Temas javascript

1)entrada de datos por teclado

2)Estructuras condicionales simples

3)Estructuras condicionales compuestas

4)Estructuas condicionales anidadas

5)Operador logico and

6) operador logico or

7)switch

8)while

9)for

10)funciones con parametros

10')funciones que retornan valor

11)funciones anonimas

13)Eventos (addeventlistener)

16)Evento click y doble click

17)Eventos mouse down y mouse up

18 )eventos mouse over y mouse out

19) mouse move

20) Evento key down, key up, keypress

21)Evento change

22)Evento focus y blur

23)Tercerparametro addeventlistener

24)evetno submit

Para la entrada de datos por teclado tenemos la función prompt. Cada vez que necesitamos ingresar un dato con ésta función aparece una ventana donde cargamos el valor. Hay otras formas más sofisticadas para la entrada de datos en una página HTML, pero para el aprendizaje de los conceptos básicos de JavaScript nos resultará más práctica esta función.

Para ver su funcionamiento analicemos este ejemplo:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var nombre;

var edad;

nombre=prompt('Ingrese su nombre:','');

edad=prompt('Ingrese su edad:','');

document.write('Hola ');

document.write(nombre);

document.write(' asi que tienes ');

document.write(edad);

document.write(' años');

</script>

</body>

</html>

La sintaxis de la función prompt es:

<variable que recibe el dato>=prompt(<mensaje a mostrar en la ventana>,<valor

inicial a mostrar en la ventana>);

La función prompt tiene dos parámetros: uno es el mensaje y el otro el valor inicial a mostrar.

----------------------------------------------------------------------------------------------------------

2)Estructuras condicionales simples

En una estructura CONDICIONAL SIMPLE por el camino del verdadero hay actividades y por el camino del falso no hay actividades. Por el camino del verdadero pueden existir varias operaciones, entradas y salidas, inclusive ya veremos que puede haber otras estructuras condicionales.

Ejemplo: Realizar la carga de una nota de un alumno. Mostrar un mensaje que aprobó si tiene una nota mayor o igual a 4:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var nombre;

var nota;

nombre=prompt('Ingrese nombre:','');

nota=parseInt(prompt('Ingrese su nota:',''));

if (nota>=4)

{

document.write(nombre+' esta aprobado con un '+nota);

}

</script>

</body>

</html>

Aparece la instrucción if en el lenguaje JavaScript. La condición debe ir entre paréntesis. Si la condición se verifica verdadera se ejecuta todas las instrucciones que se encuentran encerradas entre las llaves de apertura y cerrado seguidas al if.

Para disponer condiciones en un if podemos utilizar alguno de los siguientes operadores relacionales:

> mayor

>= mayor o igual

< menor

<= menor o igual

!= distinto

== igual

Siempre debemos tener en cuenta que en la condición del if deben intervenir una variable un operador relacional y otra variable o valor fijo.

Como queremos que en la variable 'nota' se guarde como entero lo convertimos llamando a parseInt:

nota=parseInt(prompt('Ingrese su nota:',''));

Otra cosa que hemos incorporado es el operador + para cadenas de caracteres:

document.write(nombre+' esta aprobado con un '+nota);

Con esto hacemos más corto la cantidad de líneas de nuestro programa, recordemos que veníamos haciéndolo de la siguiente forma:

document.write(nombre);

document.write(' esta aprobado con un ');

document.write(nota);

--------------------------------------------------------------------------------------------------------------------------------

3)Estructuras condicionales compuestas

Cuando se presenta la elección tenemos la opción de realizar una actividad u otra. Es decir tenemos actividades por el verdadero y por el falso de la condición. Lo más importante que hay que tener en cuenta es que se realizan las actividades de la rama del verdadero o las del falso, NUNCA se realizan las actividades de las dos ramas.

En una estructura condicional compuesta tenemos entradas, salidas, operaciones, tanto por la rama del verdadero como por la rama del falso.

Ejemplo: Realizar un programa que solicite dos números distintos y muestre el mayor de ellos:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var num1,num2;

num1=prompt('Ingrese el primer número:','');

num2=prompt('Ingrese el segundo número:','');

num1=parseInt(num1);

num2=parseInt(num2);

if (num1>num2)

{

document.write('el mayor es '+num1);

}

else

{

document.write('el mayor es '+num2);

}

</script>

</body>

</html>

La función prompt retorna un string por lo que debemos convertirlo a entero cuando queremos saber cual de los dos valores es mayor numéricamente. En el lenguaje JavaScript una variable puede ir cambiando el tipo de dato que almacena a lo largo de la ejecución del programa.

Más adelante veremos qué sucede cuando preguntamos cuál de dos string es mayor.

Estamos en presencia de una ESTRUCTURA CONDICIONAL COMPUESTA ya que tenemos actividades por la rama del verdadero y del falso.  
La estructura condicional compuesta tiene la siguiente codificación:

if (<condición>)

{

<Instruccion(es)>

}

else

{

<Instruccion(es)>

}

Es igual que la estructura condicional simple salvo que aparece la palabra clave “else” y posteriormente un bloque { } con una o varias instrucciones.

Si la condición del if es verdadera se ejecuta el bloque que aparece después de la condición, en caso que la condición resulte falsa se ejecuta la instrucción o bloque de instrucciones que indicamos después del else.

----------------------------------------------------------------------------------------------------------

4)Estructuras condicionales anidadas

Decimos que una estructura condicional es anidada cuando por la rama del verdadero o el falso de una estructura condicional hay otra estructura condicional.

Ejemplo: Confeccionar un programa que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:

Si el promedio es >=7 mostrar "Promocionado".  
Si el promedio es >=4 y <7 mostrar "Regular".  
Si el promedio es <4 mostrar "Reprobado".

Solución:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var nota1,nota2,nota3;

nota1=prompt('Ingrese 1ra. nota:','');

nota2=prompt('Ingrese 2da. nota:','');

nota3=prompt('Ingrese 3ra. nota:','');

//Convertimos los 3 string en enteros

nota1=parseInt(nota1);

nota2=parseInt(nota2);

nota3=parseInt(nota3);

var pro;

pro=(nota1+nota2+nota3)/3;

if (pro>=7)

{

document.write('promocionado');

}

else

{

if (pro>=4)

{

document.write('regular');

}

else

{

document.write('reprobado');

}

}

</script>

</body>

</html>

Analicemos el siguiente programa. Se ingresan tres string por teclado que representan las notas de un alumno, se transformas a variables enteras y se obtiene el promedio sumando los tres valores y dividiendo por 3 dicho resultado.

Primeramente preguntamos si el promedio es superior o igual a 7, en caso afirmativo por la rama del verdadero de la estructura condicional mostramos un mensaje que indique 'Promocionado' (con comillas indicamos un texto que debe imprimirse en pantalla).  
En caso que la condición nos de falso, por la rama del falso aparece otra estructura condicional, porque todavía debemos averiguar si el promedio del alumno es superior o igual a cuatro o inferior a cuatro.

Los comentarios en JavaScript los hacemos disponiendo dos barras previas al comentario (los comentario en tiempo de ejecución no son tenidos en cuenta y tienen por objetivos de documentar el programa para futuras modificaciones):

//Convertimos los 3 string en enteros

Si queremos disponer varias líneas de comentarios tenemos como alternativa:

/\*

linea de comentario 1.

linea de comentario 2.

etc.

\*/

Es decir encerramos el bloque con los caracteres /\* \*/

5)Operadores logicos

El operador &&, traducido se lo lee como "Y". Se emplea cuando en una estructura condicional se disponen dos condiciones.

Cuando vinculamos dos o más condiciones con el operador "&&" las dos condiciones deben ser verdaderas para que el resultado de la condición compuesta de Verdadero y continúe por la rama del verdadero de la estructura condicional.

Recordemos que la condición debe ir entre paréntesis en forma obligatoria.

La utilización de operadores lógicos permiten en muchos casos, plantear algoritmos más cortos y comprensibles.

Veamos un ejemplo: Confeccionar un programa que lea por teclado tres números distintos y nos muestre el mayor de ellos.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var num1,num2,num3;

num1=prompt('Ingrese primer número:','');

num2=prompt('Ingrese segundo número:','');

num3=prompt('Ingrese tercer número:','');

num1=parseInt(num1);

num2=parseInt(num2);

num3=parseInt(num3);

if (num1>num2 && num1>num3)

{

document.write('el mayor es el '+num1);

}

else

{

if (num2>num3)

{

document.write('el mayor es el '+num2);

}

else

{

document.write('el mayor es el '+num3);

}

}

</script>

</body>

</html>

Podemos leerla de la siguiente forma:  
Si el contenido de la variable num1 es mayor al contenido de la variable num2 Y si el contenido de la variable num1 es mayor al contenido de la variable num3 entonces la CONDICION COMPUESTA resulta Verdadera.  
Si una de las condiciones simples da falsa, la CONDICION COMPUESTA da Falsa y continúa por la rama del falso.  
Es decir que se mostrará el contenido de num1 si y sólo si num1>num2 y num1>num3.  
En caso de ser Falsa la condición analizamos el contenido de num2 y num3 para ver cual tiene un valor mayor.

En esta segunda estructura condicional, al haber una condición simple, no se requieren operadores lógicos.

6)Operador logico or

Traducido se lo lee como "O". Si la condición 1 es Verdadera o la condición 2 es Verdadera, luego ejecutar la rama del Verdadero.

Cuando vinculamos dos o más condiciones con el operador "O", con que una de las dos condiciones sea Verdadera alcanza para que el resultado de la condición compuesta sea Verdadero.

Ejemplo: Se carga una fecha (día, mes y año) por teclado. Mostrar un mensaje si corresponde al primer trimestre del año (enero, febrero o marzo).

Cargar por teclado el valor numérico del día, mes y año por separado.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var dia,mes,año;

dia=prompt('Ingrese día:','');

mes=prompt('Ingrese mes:','');

año=prompt('Ingrese año:','');

dia=parseInt(dia);

mes=parseInt(mes);

año=parseInt(año);

if (mes==1 || mes==2 || mes==3)

{

document.write('corresponde al primer trimestre del año.');

}

</script>

</body>

</html>

La carga de una fecha se hace por partes, ingresamos las variables dia, mes y año.

Si alguna de las condiciones simples del if da verdadero luego se muestra el mensaje:

if (mes==1 || mes==2 || mes==3)

{

document.write('corresponde al primer trimestre del año.');

}

--------------------------------------------------------------------------------------

7)switch

La instrucción switch es una alternativa para remplazar en algunas situaciones los if/else if.  
De todos modos se puede aplicar en ciertas situaciones donde la condición se verifica si es igual a cierto valor. No podemos preguntar por mayor o menor.

Con un ejemplo sencillo veremos cual es su sintaxis. Confeccionar un programa que solicite que ingrese un valor entre 1 y 5. Luego mostrar en castellano el valor ingresado. Mostrar un mensaje de error en caso de haber ingresado un valor que no se encuentre en dicho rango.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var valor;

valor=prompt('Ingrese un valor comprendido entre 1 y 5:','');

//Convertimos a entero

valor=parseInt(valor);

switch (valor) {

case 1: document.write('uno');

break;

case 2: document.write('dos');

break;

case 3: document.write('tres');

break;

case 4: document.write('cuatro');

break;

case 5: document.write('cinco');

break;

default:document.write('debe ingresar un valor comprendido entre 1 y 5.');

}

</script>

</body>

</html>

Debemos tener en cuenta que la variable que analizamos debe ir después de la instrucción switch entre paréntesis. Cada valor que se analiza debe ir luego de la palabra clave 'case' y seguido a los dos puntos, las instrucciones a ejecutar, en caso de verificar dicho valor la variable que analiza el switch.

Es importante disponer la palabra clave 'break' al finalizar cada caso. La instrucciones que hay después de la palabra clave 'default' se ejecutan en caso que la variable no se verifique en algún case. De todos modos el default es opcional en esta instrucción.

Plantearemos un segundo problema para ver que podemos utilizar variables de tipo cadena con la instrucción switch.  
Ingresar por teclado el nombre de un color (rojo, verde o azul), luego mostraremos un mensaje indicando el color ingresado:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var col;

col=prompt('Ingrese alguno de estos tres colores (rojo, verde, azul)' ,'');

switch (col) {

case 'rojo': document.write('se ingresó rojo');

break;

case 'verde': document.write('se ingresó verde');

break;

case 'azul': document.write('se ingresó azul');

break;

}

</script>

</body>

</html>

Cuando verificamos cadenas debemos encerrarlas entre comillas el valor a analizar:

case 'rojo': document.write('se ingresó rojo');

break;

---------------------------------------------------------------------------------------------------------------------------

8)while

Hasta ahora hemos empleado estructuras SECUENCIALES y CONDICIONALES. Existe otro tipo de estructuras tan importantes como las anteriores que son las estructuras REPETITIVAS.

Una estructura repetitiva permite ejecutar una instrucción o un conjunto de instrucciones varias veces.

Una ejecución repetitiva de sentencias se caracteriza por:

- La o las sentencias que se repiten.  
- El test o prueba de condición antes de cada repetición, que motivará que se repitan o no las sentencias.

Funcionamiento del while: En primer lugar se verifica la condición, si la misma resulta verdadera se ejecutan las operaciones que indicamos entre las llaves que le siguen al while.  
En caso que la condición sea Falsa continúa con la instrucción siguiente al bloque de llaves.

El bloque se repite MIENTRAS la condición sea Verdadera.

**Importante**: Si la condición siempre retorna verdadero estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, nunca finalizará el programa.

Ejemplo: Realizar un programa que imprima en pantalla los números del 1 al 100.

Sin conocer las estructuras repetitivas podemos resolver el problema empleando una estructura secuencial. Inicializamos una variable con el valor 1, luego imprimimos la variable, incrementamos nuevamente la variable y así sucesivamente. Pero esta solución es muy larga.

La mejor forma de resolver este problema es emplear una estructura repetitiva:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var x;

x=1;

while (x<=100)

{

document.write(x);

document.write('<br>');

x=x+1;

}

</script>

</body>

</html>

Para que se impriman los números, uno en cada línea, agregamos la etiqueta HTML de <br>.

Es muy importante analizar este programa:  
La primera operación inicializa la variable x en 1, seguidamente comienza la estructura repetitiva while y disponemos la siguiente condición ( x <= 100), se lee MIENTRAS la variable x sea menor o igual a 100.  
Es importante decir que NO debe haber un punto y coma al final del while, si hacemos esto estamos en presencia de un error lógico.  
Al ejecutarse la condición, retorna VERDADERO, porque el contenido de x (1) es menor o igual a 100.  
Al ser la condición verdadera se ejecuta el bloque de instrucciones que contiene la estructura while. El bloque de instrucciones contiene dos salidas al documento y una operación.

Se imprime el contenido de x y seguidamente se incrementa la variable x en uno.

La operación x = x + 1 se lee como "en la variable x se guarda el contenido de x más 1". Es decir, si x contiene 1 luego de ejecutarse esta operación se almacenará en x un 2.

Al finalizar el bloque de instrucciones que contiene la estructura repetitiva, se verifica nuevamente la condición de la estructura repetitiva y se repite el proceso explicado anteriormente.

Mientras la condición retorne verdadero, se ejecuta el bloque de instrucciones; al retornar falso la verificación de la condición, se sale de la estructura repetitiva y continúa el algoritmo, en este caso, finaliza el programa.

Lo más difícil es la definición de la condición de la estructura while y qué bloque de instrucciones se va a repetir. Observar que si, por ejemplo, disponemos la condición x >=100 ( si x es mayor o igual a 100) no provoca ningún error sintáctico pero estamos en presencia de un error lógico porque al evaluarse por primera vez la condición retorna falso y no se ejecuta el bloque de instrucciones que queríamos repetir 100 veces.

No existe una RECETA para definir una condición de una estructura repetitiva, sino que se logra con una práctica continua, solucionando problemas.

Una vez planteado el programa debemos verificar si el mismo es una solución válida al problema (en este caso se deben imprimir los números del 1 al 100 en la página), para ello podemos hacer un seguimiento del flujo y los valores que toman las variables a lo largo de la ejecución:

x

1

2

3

4

.

.

100

101 Cuando x vale 101 la condición de la estructura

repetitiva retorna falso, en este caso finaliza la estructura repetitiva.

La variable x recibe el nombre de CONTADOR. Un contador es un tipo especial de variable que se incrementa o decrementa con valores constantes durante la ejecución del programa.

El contador x nos indica en cada momento la cantidad de valores impresos en la página.

Importante: Podemos observar que el bloque repetitivo puede no ejecutarse si la condición retorna falso la primera vez.

La variable x debe estar inicializada con algún valor antes que se ejecute la operación x = x + 1.

Probemos algunas modificaciones de este programa y veamos qué cambios se deberían hacer para:

1 - Imprimir los números del 1 al 500.  
2 - Imprimir los números del 50 al 100.  
3 - Imprimir los números del -50 al 0.  
4 - Imprimir los números del 2 al 100 pero de 2 en 2 (2,4,6,8 ....100).

---------------------------------------------------------------------------------------------------------------------

9)for

Cualquier problema que requiera una estructura repetitiva se puede resolver empleando la estructura while. Pero hay otra estructura repetitiva cuyo planteo es más sencillo en ciertas situaciones.

Esta estructura se emplea en aquellas situaciones en las cuales CONOCEMOS la cantidad de veces que queremos que se ejecute el bloque de instrucciones. Ejemplo: cargar 10 números, ingresar 5 notas de alumnos, etc. Conocemos de antemano la cantidad de veces que queremos que el bloque se repita.

Por último, hay que decir que la ejecución de la sentencia break dentro de cualquier parte del bucle provoca la salida inmediata del mismo.

Sintaxis:

for (<Inicialización> ; <Condición> ; <Incremento o Decremento>)

{

<Instrucciones>

}

Esta estructura repetitiva tiene tres argumentos: variable de inicialización, condición y variable de incremento o decremento.

Funcionamiento:

1 - Primero se ejecuta por única vez el primer argumento <Inicialización>.

Por lo general se inicializa una variable.

2 - El segundo paso es evaluar la (Condición), en caso de ser verdadera se ejecuta

el bloque, en caso contrario continúa el programa.

3 - El tercer paso es la ejecución de las instrucciones.

4 - El cuarto paso es ejecutar el tercer argumento (Incremento o Decremento).

5 - Luego se repiten sucesivamente del Segundo al Cuarto Paso.

Este tipo de estructura repetitiva se utiliza generalmente cuando sabemos la cantidad de veces que deseamos que se repita el bloque.

Ejemplo: Mostrar por pantalla los números del 1 al 10.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

var f;

for(f=1;f<=10;f++)

{

document.write(f+" ");

}

</script>

</body>

</html>

Inicialmente f se la inicializa con 1. Como la condición se verifica como verdadera se ejecuta el bloque del for (en este caso mostramos el contenido de la variable f y un espacio en blanco). Luego de ejecutar el bloque pasa al tercer argumento del for (en este caso con el operador ++ se incrementa en uno el contenido de la variable f, existe otro operador -- que decrementa en uno una variable), hubiera sido lo mismo poner f=f+1 pero éste otro operador matemático nos simplifica las cosas.

Importante: Tener en cuenta que no lleva punto y coma al final de los tres argumentos del for.

El disponer un punto y coma provoca un error lógico y no sintáctico, por lo que el navegador no avisará.

10)funciones con parametros

Explicaremos con un ejemplo, una función que tiene datos de entrada.

Ejemplo: Confeccionar una función que reciba dos números y muestre en la página los valores comprendidos entre ellos de uno en uno. Cargar por teclado esos dos valores.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<script>

function mostrarComprendidos(x1,x2)

{

var inicio;

for(inicio=x1;inicio<=x2;inicio++)

{

document.write(inicio+' ');

}

}

var valor1,valor2;

valor1=prompt('Ingrese valor inferior:','');

valor1=parseInt(valor1);

valor2=prompt('Ingrese valor superior:','');

valor2=parseInt(valor2);

mostrarComprendidos(valor1,valor2);

</script>

</body>

</html>

El programa de JavaScript empieza a ejecutarse donde definimos las variables valor1 y valor2 y no donde se define la función.  
Luego de cargar los dos valores por teclado se llama a la función mostrarComprendidos y le enviamos las variables valor1 y valor2. Los parámetros x1 y x2 reciben los contenidos de las variables valor1 y valor2.

Es importante notar que a la función la podemos llamar la cantidad de veces que la necesitemos.  
Los nombres de los parámetros, en este caso se llaman x1 y x2, no necesariamente se deben llamar igual que las variables que le pasamos cuando la llamamos a la función, en este caso le pasamos los valores valor1 y valor2.

---------------------------------

funciones anonimas

var cuadrado = function(number) {

return number \* number;

}

//Llamada a la función

var resultado = cuadrado(4);

----------------------------------------------------------------------------

13)Eventos

dado esta etiqueta

<button type="submit" id="enviar">Enviar formulario</button>

queremos asociar un evetno al elemento button

document.getElementById('enviar').addEventListener('click',validar,false);

parametros

primer parametro es el nombre del evento entre appostrofo

segundo paramero es la funcion a ejecutar

tercer parametro es un booleaneo y define el flujo de eventos

14)Flujo de los Eventos

Supongamos que tenemos este marcado:

<body>

<div>

<button>HAZME CLIC</button>

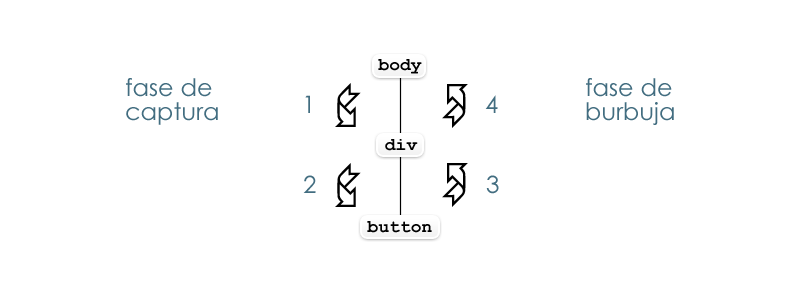
</div>

</body>

Explicar DOM

Cuando hacemos clic en el botón no sólo lo estamos haciendo sobre él, sino sobre los elementos que lo contienen en el árbol del DOM, es decir, hemos hecho clic además sobre el elemento body y sobre el elemento div[1](http://www.codexexempla.org/curso/curso_4_3_e.php#nota01). Sí sólo hay una función asignada a una escucha para el botón no hay mayor problema, pero si además hay una para el body y otra para el div, ¿cuál es el orden en que se deben lanzar las tres funciones?

Para contestar a esa pregunta existe un modelo de comportamiento, [el flujo de eventos (inglés)](http://www.w3.org/TR/2000/REC-DOM-Level-2-Events-20001113/events.html#Events-flow). Según éste, cuando se hace clic sobre un elemento, el evento se propaga en dos fases, una que es la captura —el evento comienza en el nivel superior del documento y recorre los elementos de padres a hijos— y la otra la burbuja —el orden inverso, ascendiendo de hijos a padres—. En un diagrama sería algo así:



Así, el orden por defecto de lanzamiento de las supuestas funciones sería body-div-button.

Una vez visto esto, podemos comprender el tercer parámetro de addEventListener, que lo que hace es permitirnos escoger el orden de propagación:

* true: El flujo de eventos es como el representado, y la fase de captura ocurre al lanzarse el evento. El orden de propagación para el ejemplo sería, por tanto, el indicado, body-div-button
* false: Permite saltar la fase de captura, y la propagación seguiría sólo la burbuja. Así, el orden sería button-div-body.

ejemplo con true

http://www.codexexempla.org/curso/js/prueba\_burbuja\_captura\_01.php

function alerta(){

alert(this);

}

window.onload = function (){

var elBody = document.getElementsByTagName('body')[0];

var elDiv = document.getElementById('elDiv');

var elButton = document.getElementById('elButton');

elBody.addEventListener('click',alerta,<strong>true</strong>);

elDiv.addEventListener('click',alerta,<strong>true</strong>);

elButton.addEventListener('click',alerta,<strong>true</strong>);

}

function alerta(){

alert(this);

}

window.onload = function (){

var elBody = document.getElementsByTagName('body')[0];

var elDiv = document.getElementById('elDiv');

var elButton = document.getElementById('elButton');

elBody.addEventListener('click',alerta,<strong>true</strong>);

elDiv.addEventListener('click',alerta,<strong>true</strong>);

elButton.addEventListener('click',alerta,<strong>true</strong>);

}

15)eventos segunda parte

### Eventos definidos accediendo a propiedades del objeto.

**Problema**

Implementar un formulario que solicite la carga del nombre de usuario y su clave. Mostrar un mensaje si no se ingresan datos en alguno de los dos controles.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<form method="post" action="procesar.php" id="formulario1">

Ingrese nombre:

<input type="text" id="usuario" name="usuario" size="20">

<br>

Ingrese clave:

<input type="password" id="clave" name="clave" size="20">

<br>

<input type="submit" id="confirmar" name="confirmar" value="Confirmar">

</form>

<script>

window.onload=inicio;

function inicio()

{

document.getElementById("formulario1").onsubmit=validar;

}

function validar()

{

var usu=document.getElementById("usuario").value;

var cla=document.getElementById("clave").value;

if (usu.length==0 || cla.length==0)

{

alert('El nombre de usuario o clave está vacío');

return false;

}

else

return true;

}

</script>

</body>

</html>

Con esta segunda metodología vemos que el código HTML queda limpio de llamadas a funciones JavaScript y todo el código queda dentro del bloque del script (pudiendo luego llevar todo este bloque a un archivo externo \*.js)

Lo primero que vemos es inicializar la propiedad onload del objeto window con el nombre de la función que se ejecutará cuando finalice la carga completa de la página, es importante notar que a la propiedad onload le asignamos el nombre de la función y NO debemos disponer los paréntesis abiertos y cerrados (ya que no se está llamando a la función sino le estamos pasando la dirección o referencia de la misma)

window.onload=inicio;

La función inicio es llamada por el objeto window cuando se termina de cargar la página. En esta función obtenemos la referencia del objeto formulario1 mediante el método getElementById e inicializamos la propiedad onsubmit con el nombre de la función que será llamada cuando se presione el botón submit del formulario:

function inicio()

{

document.getElementById("formulario1").onsubmit=validar;

}

Por último tenemos la función validar que verifica si los dos controles del formulario están cargados:

function validar()

{

var usu=document.getElementById("usuario").value;

var cla=document.getElementById("clave").value;

if (usu.length==0 || cla.length==0)

{

alert('El nombre de usuario o clave está vacío');

return false;

}

else

return true;

}

La misma metodología pero utilizando funciones anónimas para cada evento el código ahora queda condensado:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<form method="post" action="procesar.php" id="formulario1">

Ingrese nombre:

<input type="text" id="usuario" name="usuario" size="20">

<br>

Ingrese clave:

<input type="password" id="clave" name="clave" size="20">

<br>

<input type="submit" id="confirmar" name="confirmar" value="Confirmar">

</form>

<script>

window.onload=function() {

document.getElementById("formulario1").onsubmit=function () {

var usu=document.getElementById("usuario").value;

var cla=document.getElementById("clave").value;

if (usu.length==0 || cla.length==0)

{

alert('El nombre de usuario o clave está vacío');

return false;

}

else

return true;

}

}

</script>

</body>

</html>

Analicemos un poco el código implementado, a la propiedad onload del objeto window le asignamos una función anónima:

window.onload=function() {

...

}

En la implementación de la función anónima inicializamos la propiedad onsubmit del objeto formulario1 con otra función anónima:

document.getElementById("formulario1").onsubmit=function () {

...

}

Esta sintaxis de funciones anónimas es ampliamente utilizado.

### Modelo de eventos definidos por W3C (World Wide Web Consortium)

Este modelo de eventos se basa en la implementación de un método para todos los objetos que nos permite registrar eventos. La sintaxis del método es:

addEventListener(evento, método a ejecutar, false);

Veamos como implementamos el problema anterior utilizando este nuevo modelo de eventos:

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<form method="post" action="procesar.php" id="formulario1">

Ingrese nombre:

<input type="text" id="usuario" name="usuario" size="20">

<br>

Ingrese clave:

<input type="password" id="clave" name="clave" size="20">

<br>

<input type="submit" id="confirmar" name="confirmar" value="Confirmar">

</form>

<script>

window.addEventListener('load',inicio,false);

function inicio()

{

document.getElementById("formulario1").addEventListener('submit',validar,false);

}

function validar(evt)

{

var usu=document.getElementById("usuario").value;

var cla=document.getElementById("clave").value;

if (usu.length==0 || cla.length==0)

{

alert('El nombre de usuario o clave está vacío');

evt.preventDefault();

}

}

</script>

</body>

</html>

Lo primero que vemos es que en vez de inicializar la propiedad onload procedemos a llamar al método addEventListener:

window.addEventListener('load',inicio,false);

El primer parámetro es un string con el nombre del evento a inicializar, el segundo parámetro es el nombre de la función a ejecutar y el tercer parámetro normalmente es el valor false.

Cuando se carga completamente la página el objeto window tiene la referencia al método que se debe llamar, en nuestro caso se llama inicio. La función inicio obtiene la referencia del objeto formulario1 y procede a registrar el evento submit indicando en el segundo parámetro el nombre de la función que debe ejecutarse:

function inicio()

{

document.getElementById("formulario1").addEventListener('submit',validar,false);

}

El código de la función validar se modifica, llega como parámetro una referencia al evento y mediante este llamamos al método preventDefault si queremos que no se envíen los datos al servidor:

function validar()

{

var usu=document.getElementById("usuario").value;

var cla=document.getElementById("clave").value;

if (usu.length==0 || cla.length==0)

{

alert('El nombre de usuario o clave está vacío');

}

}

16)Evento click y doble click

Dos eventos que podemos capturar de cualquier elemento HTML son cuando un usuario hace un clic o un doble clic sobre el mismo.  
El evento click se produce cuando el usuario hace un solo clic sobre el elemento HTML y suelta inmediatamente el botón del mouse en el mismo lugar y el dblclick cuando presiona en forma sucesiva en la misma ubicación.

Para probar como registramos estos eventos implementaremos una página que muestre dos div y definiremos el evento click para el primero y el evento dblclick para el segundo.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<div style="width:200px;height:200px;background:#ffff00" id="recuadro1">

Prueba del evento click

</div>

<div style="width:200px;height:200px;background:#ff5500" id="recuadro2">

Prueba del evento dblclick

</div>

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('recuadro1').addEventListener('click',presion1,false);

document.getElementById('recuadro2').addEventListener('dblclick',presion2,false);

}

function presion1()

{

alert('se hizo click');

}

function presion2()

{

alert('se hizo doble click');

}

</script>

</body>

</html>

Primero registramos mediante la llamada al método addEventListener el evento load de la página, y dentro de la función inicio registramos los eventos click y dblclick para los dos div que definimos en la página HTML:

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('recuadro1').addEventListener('click',presion1,false);

document.getElementById('recuadro2').addEventListener('dblclick',presion2,false);

}

Como vemos lo único que hacemos en los métodos que se disparan es mostrar un mensaje:

function presion1()

{

alert('se hizo click');

}

function presion2()

{

alert('se hizo doble click');

}

17)Eventos mouse down y mouse up

Otros dos eventos relacionados con el mouse son mousedown y mouseup.  
El evento mousedown se dispara inmediatamente luego que presionamos con la flecha del mouse un elemento HTML que tiene registrado dicho evento.  
El evento mouseup se ejecuta luego de soltar el botón del mouse estando dentro del control HTML.

Para probar estos eventos implementaremos una página que contenga dos div, a un div le asignaremos el evento mousedown y al otro el evento mouseup. Cuando ocurra el evento procederemos a cambiar el texto contenido dentro del div.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<p>Presione el recuadro amarillo sin soltar el botón del mouse.</p>

<div style="width:200px;height:200px;background:#ffff00" id="recuadro1">

</div>

<p>Presione el recuadro naranja y suelte el botón del mouse.</p>

<div style="width:200px;height:200px;background:#ff5500" id="recuadro2">

</div>

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('recuadro1').addEventListener('mousedown',presion1,false);

document.getElementById('recuadro2').addEventListener('mouseup',presion2,false);

}

function presion1()

{

document.getElementById('recuadro1').innerHTML='Se presione el mouse y todavía no se soltó';

}

function presion2()

{

document.getElementById('recuadro2').innerHTML='Se presione el mouse y se soltó';

}

</script>

</body>

</html>

En la función inicio registramos los eventos mousedown y mouseup para los dos div, al div recuadro1 procedemos a registrar el evento mousedown y al div recuadro2 procedemos a registrar el evento mouseup:

function inicio()

{

document.getElementById('recuadro1').addEventListener('mousedown',presion1,false);

document.getElementById('recuadro2').addEventListener('mouseup',presion2,false);

}

El método que se dispara para el primer div procedemos a modificar todo el contenido del div accediendo a la propiedad innerHTML:

function presion1()

{

document.getElementById('recuadro1').innerHTML='Se presione el mouse y todavía no se soltó';

}

De forma similar codificamos la función que modifica el segundo div:

function presion2()

{

document.getElementById('recuadro2').innerHTML='Se presione el mouse y se soltó';

}

----------------------------------------------

18)Eventos mouse over y mouse up

El evento mouseover se dispara cuando ingresamos con la flecha del mouse a un control HTML que tiene registrado dicho evento, y mouseout se dispara cuando sacamos la flecha del mouse del control.

Es muy común utilizar estos dos eventos para producir cambios en el elemento HTML cuando ingresamos la flecha del mouse y retornar al estado anterior cuando se saca la flecha del mouse.

Es importante hacer notar que estos eventos se disparan sin tener que presionar la flecha del mouse, solo con desplazarla al interior del elemento HTML se dispara el evento mouseover.

Implementaremos un pequeño ejemplo que muestre un cuadrado, el mismo mostrará los bordes redondeados cuando ingresemos la flecha del mouse en su interior y volverá al estado anterior cuando retiremos la flecha.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<div style="width:200px;height:200px;background:#0000ff" id="recuadro1">

</div>

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('recuadro1').addEventListener('mouseover',entrada,false);

document.getElementById('recuadro1').addEventListener('mouseout',salida,false);

}

function entrada()

{

document.getElementById('recuadro1').style.borderRadius='30px';

}

function salida()

{

document.getElementById('recuadro1').style.borderRadius='0px';

}

</script>

</body>

</html>

Definimos en el HTML un div de color azul de 200 píxeles de lado:

<div style="width:200px;height:200px;background:#0000ff" id="recuadro1">

</div>

En la función inicio registramos los eventos mouseover y mouseout:

function inicio()

{

document.getElementById('recuadro1').addEventListener('mouseover',entrada,false);

document.getElementById('recuadro1').addEventListener('mouseout',salida,false);

}

El método entrada se ejecuta cuando ingresemos la flecha del mouse en el div llamado recuadro1 y procedemos en el mismo a modificar la propiedad borderRadius del estilo de este elemento (indicamos que el redondeo de los vértices del div sea de 30 píxeles):

function entrada()

{

document.getElementById('recuadro1').style.borderRadius='30px';

}

Cuando sale la flecha del mouse del div se ejecuta la función salida donde fijamos con 0 la propiedad borderRadius:

function salida()

{

document.getElementById('recuadro1').style.borderRadius='0px';

}

19) evento mouse move

El evento mousemove se dispara cada vez que desplazamos la flecha del mouse sobre el elemento HTML que esta escuchando este evento.

Este tipo de evento hay que utilizarlo con cuidado ya que puede sobrecargar el navegador, ya que este evento se dispara cada vez que desplazamos aunque sea solo un pixel.

Implementaremos para ver su funcionamiento una página que muestre un div que capture el evento mousemove y como acción incrementaremos un contador para saber la cantidad de veces que se disparó el evento.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<div style="width:200px;height:200px;background:#0000ff" id="recuadro1">

</div>

<p id="cantidad">0</p>

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('recuadro1').addEventListener('mousemove',mover,false);

}

function mover()

{

var x=parseInt(document.getElementById('cantidad').innerHTML);

x++;

document.getElementById('cantidad').innerHTML=x;

}

</script>

</body>

</html>

Disponemos un div y un párrafo donde mostramos el número 0:

<div style="width:200px;height:200px;background:#0000ff" id="recuadro1">

</div>

<p id="cantidad">0</p>

Capturamos el evento mousemove para el div:

function inicio()

{

document.getElementById('recuadro1').addEventListener('mousemove',mover,false);

}

Cada vez que se emite el evento mousemove se llama el método mover donde extraemos el valor del párrafo, lo incrementamos en uno y lo volvemos a actualizar:

function mover()

{

var x=parseInt(document.getElementById('cantidad').innerHTML);

x++;

document.getElementById('cantidad').innerHTML=x;

}

20) Evento key down, key up, keypress

Estos tres eventos son similares a los eventos del mouse: mousedown, mouseup y click pero orientados al teclado del equipo.

El evento keydown se dispara cuando presionamos cualquier tecla del teclado, el evento keyup cuando soltamos una tecla. En cuanto el evento keypress en un principio procesa tanto cuando se la presionó y soltó, el único y gran inconveniente es que la mayoría de los navegadores no dispara el evento keypress para todas las teclas del teclado.

Para probar el evento keyup implementaremos un programa que permita solo ingresar 140 caracteres y nos informe con un mensaje la cantidad de caracteres disponibles para seguir escribiendo.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<input type="text" id="texto" maxlength="140" size="140">

<br>

<p>Máxima cantidad de caracteres disponibles:<span id="cantidad">140</span></p>

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('texto').addEventListener('keyup',presion,false);

}

function presion()

{

var canti=document.getElementById('texto').value.length;

var disponibles=140-parseInt(canti);

document.getElementById('cantidad').innerHTML=disponibles;

}

</script>

</body>

</html>

Definimos un control text y lo limitamos a 140 como máximo:

<input type="text" id="texto" maxlength="140" size="140">

Disponemos un elemento span para mostrar la cantidad de caracteres como máximo a ingresar:

<p>Máxima cantidad de caracteres disponibles:<span id="cantidad">140</span></p>

Registramos el evento keyup para el control texto:

function inicio()

{

document.getElementById('texto').addEventListener('keyup',presion,false);

}

Cada vez que se suelta la tecla cuando estamos escribiendo en el control de texto procedemos a extraer el valor del control texto y obtener mediante la propiedad length la cantidad de caracteres tipeados hasta este momento y seguidamente restamos a 140 el número de caractes tipeados y procedemos a mostrarlo en el elemento spam:

function presion()

{

var canti=document.getElementById('texto').value.length;

var disponibles=140-parseInt(canti);

document.getElementById('cantidad').innerHTML=disponibles;

}

21) Evento change

El evento change se lo puede asociar a distintos elementos de formularios HTML y su comportamiento depende de cada uno.

Cuando asociamos el evento change a un control de tipo checkbox el mismo se dispara inmediatamente después que lo chequeamos o sacamos la selección. Para los controles de tipo radio también se dispara luego que cambia el estado de selección del mismo.

Para los controles select se dispara el evento change cuando cambiamos el item seleccionado.

Para los controles text y textarea se dispara cuando pierde el foco del control (porque seleccionamos otro control) y hemos cambiando el contenido del mismo.

**Problema**

Implementaremos una aplicación que muestre un alert cada vez que sucede el evento change para los controles checkbox, radio, select, text y textarea.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

Evento change del checkbox:

<input type="checkbox" id="checkbox1" name="checkbox1" checked>

<br>

Evento change del radio:<br>

<input type="radio" name="radio1" id="radioa"><br>

<input type="radio" name="radio1" id="radiob"><br>

<input type="radio" name="radio1" id="radioc"><br>

Evento change del select:

<select name="select1" id="select1">

<option value="argentina">Argentina</option>

<option value="chile">Chile</option>

<option value="uruguay">Uruguay</option>

<option value="paraguay">Paraguay</option>

<option value="bolivia">Bolivia</option>

</select>

<br>

<input type="text" name="text1" id="text1">

<br>

<textarea name="text1" id="textarea1" rows="6" cols="80"></textarea>

<br>

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('checkbox1').addEventListener('change',cambiocheckbox,false);

document.getElementById('radioa').addEventListener('change',cambioradio,false);

document.getElementById('radiob').addEventListener('change',cambioradio,false);

document.getElementById('radioc').addEventListener('change',cambioradio,false);

document.getElementById('select1').addEventListener('change',cambioselect,false);

document.getElementById('text1').addEventListener('change',cambiotext,false);

document.getElementById('textarea1').addEventListener('change',cambiotextarea,false);

}

function cambiocheckbox()

{

alert(document.getElementById('checkbox1').checked);

}

function cambioradio()

{

alert(document.getElementById('radioa').checked+' '+

document.getElementById('radiob').checked+' '+

document.getElementById('radioc').checked);

}

function cambioselect()

{

alert(document.getElementById('select1').value);

}

function cambiotext()

{

alert(document.getElementById('text1').value);

}

function cambiotextarea()

{

alert(document.getElementById('textarea1').value);

}

</script>

</body>

</html>

Creamos una serie de controles de formulario HTML en el body de la página:

Evento change del checkbox:

<input type="checkbox" id="checkbox1" name="checkbox1" checked>

<br>

Evento change del radio:<br>

<input type="radio" name="radio1" id="radioa"><br>

<input type="radio" name="radio1" id="radiob"><br>

<input type="radio" name="radio1" id="radioc"><br>

Evento change del select:

<select name="select1" id="select1">

<option value="argentina">Argentina</option>

<option value="chile">Chile</option>

<option value="uruguay">Uruguay</option>

<option value="paraguay">Paraguay</option>

<option value="bolivia">Bolivia</option>

</select>

<br>

<input type="text" name="text1" id="text1">

<br>

<textarea name="text1" id="textarea1" rows="6" cols="80"></textarea>

<br>

En la función inicio asociamos a todos los controles para el evento change un método:

function inicio()

{

document.getElementById('checkbox1').addEventListener('change',cambiocheckbox,false);

document.getElementById('radioa').addEventListener('change',cambioradio,false);

document.getElementById('radiob').addEventListener('change',cambioradio,false);

document.getElementById('radioc').addEventListener('change',cambioradio,false);

document.getElementById('select1').addEventListener('change',cambioselect,false);

document.getElementById('text1').addEventListener('change',cambiotext,false);

document.getElementById('textarea1').addEventListener('change',cambiotextarea,false);

}

La función cambiocheckbox se dispara cada vez que cambiamos el estado del checkbox (tanto cuando lo seleccionamos como cuando lo deseleccionamos):

function cambiocheckbox()

{

alert(document.getElementById('checkbox1').checked);

}

La función cambioradio se dispara cada vez que seleccionamos alguno de los tres radio:

function cambioradio()

{

alert(document.getElementById('radioa').checked+' '+

document.getElementById('radiob').checked+' '+

document.getElementById('radioc').checked);

}

La función cambioselect se dispara cuando cambiamos de item del control select:

function cambioselect()

{

alert(document.getElementById('select1').value);

}

La función cambiotext y cambiotextarea se ejecuta cuando perdemos el foco del control y hemos producido un cambio en su contenido (agregado o borrado caracteres de su interior):

function cambiotext()

{

alert(document.getElementById('text1').value);

}

function cambiotextarea()

{

alert(document.getElementById('textarea1').value);

}

22)Eventos focus y blur

El evento focus se dispara cuando se activa el control o toma foco y el evento blur se dispara cuando pierde el foco el control. Podemos capturar el evento focus y blur de un control de tipo text, textarea, button, checkbox, file, password, radio, reset y submit.

**Problema**

Confeccionar un formulario que muestre dos controles de tipo text. El que está con foco mostrar su texto de color rojo y aquel que no está seleccionado el texto se debe mostrar de color negro.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<input type="text" id="text1" name="text1" size="30">

<br>

<input type="text" id="text2" name="text2" size="30">

<script>

addEventListener('load',inicio,false);

function inicio()

{

document.getElementById('text1').addEventListener('focus',tomarfoco1,false);

document.getElementById('text2').addEventListener('focus',tomarfoco2,false);

document.getElementById('text1').addEventListener('blur',perderfoco1,false);

document.getElementById('text2').addEventListener('blur',perderfoco2,false);

}

function tomarfoco1()

{

document.getElementById('text1').style.color='#ff0000';

}

function tomarfoco2()

{

document.getElementById('text2').style.color='#ff0000';

}

function perderfoco1()

{

document.getElementById('text1').style.color='#000000';

}

function perderfoco2()

{

document.getElementById('text2').style.color='#000000';

}

</script>

</body>

</html>

En la función inicio registramos los evento fucus y blur para los dos controles text:

function inicio()

{

document.getElementById('text1').addEventListener('focus',tomarfoco1,false);

document.getElementById('text2').addEventListener('focus',tomarfoco2,false);

document.getElementById('text1').addEventListener('blur',perderfoco1,false);

document.getElementById('text2').addEventListener('blur',perderfoco2,false);

}

En los métodos tomarfoco1 y tomarfoco2 activamos el color rojo para el texto del control text:

function tomarfoco1()

{

document.getElementById('text1').style.color='#ff0000';

}

function tomarfoco2()

{

document.getElementById('text2').style.color='#ff0000';

}

En los métodos perderfoco1 y perderfoco2 procedemos a activar el color negro para los controles text:

function perderfoco1()

{

document.getElementById('text1').style.color='#000000';

}

function perderfoco2()

{

document.getElementById('text2').style.color='#000000';

}

23)Trcer parametro en addeventlistenr

Hasta ahora siempre hemos asociado una función distinta para procesar los eventos de distintos elementos HTML. Pero veremos que está permitido asociar una único función a varios eventos de distintos objetos.

Debemos definir un parámetro en la función a implementar que llega la referencia del objeto que emitió el evento.

Implementaremos un panel con un conjunto de botones que nos permitan ingresar un valor numérico presionando botones.  
Asociaremos el click de cada botón con una única función.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

<style>

.boton{

width:50px;

height:50px;

}

#resultado {

font-size:40px;

}

</style>

</head>

<body>

<input type="button" id="boton0" name="boton0" value="0" class="boton">

<input type="button" id="boton1" name="boton1" value="1" class="boton">

<input type="button" id="boton2" name="boton2" value="2" class="boton">

<input type="button" id="boton3" name="boton3" value="3" class="boton">

<input type="button" id="boton4" name="boton4" value="4" class="boton">

<input type="button" id="boton5" name="boton5" value="5" class="boton">

<input type="button" id="boton6" name="boton6" value="6" class="boton">

<input type="button" id="boton7" name="boton7" value="7" class="boton">

<input type="button" id="boton8" name="boton8" value="8" class="boton">

<input type="button" id="boton9" name="boton9" value="9" class="boton">

<br>

<div id="resultado"></div>

<script>

window.addEventListener('load', inicio, false);

function inicio() {

for (var x = 0; x <= 9; x++) {

document.getElementById('boton' + x).addEventListener('click', presion, false);

}

}

function presion(evt) {

document.getElementById('resultado').innerHTML =

document.getElementById('resultado').innerHTML + evt.target.value;

}

</script>

</body>

</html>

En el body definimos 10 botones y un div donde mostraremos todos los números ingresados.

<input type="button" id="boton0" name="boton0" value="0" class="boton">

<input type="button" id="boton1" name="boton1" value="1" class="boton">

<input type="button" id="boton2" name="boton2" value="2" class="boton">

<input type="button" id="boton3" name="boton3" value="3" class="boton">

<input type="button" id="boton4" name="boton4" value="4" class="boton">

<input type="button" id="boton5" name="boton5" value="5" class="boton">

<input type="button" id="boton6" name="boton6" value="6" class="boton">

<input type="button" id="boton7" name="boton7" value="7" class="boton">

<input type="button" id="boton8" name="boton8" value="8" class="boton">

<input type="button" id="boton9" name="boton9" value="9" class="boton">

<br>

<div id="resultado"></div>

En el método inicio asociamos los diez botones con la función presion y para reducir el código disponemos un for que se repita 10 veces y llamamos en la misma al método addEventListener (debemos ir obteniendo el id del botón concatenando 'boton' y el contador del for):

function inicio() {

for (var x = 0; x <= 9; x++) {

document.getElementById('boton' + x).addEventListener('click', presion, false);

}

}

Lo nuevo aparece en la función presion:

function presion(evt) {

document.getElementById('resultado').innerHTML =

document.getElementById('resultado').innerHTML + evt.target.value;

}

La función presion tiene un parámetro llamado evt (podemos darle cualquier nombre) y el mismo es un objeto que tiene las siguientes propiedades:

target: Referencia del objeto que generó el evento (en nuestro ejemplo alguno de los 10 botones)

type: El nombre del evento (en nuestro caso click)

button: El botón del mouse presionado (0 = izquierdo, 1 = medio, 2 = derecho)

keyCode: El caracter del teclado presionado (en caso que corresponda)

shiftKey: true o false en caso de estar presionada esta tecla.

Como en este problema debemos ir concatenando el número presionado procedemos a obtener la referencia del div y asignarle el valor actual más la propiedad value del botón presionado:

document.getElementById('resultado').innerHTML =

document.getElementById('resultado').innerHTML + evt.target.value;

24)Evento submit

Todo formulario se le puede capturar el evento submit que se dispara previo a enviar los datos del formulario al servidor.

Uno de los usos más extendidos es la de validar los datos ingresados al formulario y abortar el envío de los mismos al servidos (con esto liberamos sobrecargas del servidor)

El evento submit se dispara cuando presionamos un botón de tipo type="submit".

Para probar el funcionamiento del evento submit implementaremos un formulario que solicita la carga de una clave y la repetición de la misma. Luego cuando se presione un botón de tipo "submit" verificaremos que las dos claves ingresadas sean iguales.

<!DOCTYPE html>

<html>

<head>

<title>Ejemplo de JavaScript</title>

<meta charset="UTF-8">

</head>

<body>

<form method="post" action="procesar.php" id="formulario1">

Ingrese clave:

<input type="password" id="clave1" name="clave1" size="20" required>

<br>

Repita clave:

<input type="password" id="clave2" name="clave2" size="20" required>

<br>

<input type="submit" id="confirmar" name="confirmar" value="Confirmar">

</form>

<script>

window.addEventListener('load', inicio, false);

function inicio() {

document.getElementById("formulario1").addEventListener('submit', validar, false);

}

function validar(evt) {

var cla1 = document.getElementById("clave1").value;

var cla2 = document.getElementById("clave2").value;

if (cla1!=cla2) {

alert('Las claves ingresadas son distintas');

evt.preventDefault();

}

}

</script>

</body>

</html>

Tengamos en cuenta que la primer línea indica que se trata de una página de HTML5:

<!doctype html>

Definimos un formulario que solicita la carga de dos claves y un botón submit para enviar los datos al servidor:

<form method="post" action="procesar.php" id="formulario1">

Ingrese clave:

<input type="password" id="clave1" name="clave1" size="20" required>

<br>

Repita clave:

<input type="password" id="clave2" name="clave2" size="20" required>

<br>

<input type="submit" id="confirmar" name="confirmar" value="Confirmar">

</form>

Registramos el evento load de la página indicando que se ejecute la función inicio donde registramos el evento submit del formulario:

window.addEventListener('load', inicio, false);

function inicio() {

document.getElementById("formulario1").addEventListener('submit', validar, false);

}

La función validar extrae los contenidos de los dos "password" y verificamos si tienen string distintos en cuyo caso llamando al método preventDefault del objeto que llega como parámetro lo cual previene que los datos se envíen al servidor:

function validar(evt) {

var cla1 = document.getElementById("clave1").value;

var cla2 = document.getElementById("clave2").value;

if (cla1!=cla2) {

alert('Las claves ingresadas son distintas');

evt.preventDefault();

}

}

26)Operadores de igualdad

El operador == primero hace la conversión a un mismo tipo de dato para verificar si son iguales, en cambio el operador === llamado operador de igualdad estricta compara los valores sin hacer conversiones. Esto hace que cuando utilizamos el operador === retorne siempre false en caso que las variables que comparamos sean de distinto tipo.

El siguiente if se verifica verdadero ya que con el operador == primero se transforman los dos valores al mismo tipo de datos previos a verificar su igualdad:

var v1='55';

if (v1==55)

document.write('son iguales');

En cambio el siguiente if utilizando el operador de igualdad estricta se verifica false ya que son variables o valores de distinto tipo:

var v1='55';

if (v1===55)

document.write('son iguales');

Los operadores != y !== son los opuestos de == y ===.

El operador != retorna true si son distintos previo conversión a un mismo tipo de dato y el operador !== retorna true si los valores son distintos sin hacer una conversión previa, teniendo en cuenta que para tipos de datos distintos es true el valor retornado.

El siguiente if se verifica false:

var v1='55';

if (v1!=55)

document.write('son distintos');

En cambio utilizando el operador !== se verifica verdadero ya que son tipos de datos desiguales:

var v1='55';

if (v1!==55)

document.write('son distintos');

El operador de igualdad estricto === analiza los operandos de la siguiente forma:

* Si los dos valores son de diferente tipo luego el resultado es false.
* Si los dos valores almacen null luego el resultado es true. Lo mismo si los dos valores almacenan undefined.
* Si uno de los valores almacena el valor NaN luego la condición se verifica como false.

El operador de igualdad == analiza los operandos de la siguiente forma:

* Si los dos operandos no son del mismo tipo el operador == primero verifica si uno de los valores es null y el otro undefined luego retorna true, si uno es un número y el otro un string convierte a string el número y luego compara, si uno de los valores es true convierte este a 1 y luego compara, si uno de los valores es false convierte este a 0 (cero) y luego compara.

Algunas comparaciones:

if (true==1) //verdadero

if (true===1) //falso

if (false==0) //verdadero

if (false===0) //falso

funcion is nan

La función global isNaN (is Not a Number) verifica si el valor que le pasamos es un número válido y podemos estar seguros de operar con dicho valor. Esta función puede ser empleada inmediatamente luego de llamar a las funciones parseInt y parseFloat.

Con un pequeño ejemplo podemos ver los valores que nos retorna la función isNaN si le pasamos variables con un valor entero, float, un string con caracteres numéricos y un string con caracteres alfabéticos:

var x=10;

if (isNaN(x)) //false

alert('no es un número');

var z=10.5;

if (isNaN(z)) //false

alert('no es un número');

var edad='77';

if (isNaN(edad)) //false

alert('no es un número');

var nom='juan';

if (isNaN(nom)) //true

alert('no es un número:'+nom);