**Eventos con Jquery**

**Métodos de eventos más utilizados.**

Aunque los eventos que se usan con jQuery son en su mayoría los mismos que los usados en javascript, puede haber algunas diferencias.

jQuery define una lista de eventos y funciones para la administración de los mismos, la sintaxis por defecto para un manejador de evento es de la siguiente forma $.fn.nombreEvento.

$(**'Selector'**).nombreEvento( **function**(event){

*//funcion que administra el evento.*

*//this es el elemento que disparo el evento.*

}

Aquí es importante resaltar que **this** contendrá la instancia del elemento que disparo el evento, por ejemplos si a los enlaces(a) le agregamos el evento click, this contendrá la instancia del enlace específico sobre del cual hallamos hecho click, analice el siguiente código.

**Eventos del ratón**

**.click()** : pulsar una vez el ratón sobre un elemento:

$("#click").click(function() {  
alert("click simple");  
});

**.dblclick()** : pulsar el ratón dos veces seguidas sobre un elemento:

$("#dblclick").dblclick(function() {  
alert("doble click");  
});

**.mouseenter()** : El ratón se sitúa encima de un elemento:

$("#mouse1").mouseenter(function() {  
$(this).css("color","red");  
});

**.mouseleave()** : El ratón, que estaba situado encima de un elemento, sale de él:

$("#mouse1").mouseleave(function() {  
$(this).css("color","blue");  
});

**.hover()** : Este evento es una combinación de los dos anteriores, detecta la posición del raton, y se produce tanto cuando éste se coloca encima del elemento, como cuando sale de él. Admite dos funciones, que las escribiremos separadas por una coma, La primera de ellas se produce cuando el ratón entra, y la segunda cuando sale:

$("#hover").hover(function() {  
$(this).css("color","red");  
},  
function() {  
$(this).css("color","blue");  
});

**.mousedown()** : Momento en que presiona el botón, independientemente de si se suelta o no, vale tanto para el botón izquierdo como para el derecho.

$("#pulsaRaton").mousedown(function() {  
$(this).html("<b>mousedown</b>, mouseup: pulsado")  
});

**.mouseup()** : soltar un botón del ratón después de hacer click. El evento se produce sólo en el momento de soltar el botón.

$("#pulsaRaton").mouseup(function() {  
$(this).html("mousedown, <b>mouseup</b>: sin pulsar")  
});

**.mousemove()** : El evento se produce al mover el ratón sobre un elemento.

n1=0;  
$("#capa2").mousemove(function(){  
n1++;   
$("#mousemove").html("mousemove: "+n1+" movimientos");  
});

**.mouseover()** : El evento se produce al situarse el ratón sobre un elemento. El evento puede producirse varias veces mientras está encima del elemento.

n2=0;   
$("#capa2").mouseover(function(){  
n2++;  
$("#mouseover").html("mouseover: "+n2+" veces");  
});

*Practica eventos II*

mousemove: 0 movimientos

mouseover: 0 veces

toggle: click para cambiar color

tecla pulsada

**.toggle()** : El evento se produce al hacer click con el ratón. Tiene la particularidad de que admite varias funciones, que se van alternando en su ejecución cuando el usuario realiza clicks.

$("#toggle").toggle(function() {  
$(this).css("color","red");  
},  
function() {  
$(this).css("color","blue");  
},  
function() {  
$(this).css("color","green");  
});

**Eventos del teclado**

**.keydown()** : El evento se produce en el momento que se presiona una tecla, independientemente de si se libera la presión o se mantiene. Se produce una única vez en el momento exacto de la presión.

**.keypress()** : Se produce al tener pulsada una tecla, Si se mantiene pulsada, el evento se produce varias veces, de la misma manera que se escriben varios caracteres al mantenerla pulsada. Se corresponde con el hecho de tener la tecla pulsada.

**.keyup()** : El evento se produce en el momento de dejar de presionar una tecla que teníamos pulsada.

Los eventos de teclado se suelen aplicar al documento (document), y no a un elemento concreto. veamos el código del ejemplo del cuadro de la derecha:

$(document).keydown(function(){  
$("#teclado").html("tecla pulsada");  
});  
$(document).keyup(function(){  
$("#teclado").html("tecla sin pulsar");  
});

### Ejemplo de eventos de teclado:

$('#target').keyup(function(event) {

var msg = 'El evento keyup() fue llamado para la tecla ' + event.which;

console.log(msg);

}).keydown(function(event) {

if (event.which == 13) {

event.preventDefault();

}

});

**Eventos combinados teclado-ratón**

Son los eventos que controlan el "foco" o elemento seleccionado. Podemos llevar el foco a un elemento mediante el ratón (haciendole click), o mediante el teclado (tabulador o inter). En una página los elementos que admiten tener el foco suelen ser los elementos de formulario.

**.focusin()** : El elemento adquiere el foco de la aplicación.

**.focusout()** : El elemento pierde el foco de la aplicación.

**.focus()** : El elemento tiene el foco de la aplicación. Igual que el evento "onfocus" de javascript.

**Lenguaje de programación Java Script**

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Código: HTML - Hola Mundo!

<!DOCTYPE html>

<html>

<head>

<title>Hello World!</title>

</head>

<body>

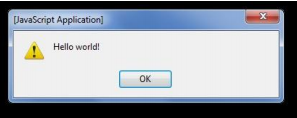
<script>

alert('Hello world!');

</script>

</body>

</html>



**Variables**

En pocas palabras, una variable es un espacio de almacenamiento en un ordenador para grabar cualquier tipo de datos como una cadena de caracteres, un valor numérico o estructuras un poco más específicas.

Para declarar una variable, simplemente hay que escribir la siguiente línea:

Código: JavScript

var miVariable;

Código: JavaScript

var miVariable;

miVariable = 2;

El signo = se utiliza para asignar un valor a la variable, aquí le hemos asignado el número 2.

Cuando das un valor a una variable, decimos que se trata de una asignación, ya que asigna un valor a la variable.

Es posible simplificar el código en una sola línea:

Código: JavaScript

var miVariable = 5.5 / / Como puedes ver, los números decimales se

separan con un punto

Del mismo modo, puedes declarar y asignar variables en una sola línea:

Código: JavaScript

var miVariable1, miVariable2 = 4, miVariable3;

Aquí hemos declarado tres variables en una fila, pero sólo la segunda tiene valor asignado.

Y la última posibilidad, que puede ser útil de vez en cuando:

Código: JavaScript

var miVariable1, miVariable2;

miVariable1 = miVariable2 = 2;

Ambas variables ahora contienen el mismo número 2. Puedes hacer lo mismo con tantas variables como desees.

**El tipo numérico (número):** representa cualquier número, ya sea un entero, un número negativo, en notación científica, etc. En pocas palabras, este es el tipo de números.

**Cadenas de caracteres (alias string):** Este tipo representa texto. Puede asignarse de dos maneras diferentes.

Código: JavaScript

var text1 = "Mi primer texto" / / Con comillas

var text2 = 'Mi segundo mensaje' / / Con apóstrofos

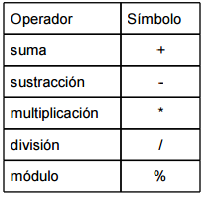
**Booleanos (booleano):** son un tipo particular de que se tratará más adelante. Mientras tanto, en pocas palabras, un tipo booleano permite dos estados verdadero o falso. Estos dos estados se puede escribir como sigue:

Código: JavaScript

var EsVerdader = true;

var EsFalso = false;

**Operadores aritméticos**



Algunos cálculos sencillos

Programar el cálculo es casi tan fácil como en una calculadora, por ejemplo:

Código: JavaScript

var resultado = 3 + 2;

alert (resultado) / / Muestra « 5 »

Así que puedes hacer cálculos con dos números, es bueno, pero con dos variables que contienen números en sí es más útil:

Código: JavaScript

var número1=3, número2 = 2, resultado;

resultado = numero1 \* numero2;

alert (resultado) / / Muestra: « 6 »

Podemos ir aún más lejos al escribirlo como cálculos con operadores múltiples y variables:

Código: JavaScript

var divisor = 3, resultado1, resultado2, resultado3;

resultado1 = (16 + 8) / 2 - 2; / / 10

resultado2 = resultado1 / divisor;

resultado3 = resultado1 % divisor;

alerta (resultado2) / / Resultado de la división: 3,33

**Interactuar con el usuario**

La concatenación es un buen momento para introducir la primera interacción con el usuario a través de la función prompt (). He aquí cómo se usa:

Código: JavaScript

var NombreUsuario = prompt('Introduce nombre:');

alert(NombreUsuario); // Muestra el nombre introducido

**Conversión de una cadena de caracteres en número**

Ahora trataremos de hacer una adición con números proporcionados por el usuario:

Código: JavaScript

var primero, segundo, resultado;

primero = prompt ('Introduzca el primer número: ");

segundo = prompt ('Introduzca el segundo número:');

resultado = primero + segundo;

alert (resultado);

**Conversión de un número en una cadena**

Para concluir esta sección, veremos cómo convertir un número en una cadena. Ya es posible concatenar un número y una cadena sin conversión, pero no dos números, tal como se les añadió debido a la utilización de +. Por lo tanto necesito convertir un número en cadena.

Código: JavaScript

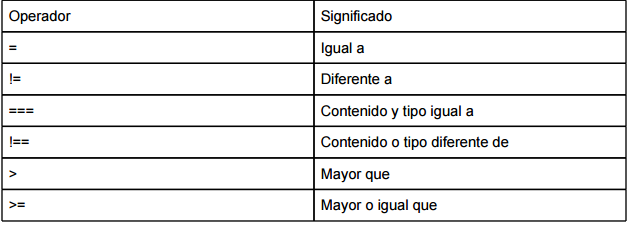
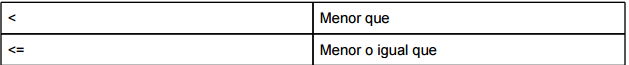
var texto, numero1 = 4, numero2 = 2;

text = numero1 + ’ ’ + numero2;

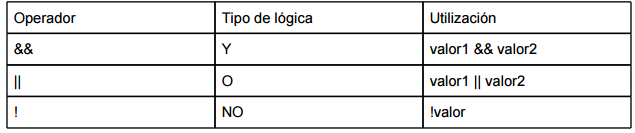
alert (texto) / / Muestra: « 42 »

**Los operadores de comparación**

Como su nombre indica, estos operadores pueden realizar comparaciones entre diferentes valores entre ellos. En total, hay muchos, ocho, aquí están:



**Operadores lógicos**



**Condición "if else"**

Código: JavaScript

if (true) {

alert ("Este mensaje se muestra bien.");

}

if (false) {

alert ("No hay necesidad de insistir, este mensaje no se mostrará.");

}

**La condición "switch"**

Un ejemplo: tenemos un mueble con cuatro cajones, cada uno conteniendo diferentes objetos, y es necesario que el usuario pueda conocer el contenido de cada cajón, que ha sido cifrado. Si lo hiciéramos con if else sería muy largo y tedioso:

Código: JavaScript

var cajon = parseInt (prompt ('Elegir el cajón abierto (1-4): '));

if (cajon == 1) {

alert ('Contiene varias herramientas de dibujo: papel, lápices, etc. ')

} else if (cajon == 2) {

alert ('Contenido hardware: cables, componentes, etc. ')

} else if (cajon == 3) {

alert ('¿Ah, el cajón está cerrado malo?');

} else if (cajon == 4) {

alert ('Contiene la ropa: camisas, pantalones, etc. ')

Else {}

alert ("La noticia del día: el gabinete contiene sólo cuatro cajones y

hasta que se demuestre lo contrario, los cajones negativos no existen ").;

}

**Bucles**

**Bucle while**

Un bucle es una estructura condicional, similar a lo visto previamente, excepto que se trata de repetir una serie de instrucciones. La repetición es hasta que lo indique el bucle.

Cada vez que el bucle se repite se efectúa una iteración (que en realidad es un sinónimo de repetición).

Código: JavaScript

while (condicion) {

sentencia\_1;

sentencia\_2;

sentencia\_3;

}

**Do while**

Este bucle se asemeja mucho al bucle while, excepto que en este caso el bucle se ejecuta siempre por lo menos una vez. En el caso de un bucle while, si la condición no es válida, el bucle no es ejecutado Con do while, el bucle se ejecuta una vez, entonces la condición se comprueba para determinar si el bucle debe continuar. La sintaxis de un bucle do while es:Código: JavaScript

do {

sentencia\_1;

sentencia\_2;

sentencia\_3;

} while (condición);

**Bucle for**

El bucle for se ve en su aplicación como parecido al bucle while, pero su arquitectura parece complicado al principio. El bucle for es un bucle que funciona de forma bastante simple, pero muy compleja para los principiantes debido a su sintaxis.

Código: JavaScript

for (inicio; condicion, incremento) {

sentencia\_1;

sentencia\_2;

sentencia\_3;

}