utiliza caché coloca copias de los documentos descargados durante cierto tiempo; cuando el navegador detecta que una referencia la tiene en caché no la descarga desde el servidor sino que utiliza la copia local. Para actualizar la caché en IE General / Herramientas / Opciones de Internet.
→ Eliminar las páginas temporales de Internet.

# Integrar el Código JavaScript con HTML.

## 1. JavaScript en elementos HTML.

Se usa, para controlar los eventos asociados a un elemento HTML concreto.

```
<!DOCTYPE html>
<html>
        <head>
                <meta http-equiv="Content-Type"  content="text/html;  charset=iso-8859-1" />
                  <title> Ejemplo 1 </title>
        </head>
        <body>
                  <p  onclick="alert('Prueba');"> Ejemplo 1: Código en los Atributos</p>
        </body>
</html>
```

**Desventaja**: El mantenimiento y modificación del código puede resultar complicado.

## 2. JavaScript en el mismo documento HTML.

Definimos el código entre las etiquetas <head> y </head>

```
<!DOCTYPE html>
<html>
<head>
                <meta http-equiv="Content-Type"  content="text/html;  charset=iso-8859-1" />
                  <title> Ejemplo 1 </title>
                <script type="text/javascript">
                                alert ("Prueba de JavaScript");
                </script>
</head>
<body>
                <p  onclick="alert('Prueba');"> Ejemplo 2: Código Embebido</p>
</body>
</html>
```

**Desventaja**: Si ese fragmento de código lo queremos usar en otras páginas, debemos incluirlo en cada una de ellas, lo cual es un inconveniente cuando tenemos que realizar modificaciones de dicho código.

## 3. JavaScript en un Archivo Externo.

Las mismas instrucciones JavaScript incluidas entre <script></script> se pueden almacenar en un fichero externo con extensión .js.

**Archivo:** "mensaje.js"

```
alert("Prueba de JavaScript");
```

**Código HTML y enlazarlo con el Archivo "mensaje.js"**

```
<!DOCTYPE html>
<html>
<head>
        <meta http-equiv="Content-Type"  content="text/html;  charset=iso-8859-1" />
        <title> Ejemplo 3 </title>
        <script  type="text/javascript"  src="/inc/mensaje.js"> </script>
</head>
<body>
        <h1>Ejemplo 3: Fichero Externo <h1>
</body>
</html>
```

**Importante:**

Sólo puede enlazarse un archivo en cada etiqueta. Aunque puedo poner muchas etiquetas <script>. Así, facilitamos la reutilización y modularización del código.

El Atributo "type" nos permite decir al navegador cuál es el usado para codificar el script.

## Sintaxis.

Este lenguaje distingue entre Mayúsculas y Minúsculas.

Se ignoran: Tabulaciones, espacios en blanco, saltos de línea presentes dentro del código.

**Insertar Comentarios en en Código:**

```
<script type="text/javascript">
        // Este modo permite comentar una sola línea
        /* Varias líneas
                comentadas */
</script>
```

Se suele insertar un (;) al final de cada instrucción de JavaScript, podemos omitirlo si cada instrucción se encuentra en una línea diferente. (Es preferible usar el ;).

Nomenclatura de Variables: Prefijo tipo de dato + Nombre (contenido) separa palabras Mayúscula.
txtLíneaDetalle, dtFechaEntrada, intContadorLíneas, bRetorno , strLíneaDetalle, nContadorLínea.

**Nomenclatura de Funciones:** Prefijo Acción Principal + Descripción de la función separadas Mayúscula.

Get: Obtiene valor. GetNro(), obtenerNro()
Set: Asigna valor. SetNro(), asignarNro()
Is: Devuelve verdadero o falso. IsPar(), esPar()
Add: Añade un elemento. AddElemento(), agregarElemento()
Print: Imprime un objeto. PrintFormulario(), imprimirFormulario()
isObjetoEmpleado() , getColor()

**Palabras Reservadas:**

JavaScript tiene una serie de palabras que no podemos usar para definir nombres de variables, funciones ó etiquetas.

Break, case, catch, continue, debugger, default, delete, do, else, finally, for, function, if, in, instanceof, new, return, switch, this, throw, try, typeof, var, void, while, with.

- En http://www.ecmascript.org podemos ver las palabras reservadas del estándar JavaScript.

**Tipos de Datos:**

Especifica el valor que se guardará en una variable.

- Tipos de Datos Primitivos: Números, Cadenas de Texto, Valores Booleanos.
- Tipos de Datos Compuesto: Objeto (colección de valores primitivos)

*Números:* Sólo hay un tipo de datos numérico. (**double** ó punto flotante).

Para usar un valor en base 16 iniciamos con 0x seguido de los dígitos hexadecimales.

*Cadenas de Texto:* **String** (Las cadenas de texto van entre “”, ` `, comillas dobles ó simples)

Secuencias de Escape: Para representar símbolos de la sintaxis (uso \ )

\\, \' , \” , \n , \t , \v , \f , \r , \b

Valores Booleanos: Sólo admite dos valores (**true, false**)

**Variables:**

- Declaración: Para definir la variable usamos la palabra reservada “var”

```
var variable_1;
var variable_2, variable_3;
```

(De momento no tienen asignado ningún valor inicial).

- Inicialización:

```
var variable_1 = 10;
var variable_2 = variable_1 + 10;
var variable_3 = prompt ( ' Introduce un valor: ' );
```

**Operadores:**

- Aritméticos: + , - , * , / , % , ++ , -
- Lógicos: && , || , !
- Asignación: += , -= , *= , /= , %=
- Comparación: < , <= , == , > , >= , != , === , !==
- Condicional: ?:

## Ejemplo 1.

Ejecuta el siguiente código para ver el tipo de dato asociado a las variables:

```
var var1;
document.write (typeof(var1));      //tipo undefined
var1 = 5;
document.write (typeof(var1));      //tipo number
var1 = “Texto”;
document.write (typeof(var1));      //tipo string
var1 = 8.5;
document.write (typeof(var1));      //tipo number
var1 = true;
document.write (typeof(var1));      //tipo boolean
```

## Sentencias Condicionales.

- **IF**

```
if (expresión_1) {
        instrucciones_1;
 } else  {
        instrucciones_2;
 }
```

- **SWITCH**

```
switch (expresión) {
        case (valor1):
                Instrucciones a ejecutar si expresion = valor1;
                document.write("Página principal");
        break;
        case (valor2):
                Instrucciones a ejecutar si expresion = valor2;
        break;
        default:
                Instrucciones a ejecutar si expresión es diferente a los valores de antes;
                document.write("valor no esperado");
}
```

## Control de Bucles.

- **WHILE**

```
while (expresión) {
        instrucciones
}
```

```
do {
        instrucciones
} while  (expresión)
```

- **DO-WHILE**

```
do  {
   ....
} while  (condición)
```

- **FOR**

```
for (valor_inicial ; expresión_condicional ; incremento_ó_decremento_de_la_variable) {
        cuerpo_del_bucle
}
```

## Funciones.

```
Function  nombreFunción  ( [],[]) {
        ....
        [return  valorRetorno];
}
```

## Ejemplo 2.

```
<!DOCTYPE HTML>
<!-- Ejemplo 1-->
<html>
    <head>
        <title>Primer documento (file003.html)</title>
        <script language=javascript>
            function aviso() {
                alert("Hola, soy un mensaje generado por JavaScript");
            }
        </script>
    </head>
    <body>
        En este documento se muestra la interacción entre HTML y JavaScript.<br>
        Las funciones JavaScript se analizan en el momento de la invocación. <br>
        El código que no pertenezca a una función se analiza en la carga <br>
        del documento, siguiendo la propia secuencia de aparición dentro <br>
        del documento. <br>
        <script language=javascript>
            var miFecha = new Date();
            document.write("Hoy es " + miFecha.getDate() + "/" + (miFecha.getMonth()+1) + "/"  + miFecha.getYear());
        </script>
        <input type="button" name="clicAviso" value="Invocar funcióndel aviso"  onclick="aviso()">
    </body>
</html>
```

– Como vemos en el bloque <head> definimos una función que puede ser invocada desde cualquier parte del documento, pero solo se ejecutará cuando la invoquemos y por tanto sólo será interpretada cuando la llamemos.

– Dentro del <body> hemos impreso en el documento la fecha del día, como no está dentro de una función se va a ejecutar en el momento de cargar la página.

– En los elementos HTML se pueden definir gestores de eventos por ejemplo: “onclick” (onmouseover, onmouseout, onload, onchange, onsubmit, onfocus, onblur, onunload)

## Ejemplo 3.

**Verificar si está activado JavaScript, para ello usa el código:**

```
<script language=javascript>
        alert (“¡OK  JavaScript!”);
</script>
<noscript>
        No está activado JavaScript.        // Aquí va el contenido html
</noscript>
```

## Práctica 1.

1. ¿Qué hay que hacer para instalar JavaScript?
2. ¿Se puede acceder a una base de datos desde una función JavaScript?
3. ¿Se puede leer un archivo de texto desde código JavaScript?
4. JavaScript es un lenguaje que se ejecuta compilado. ¿Verdadero o Falso?
5. JavaScript tiene algunas similitudes con Java pero son dos lenguajes diferentes y con objetivos distintos. ¿Verdadero o Falso?.
6. Un mismo código JavaScript genera los mismos resultados en todos los navegadores. ¿Verdadero ó Falso?.
7. Para probar una aplicación Web con JavaScript y que se ejecutará en Internet es suficiente con probarla en la última versión de IE. ¿Verdadero ó Falso?

**Solución:**

1. Nada. JavaScript está integrado en todos los navegadores actuales (IE, Firefox, Opera, Chrome, Netscape, etc.). Lo que sí puede ser necesario es simplemente activar la opción del uso de JavaScript dentro del navegador.
2. No, no se puede ni leer ni escribir en una base de datos directamente desde código JavaScript; estas acciones se deben realizar mediante un lenguaje del servidor, por ejemplo, ASP o PHP.
3. No, está prohibida la lectura y escrituras de archivos de todo tipo desde código JavaScript. Si se quiere realizar este tipo de operaciones en el disco del servidor se debe apelar a uno de los lenguajes del servidor, por ejemplo, ASP o PHP. Si se quiere realizar este tipo de operaciones en el cliente se debería utilizar código Java o funciones ActiveX, pero son acciones que el cliente debe autorizar.
4. Falso.
5. Verdadero.
6. Falso. Existen casos en que la compatibilidad puede ser total, por lo que la respuesta Verdadero en esos casos sería correcta, pero la respuesta correcta es Falso porque las implementaciones de JavaScript son diferentes entre los distintos navegadores, por lo que es probable que tengamos que programar algunas instrucciones específicas para las distintas versiones.
7. Falso.

## Práctica 2.

Evaluar una expresión de tipo cadena usando la función **eval()**

Será un formulario de entrada donde el usuario introduce una formula aritmética y con un clic obtiene el resultado, osea crearemos una calculadora.

**Solución**

```
<!DOCTYPE HTML>
<!-- archivo Parte 3/Practica 2 -->
<html>
	<head>
	<title>Calculadora muy fácil y simple (file004.html)</title>
		<script language=javascript>
		function fnCalcular(str) {
		    var iResultado = eval(str);
		    document.miForm.miResultado.value = iResultado;

		}
		</script>
	</head>
	<body>
		 <form name="miForm">Introducir una fórmula aritmética::<br>
			<input type=text" name="miExpresión" value="" size=80>
			<p> El resultado es: <input type="text" name="miResultado" value=0>
			<input type="button" name="miButton" value="Calcular"
			onclick="fnCalcular(miForm.miExpresión.value)"
		 </form>
	</body>
</html>
```