Introducción JavaScript

Funcion que se ejecuta al cargar la pagina

```
window.onload= function(){    }
```

Otra opcion seria:

```
windows.addEventListener('load', cargaEventos);

function cargaEventos()
 document.getElementById("par").addEventListener("Click,mifuncion");
 document.getElementById("cierra").addEventListener("Click,cierra");
 document.getElementById("test").addEventListener("Click,test");
```

Corre una lista y mira si se ha pulsado el elemento

```
var li=document.getElementsByTagName("li");

for(var i=0; i<li.length; i++){
  li[i].addEventListener("click",elimina);
}
```

Javascript

- Javascript es un **lenguaje de programación** que se utiliza para añadir **pequeños** programas dentro del código de una página **web**.

- Se trata de un lenguaje ejecutado **desde el navegador**, al contrario que, por ejemplo, PHP, que se ejecuta en el servidor. Este hecho hace que sea el lenguaje a utilizar a la hora de añadir **efectos** y comportamientos **dinámicos** a nuestra página.

- La sintaxis de Javascript es **muy similar** a la de PHP, y por tanto únicamente se trataran en profundidad las peculiaridades que lo diferencian de él.

Javascript

- El código Javascript irá situado siempre entre las etiquetas HTML **<script>…</script>**. El formato habitual es el siguiente:

<script>
código javascript…
</script>

- Se puede insertar un script en cualquier lugar del código, aunque lo más común es encontrarlo al **inicio** de la página, dentro de la etiqueta **<head>**.

Javascript

- Definir JavaScript en un archivo externo

```
<script src="/js/codigo.js"></script>
```

Archivo codigo.js

```
alert("Un mensaje de prueba");
```

- Incluir JavaScript en los elementos HTML

```
<p onclick="alert('Un mensaje de prueba')">Un párrafo de texto.</p>
```

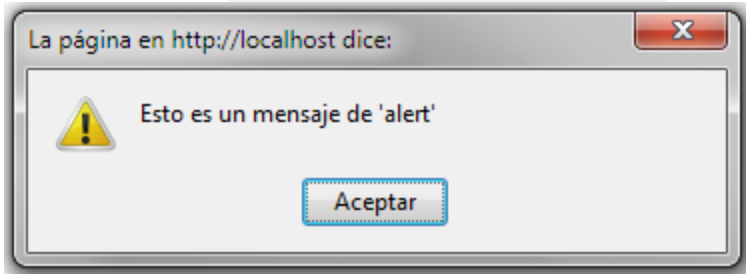
- Etiqueta noscript

la etiqueta <noscript> para mostrar un mensaje al usuario cuando su navegador no puede ejecutar JavaScript.

Javascript

- La función **alert**
  - Se trata de una función que nos permite mostrar un **mensaje informativo** en nuestra página web.

alert("Texto a mostrar");

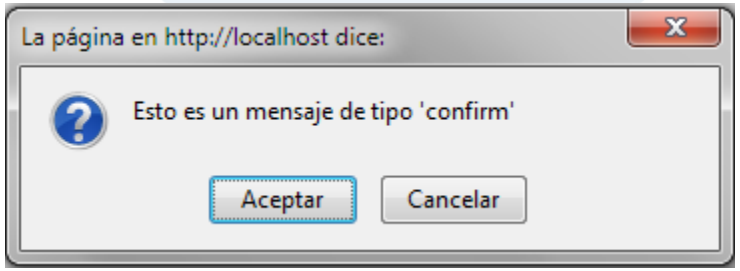


  - Suele utilizarse para mostrar **información** al usuario sobre sus acciones, como algún **error** en formularios, o **información corta** que se quiera destacar.

Javascript

- La función **confirm**
  - Función similar a alert, que además de mostrar un texto informativo, plantea una **elección** al usuario, mostrando los botones **Aceptar** y **Cancelar**.

    confirm("Texto a mostrar");



  - Se puede asignar el resultado **a una variable**, ya que la función devuelve un valor **booleano, true o false**. Dependiendo del valor recibido, se pueden programar distintas acciones.

Ejercicios

1. Añadid a alguna de las páginas web que tenéis hechas un mensaje de bienvenida que se muestre mediante **alert**, de Javascript.

2. Modificad el mensaje para que sea de tipo **confirm.**

Javascript

- Palabras reservadas:

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

- Las normas básicas que definen la sintaxis de JavaScript son las siguientes
  - No se tienen en cuenta los espacios en blanco y las nuevas líneas.
  - Se distinguen las mayúsculas y minúsculas.
  - No se define el tipo de las variables.
  - Se pueden incluir comentarios.
    - // comentario de sola línea
    - /* Los comentarios de varias líneas */

Programación básica

- Variables

La palabra reservada **var** solamente se debe indicar al definir por primera vez la variable, lo que se denomina **declarar** una variable.

Si cuando se declara una variable se le asigna también un valor, se dice que la variable ha sido **inicializada**.

```
var numero_1 = 3;
var numero_2 = 1;
var resultado = var numero_1 + var numero_2;
```

Programación básica

El nombre de una variable también se conoce como **identificador** y debe cumplir:

- Sólo puede estar formado por letras, números y los símbolos $ (dólar) y _ (guión bajo).
- El primer
- carácter no puede ser un número.

```
var $numero1;
var _$letra;
var $$$otroNumero;
var $_a__$4;
var 1numero;      // Empieza por un número
var numero;1_123; // Contiene un carácter ";"
```

Programación básica

- Tipos de variables

Numéricas

var iva = 16; *// variable tipo entero*
var total = 234.65; *// variable tipo decimal*

Cadenas de texto

var mensaje = "Bienvenido a nuestro sitio web";
var nombreProducto = 'Producto ABC';
var letraSeleccionada = 'c‘;

Se pueden definir con comilla simple o comilla doble

Programación básica

Para incluir caracteres especiales dentro de una cadena de texto.

Si se quiere incluir...	Se debe incluir...
Una nueva línea	\n
Un tabulador	\t
Una comilla simple	\'
Una comilla doble	\"
Una barra inclinada	\\

```
var texto1 = 'Una frase con \'comillas simples\' dentro';
var texto2 = "Una frase con \"comillas dobles\" dentro";
```

Programación básica

Arrays

Para definir un array, se utilizan los caracteres [y] para delimitar su comienzo y su final y se utiliza el carácter , (coma) para separar sus elementos:

var nombre_array = **[**valor1**,** valor2**,** ...**,** valorN**]**;

var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes", "Sábado", "Domingo"];

Cada elemento se accede indicando su posición dentro del array.

var diaSeleccionado = dias[0]; *// diaSeleccionado = "Lunes"*

var otroDia = dias[5]; *// otroDia = "Sábado"*

* las posiciones de los elementos empiezan a contarse en el 0 y no en el 1.

Programación básica

Booleans

Una variable de tipo *boolean* almacena un tipo especial de valor que solamente puede tomar dos valores: true (verdadero) o false (falso).

```
var clienteRegistrado = false;
var ivaIncluido = true;
```

Programación básica

- Operadores

Asignación

```
var numero1 = 3;
var numero2 = 4;
/* Error, la asignación siempre se realiza a una variable,
por lo que en la izquierda no se puede indicar un número */
5 = numero1;
// Ahora, la variable numero1 vale 5
numero1 = 5;
// Ahora, la variable numero1 vale 4
numero1 = numero2;
```

Programación básica

Incremento y decremento

Estos dos operadores solamente son válidos para las variables numéricas

```
var numero = 5;
++numero;    // numero = numero + 1;
alert(numero); // numero = 6

var numero = 5;
--numero; // numero = numero - 1;
alert(numero); // numero = 4
```

Programación básica

Si el operador ++ se indica como prefijo del identificador de la variable, su valor se incrementa **antes** de realizar cualquier otra operación. Si el operador ++ se indica como sufijo del identificador de la variable, su valor se incrementa **después** de ejecutar la sentencia en la que aparece.

```
var numero1 = 5;
var numero2 = 2;
numero3 = numero1++ + numero2;
// numero3 = 7, numero1 = 6
```

Dos operaciones en uno:
1ro - 5+2=7
2do - numero1= 5+1

```
var numero1 = 5;
var numero2 = 2;
numero3 = ++numero1 + numero2;
// numero3 = 8, numero1 = 6
```

numero1 = 5+1 = 6
6+2=8

Programación básica

Matemáticos

JavaScript permite realizar manipulaciones matemáticas sobre el valor de las variables numéricas. Los operadores definidos son: suma (+), resta (-), multiplicación (*), división (/) y modulo (%).

Resto de la division

```
var numero1 = 10;
var numero2 = 5;

resultado = numero1 / numero2; // resultado = 2
resultado = 3 + numero1; // resultado = 13
resultado = numero2 – 4; // resultado = 1
resultado = numero1 * numero 2; // resultado = 50
resultado = numero1 % numero2; // resultado = 0

numero1 = 9;
numero2 = 5;
resultado = numero1 % numero2; // resultado = 4
```

Programación básica

Matemáticos

Los operadores matemáticos también se pueden combinar con el operador de asignación para abreviar su notación.

```
var numero1 = 5;

numero1 += 3; // numero1 = numero1 + 3 = 8
numero1 -= 1; // numero1 = numero1 - 1 = 4
numero1 *= 2; // numero1 = numero1 * 2 = 10
numero1 /= 5; // numero1 = numero1 / 5 = 1
numero1 %= 4; // numero1 = numero1 % 4 = 1
```

Programación básica

Relacionales

Los operadores relacionales son idénticos a los que definen las matemáticas: mayor que (>), menor que (<), mayor o igual (>=), menor o igual (<=), igual que (==) y distinto de (!=). El resultado de todos estos operadores siempre es un valor booleano.

```
var numero1 = 3;
var numero2 = 5;
resultado = numero1 > numero2; // resultado = false
resultado = numero1 < numero2; // resultado = true

numero1 = 5;
numero2 = 5;
resultado = numero1 >= numero2; // resultado = true
resultado = numero1 <= numero2; // resultado = true
resultado = numero1 == numero2; // resultado = true
resultado = numero1 != numero2; // resultado = false
```

Programación básica

Relacionales

Los operadores relacionales también se pueden utilizar con variables de tipo cadena de texto.

```
var texto1 = "hola";
var texto2 = "hola";
var texto3 = "adios";

resultado = texto1 == texto3; // resultado = false
resultado = texto1 != texto2; // resultado = false
resultado = texto3 >= texto2; // resultado = false
```

* a es menor que b, b es menor que c, A es menor que a, etc.

Programación básica

- Lógicos

El resultado de cualquier operación que utilice operadores lógicos siempre es un valor lógico o *booleano*.

Negación

! Se utiliza para obtener el valor contrario al valor de la variable.

```
var visible = true;
alert(!visible); // Muestra "false" y no "true"
```

Programación básica

Negación

variable	!variable
true	false
false	true

- Si la variable contiene un número, se transforma en false si vale 0 y en true para cualquier otro número (positivo o negativo, decimal o entero).
- Si la variable contiene una cadena de texto, se transforma en false si la cadena es vacía ("") y en true en cualquier otro caso.

```
var cantidad = 0;
vacio = !cantidad; // vacio = true
cantidad = 2;
vacio = !cantidad; // vacio = false
var mensaje = "";
mensajeVacio = !mensaje; // mensajeVacio = true
mensaje = "Bienvenido";
mensajeVacio = !mensaje; // mensajeVacio = false
```

Programación básica

AND

variable1	variable2	variable1 && variable2
true	true	true
true	false	false
false	true	false
false	false	false

La operación lógica AND (**&&**) obtiene su resultado combinando dos valores booleanos

```
var valor1 = true;
var valor2 = false;
resultado = valor1 && valor2; // resultado = false
valor1 = true;
valor2 = true;
resultado = valor1 && valor2; // resultado = true
```

Programación básica

OR

variable1	variable2	variable1 || variable2
true	true	true
true	false	true
false	true	true
false	false	false

La operación lógica OR (||) obtiene su resultado combinando dos valores booleanos.

```
var valor1 = true;
var valor2 = false;
resultado = valor1 || valor2; // resultado = true
valor1 = false;
valor2 = false;
resultado = valor1 || valor2; // resultado = false
```

* Ejercicio 4

Programación básica

- Estructuras de control de flujo

Son instrucciones del tipo "*si se cumple esta condición, hazlo; si no se cumple, haz esto otro*". También existen instrucciones del tipo "*repite esto mientras se cumpla esta condición*".

Estructura if

```
if(condicion) {
...
}
```

Ejemplo:

```
var mostrarMensaje = true;

if(mostrarMensaje == true) {
alert("Hola Mundo");
}
```

```
if(mostrarMensaje) {
    alert("Hola Mundo");
}
```

Programación básica

Estructura if...else

```
if(condicion) {
...
}
else {
...
}
```

Ejemplo:

```
var edad = 18;
if(edad >= 18) {
    alert("Eres mayor de edad");
}else {
    alert("Todavía eres menor de edad");
}
```

La estructura if...else if ... se puede encadenar para realizar varias comprobaciones seguidas.

Programación básica

Estructura switch

```
switch (expression) {
case (value1) :
// Code for value1
break;
case (value2) :
// Code for value2
break;
...
default :
// Code for other values
}
```

La estructura switch, se utiliza para agilizar los casos en los que la variable que estamos evaluando (en el ejemplo expression) puede tomar múltiples valores.

El switch trabaja de la misma manera que lo harían sucesivos if... else if...

En los "case" pondremos los possibles valores que pueda tener la variable evaluada.

```
var edad = promt ("introduce tu edad")
```

* Ejercicio 5 y 6

Programación básica

Estructura for

La estructura for permite realizar repeticiones (también llamadas bucles) de una forma muy sencilla.

```
for(inicializacion; condicion; actualizacion) {
...
}
```

- La "inicialización" es la zona en la que se establece los valores iniciales de las variables que controlan la repetición.
- La "condición" es el único elemento que decide si continua o se detiene la repetición.
- La "actualización" es el nuevo valor que se asigna después de cada repetición a las variables que controlan la repetición.

Programación básica

Estructura for

```
var mensaje = "Hola, estoy dentro de un bucle";
for(var i = 0; i < 5; i++) {
alert(mensaje);
}
```

i++ --> Cta de 0 a 4
++i --> Cta de 1 a 5

Estructura for...in

```
for(indice in array) {
...
}
```

Ejemplo:

```
var dias = ["Lunes", "Martes", "Miércoles", "Jueves", "Viernes",
"Sábado", "Domingo"];
```

```
for(i in dias) {
alert(dias[i]);
}
```

```
for(var i=0; i<7; i++) {
    alert(dias[i]);
}
```

Programación básica

Estructura while

La estructura while permite crear bucles que se ejecutan ninguna o más veces, dependiendo de la condición indicada.

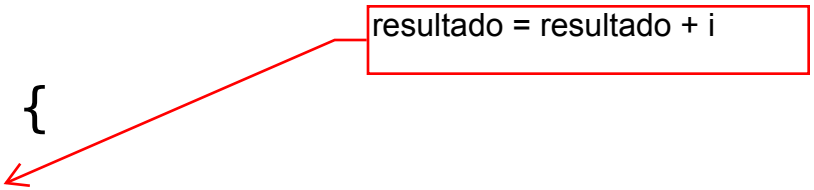
Su definición formal es:

```
while(condicion) {
...
}
```

Ejemplo:

```
var resultado = 0;
var numero = 100;
var i = 0;

while(i <= numero) {
    resultado += i;
    i++;
}
```



Programación básica

Estructura do...while

El bucle de tipo do...while es muy similar al bucle while, salvo que en este caso **siempre** se ejecutan las instrucciones del bucle al menos la primera vez.

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

Ejemplo:

```
var resultado = 1;
var numero = 5;

do {
    resultado *= numero; // resultado = resultado * numero
    numero--;
} while(numero > 0);
```

Programación básica

Sentencia break y continue

- La **sentencia break** sirve para interrumpir la ejecución de un bloque iterativo en cualquier momento.

- La sentencia **continue** también sirve para alterar el funcionamiento de los bloques iterativos, pero en vez de interrumpir la ejecución del bloque iterativo, se inicia una iteración nueva.

```
Salta un calculo del bucle y continua
con el bucle

  if (i==2){   continue   }

No usaria el 2
```

* Ejercicio 7

Programación básica

Funciones útiles para cadenas de texto

length, calcula la longitud de una cadena de texto:
var mensaje = "Hola Mundo";
var numeroLetras = mensaje.length; *// numeroLetras = 10*

+ y concat(), se emplea para concatenar varias cadenas de texto
var mensaje1 = "Hola";
var mensaje2 = " Mundo";

var mensaje = mensaje1 + mensaje2; *// mensaje = "Hola Mundo"*
var mensaje2 = mensaje1.concat(" Mundo"); *// mensaje2 = "Hola Mundo"*
* Las cadenas de texto también se pueden unir con variables numéricas.

string.**replace(searchvalue, newvalue)**

The replace() method searches a string for a specified value, or a regular expression, and returns a new string where the specified values are replaced.

toUpperCase(), transforma todos los caracteres de la cadena a sus correspondientes caracteres en mayúsculas:

```
var mensaje1 = "Hola";
var mensaje2 = mensaje1.toUpperCase(); // mensaje2 = "HOLA"
```

toLowerCase(), transforma todos los caracteres de la cadena a sus correspondientes caracteres en minúsculas:

```
var mensaje1 = "HolA";
var mensaje2 = mensaje1.toLowerCase(); // mensaje2 = "hola"
```

charAt(posicion), obtiene el carácter que se encuentra en la posición indicada:

```
var mensaje = "Hola";
var letra = mensaje.charAt(0); // letra = H
letra = mensaje.charAt(2); // letra = l
```

Programació

string.**indexOf**(searchvalue, start)

busca palabras tambien:

```
var str = "Hello world, welcome to the universe.";
var n = str.indexOf("welcome");
```

indexOf(caracter), calcula la posición en la que se encuentra el carácter indicado dentro de la cadena de texto. Si el carácter se incluye varias veces dentro de la cadena de texto, se devuelve su primera posición empezando a buscar desde la izquierda.

```
var mensaje = "Hola";
var posicion = mensaje.indexOf('a'); // posicion = 3
posicion = mensaje.indexOf('b'); // posicion = -1
```

lastIndexOf(caracter), calcula la última posición en la que se encuentra el carácter indicado dentro de la cadena de texto.

string.lastIndexOf(searchvalue, start)

```
var mensaje = "Hola";
var posicion = mensaje.lastIndexOf('a'); // posicion = 3
posicion = mensaje.lastIndexOf('b'); // posicion = -1
```

Programación básica

split(separador), convierte una cadena de texto en un array de cadenas de texto.

```
var mensaje = "Hola Mundo, soy una cadena de texto!";
var palabras = mensaje.split(" "); // palabras = ["Hola", "Mundo,", "soy", "una", "cadena", "de", "texto!"];

var palabra = "Hola";
var letras = palabra.split(""); // letras = ["H", "o", "l", "a"]
```

Programación básica

substring(inicio, final), extrae una porción de una cadena de texto. El segundo parámetro es opcional.

```
var mensaje = "Hola Mundo";

var porcion = mensaje.substring(2); // porcion = "la Mundo"
porcion = mensaje.substring(5); // porcion = "Mundo"
var porcion = mensaje.substring(-2); // porcion = "Hola Mundo“

var porcion = mensaje.substring(1, 8); // porcion = "ola Mun"
porcion = mensaje.substring(3, 4); // porcion = "a“

var porcion = mensaje.substring(5, 0); // porcion = "Hola "
porcion = mensaje.substring(0, 5); // porcion = "Hola "
```

Programación básica

Funciones útiles para arrays

length, calcula el número de elementos de un array:
var vocales = ["a", "e", "i", "o", "u"];
var numeroVocales = vocales.length; // numeroVocales = 5

`Junta arrays, manteniendo formato array`

concat(), se emplea para concatenar los elementos de varios arrays:
var array1 = [1, 2, 3];
array2 = array1.concat(4, 5, 6); // array2 = [1, 2, 3, 4, 5, 6]
array3 = array1.concat([4, 5, 6]); // array3 = [1, 2, 3, 4, 5, 6]

`junta arrays y crea una sola variable texto`

join(separador), une todos los elementos de un array para formar una cadena de texto:
var array = ["hola", "mundo"];
var mensaje = array.join(""); *// mensaje = "holamundo"*
mensaje = array.join(" "); *// mensaje = "hola mundo"*

Programación básica

Elimina ultimo elemento de array

pop(), elimina el último elemento del array y lo devuelve. El array original se modifica y su longitud disminuye en 1 elemento.

```
var array = [1, 2, 3];
var ultimo = array.pop(); // ahora array = [1, 2], ultimo = 3
```

Añade al final de la array

push(), añade un elemento al final del array. El array original se modifica y aumenta su longitud en 1 elemento.

```
var array = [1, 2, 3];
array.push(4); // ahora array = [1, 2, 3, 4]
```

Elimina primer elemento de array

shift(), elimina el primer elemento del array y lo devuelve. El array original se ve modificado y su longitud disminuida en 1 elemento.

```
var array = [1, 2, 3];
var primero = array.shift(); // ahora array = [2, 3], primero = 1
```

Programación básica

unshift(), añade un elemento al principio del array. El array original se modifica y aumenta su longitud en 1 elemento.
var array = [1, 2, 3];
array.unshift(0); *// ahora array = [0, 1, 2, 3]*

reverse(), modifica un array colocando sus elementos en el orden inverso a su posición original:
var array = [1, 2, 3];
array.reverse(); *// ahora array = [3, 2, 1]*

array.**slice(start, end)**

The slice() method returns the selected elements in an array, as a new array object.
Rompe relacion array con su array padre. Evita que este ligadas en los cambios

Programación básica

Funciones útiles para números

Infinity, hace referencia a un valor numérico infinito y positivo (también existe el valor –Infinity para los infinitos negativos)

Para saber si es multiplo de.
Sabes si en entero o no

```
var numero1 = 10;
var numero2 = 0;
alert(numero1/numero2); // se muestra el valor Infinity
```

toFixed(digitos), devuelve el número original con tantos decimales como los indicados por el parámetro digitos y realiza los redondeos necesarios.

```
var numero1 = 4564.34567;
numero1.toFixed(2); // 4564.35
numero1.toFixed(6); // 4564.345670
numero1.toFixed(); // 4564
```

Programación básica

NaN, (del inglés, "*Not a Number*") se emplea para indicar un valor numérico no definido.

Pasa texto a numero:	**parseInt**(string, radix)
Pasa numero a texto:	number.**toString**(radix)

```
var numero1 = 0;
var numero2 = 0;
alert(numero1/numero2); // se muestra el valor NaN“
```

isNaN(), permite proteger a la aplicación de posibles valores numéricos no definidos

```
var numero1 = 0;
var numero2 = 0;
if(isNaN(numero1/numero2)) {
    alert("La división no está definida para los números indicados");
}else {
    alert("La división es igual a => " + numero1/numero2);
}
```

Programación básica

Funciones

- Una **función e**s un trozo de código que ejecuta una tarea determinada. Esta tarea está constituida por una serie de instrucciones.

- Las funciones evitan al programador tener que repetir el mismo código varias veces y hacen más inteligible el funcionamiento de un programa.

Programación básica

Las funciones se definen mediante la palabra reservada function, seguida del nombre de la función.

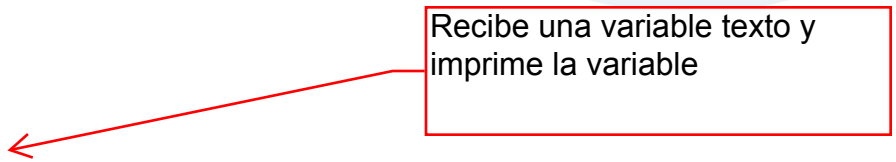
```
function nombre_funcion() {
...
}
```

El nombre de la función se utiliza para *llamar o invocar* a esa función cuando sea necesario. los símbolos { y } se utilizan para encerrar todas las instrucciones que pertenecen a la función

Javascript

- **Funciones**
  - Normalmente, se declaran las funciones Javascript en **la cabecera** de la página, es decir, englobada dentro de las etiquetas **<head>**. Su sintaxis es la siguiente:

```
<head>
...
<script type='text/javascript'>
function  mostrar_un_alert(texto){
      alert(texto);
}
...
</script>
...</head>
```



Javascript

- **Funciones**
  - Para llamar a las funciones, lo haremos de la misma manera que con las funciones ya definidas, como alert:

    <td width="500" onClick="mostrar_un_alert('Bienvenidos');">

  - Al igual que hacíamos con los estilos, es muy común **guardar funciones javascript en archivos independientes**. Estos archivos tendran extension **.js**, y los añadiremos en nuestras páginas con el siguiente código en la cabecera.

<script type="text/javascript" src="funciones.js">
</script>

Programación básica

```
var resultado;

var numero1 = 3;
var numero2 = 5;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);

numero1 = 10;
numero2 = 7;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);

numero1 = 5;
numero2 = 8;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
...
```

```
function suma_y_muestra() {
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
}

var resultado;

var numero1 = 3;
var numero2 = 5;

suma_y_muestra();

numero1 = 10;
numero2 = 7;

suma_y_muestra();

numero1 = 5;
numero2 = 8;

suma_y_muestra();
...
```

Programación básica

La mayoría de funciones deben acceder al valor de algunas variables para producir sus resultados. Las variables que necesitan las funciones se llaman *argumentos*.

Los argumentos se indican dentro de los paréntesis que van detrás del nombre de la función y se separan con una coma (,).

function nombre_funcion(**arg1, arg2, …, argn**) {
…
}

Al invocar la función, el número de argumentos que se pasa a lafunción debería ser el mismo que el número de argumentos que ha indicado la función. El orden de los argumentos es fundamental

Programación básica

```
var resultado;

var numero1 = 3;
var numero2 = 5;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);

numero1 = 10;
numero2 = 7;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);

numero1 = 5;
numero2 = 8;

// Se suman los números y se muestra el resultado
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
...
```

```
function suma_y_muestra(numero1, numero2) {
resultado = numero1 + numero2;
alert("El resultado es " + resultado);
}

var resultado;

var numero_1 = 3;
var numero_2 = 5;

suma_y_muestra(numero_1, numero_2);

numero1 = 10;
numero2 = 7;

suma_y_muestra(numero1, numero2);

~~numero1 = 5;~~
~~numero2 = 8;~~

suma_y_muestra(5, 8);
```

Los nombres de las variables de la llamada no tienen que tener el mismo nombre.

Programación básica

Las funciones no solamente puede recibir variables y datos, sino que también pueden devolver los valores que han calculado. Para devolver valores dentro de una función, se utiliza la palabra reservada **return**.

```
function calculaPrecioTotal(precio) {
    var impuestos = 1.16;
    var gastosEnvio = 10;
    var precioTotal = ( precio * impuestos ) + gastosEnvio;
    return precioTotal;
}
// El valor devuelto por la función, se guarda en una variable
var precioTotal = calculaPrecioTotal(23.34);

// Seguir trabajando con la variable "precioTotal"
```

llamada a la funcion, dando la cifra 23.34 como variable precio

No tiene por que tener este nombre como variable

Todas las instrucciones que se incluyen después de un return se ignoran y por ese motivo la instrucción return suele ser la última de la mayoría de funciones.

Programación básica

Ámbito de las variables

Variable local solamente está definida dentro de la función

la variable solo se puede usar dentro de la funcion.
Si se usa fuera no funciona

Ejemplo:

```
function creaMensaje() {
    var mensaje = “Mensaje de prueba”;
}
creaMensaje();
alert(mensaje);
```

Cualquier instrucción que se encuentre dentro de la función puede hacer uso de esa variable, pero todas las instrucciones que se encuentren en otras funciones o fuera de cualquier función no tendrán definida la variable mensaje.

Programación básica

Ámbito de las variables

Variable global, está definida en cualquier punto del programa (incluso dentro de cualquier función).

La variable se puede usar fuera y dentro de la funcion

```
var mensaje = “Mensaje de prueba”;
function muestraMensaje() {
alert(mensaje);
}
```

¿Qué sucede si una función define una variable local con el mismo nombre que una variable global que ya existe?

Programación básica

```
var mensaje = "gana la de fuera";

function muestraMensaje() {
var mensaje = "gana la de dentro";
alert(mensaje);
}

alert(mensaje);
muestraMensaje();
alert(mensaje);

//resultados
gana la de fuera
gana la de dentro
gana la de fuera
```

Variable global

Variable local. Al definirse dentro de la funcion. solo sirve para dentro la funcion

```
var mensaje = "gana la de fuera";

function muestraMensaje() {
mensaje = "gana la de dentro";
alert(mensaje);
}

alert(mensaje);
muestraMensaje();
alert(mensaje);

//resultados
gana la de fuera
gana la de dentro
gana la de dentro
```

Variable global

se da un valor. Pero no se define la funcion. Por lo que manda la varaiable global

Ejercicios

- Coged un formulario de ejemplo, cread una función que pregunte al usuario si está seguro de enviar el formulario, y en caso afirmativo, que se envíe.
  - La función para enviar un formulario es:

document.nombre_formulario.submit();

* Ejercicio 8, 9 y 10

DOM

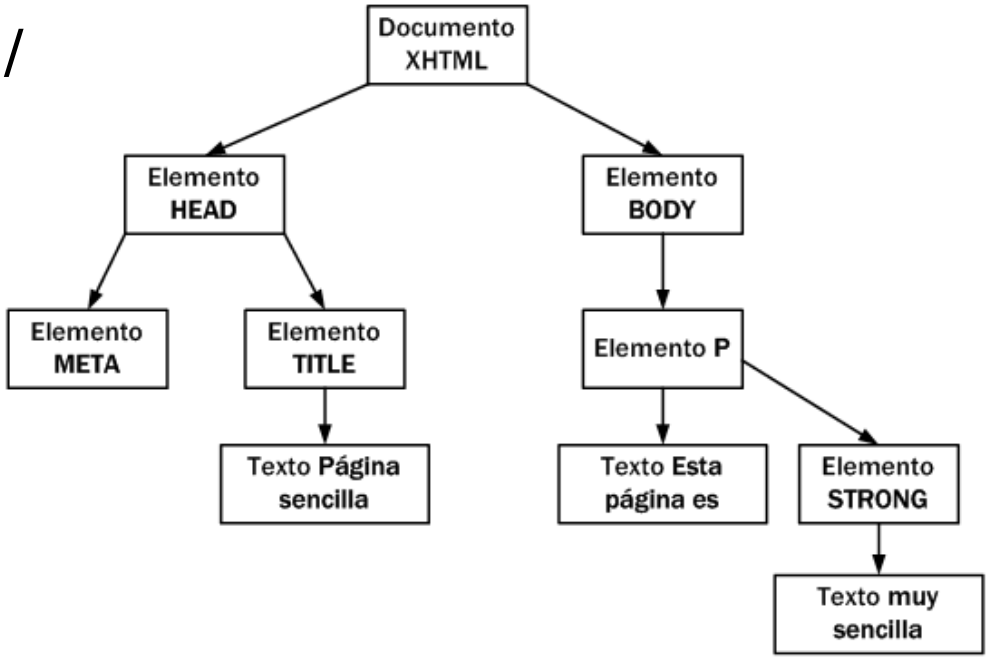
Document Object Model o DOM permite a los programadores web acceder y manipular las páginas XHTML como si fueran documentos XML.

DOM transforma todos los documentos XHTML en un conjunto de elementos llamados *nodos*, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos. Por su aspecto, la unión de todos los nodos se llama "*árbol de nodos*".

El árbol DOM se construye completamente, después de que la página XHTML se cargue por completo.

DOM

```
<!DOCTYPE html>
<html
xmlns="http://www.w3.org/
1999/xhtml">
<head>
<meta http-equiv="Content-
Type" content="text/html;
charset=iso-8859-1" />
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es
<strong>muy
sencilla</strong></p>
</body>
</html>
```



Árbol de nodos generado automáticamente por DOM a partir del código XHTML de la página

DOM

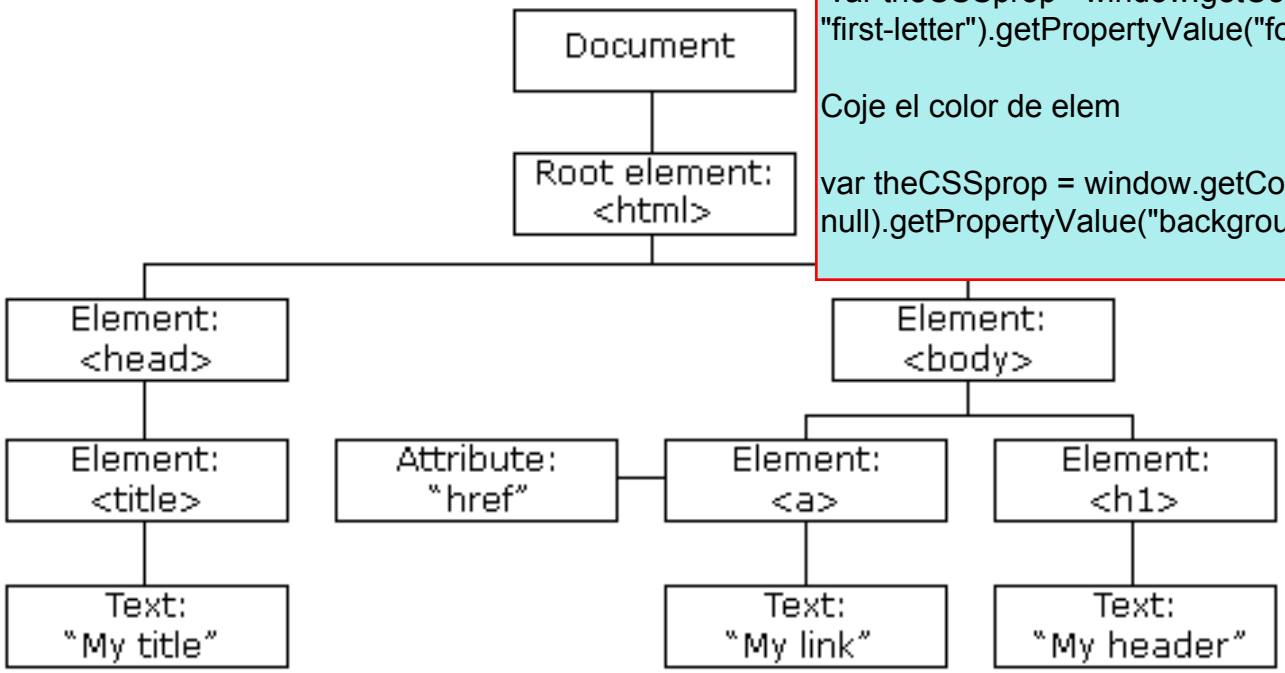
getComputedStyle
getPropertyValue

Coje de elem el estilo de la primera letra y mira que tamaño tiene

```
var theCSSprop =window.getComputedStyle(elem, "first-letter").getPropertyValue("font-size");
```

Coje el color de elem

```
var theCSSprop = window.getComputedStyle(elem, null).getPropertyValue("background-color")
```



DOM

Con el object model, JavaScript obtiene todo el poder que necesita para crear HTML dinámico:

- JavaScript puede cambiar todos los elementos HTML.
- JavaScript puede cambiar todos los atributos HTML.
- JavaScript puede cambiar todos los estilos CSS.
- JavaScript puede eliminar elementos y atributos de la página.
- JavaScript puede añadir nuevos elementos y atributos.
- JavaScript puede crear nuevos eventos en la página HTML.

DOM

- **document**

Objeto javascript que hace referencia al **contenido total** de nuestra página. Todas sus propiedades las podemos encontrar en el siguiente enlace:

http://www.w3schools.com/jsref/dom_obj_document.asp

DOM

Las funciones DOM se utilizan para acceder de forma directa a cualquier nodo del árbol.

getElementsByTagName(nombreEtiqueta) obtiene todos los elementos de la página XHTML cuya etiqueta sea igual que el parámetro que se le pasa a la función.

```
var parrafos = document.getElementsByTagName("p");
var primerParrafo = parrafos[0];
```

Coje todos los "p" y lo pone en una array

getElementsByClassName(nombreClase) devuelve una lista de todos los elementos de la clase especificada.

los CLASS se pueden usar mas de una vez en HTML

```
var parrafoEspecial =
document.getElementsByClassName("especial");
<p class="especial">...</p>
```

Coje todos los class "especial" y lo pone en una array

DOM

los ID solo se pueden usar una vez

getElementById(nombreAtributo) devuelve el elemento XHTML cuyo atributo id coincide con el parámetro indicado en la función.

NO devuelve una array, porque solo puede haber uno

```
<div id="cabecera">
    <p>Primer párrafo de la cabecera</p>
    <p>Segundo párrafo de la cabecera</p>
</div>
<p id="demo"></p>

var cabe = document.getElementById("cabecera");
var parrafos = cabe.getElementsByTagName("p");
document.getElementById("demo").innerHTML =
 parrafos[0].innerHTML;
```

DOM

querySelectorAll(nombreEtiqueta.selector) devuelve los elementos HTML que coincidan con el nombre de la etiqueta y el valor del selector.

```
document.querySelectorAll("h2, div, span")
document.querySelectorAll("div > p")   - Todos los P que estan dentro de un Div
document.querySelectorAll("*")          - Selecciona todo
```

```
<p class="intro">Primer párrafo.</p>
<p class="intro">Segundo párrafo.</p>
<p id="demo"></p>

<script>
    var x = document.querySelectorAll("p.intro");
    document.getElementById("demo").innerHTML =
x[0].innerHTML;
</script>
```

```
<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

document.querySelectorAll("a[href='http://www.wikipedia.org']");
```

```
<a href="https://www.w3schools.com">w3schools.com</a>
<a href="http://www.disney.com" target="_blank">disney.com</a>
<a href="http://www.wikipedia.org" target="_top">wikipedia.org</a>

 var x = document.querySelectorAll("a[target]"); -llamaria a: a target
```

DOM

Acceso directo a los atributos XHTML

Los atributos XHTML de los elementos de la página y las propiedades CSS se transforman automáticamente en propiedades de los nodos. Para acceder a su valor, simplemente se indica el nombre del atributo XHTML detrás del nombre del nodo.

```
var enlace = document.getElementById("enlace");
alert(enlace.href); // muestra http://www...com
<a id="enlace" href="http://www...com">Enlace</a>
```

Javascript

HTML

```
var imagen = document.getElementById("imagen");
alert(imagen.style.margin);
<img id="imagen" style="margin:0; border:0;" src="logo.png" />
```

DOM

La transformación del nombre de las propiedades CSS compuestas consiste en eliminar todos los guiones medios (-) y escribir en mayúscula la letra siguiente a cada guion medio.
line-height se transforma en **lineHeight**
border-top-style se transforma en **borderTopStyle**

Ejemplo de utilización:
document.getElementById("myDiv").style.borderTopStyle = "dotted";

Es lo mismo que poner:

var nueva = document.getElementById("myDiv");
nueva. style.borderTopStyle = "dotted";

DOM utiliza el nombre className para acceder al atributo class de XHTML
var parrafo = document.getElementById("parrafo");
alert(parrafo.className); // muestra "normal“

...

* Ejercicio 11

DOM

Se puede almacenar en una variable:
ventana = window.open ()

funcion cierra (){
    ventana.close()
}

- **window.location**
  - Función del objeto **window** que nos permite cambiar la dirección de la página: (ver más funciones en http://www.w3schools.com/js/js_window_location.asp)

window.location="http://www.paginanueva.com";

- **window.open**

Remplaza ID por la informacion del texto:
HTML -->
JS --> document.getElementById("informacion").**innerHTML** = "Pinchado el enlace";

  - Función similar a la anterior, pero que abre la dirección en una nueva ventana:

window.open("pagina.html","nombrepagina","width=500,height=200");

  - Fuente de 'pop-ups' publicitarios, algunos navegadores **la bloquean**.

Ejercicios

1. Mostrad un mensaje de tipo **confirm**, y dependiendo de la respuesta, redirigir al usuario a una página u otra. Podéis basaros en una página principal que abra el html de la lista horizontal o vertical según la respuesta del confirm (la lista horizontal y vertical la tenemos en los ejercicios 7.1 y 7.2 de CSS).

2. Modificad una página existente para que, en cuanto la abramos, se abra con otro contenido.

DOM

Creación de nuevos elementos HTML (nodos)

1. Creación de un nuevo elemento (nodo) <p>
var para = document.createElement("p");

2. Añadir texto al elemento <p>, primero debemos crear un nodo de texto para después añadirlo al elemento <p>
var node = document.createTextNode("This is a new paragraph.");
para.appendChild(node);

3. Finalmente tenemos que añadir el elemento nuevo a uno existente. Primero buscamos el elemento y luego lo añadimos.
var element = document.getElementById("div1");
element.appendChild(para);

DOM

Código de ejemplo:

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph.</p>
</div>

<script>
  var para = document.createElement("p");
  var node = document.createTextNode("This is new.");
  para.appendChild(node);
  var element = document.getElementById("div1");
  element.appendChild(para);
</script>
```

DOM

Eliminación de elementos HTML existentes (nodos)

```
<div id="div1">
<p id="p1">This is a paragraph.</p>
<p id="p2">This is another paragraph
</div>
```

Eliminia "li" al pulsarlos:

```
function elimina(elemento){
   var elli =  elemento.parentElement;
   var elul =  elli.parentElement;
   elul.removeChild(elli);
}

<ul>
  <li><a onclick="elimina(this);" href="#"> 1 </a></li>
  <li><a onclick="elimina(this);" href="#"> 2 </a></li>
  <li><a onclick="elimina(this);" href="#"> 3 </a></li>
</ul>
```

Si queremos borrar el elemento con id

1. Encontrar el elemento padre:

```
var parent = document.getElementById("div1");
```

2. Encontrar el elemento <p> con id="p1" y borrarlo des del padre:

```
var child = document.getElementById("p1");
parent.removeChild(child);
```

DOM

Eliminación de elementos HTML existentes (nodos)

Otra forma de hacer lo mismo, sería buscar el elemento a borrar y utilizanzo su propiedad parentNode encontrar el padre:

```
var child = document.getElementById("p1");
child.parentNode.removeChild(child);
```

Para ver otras opciones:
http://www.w3schools.com/js/js_htmldom_nodes.asp

Javascript

- **Eventos en Javascript**
  - Un evento se produce cada vez que un usuario interactúa de alguna manera con una página. A estas acciones, se les puede asociar código Javascript, que se “activará” en el momento en que se detecten
  - Los eventos son el modo que tenemos de controlar las acciones de los usuarios de la página, y en base a ellos definiremos un comportamiento u otro mediante el uso de funciones Javascript.

Javascript

- **Manejadores de eventos**
  - Los manejadores de eventos se sitúan en el mismo lugar que los **atributos** de las diferentes etiquetas, y hay tantos como posibles comportamientos puede tener el usuario sobre dicha etiqueta.

    <td width="500" onClick="alert('se ha hecho click');">
    …
    </td>

  - A **todos** los tipos de etiqueta se les puede añadir manejadores de eventos, aunque algunas estarán más indicadas para ello que otras, sobretodo los campos de formulario.

Lista eventos completa:
https://www.w3schools.com/jsref/dom_obj_event.asp

Javascript

Evento	Descripción	Elementos para los que está definido
onblur	Deseleccionar el elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onchange	Deseleccionar un elemento que se ha modificado	<input>, <select>, <textarea>
onclick	Pinchar y soltar el ratón	Todos los elementos
ondblclick	Pinchar dos veces seguidas con el ratón	Todos los elementos
onfocus	Seleccionar un elemento	<button>, <input>, <label>, <select>, <textarea>, <body>
onkeydown	Pulsar una tecla (sin soltar)	Elementos de formulario y <body>
onkeypress	Pulsar una tecla	Elementos de formulario y <body>
onkeyup	Soltar una tecla pulsada	Elementos de formulario y <body>
onload	La página se ha cargado completamente	<body>

Javascript

Evento	Descripción	Elementos para los que está definido
onmousedown	Pulsar (sin soltar) un botón del ratón	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseout	El ratón *"sale"* del elemento (pasa por encima de otro elemento)	Todos los elementos
onmouseover	El ratón *"entra"* en el elemento (pasa por encima del elemento)	Todos los elementos
onmouseup	Soltar el botón que estaba pulsado en el ratón	Todos los elementos
onreset	Inicializar el formulario (borrar todos sus datos)	<form>
onresize	Se ha modificado el tamaño de la ventana del navegador	<body>
onselect	Seleccionar un texto	<input>, <textarea>
onsubmit	Enviar el formulario	<form>
onunload	Se abandona la página (por ejemplo al cerrar el navegador)	<body>

Javascript

- **Manejadores más comunes**
  - **onClick / onDblClick**: reacciona al hacer clic (o doble clic en el segundo caso) con el ratón sobre el elemento
  - **onMouseOver / onMouseOut:** reaccionan al pasar el ratón por encima del elemento y al quitarlo, respectivamente
  - **onMouseDown / onMouseUp:** activado al pulsar el botón del ratón y al soltarlo, respectivamente
  - **onMouseMove:** activado al mover el cursor por la página

Javascript

- **Manejadores más comunes**
  - **onKeyUp / onKeyDown**: sucede al apretar una tecla y al soltarla, respectivamente
  - **onKeyPress:** activado al dejar pulsada una tecla durante un tiempo
  - **onChange:** producido al modificar un elemento de formulario
  - **onLoad / onUnload:** provocado por la carga del elemento y su cierre, respectivamente. Muy común su uso en la etiqueta <body>

Javascript

- **Manejadores más comunes**
  - **onMove / onResize**: sucede al mover la ventana del navegador o al cambiar su tamaño
  - **onSubmit / onReset:** activado al enviar un formulario o al reiniciarlo
  - **onFocus / onBlur:** producido al tener el elemento el foco de la aplicación (seleccinado por el ratón), y al perderlo

Javascript

La variable **this**, hace referencia al elemento XHTML que ha provocado el evento.
Ejemplo: modificar el color de los bordes

```
<div id="contenidos" style="width:150px; height:60px; border:thin solid silver"
onmouseover="document.getElementById('contenidos').style.borderColor='black';"
onmouseout="document.getElementById('contenidos').style.borderColor='silver';">
      Sección de contenidos...
</div>

<div id="contenidos" style="width:150px; height:60px; border:thin solid silver"
onmouseover="this.style.borderColor='black';"
onmouseout="this.style.borderColor='silver';">
      Sección de contenidos...
</div>
```

Ejercicios

1. Modificad un formulario de manera que, antes de enviarse, aparezca un **alert** advirtiendo al usuario sobre la acción.

2. Cread otra página en la que, al pasar por encima de una imagen, se muestre un **alert** advirtiendo que está prohibida su copia.

3. Modificad un formulario con campos de texto, y añadid a algunos de ellos **alerts** cuando se **cambie** su contenido, y para cuando **pierdan el foco**.

4. Por último, hacer que la página muestre un mensaje de despedida cuando **la cerremos**.

```
window.onbeforeunload =function(){
    return "gracias poer su visita";
```

Ejercicios

- Añadid un botón al formulario que ponga todos los campos de texto **de color rojo**.
  - Para modificar el estilo de un campo:

document.n_form.n_campo.className="nombre_estilo";
document.getElementById("n_campo").className="nombre_estilo";

Ejercicios

- Esta vez, haced que el formulario compruebe si el campo «Email» está rellenado (no utilizar el atributo required). Si está vacío, que muestre una advertencia, ponga el cursor sobre él y cambie el color de fondo a rojo (utilizar backgroundColor).
  - La función para obtener el valor de un campo de texto:

document.getElementById("n_campo").value

  - La función para que un campo obtenga el foco es:

document.nombre_formulario.nombre_campo.focus();

Ejercicios

- Aquí Cread un **nuevo campo para la edad**, y añadid una función que compruebe que se escribe un **número** en ese campo cada vez que se modifique su valor.
- En caso contrario, **se borrará el contenido del campo, se pondrá en rojo**, **y el cursor se situará sobre él**. Cuando se haya escrito un valor correcto de nuevo, se volverá al color normal.

  - Para comprobar que una cadena es no numérica, usamos la función **isNaN**. Esta función devuelve un booleano, es decir, **true** o **false**, y por tanto podemos utilizarla como condición en un *if* o un *while*:

isNaN('cadena_de_texto');

EventListener

El método addEventListener() relaciona un evento a un elemento.
Este método no sobreescribe los eventos que pueda tener el elemento.
A parte de añadir un evento a un elemento nos permite:

- Añadir más de un evento del mismo tipo a un mismo elemento (por ejemplo dos eventos click)
- Añadir un evento a un objeto DOM.
- Facilita el control de como reaccionan los eventos al bubbling https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture

Cuando usamos el método addEventListener(), JavaScript queda separado del HTML construido, eso nos permite una mejor lectura del HTML y tener el control de los eventos fuera del HTML.
También podemos eliminar un EventListener usando el método removeEventListener().

EventListener

Sintaxis:

element.addEventListener(event, function, useCapture);

- El primer parámetro es el tipo de evento (como "clic" o "mousedown").
- El segundo parámetro es la función que queremos llamar cuando se produce el evento.
- El tercer parámetro es un valor booleano que especifica si se debe utilizar el bubbling evento o captura de eventos. Este parámetro es opcional.

Nota: no se utiliza el prefijo "on" para el evento: usar "click" en lugar de "onclick".

Reemplaza al ONCLICK

JS

```
mifuncion(){
   alert("hola");
}

document.getElementById("par").addEventListener("click",mifuncion);
```

HTML

```
<p id="par">hola</p>
```

EventListener

Ejemplos:

- Agregar un controlador de eventos a un Elemento

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add2

- Añadir Muchos controladores de eventos al mismo elemento

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_add_many2

- Agregar un controlador de eventos al objeto de ventana

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_dom

- Paso de parámetros

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_parameters

- Evento bubbling o captura de eventos

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener_usecapture

- Eliminar evento

https://www.w3schools.com/js/tryit.asp?filename=tryjs_addeventlistener

Formularios

El objeto **document** permite acceder directamente a cualquier formulario mediante su atributo **name**:

```
var formularioPrincipal = document.formulario;
var formularioSecundario = document.otro_formulario;
```

```
<form name="formulario" >
  ...
</form>

<form name="otro_formulario" >
  ...
</form>
```

Formularios

Los elementos de los formularios también se pueden acceder directamente mediante su atributo name:

```
var formularioPrincipal = document.formulario;
var primerElemento = document.formulario.elemento;
<form name="formulario">
  <input type="text" name="elemento" />
</form>
```

También se puede acceder a los formularios y a sus elementos utilizando las funciones DOM de acceso directo a los nodos

```
var formularioPrincipal = document.getElementById("formulario");
var primerElemento = document.getElementById("elemento");
<form name="formulario" id="formulario" >
  <input type="text" name="elemento" id="elemento" />
</form>
```

Formularios

Las propiedades de los elementos de un formulario:

- **type**: indica el tipo de elemento que se trata.

- **form**: es una referencia directa al formulario al que pertenece el elemento.
  `document.getElementById("id_del_elemento").form`

- **name**: obtiene el valor del atributo name de XHTML. Solamente se puede leer su valor, por lo que no se puede modificar.

- **value**: permite leer y modificar el valor del atributo value de XHTML. Para los campos de texto (<input type="text"> y <textarea>) obtiene el texto que ha escrito el usuario. Para los botones obtiene el texto que se muestra en el botón.

Formularios

Los eventos más utilizados en el manejo de los formularios:

- **onclick**: evento que se produce cuando se pincha con el ratón sobre un elemento.

- **onchange**: evento que se produce cuando el usuario cambia el valor de un elemento de texto (<input type="text"> o <textarea>). También se produce cuando el usuario selecciona una opción en una lista desplegable (<select>).

- **onfocus**: evento que se produce cuando el usuario selecciona un elemento del formulario.

- **onblur**: evento complementario de onfocus, ya que se produce cuando el usuario ha *deseleccionado* un elemento por haber seleccionado otro elemento del formulario.

Formularios

Obtener el valor de los campos de formulario:
Cuadro de texto y textarea

```
<input type="text" id="texto" />
var valor = document.getElementById("texto").value;

<textarea id="parrafo"></textarea>
var valor = document.getElementById("parrafo").value;
```

Formularios

Obtener el valor de los campos de formulario:
Radiobutton

```
<input type="radio" value="si" name="pregunta"          id="pregunta_si"/> SI
<input type="radio" value="no" name="pregunta" id="pregunta_no"/> NO
<input type="radio" value="nsnc" name="pregunta" id="pregunta_nsnc"/>
NS/NC

if (document.getElementById('pregunta_si').checked) {
      alert(" Elemento: " + document.getElementById('pregunta_si').value + "
seleccionado.";
else if (document.getElementById('pregunta_si').checked) {
      ...
}
```

Formularios

Obtener el valor de los campos de formulario:
Checkbox

```
<input type="checkbox" value="condiciones" name="condiciones" id="condiciones"/> He leído y acepto las condiciones
<input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/> He leído la política de privacidad

var elemento = document.getElementById("condiciones");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);

elemento = document.getElementById("privacidad");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);
```

Formularios

Obtener el valor de los campos de formulario:

Select

```
<select id="opciones" name="opciones">
  <option value=“a">Primer valor</option>
  <option value=“b">Segundo valor</option>
  <option value=“c">Tercer valor</option>
  <option value=“d">Cuarto valor</option>
</select>
```

options, es un array creado automáticamente por el navegador para cada lista desplegable y que contiene la referencia a todas las opciones de esa lista.

selectedIndex, cuando el usuario selecciona una opción, el navegador actualiza automáticamente el valor de esta propiedad, que guarda el índice de la opción seleccionada. El índice hace referencia al array options

Formularios

Obtener el valor de los campos de formulario:

Select

```
// Obtener la referencia a la lista
var lista = document.getElementById("opciones");

// Obtener el índice de la opción que se ha seleccionado
var indiceSeleccionado = lista.selectedIndex;
// Con el índice y el array "options", obtener la opción seleccionada
var opcionSeleccionada = lista.options[indiceSeleccionado];

// Obtener el valor y el texto de la opción seleccionada
var textoSeleccionado = opcionSeleccionada.text;
var valorSeleccionado = opcionSeleccionada.value;
```

Formularios

Establecer el foco en un elemento

Para asignar el foco a un elemento de XHTML, se utiliza la función focus():

```
document.getElementById("primero").focus();
```

```
<form id="formulario" action="#">
  <input type="text" id="primero" />
</form>
```

Formularios

Validación

Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario. En esta función, se comprueban si los valores que ha introducido el usuario cumplen las restricciones impuestas por la aplicación.

<form action="" method="" id="" name="" **onsubmit="return validacion()"**>
...
<input type="sumbit“ value="Enviar" />
</form>

Formularios

```
function validacion() {
  if (condicion que debe cumplir el primer campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  else if (condicion que debe cumplir el segundo campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  ...
  else if (condicion que debe cumplir el último campo del formulario) {
    // Si no se cumple la condicion...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  // Si el script ha llegado a este punto, todas las condiciones
  // se han cumplido, por lo que se devuelve el valor true
  return true;
}
```

Formularios

Validar un campo de texto obligatorio

Se trata de forzar al usuario a introducir un valor en un cuadro de texto o textarea en los que sea obligatorio.

```
valor = document.getElementById("campo").value;
if (valor.length == 0 || /^\s+$/.test(valor) ) {
  return false;
}
```

- La condición **valor.length == 0** obliga a que el texto introducido tenga una longitud superior a cero caracteres, esto es, que no sea un texto vacío
- La condición **(/^\s+$/.test(valor))** (expresión regular) obliga a que el valor introducido por el usuario no sólo esté formado por espacios en blanco.

Formularios

Validar un campo de texto con valores numéricos

Se trata de obligar al usuario a introducir un valor numérico en un cuadro de texto.

```
valor = document.getElementById("campo").value;
if( isNaN(valor) ) {
  return false;
}
```

Ejemplos:

```
isNaN(3);          // false
isNaN(3.3545);     // false
isNaN(+23.2);      // false
isNaN("-23.2");    // false
isNaN("23a");      // true
isNaN("23.43.54"); // true
```

Formularios

Validar que se ha seleccionado una opción de una lista

Se trata de obligar al usuario a seleccionar un elemento de una lista desplegable.

```
indice = document.getElementById("opciones").selectedIndex;
if( indice == -1 ) { //si ninguna opción es seleccionada la propiedad
                     //selectedIndex tendrá valor -1
  return false;
}
```

```
<select id="opciones" name="opciones">
  <option value="">- Selecciona un valor -</option>
  <option value="1">Primer valor</option>
  <option value="2">Segundo valor</option>
</select>
```

Formularios

Validar una dirección de email

Se trata de obligar al usuario a introducir una dirección de email con un formato válido. Por tanto, lo que se comprueba es que la dirección parezca válida, ya que no se comprueba si se trata de una cuenta de correo electrónico real y operativa.

```
valor = document.getElementById("campo").value;
if( !/^([\da-z_\.-]+)@([\da-z\.-]+)\.([a-z\.]{2,6})$/.test(valor)) ) {
  return false;
}
```

Formularios

Validar un DNI

Se trata de comprobar que el número proporcionado por el usuario se corresponde con un número válido de Documento Nacional de Identidad o DNI.

```
valor = document.getElementById("campo").value;
var letras = ['T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B', 'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E', 'T'];

if( !(/^\d{8}[A-Z]$/.test(valor)) ) {
  return false;
}
if(valor.charAt(8) != letras[(valor.substring(0, 8))%23]) {
  return false;
}
```

La validación no sólo debe comprobar que el número esté formado por ocho cifras y una letra, sino que también es necesario comprobar que la letra indicada es correcta para el número introducido.

Formularios

Validar un número de teléfono

```
valor = document.getElementById("campo").value;
if( !(/^\d{9}$/.test(valor)) ) {
  return false;
}
```

La condición **/^\d{9}$/** se basa en el uso de expresiones regulares, que comprueban si el valor indicado es una sucesión de nueve números consecutivos.

Formularios

Validar un número de teléfono
Expresiones regulares que se pueden utilizar para otros formatos de número de teléfono.

Número	Expresión regular	Formato
900900900	/^\d{9}$/	9 cifras seguidas
900-900-900	/^\d{3}-\d{3}-\d{3}$/	9 cifras agrupadas de 3 en 3 y separadas por guiones
900 900900	/^\d{3}\s\d{6}$/	9 cifras, las 3 primeras separadas por un espacio
900 90 09 00	/^\d{3}\s\d{2}\s\d{2}\s\d{2}$/	9 cifras, las 3 primeras separadas por un espacio, las siguientes agrupadas de 2 en 2
(900) 900900	/^\(\d{3}\)\s\d{6}$/	9 cifras, las 3 primeras encerradas por paréntesis y un espacio de separación respecto del resto
+34 900900900	/^\+\d{2,3}\s\d{9}$/	Prefijo internacional (+ seguido de 2 o 3 cifras), espacio en blanco y 9 cifras consecutivas

Formularios

Validar que un checkbox ha sido seleccionado

Si un elemento de tipo checkbox se debe seleccionar de forma obligatoria, JavaScript permite comprobarlo de forma muy sencilla:

```
elemento = document.getElementById("campo");
if( !elemento.checked ) {
  return false;
}
```

Si se trata de comprobar que todos los checkbox del formulario han sido seleccionados, es más fácil utilizar un bucle

Formularios

Validar que un radiobutton ha sido seleccionado
La comprobación que se realiza es que el usuario haya seleccionado algún radiobutton de los que forman un determinado grupo.

```
If (document.getElementById('pregunta_si').checked ||
document.getElementById('pregunta_no').checked){
      return true;
}else {
      return false;
}
```

Ejercicios

- Finalmente, cread una función Javascript que compruebe **todos los campos del formulario**, a excepción del de comentarios, que será opcional.

- Si encuentra un campo vacío, **mostrará un mensaje de alerta, pondrá el campo en rojo y situará el cursor sobre él.** En caso contrario, lo pondrá al color normal y comprobará el siguiente.

- Cuando todos hayan sido comprobados, se enviará finalmente el formulario.

Formularios

Evitar el envío duplicado de un formulario

```
<form id="formulario" action="#">
  ...
      <input type="button" value="Enviar" onclick="this.disabled=true;
this.value='Enviando...'; this.form.submit()" />
</form>
```

Cuando se pulsa sobre el botón de envío del formulario, se produce el evento onclick sobre el botón.

1. Se deshabilita el botón mediante la instrucción this.disabled = true.
2. Se cambia el mensaje que muestra el botón. Del original "Enviar" se pasa al más adecuado "Enviando..."
3. Se envía el formulario mediante la función submit() en la siguiente instrucción: this.form.submit().

Formularios

Limitar el tamaño de caracteres de un textarea

En los campos de formulario de tipo textarea no se puede limitar el máximo número de caracteres que se pueden introducir, de forma similar al atributo maxlength de los cuadros de texto normales.

Algunos eventos (como onkeypress, onclick y onsubmit) se puede evitar su comportamiento normal si se devuelve el valor false.

```
<textarea onkeypress="return false;"></textarea>
```

El navegador no ejecuta el comportamiento por defecto del evento, es decir, la tecla presionada no se transforma en ningún carácter dentro del textarea.

Sin embargo, si un evento devuelve el valor true, su comportamiento es el habitual:

```
<textarea onkeypress="return true;"></textarea>
```

Formularios

Limitar el tamaño de caracteres de un textarea

Para limitar el número de caracteres que se pueden escribir en un elemento de tipo textarea: se comprueba si se ha llegado al máximo número de caracteres permitido y en caso afirmativo se evita el comportamiento habitual del evento.

```
function limita(maximoCaracteres) {
  var elemento = document.getElementById("texto");
  if(elemento.value.length >= maximoCaracteres ) {
    return false;
  }
  else {
    return true;
  }
}

<textarea id="texto" onkeypress="return limita(100);"></textarea>
```

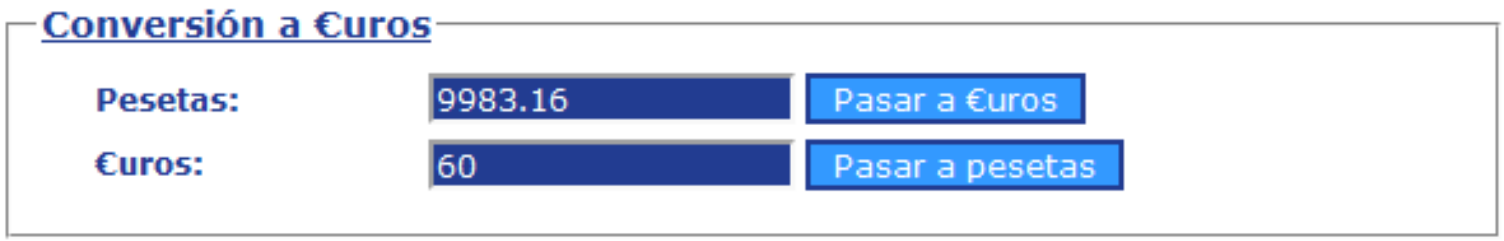
Formularios

Ejemplo

Mejorar el ejemplo anterior indicando en todo momento al usuario el número de caracteres que aún puede escribir.

Ejercicios

- **Conversor de Euros**
  - Cread un formulario que, mediante Javascript, sea capaz de traducir un importe de pesetas a euros, y viceversa. La aplicación deberá comprobar que el campo no está vacío, y que contiene un valor numérico.



* Ejercicio 14

Ejercicios

1. Cread una página como la siguiente. Hacer clic en los botones implicará modificar el título de la página, mientras que pasar por encima de las áreas de color inferiores hará que cambie el color de fondo de la página.

- **document.body.style.backgroundColor:** color de fondo de la etiqueta <body>
- **document.title:** título de la página, contenido de <title>

Javascript

- **document.*images***
  - Para acceder a las propiedades de una imagen, lo haremos mediante el objeto ***images***, de document. Así, para cambiar el atributo 'src' de una imagen, por ejemplo, lo haríamos así:

    document.images.nombre_imagen.src='img/img1.jpg';

- **Modificar contenido de una etiqueta div**
  - Para modificar el contenido de un div, tendremos que acceder al mismo mediante la siguiente fórmula:

document.getElementById('id_div').innerHTML='nuevo contenido';

(la fórmula **getElementById(..)** servirá también para **todos los elementos** que ya hemos visto)

Ejercicios

- **Visor de imágenes**
  - Cread una página con miniaturas de imágenes, y una imagen más grande con un texto descriptivo. Cuando se haga clic sobre cada una de las miniaturas, se cambiará la imagen grande y el texto que la describe por el de ésta. Cuando se carge la página se cargará la primera imagen.

Obteniendo información del event

JavaScript permite obtener información sobre el ratón y el teclado mediante un objeto especial llamado event.

En los navegadores tipo Internet Explorer, el objeto event se obtiene directamente mediante:

```
var evento = window.event;
```

En el resto de navegadores, el objeto event se obtiene mágicamente a partir del argumento que el navegador crea automáticamente:

```
function manejadorEventos(elEvento) {
    var evento = elEvento;
}
```

La forma correcta de obtener el objeto event en cualquier navegador:

```
function manejadorEventos(elEvento) {
    var evento = elEvento|| window.event;
}
```

Obteniendo información del event

Información sobre el evento

La propiedad type indica el tipo de evento producido, lo que es útil cuando una misma función se utiliza para manejar varios eventos:

```
var tipo = evento.type;
```

La propiedad type devuelve el tipo de evento producido, que es igual al nombre del evento pero sin el prefijo on.

Obteniendo información del event

Ejemplo: Ejemplo en el que se resaltaba una sección de contenidos al pasar el ratón por encima:

```
function resalta(elEvento) {
	var evento = elEvento || window.event;
	switch(evento.type) {
		case 'mouseover':
			this.style.borderColor = 'black';
			break;
		case 'mouseout':
			this.style.borderColor = 'silver';
			break;
	}
}
window.onload = function() {
	document.getElementById("seccion").onmouseover = resalta(event);
	document.getElementById("seccion").onmouseout = resalta(event);
}
```

Obteniendo información del event

Información sobre los eventos de teclado

Existen tres eventos diferentes para las pulsaciones de las teclas (onkeyup, onkeypress y onkeydown).

- **Onkeydown**: se corresponde con el hecho de pulsar una tecla y no soltarla.
- **Onkeypress**: es la propia pulsación de la tecla.
- **Onkeyup**: hace referencia al hecho de soltar una tecla que estaba pulsada.

Las propiedades diferentes de todos los eventos de teclado: **keyCode**, **charCode**.

Para convertir el código de un carácter al carácter que representa la tecla que se ha pulsado, se utiliza la función **String.fromCharCode().**

Obteniendo información del event

Ejemplo

El siguiente código de JavaScript permite obtener de forma correcta en cualquier navegador el carácter correspondiente a la tecla pulsada:

```
function manejador(elEvento) {
    var evento = elEvento || window.event;
    var caracter = evento.charCode || evento.keyCode;
    alert("El carácter pulsado es: " +
String.fromCharCode(caracter));
}
document.onkeypress = manejador(event);
```

Obteniendo información del event

Información sobre los eventos de teclado

Existen dos tipos de teclas:

- Las teclas normales (como letras, números y símbolos normales).
- Las teclas especiales (como ENTER, Alt, Shift, etc.)

Las propiedades **altKey**, **ctrlKey** y **shiftKey** almacenan un valor booleano que indica si alguna de esas teclas estaba pulsada al producirse el evento del teclado.

```
if(evento.altKey) {
    alert('Estaba pulsada la tecla ALT');
}
```

Obteniendo información del event

Ejercicio:
Restringir los caracteres permitidos en un cuadro de texto
Bloquear algunos caracteres determinados en un cuadro de texto utilizando el evento onkeypress.
Cuando se pulsa una tecla, se comprueba si el carácter de esa tecla se encuentra dentro de los caracteres permitidos para ese elemento <input>.
Ejemplo:

```
// Sólo números
<input type="text" id="texto" onkeypress="return permite(event, 'num')" />

// Sólo letras
<input type="text" id="texto" onkeypress="return permite(event, 'car')" />

// Sólo letras o números
<input type="text" id="texto" onkeypress="return permite(event,
'num_car')" />
```

Obteniendo información del event

```
function permite(elEvento, permitidos) {
  // Variables que definen los caracteres permitidos
  var numeros = "0123456789";
  var caracteres = " abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
  var numeros_caracteres = numeros + caracteres;
```

// Seleccionar los caracteres a partir del parámetro de la función (numeros, caracteres o numeros_caracteres)

...

// Obtener la tecla pulsada

...

// Comprobar si la tecla pulsada se encuentra en los caracteres permitidos

...

}

Obteniendo información del event

Ejercicio:
Recupera el formulario de la página 112 y realiza los cambios que sean necesarios para que una vez se hayan superado el número máximo de carácteres en el texarea nos permita pulsar las teclas Backspace, Supr. y las flechas horizontales (8 = BackSpace, 46 = Supr, 37 = flecha izquierda, 39 = flecha derecha)
var teclas_especiales = [8, 37, 39, 46];

Obteniendo información del event

Información sobre los eventos de ratón

Es posible obtener la posición del ratón respecto de la pantalla del ordenador:

```
var coordenadaX = evento.screenX;
var coordenadaY = evento.screenY;
```

Respecto la esquina de la pantalla

respecto de la ventana del navegador:

```
var coordenadaX = evento.clientX;
var coordenadaY = evento.clientY;
```

Respecto donde se ha pulsado

y respecto de la propia página HTML (que se utiliza cuando el usuario ha hecho *scroll* sobre la página). Internet Explorer no proporciona estas coordenadas de forma directa.

Obteniendo información del event

Información sobre los eventos de ratón

Es necesario detectar si el navegador es de tipo Internet Explorer y en caso afirmativo realizar un cálculo sencillo:

```
// Detectar si el navegador es Internet Explorer
var ie = navigator.userAgent.toLowerCase().indexOf('msie')!=-1;

if(ie){
    coordenadaX = evento.clientX + document.body.scrollLeft;
    coordenadaY = evento.clientY + document.body.scrollTop;
}else{
    coordenadaX = evento.pageX;
    coordenadaY = evento.pageY;
}
```

* Ejercicio 15

Relojes

Para crear y mostrar un reloj, se debe utilizar el objeto interno **Date()** para crear fechas/horas.

```
Var fechaHora = new Date();

document.getElementById("reloj").innerHTML = fechaHora;

<div id="reloj" />
```

Mon May 04 2009 13:36:10 GMT+0200

```
var fechaHora = new Date();

var horas = fechaHora.getHours();
var minutos = fechaHora.getMinutes();
var segundos = fechaHora.getSeconds();

document.getElementById("reloj").innerHTML = horas+':'+minutos+':'+segundos;

<div id="reloj" />
```

20:9:21

Relojes

Si la hora, minuto o segundo son menores que 10, JavaScript no añade el 0 por delante

```
if(horas < 10) { horas = '0' + horas; }
if(minutos < 10) { minutos = '0' + minutos; }
if(segundos < 10) { segundos = '0' + segundos; }
```

Si se quiere mostrar el reloj con un formato de 12 horas en vez de 24:

```
var sufijo = ' am';
if(horas > 12) {
  horas = horas - 12;
  sufijo = ' pm';
}
```

Relojes

Para completar el reloj, sólo falta que se actualice su valor cada segundo.
setInterval(nombreFuncion, milisegundos);

La función **setInterval** () permite ejecutar una función (nombreFuncion) infinitas veces de forma periódica con un lapso de tiempo entre ejecuciones de tantos milisegundos como se hayan establecido.

setTimeOut

The setTimeout() method calls a function or evaluates an expression after a specified number of milliseconds.