[1. Introducción 1](#_Toc59604961)

[2. Comenzemos a programar 1](#_Toc59604962)

[3. Llamadas a funciones por medio de eventos 3](#_Toc59604963)

[4. Variables 5](#_Toc59604964)

[5. Operadores 7](#_Toc59604965)

[6. Introducción de datos 7](#_Toc59604966)

[7. Sentencias de control 8](#_Toc59604967)

[8. Animación. 10](#_Toc59604968)

[9. Objetos Definidos 13](#_Toc59604969)

[10. Funciones 15](#_Toc59604970)

# 1. Introducción

JavaScript es una de las múltiples aplicaciones que han surgido para extender las capacidades del Lenguaje HTML, no se utiliza para crear programas, sino para mejorar las páginas Web.

Para escribir un JavaScript debe recordarse lo siguiente:

1. Los JavaScript se forman dentro de funciones.
2. Las funciones se especifican entre las etiquetas <script> y </script>
3. Las etiquetas <script> deben colocarse entre las etiquetas <head> y </head>
4. Las etiquetas <title> no pueden estar colocadas entre las de <script>
5. La llamada a la función JavaScript se hace a través de un evento de un elemento del documento.

# 2. Comenzemos a programar

alert (“Texto en la ventana”);



1. Llamada de una función desde un elemento del documento.

<%@ LANGUAGE=JavaScript%>

<html>

<head>

**<script>**

**function bienvenida()**

**{ alert (“Bienvenida a todos”);**

**}**

**</script>**

<title>Nuestro primer JavaScript</title>

<head>

<body onLoad=**bienvenida**();>

</body>

</html>

1. Llamada a una función desde otra función

<%@ LANGUAGE=JavaScript%>

<html>

<head>

<script>

function bienvenida()

{ alert (“Bienvenida a todos”);

**rehola( );**

**}**

**function rehola()**

**{ alert(“Hola \n de nuevo”); }**

</script>

<title>Nuestro primer JavaScript</title>

<head>

<body onLoad=**bienvenida**();>

</body>

</html>

Nota:

El grupo “\n” genera un salto de línea.

# 3. Llamadas a funciones por medio de eventos

Un evento es un mecanismo por el cual podemos detectar las acciones que realiza el usuario, en el primer ejemplo de JavaScript se carga la función por medio del evento “Load”, la siguiente tabla muestra los eventos y manipuladores de JavaScript

|  |  |
| --- | --- |
| **Evento** | **Se ejecuta cuando…** |
| onLoad | Termina de cargar una página o freame (entra) |
| onUnload | Descarga una página o frame (salir) |
| onAbort | Aborta la carga de una página |
| onError | Producirse algún error en la carga de la página |
| onMouseOver | Pasar el ratón por encima de un objeto |
| onMouseOut | Dejar de estar el ratón sobre el objeto |
| onMouseMove | Mover el ratón por el documento |
| onKeyUp | Presionar una tecla |
| onClick | Hacer clic con el ratón |
| onResize | Dimensionar la ventana del navegador |
| onMove | Mover la ventana del navegador |
| onChange | Modificar texto en un control de edición, al perder el foco, usualmente con controles tipo texto o botones radio al cambiar su valor. |
| onSelect | Seleccionar texto en un control de edición. |
| onFocus | Situar el foco en un control |
| onBlur | Perder el foco un control |
| onSubmit | Enviar un formulario |
| onReset | Inicializar un formulario |

Veamos el siguiente ejemplo:

<%@ LANGUAGE=JavaScript%>

<html>

<head>

<script>

function hola( ) {alert (“Hola Mundo”) ; }

function pulsar( ) {alert(“Tocó el botón”);}

function foco( ) {alert(“Foco en la primera caja”);}

function tecla( ) {alert(“Pulsó una tecla”);}

function adios( ) {alert (“Adios Mundo” ) ; }

function cambio( ) {alert(“Cambio de tamaño”);}

</script>

<title>Autor: El Mundo </title>

</head>

<body onLoad=hola( ); onUnload=adios( ); onResize=cambio( );>

<input type=”button” value=”Botón” onClick=pulsar( );>

<input type=”text” size=”5” onFocus=foco( );>

<input type=”text” size=”5” onKeyPress=tecla( );>

</body>

</html>

Las instrucciones anteriores harán lo siguiente

* Al momento de entrar a la página aparecerá el mensaje de “Hola Mundo”
* Aparecerán 3 objetos, 1 botón y 2 cajas para escribir texto
* Si presiona el botón aparecerá el mensaje “Tocó el botón”
* Al posicionarse en la primera caja de texto aparecerá el mensaje “Foco en la primera caja”
* Al posicionarse en la segunda caja, cada vez que presione un tecla para escribir un texto aparecerá el mensaje “Pulsó una tecla”
* Si cambia el tamaño de la página aparecerá el mensaje “Cambio de tamaño”
* Y por último, al cerrar la página surgirá el último mensaje “Adios Mundo”

a. Para los botones de tipo radio:

<input type=”radio” name “R1” **onClick=juego(1);** >

Cada opción de botón genera el valor.

b. En un texto de introducción de password:

<input type=”password” name=”secre” size=”5” **onKeyUp=pasa( );** >

El valor de la clave se encuentra en secre.value.

1. Utilizando un grupo de instrucciones para un combo:

<select size=”1” name=”lugares” **onChange=viaje( ) ;** >

<option value=”1”>Quito</option>

<option value=”2”>Moscu></option>

</select>

El valor se guarda en la variable destino.value

1. Con un control de tipo imagen:

<img name=”imagen” src =”dibujo.gif”  **onMouseOver= pasa(1)** ;

**onMouseOut= pasa(2);** >

El valor se encuentra en imagen.src el cual posee la ubicación del dibujo a desplegar, en la instrucción anterior el dibujo cambia dependiendo de si posiciona el mouse y lo retira.

No es necesario que una función sea llamada por un evento, esta puede ejecutarse como un proceso independiente, para ello debe hacerse la llamada a la función del mismo lugar donde se declaran las variables globales.

Más adelante, en el apartado de animación se presenta un ejemplo.

# 4. Variables

Las variables son declaradas en el mismo apartado que las funciones, o sea entre <script> y </script> por su alcance pueden ser:

* 1. Globales: Si se les declara fuera de una función.
  2. Locales: Si son declaradas entre de una función.

Los controles definidos entre <body> y </body> pueden ser utilizados como variables globales por todas las funciones, en general la mayoría solo debe hacerse referencia a su valor por medio de su nombre, la sintaxis es en general:

**Nombre\_control. value**

Si fuera un control de tipo imagen la sintaxis sería:

**Control\_imagen.src**

Por otra parte, los tipos de datos que pueden tener una variable son:

Numérico

Booleano

String o carácter

Veamos algunos ejemplos

var fecha **=new Date( );**

var letras **=new String( );**

var caracter1=”12”;

var numero1=10;

x = caracter1 + numero1; // daría como resultado el string “1210”

x = numero1 + caracter1; // daría como resultado el número 22

Para evitar problemas de conversión se pueden utilizar métodos como los siguientes:

|  |  |
| --- | --- |
| **Tipo de conversión** | **Sintaxis** |
| De texto a número entero | X = parseInt( texto ); |
| De texto a decimal | X = parseFloat ( texto ); |
| De número a texto | Tex1 = X ; |

Las variables pueden ser pasadas entre funciones, veamos un ejemplo

<html>

<head>

<script>

function opt(valor)

{

if (valor = = 1)

alert (“A pantalla 1”);

else

alert (“A pantalla 2“);

}

</script>

<title>Nuevo ejemplo</title>

</head>

<body>

<a ref.=”Uno.htm” onMouseOver=opt(1); >ir a uno</a>

<a ref.=”Dos.htm” onMouseOver=opt(2); >ir a dos</a>

</body>

</html>

Cuando el mouse pase sobre uno de los mensajes aparecerá su respectivo mensaje, si es pantalla 1 o 2

También las variables pueden ser utilizadas dentro de una matriz, la cual es una colección de variables del mismo tipo que tienen un nombre común. A un elemento específico de una matriz se accede mediante su índice. Todas las matr5ices tienen como primer índice el valor “0”. La sintaxis es:

Para declarar:

var arreglo\_nombre = new Array( );

Para referir

arreglo\_nombre[ índice ] = valor;

Nota:

No se especifica el límite para el arreglo, y el índice debe ser un entero, o una variable de tipo numérico entero.

# 5. Operadores

JavaScript define TRES tipos de operadores: aritméticos, relacionales y lógicos.

Aritméticos

|  |  |
| --- | --- |
| **OPERADOR** | **DESCRIPCION** |
| - | Resta |
| + | Suma |
| \* | Multiplica |
| / | Divide |
| % | Resto de una división |
| -- | Decrementa en 1 |
| ++ | Incrementa en 1 |
| VARI+=VALOR | Incrementa el valor de vari |
| VARI-=VALOR | Decrementa el valor de vari |
| VARI\*=VALOR | Multiplica el valor por vari |

Relacionales y lógicos

|  |  |
| --- | --- |
| **OPERADOR** | **DESCRIPCION** |
| < | Menor |
| <= | Menor o igual |
| > | Mayor |
| >= | Mayor o igual |
| = = | Igual |
| ! = | Distinto |
| && | “And” lógico |
| | | | “Or” lógico |
| ! | No (Not) |

# 6. Introducción de datos

Para introducir datos se puede realizar de dos formas: por medio de una caja de texto y por medio de un prompt.

Caja de texto

<input type=”text” name=”cajaText” size=”20”>

En las cajas de texto no sólo se asigna valores al momento de la digitación pueden ser asignados valores de la siguiente forma:

Variable = cajaText.value;

cajaText.value = valor o variable;

cajaText.focus( ); // Se posiciona en la caja

Ventana Prompt

Variable = prompt (“Texto de la ventana”, “valor inicial”);

Una vez se le da aceptar a la caja, el valor es almacenado en la variable.

# 7. Sentencias de control

Es la forma como JavaScript maneja el flujo de ejecución.

IF - ELSE

if (expresión – booleana )

{

sentencia (s) ;

}

[else] // opcional

{

sentencia (s) ;

}

SWITCH

switch (expresión) {

case contante1 :

sentencia (s);

break;

case contante2 :

sentencia (s);

break:

…..

[default:]

sentencia (s) ;

}

WHILE

[inicialización de variables]

while (condición [es] )

{

sentencia (s) ;

[iteración ; ]

(cambio de condiciones) ;

}

DO WHILE

[inicialización de variables]

do {

{ sentencia (s) ;

[iteración ; ]

(cambio de condiciones) ;

} while (condición [es] )

FOR

for ( inicio ; cond\_fin ; iteración )

{

sentencia (s) ;

}

Rupturas:

Toda iteración puede romperse (principalmente en un while) con la instrucción break, pero también puede hacerse por medio de una ventana de opciones, la sintaxis es:

var\_booleana = confirm (“¿Desea Finalizar?”);



# 8. Animación.

La animación en JavaScript puede ser de texto, imágenes o ambas cosas interactuando. En todos los casos para conseguirla se trabaja del mismo modo.

Lo primero se prepara las etiquetas del texto o de la imagen dándoles un nombre (con los mismos requisitos que las variables) por medio del atributo “name” en caso de las imágenes e id en caso de texto. De esta manera podremos hacer referencia al elemento deseado desde la función encargada de ejecutar la animación.

Para utilizar las animaciones es importante tener en cuenta los estilos, a continuación presentamos una tabla de estilos:

|  |  |
| --- | --- |
| **Sintaxis** | **Descripción** |
| style= “Posición: absoluta; top: pos; left: pos | Posibilita el cambio de posición |
| style= “color:nombre de color” | Posibilita el cambio de color |
| style=”visibility: hidden o visible | Posibilita mostrar y ocultar |

Una vez construida y modificada la o las etiquetas, las variaciones que sufran sus estilos o cualquier otro de sus atributos se realizaran en la función que tengamos preparada para ello. Para todos los cambios que realicemos la sintaxis a seguir es:

Nombre\_etiqueta.style.estilo=valor ;

Nombre\_etiqueta.atributo=valor ;

Si nuestra animación avanza según un intervalo de tiempo, debemos utilizar el método **setTimeout( )**  para establecer el intervalo de tiempo y la llamada a la función. Su sintaxis es:

setTimeout (“funcion\_nombre( ) “, intervalo en milisegundos ) ;

Veamos un ejemplo

<html>

<head>

<script>

var horizontal = 12;

setTimeout (“mueve( )”,500);

function mueve( )

{

horizontal+=10;

TEXTO1.style.left = horizontal;

if (horizontal>=200) horizontal=12 ;

setTimeout (“mueve( )”, 500);

}

</script>

</head>

<body>

<p id=”TEXTO1” style=”position=absolute; top:16; left:12”>

MOVERSE

</p>

</body>

</html>

El programa presentará en la parte superior la palabra “MOVERSE” que se desplazará hacia la derecha hasta la mitad de la pantalla y luego volverá a iniciar.

<html>

<head>

<script>

</script>

</head>

<body>

</body>

</html>

Otro ejemplo.

<html>

<head>

<script>

var estado=true;

setTimeout (“ver( )”,500);

function ver( )

{

estado=!estado;

if (estado)

TEXTO1.style.visibility=”visible”;

else

TEXTO1.style.visibility=”hidden”;

setTimeout (“ver( )”, 500);

}

</script>

</head>

<body>

<p id=”TEXTO1” style=”visibility:visible”>TEXTUX</p>

</body>

</html>

En el ejemplo anterior notamos que la función no es llamada por ningún evento, y esta queda activada en un lazo infinito porque se vuelve a llamar, haciendo que la palabra “TEXTUS” aparezca y desaparezca.

Veamos el último ejemplo de animación, donde las imágenes se moverán verticalme.

<html>

<head>

<script>

// Para que cargue la imagen desde el principio

var dibujo=new Image;

dibujo.src=”pic2.gif”;

var vertical=35;

var ida=true;

var estado=true;

setTimeout (“mover( )”,200);

function mover( )

{ estado=!estado;

if (estado)

img.src=”tenista1.gif”;

else

img.src=”tenista2.gif”;

if (vertical<=200 && ida)

vertical+=10;

else

{ vertical- =10;

ida=false;

if (vertical<=35)

ida=true;

}

img.style.top=vertical;

setTimeout (“mover( )”, 200);

}

</script>

</head>

<body>

<img name=”img” src =”pic1.gif” style=”position:absolute; left:12; top:25” >

</body>

</html>

# 9. Objetos Definidos

Cuando se carga un documento en el navegador, se crean automáticamente una colección de Objetos predefinidos, útiles para describir el documento sobre el que se trabaja, la ventana del navegador y todo tipo de elementos de las páginas web. Se agrupan en los objetos windows, document, history, navigator y screen. También hay toda una colección de objetos para utilidades varias.

Windows:

Nos permite definir las características de la ventana del navegador o de las ventanas que construyamos nuevas. A continuación tenemos los métodos mediante los cuales podremos definir sus características.

|  |  |  |
| --- | --- | --- |
| **METODO** | **DESCRIPCION** | **SINTAXIS** |
| Open | Abrir ventanas | var=window.open(“url”,”name”,”atrbs”); |
| Close | Cerrar ventanas | var.close( ); |
| Opener | Indica si se abrió | var\_boolean=var:opener |
| Closed | Indica si se cerró | var\_boolean=var.closed |
| Location | Enlaza con una página | var.Location=”url”; |
| Print | Imprime el documento | var.Print( ); |
| Alert | Abre ventana alert | var.alert(datos); |
| Confirm | Abre ventana confirm | var.confirm(datos); |
| Prompt | Abre ventana prompt | var.prompt(datos, “val inicio”); |
| Status | Texto en barra estado | var.status=”mensaje”; |
| showModalDialog | Crea ventana modal | var=window.showModalDialog(“url”,”atrbs”); |

Atributos de SHOWMODALDIALOG

|  |  |
| --- | --- |
| **ATRIBUTO** | **ELEMENTO** |
| dialogWidth:valor | Define el ancho |
| dialogHeight:valor | Define el alto |
| dialogTop:valor | Define posición superior |
| dialogLeft:valor | Define posición inferior |

Todos los atributos deberán ir dentro de las comillas y separados por un espacio entre si, todos son opcionales.

Los atributos de OPEN

|  |  |
| --- | --- |
| **ATRIBUTO** | **ELEMENTO** |
| toolbar=[yes / no] | Barra de herramientas |
| location=[yes / no] | Barra de direcciones |
| directories=[yes / no] | Histórico |
| channelmode= =[yes / no] | Barra de canales |
| menubar=[yes / no] | Barra de menús |
| status=[yes / no] | Barra de estado |
| scrollbars=[yes / no] | Barra de Scroll |
| resizable=[yes / no] | Dimensionable |
| width=pixels | Ancho de ventana |
| height=pixels | Alto de ventana |
| fullscreen=[yes / no] | Maximizada |
| top=pixels | Posición superior |
| left=pixels | Posición izquierda |

Todos los atributos deberán ir dentro de las comillas y separados por un espacio entre si, todos son opcionales.

Veamos un ejemplo de cómo crear una ventana y luego cerrarla

<html>

<head>

<script>

var v1;

function abre( )

{ v1=window.open("hola2.htm", "v", "status=yes resizable=yes");

v1.status="Ventana creada ";

status="Ventana";

}

function cierra( )

{ v1.close( ); }

</script>

</head>

<body onLoad=abre();>

<input type="button" value="Cerrar" onClick=cierra( );>

</body>

</html>

Document:

Objeto dependiente de window, es quien contiene las propiedades para trabajar con el documento y su contenido, es decir, la página web. Sus métodos pueden ser usados también por window. Los que normalmente se trabaja son:

|  |  |  |
| --- | --- | --- |
| **METODO** | **DESCRIPCION** | **SINTAXIS** |
| write | Escribe en el documento | Document.write(dato); |
| writeln | Escribe y salta de línea | Document.writeln(dato); |
| alinkColor | Color de enlace(sin usar) | Document.alinkColor=”color”; |
| linkColor | Color de enlace (activo) | Document.linkColor=”color”; |
| vlinkColor | Color de enlace (usado) | Document.vlinkColor=”color”; |
| bgColor | Color de fondo | Document.bgColor=”color”; |
| fgColor | Color del texto | Document.fgColor=”color”; |
| referrer | Url del documento anterior | Var=document.referrer; |
| location | Url del documento actual | Var=docuement.location; |
| lastModified | Fecha modificación | Var=document.lastModified; |

Screen:

Objeto por el cual podemos conocer la configuración y tipo de tarjeta gráfica que tiene el usuario.

|  |  |  |
| --- | --- | --- |
| METODO | DESCRIPCION | SINTAXIS |
| height | Altura de pantalla | Var=screen.height |
| width | Ancho de pantalla | Var=screen.width |
| colorDepth | Bits por píxel | Var=screen.colorDepth |

# 10. Funciones

Funciones para fecha y hora

|  |  |
| --- | --- |
| **METODO** | **DESCRIPCION** |
| Objeto.toGMTString( ) ; | Obtiene hora formato GMT |
| Objeto.getDate ; | Obtiene la fecha del sistema |
| Objeto.getMonth( ) ; | Obtiene el mes |
| Objeto.getYear( ) ; | Obtiene el año |
| Objeto.getHours( ); | Obtiene la hora |
| Objeto.getMunutes( ); | Obtiene los minutos |
| Objeto.getSeconds( ); | Obtiene los segundos |

El objeto tiene que ser definido de tipo date es decir

**Var Objeto=new Date( );**

Funciones aritméticas

|  |  |
| --- | --- |
| **METODO** | **DESCRIPCION** |
| Var=Math.sin(valor) ; | Seno de un ángulo expresado en radianes |
| Var=Math.cos(valor) ; | Coseno de un ángulo expresado en radianes |
| Var=Math.tan(valor); | Tangente de un ángulo expresado en radianes |
| asin( ), acos( ), atan( ) | Igual a los anteriores pero devuelven arcos |
| Var=Math.abs(valor) ; | Valor absoluto |
| Var=Math.log(valor) ; | Logaritmo decimal |
| Var=Math.max(val1, val2); | Devuelve el mayor entre val1 y val2 |
| Var=Math.min(val1, val2); | Devuelve el menor entre val1 y val2 |
| Var=Math.pow(base, expon); | Potencia de base |
| Var=Math.sqrt(valor); | Raíz cuadrada del valor |
| Var=Math.round(valor); | Redondea un valor con decimales |
| Var=Math.random( ) \* Inter. | Número aleatorio entre 0 y el intervalo |
| parseInt |  |

Funciones con cadenas de caracteres.

|  |  |
| --- | --- |
| **METODO** | **DESCRIPCION** |
| Objeto / var.length ; | Devuelve la longitud de la cadena del objeto o variable |
| Objeto / var.CharAt(indice) ; | Devuelve la letra que esté en la posición del índice |
| Objeto / var.subString(i1, i2) | Devuelve el texto comprendido entre posición i1 e i2 |
| Objeto / var.indexof (letra) | Devuelve el índice de la letra buscada |
| Objeto / var.replace(letr1, letr2); | Reemplaza letr1 por letr2 |
| Objeto / var.toLowerCase( ); | Transforma a minúsculas el texto del objeto |
| Objeto / var.toUpperCase( ); | Transforma a mayúsculas el texto del objeto |