Introducción al JavaScript

Autor: Universidad de Navarra

[Ver curso online]

Presentación del curso

Javascript, uno de los lenguajes más empleados en Internet, le permitirá construir webs más dinámicas y completas.
Con este curso gratis, MailxMail.com y el Centro de Tecnología Informática de la Universidad de Navarra le brindan la oportunidad de acercarse a esta completa herramienta que ha modificado la red en sólo unos años. Sus sitios de Internet no volverán a ser lo mismo.

Visita más cursos como este en mailxmail:
[http://www.mailxmail.com/cursos-informatica]
[http://www.mailxmail.com/cursos-programacion]

¡Tu opinión cuenta! Lee todas las opiniones de este curso y déjanos la tuya:
[http://www.mailxmail.com/curso-introduccion-javascript/opiniones]

Cursos similares

Cursos	Valoración	Alumnos	Vídeo
Introducción a Oracle En el curso se introducen los conceptos básicos para saber que es un Gestor de Bases de Datos Relacional, y la utilidad de Oracle para este cometido.... [01/02/06]		17.514	
Introducción al lenguaje Pascal Pascal es un lenguaje de alto nivel y de propósito general (es aplicable a un gran número de aplicaciones diversas) desarrollado por el profesor suizo Niklaus Wirth como ... [01/03/06]		19.357	
Guía para instalar FreeBSD Guia para instalar FreeBSD como servidor WWW, usando Apache, Mysql, PHP. Configurando FreeBSD, Compilando el Kernel, Activando el Sonido en FreeBSD, Actualizando los port... [06/04/06]		1.140	
Webs dinámicas con PHP El lenguaje PHP es un lenguaje de programación de estilo clásico, nada que ver con el HTML, XML o WML. Se parece mucho más al Java o Javascript pero, a diferencia de esto... [10/09/04]		12.181	
PHP y MySQL. Aplicaciones Web: base de datos MySQL I (séptima parte) Programación de aplicaciones Web con PHP y MySQL Ahora estudiaremos la Base de Datos MySQL. Te explicaremos la instalación, la configuraci&oacu... [02/12/08]		1.667	

1. Introducción

[http://www.mailxmail.com/curso-introduccion-javascript/introduccion]

El **JavaScript** permite crear aplicaciones específicamente orientadas a su funcionamiento en la red Internet. Usando **JavaScript**, se pueden crear páginas HTML dinámicas que procesen la entrada del usuario y que sean capaces de gestionar datos persistentes usando objetos especiales, archivos y bases de datos relacionales.

Con **JavaScript** se pueden construir aplicaciones que varían desde la gestión de la información corporativa interna y su publicación en Intranets hasta la gestión masiva de transacciones de comercio electrónico.

Aunque se trata de algo que se sale del alcance de este curso, es importante reseñar que JavaScript puede utilizar una tecnología propietaria de Netscape, denominada LiveConnect; con el propósito de que las aplicaciones JavaScript puedan tener acceso a aplicaciones basadas en objetos distribuidos CORBA y Java.

En cualquier caso, es importante señalar que, pese a la similitud de nombres, JavaScript no es Java.

Las aplicaciones cliente y servidor en JavaScript comparten el mismo núcleo de lenguaje. Este núcleo se corresponde con ECMA-262, el lenguaje de scripts estándar de la Oficina de Estándares de la Unión Europea, con algunos añadidos extra. **Aunque Aunque Javascript de cliente y de servidor comparten el mismo conjunto base de funciones y características; en algunos casos se utilizan de distinta forma.** Los Los componentes de JavaScript son los siguientes:

-Núcleo de JavaScript (Core JavaScript).

-JavaScript para Cliente.

-JavaScript para Servidor.

JavaScript para cliente engloba el núcleo del lenguaje y algunos elementos adicionales como, por ejemplo, una serie de objetos predefinidos que sólo son relevantes para la ejecución de JavaScript en el contexto de un cliente Web.

Así mismo, JavaScript para servidor incluye también el núcleo de lenguaje y los objetos predefinidos y funciones necesarias para el correcto funcionamiento en el marco de un servidor.

El código JavaScript para cliente se integra directamente en páginas HTML y es interpretado, en su totalidad, por el cliente Web en tiempo de ejecución.

Puesto que con frecuencia es necesario ofrecer el mayor rendimiento posible, las aplicaciones JavaScript desarrolladas para servidores se pueden compilar antes de

instalarlas en dichos servidores.

2. Javascript para aplicaciones cliente

[http://www.mailxmail.com/...curso-introduccion-javascript/javascript-aplicaciones-cliente]

Los clientes Web que soportan JavaScript, tales como el Netscape Navigator/Communicator (desde la versión 2.0) o el Microsoft Internet Explorer (desde la versión 3.0) pueden interpretar sentencias JavaScript colocadas en un documento HTML.

Cuando el cliente Web solicita una página de este tipo, el servidor envía por la red al cliente el contenido completo del documento; incluyendo todos los códigos HTML y las sentencias JavaScript que pudieran existir en éste.

El cliente lee entonces la página de forma secuencial, desde el principio hasta el final, representando visualmente los códigos HTML y ejecutando las sentencias JavaScript conforme avanza el proceso de lectura e interpretación.

Las sentencias JavaScript colocadas en una página Web pueden dar respuesta a eventos de usuario, tales como la pulsación de un botón del ratón (clic), la entrada de datos en un formulario y la navegación por una página.

Por ejemplo, se puede crear una función JavaScript que permita verificar que la información introducida por el usuario en un campo de entrada de datos de un formulario (número de teléfono, código postal, número de tarjeta de crédito, etc.) tienen el formato correcto.

En este caso, lo importante es que, sin necesidad de realizar ninguna transmisión de datos por la red, se puede validar dicha información, mostrando al usuario un cuadro de diálogo en caso de que ésta sea incorrecta.

3. Introducción de Javascript en documentos Html

[http://www.mailxmail.com/...o-introduccion-javascript/introduccion-javascript-documentos-html]

Se ofrece aquí un primer ejemplo en el que se ilustra la integración directa de código JavaScript en un documento HTML:

<HTML>

<HEAD>

<TITLE>Primer ejemplo de JavaScript</TITLE>

</HEAD>

<BODY>

Esto es texto normal de un documento HTML

<SCRIPT LANGUAGE="JavaScript"> docurnent.write("Texto generado desde JavaScript")

</SCRIPT>

Esto es, de nuevo, HTML

</body>

</HTML>

Como se puede observar, este ejemplo tiene la apariencia de un documento HTML estándar. La única novedad viene dada por la presencia del fragmento correspondiente al código JavaScript:

<SCRIPT LANGUAGE="JavaScript"> document.write('Texto generado desde Javascript")

</SCRIPT>

Para poder ver el resultado de su ejecución, bastará con cargar dicho documento con cualquiera de los clientes Web antes mencionados.

La salida, como se aprecia en las figuras adjuntas (para ambos clientes Web), se compone de tres líneas de texto:

- Esto es texto normal de un documento HTML
- Texto generado desde JavaScript
- De nuevo HTML

En realidad no se trata de un **script** útil, puesto que todo lo que ofrece (mostrar una línea de texto) se podría haber hecho en HTML directamente y, sin duda, con mayor comodidad. Sólo se trata de mostrar el funcionamiento del código **<SCRIPT>.**

En efecto, cualquier elemento que quede delimitado por los códigos **<SCRIPT> y </SCRIPT>** se considera código **JavaScript**.

En este caso particular, se ha utilizado **document.write**, una de las funciones más importantes de JavaScript, que permite escribir algo en el documento actual (en este caso, el documento HTML que contiene el ejemplo).

4. Archivos de código Javascript

[http://www.mailxmail.com/curso-introduccion-javascript/archivos-codigo-javascript]

El atributo **SRC** del código **SCRIPT** del lenguaje HTML permite especificar un archivo que contiene el código JavaScript (en lugar de incrustar el código JavaScript en el documento HTML).

Por ejemplo:

<HEAD>

<SCRIPT SRC="comun.js"> ...

</SCRIPT> </HEAD> <BODY> ...

Este atributo es especialmente útil para compartir funciones entre numerosos documentos HTML.

Las sentencias JavaScript del interior de un código **<SCRIPT SRC= ... >** se ignoran a menos que la inclusión cause un error.

Se incluye una sentencia que muestre un mensaje de error en caso de no poder cargar el archivo de código.

Por ejemplo: **<script src="http://www.mienlace.es/funciones/funcion1.js">**

Los archivos JavaScript externos sólo pueden tener código JavaScript y ninguna sentencia HTML.

Clientes que no soportan JavaScript.- Estos clientes no admiten la etiqueta HTML **<script>.** Consideran la etiqueta **SCRIPT** y todo su contenido como texto normal, por ello, se hace preciso ocultar el código a clientes que no lo soporten. Para evitar esto, se utilizan los comentarios de HTML entre las etiquetas **<SCRIPT> y </SCRIPT>:**

<SCRIPT LANGUAGE="JavaScript">

</SCRIPT>

Recuerda que es la forma de insertar comentarios en HTML.

Otra forma de conocer si un cliente soporta JavaScript es insertar el código **<NOSCRIPT>...</NOSCRIPT>.** De modo, los navegadores que no soporten JavaScript ejecutan las sentencias HTML alternativas incluidas dentro de esta

etiqueta.

5. JavaScript para aplicaciones del servidor

[http://www.mailxmail.com/...curso-introduccion-javascript/javascript-aplicaciones-servidor]

En el servidor JavaScript también está integrado en páginas HTML. Las sentencias de **JS** **JS** del servidor pueden realizar multitud de tareas:

-Conectarse a bases de datos relacionales de varios fabricantes.

-Compartir información entre usuarios de una aplicación.

-Acceder a los ficheros del servidor.

-Comunicarse con otras aplicaciones a través de **LIVECONNECT y JAVA.**

Las aplicaciones JavaScript del servidor se compilan generando archivos binarios.

Existen servicios especiales de JavaScript en el servidor:

-Servicio de Gestión de Sesiones.

-Servicio de Bases de Datos LiveWire.

JavaScript y Java.-

JavaScript	Java
Interpretado por el cliente	Copilado (bytecodes). Se descarga del servidor y se ejecuta en el cliente
Orientado a Objetos	Basado en clases
El código se integra e incrusta en documentos HTML	Se utilizan APPLETS. Se accede a ellos desde documentos
Los tipos de datos de las variables no se declaran	Es necesario definir los tipos de datos a las variables
No se puede escribir automáticamente en el disco duro	No se puede escribir automáticamente en el disco duro

6. Valores

[http://www.mailxmail.com/curso-introduccion-javascript/valores]

JavaScript reconoce los siguientes valores:

-Valores numéricos.

-Valores lógicos (true, false)

-Cadenas de caracteres

-Valor null

-Valor no definido (undefined)

Además de estos valores, existen otros elementos propios de JavaScript como los objetos y las funciones.

JavaScript trata de forma dinámica los datos y se puede realizar la siguiente operación:

var unValor=50

Y después asignar a un **Valor** un valor de tipo cadena de caracteres:

unValor="Ahora está lloviendo"

Variables.- Las variables pueden comenzar por un carácter o un subrayado bajo (_).

Cuando a una variable no se le asigna un valor, tiene valor indefinido (undefined). Si se le pone un valor, pueden ocurrir dos cosas:

-Si fue declarada sin **"var",** se produce un error en tiempo de ejecución.

-Si fue declarada con **"var",** devuelve el valor **NaN (Not a Number).**

Veamos un ejemplo:

```
function f1(){
return y-2;
}

f1() // Esta llamada a f1 provoca un error en tiempo de ejecución
```

```
function f2(){
return var y-2;
}

f2() // devuelve el valor NaN
```

Podemos utilizar el valor "undefined" para ponerle valor a una expresión:

```
var miVar;

if(miVar==undefined){
hazunacosa();}
else{
hazotracosa();}
```

El valor "undefined" se comporta como falso cuando se usa como tipo booleano.

Las variables pueden ser:

Globales: cuando se le asigna un valor fuera de una función. El uso de "var" es opcional.

Locales: Se realiza la operación de asignación dentro de una función. El uso de "var" es obligatorio.

Por último, es bueno saber que se puede acceder a una variable de un documento HTML de un FRAME desde otro:

```
parent.miVar
```

7. Los literales

[http://www.mailxmail.com/curso-introduccion-javascript/literales]

Un literal es un valor fijo que se especifica en el **script**.

Existen varios tipo de literales que explicaremos a continuación:

-Arrays

-Booleanos

-Coma Flotante

-Enteros

-Objetos

-Cadenas

1.Arrays: conjunto de 0 o más expresiones encerradas entre corchetes ([]).

Ejemplo:

coches=["BMW","Mercedes","Audi","Volvo"]

Coches es un **array** de 4 elementos.

Podemos dejar elementos del **array** vacíos:

ciudades=["Madrid",,"Valladolid"]

ciudades=[,,"Pamplona"]

ciudades=["Madrid","Pamplona",,]

2.Los boléanos: tienen 2 valores booleanos: true y false.

3.Coma flotante: un literal de coma flotante puede tener las siguientes partes:

Un entero decimal

Un punto decimal(".")

Una parte fraccionaria

Un exponente ("e" o "E")

Un literal de coma flotante debe tener al menos un dígito y un punto decimal o una **"e' "e"(o "E").**

Ejemplos:

3.1415

-6.23E11

.2e10

2E-10

8. Más clases de literales

[http://www.mailxmail.com/curso-introduccion-javascript/mas-clases-literales]

Continuamos hablando de los tipos de literales.

4. Los enteros: los literales enteros se pueden expresar como:

Decimales: del 0 al 9.

Octales: comienzan por 0. Del 0 al 7.

Hexadecimales: comienzan por "0x" o "0X". Del 0 a 9 y a(o A) a f(o F).

5. Los objetos: conjunto de cero o más parejas de parejas nombre: **valor.**

Ejemplo:

avión ={marca:"Boeing",modelo:"747",npasajeros:"450"}

Para referirnos desde JavaScript a una propiedad del objeto avión:

document.write(avión.modelo)

6.Las cadenas: el literal de cadena es una secuencia de caracteres entre los signos de comillas simples o dobles (¿ o ¿).

En un literal de tipo cadena podemos utilizar los métodos del objeto **String**.

Existe un conjunto de caracteres que comienzan por **\ .** Algunos de ellos son:

\ b: retroceso

\ f: avance de línea

\ n: salto de línea

\ r: return

\ t: tabulación

\ ' , \ ", \ \ : los símbolos **', "** y **** respectivamente.

9. Objetos

[http://www.mailxmail.com/curso-introduccion-javascript/objetos]

Definimos como objeto, una entidad con una serie de propiedades que definen su estado y unos métodos (funciones) que actúan sobre esas propiedades.

La forma de acceder a una propiedad de un objeto es la siguiente:

nombreobjeto.propiedad

También se puede referir a una propiedad de un objeto por su índice en la creación. Los índices comienzan por 0:

casa[0]=casa.localidad

casa[1]=casa.superficie

casa[2]=casa.precio

Vamos a crear un objeto con una serie de propiedades. La forma de crear un objeto es la siguiente:

-Crear una función constructora

```
function casa(localidad,superficie,precio){
this.localidad=localidad
this.superficie=superficie
this.precio=precio
}
```

Instanciar objetos con "new"

```
casa1=new casa("Pamplona",90,15000000)
casa2=new casa("Bilbao",110,23000000)
```

Dos observaciones importantes:

-Gracias a **new** creamos nuevos objetos con las propiedades de los ya creados.

"this" hace referencia al propio objeto.

-A un objeto se le pueden seguir añadiendo propiedades tras ser definido, aunque

es una práctica que no se aconseja, pues todos los objetos ya creados hasta entonces añaden también esa propiedad con valor nulo. Para ello se utiliza la palabra prototype:

casa.prototype.año=null

casa.año="1980"

10. Otra forma de crear objetos en forma literal

[http://www.mailxmail.com/...curso-introduccion-javascript/forma-crear-objetos-forma-literal]

Otra forma de crear objetos de forma literal es la siguiente:

nombreobjeto={propiedad1:valor;propiedad2:valor;.....propiedadN:valor}

Finalmente, sólo añadir que una propiedad puede estar formada por varias subpropiedades de modo que, para referenciarlas deberemos seguir una notación similar a la anterior:

nombreobjeto.nombrepropiedad.nombreSubpropiedad

Ejemplo:

casa1={localidad:"Pamplona",precio:15000000,superficie:{interior:90,terraza:10}}

En este ejemplo, casa1 es un objeto en el que, la propiedad "superficie", está a su vez formada por dos subpropiedades: interior y terraza.

Cómo se crean métodos.- Veamos ahora cómo se crean los métodos:

Un método no es más que una función asociada a un tipo de objeto:

objeto.nombremétodo=nombrefunción

Veamos un ejemplo. Este método sirve para mostrar las propiedades de un objeto casa:

```
function VerCasa(){
var mostrar="La casa está en " + this.localidad + ", tiene " + this.superficie +" m2 y cuesta " +this.precio +" ptas."

return(mostrar)
}
```

Si en la función constructora de casa añadimos:

this.VerCasa=VerCasa

Ya tenemos un método llamado **VerCasa** que nos permite ver las propiedades de cualquier objeto de tipo casa.

Una vez creado un objeto, si lo que queremos es eliminarlo, sólo tenemos que llamar al operador **"delete".**

delete casa

Este operador es interesante, porque, si consigue eliminar el objeto, devuelve **"true"** si se le pone valor, por lo que se puede saber cuando un objeto ha sido borrado satisfactoriamente.

11. Los operadores (I)

[http://www.mailxmail.com/curso-introduccion-javascript/operadores-1]

Existen varios tipos de operadores en JavaScript:

1.Asignación: este tipo de operador se utiliza para asignar valores a las variables.

var resultado=50

Asigna a la variable "resultado" el valor 50.

Existen abreviaturas de algunas operaciones de asignación:

x + = y x = x + y
x - = y x = x - y
x * = y x = x * y
x % = y x = x % y
x / = y x = x / y

2.Comparación: en JavaScript, se pueden comparar variables de distinto tipo, pues es capaz de forzar conversiones:

= = Devuelve **true** si son iguales. Fuerza conversiones de tipo.

!= Devuelve **true** si son distintos. Fuerza conversiones de tipo.

= = = Devuelve **true** si son iguales y del mismo tipo.

! = = Devuelve **true** si son distintos o de distinto tipo.

> Devuelve **true** si la variable de la izquierda es mayor que la variable de la derecha

< Devuelve **true** si la variable de la derecha es mayor que la variable de la izquierda

> = Devuelve **true** si la variable de la izquierda es mayor o igual que la variable de la derecha

< = Devuelve **true** si la variable de la izquierda es menor o igual que la variable de la derecha

12. Los operadores (II)

[http://www.mailxmail.com/curso-introduccion-javascript/operadores-2]

Continuamos hablando de los diferentes tipos de operadores:

3. Aritméticos: los operadores aritméticos, a partir de varios operandos, devuelven un solo valor; resultado de la operación realizada con los anteriores operandos.

En JavaScript existe notación postfija y prefija, por lo que variable++ y ++variable son dos formas distintas de incrementar una variable. En primer lugar, se procesa la variable, y luego se incrementa. Sin embargo, en el segundo caso, primero se incrementa la variable y después se procesa.

>% Binario: devuelve el resto de una división.

+ + Unitario: incrementa el valor de la variable.

- - Unitario: disminuye el valor de una variable.

- Unitario: cambia el signo de una variable.

4.Lógicos: los operadores lógicos devuelven un valor binario.

&&

AND

| |

OR

!

NOT

Es importante saber que si en una evaluación ya se conoce el resultado, no se pone valor a los demás términos:

true || devuelve true.

false && devuelve false.

13. Las sentencias - las condicionales

[http://www.mailxmail.com/curso-introduccion-javascript/sentencias-condicionales]

Las sentencias en JavaScript se dividen en varios tipos:

1.Condicionales: las sentencias condicionales sin **"if"** y **"switch"**. La sintaxis de **"if"** es la siguiente:

```
if(condición){
acciones
}
else{
acciones
}
```

Ejemplo:

```
var i=5

if(i>5){
document.write("i es mayor que 5")
}
else{
document.write("i es menor o igual que 5")
}
```

La sentencia **"switch"** toma una variable, y la evalúa según unos posibles valores:

```
switch(variable){
case valor1:
acciones1;
break;
case valor2:
acciones2;
break;
.....
```

.....

case valorN:

accionesN;

break;

default acciones;

}

Veamos cada una de las partes:

case valor1: en el caso de que la variable tenga el valor **"valor1",** realizará las acciones **"acciones1".**

break: si no se incluye esta sentencia después de cada "case", se realizarían todos los cases del **"switch"** hasta el final. De este modo, sólo se realizarán las acciones referentes al "case" concreto.

Default: si el valor de la variable no concuerda con ningún case, se realizarán las acciones de **default.**

En la próxima lección le explicaremos el segundo tipo de sentencia: **los bucles.**

14. Las sentencias - Los bucles

[http://www.mailxmail.com/curso-introduccion-javascript/sentencias-bucles]

Veamos el segundo tipo de sentencia: los bucles.

Bucles.- Estas sentencias se caracterizan porque el flujo puede pasar varias veces por ellas hasta que se cumple una condición.

"for" representa una o varias sentencias que se repiten un número determinado de veces:

for(inicio;final;incremento){

acciones

}

"do...while" es un bucle que al menos se recorre una vez, antes de analizar la condición al final.

do(condición){

acciones

}while(condición)

"while" es una sentencia de bucle que puede que no se realice ninguna vez, pues la condición se evalúa al principio del bucle.

while(condición){

acciones

}

Con la sentencia **"break",** se puede salir de una sentencia de bucle sin limitaciones.

Con la sentencia **"continue",** se termina el bucle actual y se comienza con el siguiente.

Manipulación de Objetos.- Existen dos sentencias **(for y with)** que permiten acceder a las propiedades de un objeto de forma rápida.

La sentencia **"for"** recorre todas y cada una de las propiedades de un objeto con un índice.

for(variable en objeto){

acciones

}

"with" establece un objeto por defecto al que aplica un conjunto de acciones:

with(objeto){

acciones

}

La forma de introducir comentarios en **JavaScript** es con **/ /**. Si se trata de una línea, los comentarios se introducirán con **/* ...*/,** si queremos tomar un bloque como comentario.

15. Las funciones

[http://www.mailxmail.com/curso-introduccion-javascript/funciones]

Una función es un elemento del programa creado con la finalidad de realizar una determinada acción. Una función puede ser llamada desde otra.

En JavaScript, las funciones se definen en la cabecera del documento HTML. Su sintaxis es:

```
function nombreFunción([parámetros]){
acciones
}
```

Veamos un ejemplo: el siguiente **script** es una función que toma los valores de un formulario, y devuelve en un cuadro de texto del mismo el valor de la primera casilla elevado a la potencia de la segunda:

```
<script>
function potencia(){
var i=0;
var resul=1;
if(document.cálculo.elevado.value==0)
document.cálculo.resultado.value=resul;
else
{
resul=document.cálculo.base.value;
for(i=1;i<document.cálculo.elevado.value;i++)
resul=resul*document.cálculo.base.value;
}
document.cálculo.resultado.value=resul;
}
</script>
```

Importante: las funciones siempre irán situadas en la cabecera del documento HTML.

Se debe recordar cómo se accede a los diferentes elementos de un documento **HTML**

HTML. Primero, **document**, después los nombres de los distintos elementos que existen en ese elemento:

- Cálculo: es el valor que se le ha dado al atributo **"name"** del formulario.

-Resultado, base, elevado: es el valor que se le ha dado al atributo **"name"** de una de las cajas de texto del formulario.

Veamos la parte **HTML** de este documento:

<form name="cálculo">

Número: <input type="text" name="base">

Potencia: <input type="text" name="elevado">

Resultado: <input type="text" name="resultado">

<input type="button" name="poten" Value="Calcular potencia" onClick="potencia()">

</form>

16. Los parámetros de las funciones

[http://www.mailxmail.com/curso-introduccion-javascript/parametros-funciones]

En JavaScript, los parámetros de las funciones se pasan por valor, es decir, si una variable cambia de valor en la función, fuera de ella sigue teniendo el mismo valor que cuando entró en ella.

Si una función devuelve su resultado con **return**, la función debe ser asignada o formar parte de una respuesta. En caso contrario, si una función no devuelve un valor con **"return",** puede ser llamada sin ser asignada.

Los argumentos de las funciones se gestionan con un array propio de cada una de ellas. Al array se accede con "nombreFunción.arguments[i]", donde "i" es un índice que comienza por 0.

Para conocer el número de parámetros, podemos utilizar: **"arguments.length".**

Existe un conjunto de funciones predefinidas, veamos algunas de ellas:

Fecha:

getDate(): día del mes

getDay(): día de la semana

getMonth(): día del mes

getYear(): año (con dos dígitos)

getTime(): milisegundos transcurridos desde 1/1/1970

getHours(): hora entre 0 y 23

getMinutes(): minutos entre 0 y 59

getSeconds(): segundos entre 0 y 59

Existen las mismas funciones pero con "set" en lugar de "get", de modo que nos permiten cambiar el valor de esas variables.

17. Las ventanas

[http://www.mailxmail.com/curso-introduccion-javascript/ventanas]

Tenemos tres formas de mostrar un mensaje al usuario en una ventana de aviso:

Alert: muestra el contenido de lo que le pasamos en una ventana con un botón de aceptar.

alert("Esto es una prueba")

Confirm: muestra un mensaje como **"alert",** pero con dos botones: aceptar y cancelar. Si se evalúa, al pulsar aceptar devuelve **true** y con cancelar, **false**.

Prompt: muestra una ventana donde podemos escribir un valor, de modo que después pueda ser asignado a una variable.

prompt("mensaje",valor por defecto)

Matemáticas.- Se trata de las propiedades de **Math**:

abs(): valor absoluto

max(v1,..,vn): valor máximo

min(v1,..,vn): valor mínimo

round(): redondear

exp(): exponencial

log(): logaritmo

pow(base,exponente): potencia

sqrt(): raíz cuadrada

sin(): seno

cos(): coseno

tan(): tangente

asin(): arcoseno

acos(): arcocoseno

atan(): arcotangente

18. Gestión de eventos

[http://www.mailxmail.com/curso-introduccion-javascript/gestion-eventos]

Evento se considera cualquier acción que el usuario realiza con el sistema: hacer click, posicionarse con el ratón en un lugar determinado, enviar un formulario, posicionarse en un cuadro para texto,....

Para referirnos a un evento en HTML, el nombre del evento irá precedido por "on". Por ejemplo, el gestor de eventos de**"Click"** será **"onClick".**

La forma de llamar a un evento es la siguiente. Imaginamos que tenemos un botón en un formulario, y queremos que al pulsarlo realice una acción determinada:

<form ....>

<input type="button" onClick="función([parámetros])">

</form ....>

Con esta acción, asociamos al evento click sobre el botón las acciones que realice la función.

Veamos los eventos que conoce JavaScript:

DragDrop: arrastrar un objeto a la ventana del navegador.

Error: se produce un error en la carga de un documento.

Focus: el usuario se posiciona en una ventana o cuadro de texto de un formulario.

KeyDown: se pulsa una tecla.

KeyPress: se pulsa o libera una tecla

KeyUp: se libera una tecla

Load: se carga un documento en el navegador

MouseDown: se pulsa un botón del ratón

MouseMove: se mueve el cursor

MouseOver: el puntero del ratón se posiciona sobre un enlace

MouseOut: el puntero del ratón sale de un enlace o imagen mapa

MouseUp: se libera un botón del ratón.

Move: se mueve la ventana. Esta acción también la puede realizar el script.

Reset: se pulsa sobre el botón **reset** del formulario.

Resize: las dimensiones de la ventana cambian.

Select: se selecciona una de las opciones de un cuadro combo del formulario.

Submit: se pulsa el botón **submit** del formulario.

UnLoad: el usuario sale de la página.

Recuerde que para llamar a los eventos, se debe anteponer **"on"** al nombre del evento.

En la próxima lección veremos ejemplos de eventos.

19. Ejemplos de eventos

[http://www.mailxmail.com/curso-introduccion-javascript/ejemplos-eventos]

Veamos un ejemplo del evento **Click:**

```
<script>
function contar(objetoSelect){
var seleccionadas=0
for (var i=0;i < objetoSelect.options.length;i++){
if (objetoSelect.options[i].selected)
seleccionadas++;
}
return seleccionadas;
}
</script>
```

El resto de código HTML es el siguiente:

```
<BODY BGCOLOR="#FFFFFF">

<form name="formulario">

Selecione los temas que son de su interés
y pulse después el botón

<select name="temas" múltiple>
<option selected>Informática
<option>Naturaleza
<option>Música
<option>Deportes
<option>Economía
<option>Coleccionismo
</select>

<input type="button" value="¿Cuántos hay selecionados?"
```

```
onclick="alert('Número de opciones selecionadas: ' +
contar(document.formulario.temas))">

</form>
```

Vamos a estudiar este ejemplo con detenimiento:

Este formulario cuenta el número de opciones de un cuadro combo.

Hemos definido un formulario llamado formulario que posee un cuadro combo (llamado temas) y un

botón.

Si nos fijamos en el botón, veremos que con el evento Click llama a la función definida previamente contar.

Esta función toma como parámetro el cuadro combo (document.formulario.temas) y lo procesa, devolviendo un valor con **return**

Obsérvese también que el resultado no se escribe en un cuadro de texto, sino que usamos **alert** para mostrar un mensaje en otro cuadro, que concatena un literal y el resultado de la función contar.

20. La captura

[http://www.mailxmail.com/curso-introduccion-javascript/captura]

JavaScript permite definir eventos y asignarlos a objetos por encima de los elementos donde nacen dichos eventos.

Para definir estos eventos, los objetos **window**, **document** y **layer** utilizan los siguientes métodos:

-captureEvents: captura eventos del tipo que se especifique.

-releaseEvents: ignora la captura del tipo especificado.

-routeEvent:: envía el evento capturado a un objeto.

Ahora vamos a ver la secuencia de captura, definición y activación de un gestor de eventos:

Debemos especificar el tipo de eventos que queremos capturar:

window.captureEvent(Event.CLICK [| Event.* | Event.*])

De este modo, todos los eventos de tipo CLICK que se produzcan serán capturados. Nótese que se pueden especificar varios eventos, siempre separados por **|** que significa **OR.**

Tras capturar dicho(s) evento(s), deberemos especificar una función que realice las acciones asociadas a dicho evento:

function evento_nombre([parámetros]){

acciones

return true o false

La función devolverá **true** cuando la acción sea posible y **false** en caso contrario.

Ya tenemos el evento capturado y la función asociada, sólo queda asignar al evento la función especificada:

window.onClick=evento_nombre;

21. Las cookies

[http://www.mailxmail.com/curso-introduccion-javascript/cookies]

Una COOKIE es un pequeño elemento de información que se almacena en el cliente como parte del archivo **cookies.txt.**

Una **cookie** puede estar gestionada de tres formas:

1.Con un CGI, programa residente en la parte servidora.

2.Desde JavaScript, con **document.cookie**

3.Con un objeto del servidor llamado **client** dedicado a las labores de mantenimiento.

La forma de acceder a una **cookie** es con la propiedad **cookie** de **document**, de modo que devuelve una cadena que contiene todas las **cookies** existentes en nuestro sistema.

La forma más sencilla de enviar una **cookie** con JavaScript es usando la instrucción:

document.cookie='cookie'

Las funciones.- Para acceder a una **cookie** en concreto, podemos utilizar las siguientes funciones.

Esta función devuelve una cadena con el valor de la **cookie** o **null** si no la encuentra:

```
function getCookie(name){
var cname=name + "=";
var dc=document.cookie;
if(dc.length>0{
begin=dc.indexOf(cname);
if(begin!=-1){
begin+=cname.length;
end=dc.indexOf(";",begin);
if(end==-1)
end=dc.length;
return(dc.substring(bgin,end));
}
```

```
}
}
```

22. Configurar y borrar una cookie

[http://www.mailxmail.com/curso-introduccion-javascript/configurar-borrar-cookie]

Para configurar una cookie podemos utilizar la siguiente función:

```
function setCookie(name,values,expires,path,domain,secure){
document.cookie=name+ "=" +escape(value) +
((expires==null)?"":";expires="+expires.toGMTString())
+((path==null)?"":";path=" + path)
+((domain==null)?"":";domain="+domain)
+((secure==null)?"":";secure");
}
```

Para borrar una cookie usaremos una función como la siguiente:

```
function delCookie(name,path,domain){
if(getCookie(name)){
document.cookie=name+"="
+((path==null)?"":";path="+path)
+((domain==null)?"":";domain="+domain)+";
expires=Thu,01-Jan-70 00:00:01 GMT";
}
}
```

Limitaciones.- Las cookies tienen unas limitaciones:

-Máximo de 300 cookies en el archivo **cookies.txt**. Si se excede ese límite, se eliminan las menos recientes.

-Máximo de 4 Kb por cookie. Si se excede el límite, se trunca la cookie dejando intacto el nombre, siempre que éste no exceda los 4Kb.

-20 cookies por servidor o dominio. Si se excede ese límite, se eliminan las menos recientes.

23. Lista de Scripts

[http://www.mailxmail.com/curso-introduccion-javascript/lista-scripts]

Veamos algunos scripts que puede realizar para avisar al usuario:

-Aviso tras abrir la página:

```
<script language="JavaScript">

// Función que devuelve un aviso.
function loadalert ()
{alert("Hola, este mensaje se carga al abrir la página")
}
// Llamaremos a esta función en el evento OnLoad de la página, es decir, cuando la
página se cargue.
</script>
```

-Confirmar el salto en un enlace:

```
<script>
function rusure(){
question = confirm("Bien hecho")
if (question !="0"){
self.location = "../nuevae.html"
}
}
// Esta función nos muestra un texto en la pantalla, de modo que si aceptamos el texto
(!=0), enlazaremos con otra página
</script>
```

-Enlace a otra ventana si confirma el aviso

```
<script>
function winconfirm(){
```

```
question = confirm("Va a saltar a una ventana nueva")
if (question != "0"){
window.open("../nuevav.html", "NewWin",
"toolbar=yes,location=yes,directories=no,status=no,menubar=yes,scrollbars=yes
resizable=no,copyhistory=yes,width=435,height=260")
}
}
</script>
```

Si desea saber más sobre otros sobres Scripts de información al usuario http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/ealert.html, uso del ratón http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/eraton.html, botones y formularios http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/ebf.html, cuadros combo http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/ecombo.html, sucesos aleatorios http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/erandom.html, relojes y fechas http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/ereloj.html o referentes a usuario http://www.unav.es/cti/manuales/TutorialJavaScript/scripts/euser.html pulse encima de cada una de las opciones.

Guía para instalar FreeBSD

Guia para instalar FreeBSD como servidor WWW, usando Apache, Mysql, PHP. Configurando FreeBSD, Compilando el Kernel, Activando el Sonido en FreeBSD, Actualizando los port...
[06/04/06]

1.140

Procesos en C. Sincronización (primera parte)

Curso de informática sobre sincronización de procesos en C y señales en linux que te ofrece la posibilidad de comprender los mecanismos de comunicación entre procesos inf...
[21/10/08]

1.741

Webs dinámicas con PHP

El lenguaje PHP es un lenguaje de programación de estilo clásico, nada que ver con el HTML, XML o WML. Se parece mucho más al Java o Javascript pero, a diferencia de esto...
[10/09/04]

12.181