JavaScript

Índice

[Conceptos básicos de JavaScript 3](#_Toc63075952)

[Historia 3](#_Toc63075953)

[Funcionamiento 4](#_Toc63075954)

[Sintaxis 4](#_Toc63075955)

[Elementos básicos del lenguaje 4](#_Toc63075956)

[Variables 5](#_Toc63075957)

[Tipo de datos 5](#_Toc63075958)

[Constantes 6](#_Toc63075959)

[Operadores 6](#_Toc63075960)

[Tabla de ejercicios 8](#_Toc63075961)

[Funciones y estructuras de control 9](#_Toc63075962)

[Estructuras de control 9](#_Toc63075963)

[Bucle for 9](#_Toc63075964)

[Bucle while 10](#_Toc63075965)

[Bucle Do While 10](#_Toc63075966)

[Funciones 10](#_Toc63075967)

[Retornar valores 11](#_Toc63075968)

[Funciones predefinidas 11](#_Toc63075969)

[Objetos y arrays 13](#_Toc63075970)

[Objetos en JavaScript 13](#_Toc63075971)

[Propiedades y métodos 13](#_Toc63075972)

[Objetos predefinidos 14](#_Toc63075973)

[El objeto window 14](#_Toc63075974)

[El objeto document 15](#_Toc63075975)

[El objeto location 16](#_Toc63075976)

[El objeto screen 17](#_Toc63075977)

[El objeto de navigator 17](#_Toc63075978)

[El objeto History 18](#_Toc63075979)

[El objeto Date 19](#_Toc63075980)

[El objeto String 20](#_Toc63075981)

[El objeto Array 21](#_Toc63075982)

# Conceptos básicos de JavaScript

## Historia

A principios de los años 90, la mayoría de los usuarios que se conectaban a Internet lo hacían con módems a una velocidad máxima de 28.8 kbps. En esa época, empezaban a desarrollarse las primeras aplicaciones web y, por tanto, las páginas web comenzaban a incluir formularios complejos.

Con unas aplicaciones web cada vez más complejas y una velocidad de navegación tan lenta, surgió la necesidad de un lenguaje de programación que se ejecutara en el navegador del usuario. De esta forma, si el usuario no rellenaba correctamente un formulario, no se le hacía esperar mucho tiempo hasta que el servidor volviera a mostrar el formulario indicando los errores existentes.

***Brendan Eich***, un programador que trabajaba en Netscape pensó que podría solucionar este problema adaptando otras tecnologías existentes (como Striptease) al navegador Netscape Navigator 2.0, que iba a lanzarse en 1995. Inicialmente, Eich denominó a su lenguaje LiveScript.

Posteriormente, Netscape firmó una alianza con Sun Microsystems para el desarrollo del nuevo lenguaje de programación. Además, justo antes del lanzamiento Netscape decidió cambiar el nombre por el de JavaScript. La razón del cambio de nombre fue exclusivamente por marketing, ya que Java era la palabra de moda en el mundo informático y de Internet de la época.

La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales.

Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo ECMA European Computer Manufacturers Association).

ECMA creó el comité TC39 con el objetivo de "estandarizar de un lenguaje de script multiplataforma e independiente de cualquier empresa". El primer estándar que creó el comité TC39 se denominó ECMA-262, en el que se definió por primera vez el lenguaje ECMAScript.

Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

La organización internacional para la estandarización (ISO) adoptó el estándar ECMA-262 a través de su comisión IEC, dando lugar al estándar ISO/IEC-16262.

## Funcionamiento

JavaScript es un lenguaje de programación interpretado, lo que significa que no es necesario compilarlo previamente para su ejecución. Sera el propio navegador quien interprete el código y lo ejecute directamente, por lo tanto, podremos decir que su entorno de ejecución es el navegador. Métodos de integración de código JavaScript en un documento HTML:

* Etiqueta <script>: Es un método similar al usado en CSS con la etiqueta <style>, con la diferencia de que la etiqueta <script>, si es necesario, podrá definirse prácticamente en cualquier lugar de nuestro código HTML.
* Fichero externo: Define nuestro código JavaScript en un fichero externo con extensión .js.

## Sintaxis

JavaScript es un lenguaje que distingue entre mayúsculas y minúsculas, es lo que se conoce como case sensitive. No es obligatorio terminar cada instrucción o sentencia con el carácter punto y coma. Y JavaScript permite la introducción de comentarios, en una línea con // y con saltos de línea con /\*\*/.

## Elementos básicos del lenguaje

### Variables

Para definir nuestras variables, JavaScript hace uso de la palabra reservada var. Al declarar las variables, tendremos la posibilidad de declararlas como vacías o bien ya inicializadas si conocemos su valor.

El ámbito de las variables es la zona del código desde donde estas variables serán accesibles, por ejemplo, una variable que ha sido declarada en una función a no estará disponibles para el código de la función b. Existen dos tupos de variables: variables globales y variables locales.

* Variables locales: Son aquellas que únicamente son referenciadas dentro del código donde fueron definidas.
* Variables globales: Se pueden referenciarse o ser accesibles desde cualquier punto de nuestro código.

### Tipo de datos

Los tipos de datos que se pueden almacenar en una variable son numerosos: cadenas de texto, números enteros, reales, negativos y valores booleano0s, entre otros. En JavaScript, al ser un lenguaje interpretado, el tipo de variable se decide en tiempo de ejecución dependiendo del tipo de datos asignado a la variable. Los tipos de variables se pueden dividir en tres grandes grupos: numéricos, texto y booleanos.

#### Variables de tipo numérico

Son variables que únicamente pueden almacenar valores numéricos, ya sean enteros positivos, negativos, números decimales o reales. Para separar la parte entera de la decimal es obligatorio usar el punto.

#### Variables de tipo texto

Estas variables almacenan cadenas de texto de cualquier longitud, ya sean palabras, frases o párrafos completos. Para asignar un valor a este tipo de variables, hay que encerrar la cadena de texto entre comillas dobles o simples.

#### Variables de tipo booleano

Las variables booleanas son aquellas que almacenan un tipo de dato especial que únicamente puede aceptar dos valores: true o false.

### Constantes

Las constantes son valores que, una vez han sido declarados o inicializados, no modifican su valor durante toda la ejecución del programa. Para definir las constantes, utilizaremos la palabra reservada const seguida del nombre de la constante. Nombre que por convenio se suele escribir en mayúsculas.

### Operadores

Los operadores son elementos del lenguaje que nos permite manipula y comparar variables o datos para la toma de decisiones, son elementos claves en las estructuras de control de cualquier lenguaje de programación y, por lo tanto, una parte fundamental de nuestro código. Grupos de operadores: operadores aritméticos, de asignación, de cadena de texto, lógicos y condicionales.

#### Operadores aritméticos

Este operador se utiliza para realizar cálculos matemáticos simples con nuestras variables o con cualquier valor numérico. Son: suma, resta, multiplicación, división e incremento/decremento.

#### Operadores de asignación

* Asignación: Es un operador que no necesita prácticamente explicación, permite asignar el valor situado en la parte derecha del igual a la variable situada en la parte izquierda.
* Asignación aritmética: Operador muy útil para unificar en una misma instrucción de este tipo como operaciones matemáticas hemos estudiado:
  + Asignación y suma: +=.
  + Asignación y resta: -=.
  + Asignación y producto: \*=.
  + Asignación y división: /=.
  + Asignación y modulo: %=.

#### Operadores de cadena

JavaScript solo posee un operador de manipulación de cadena de texto, es el operador de concatenación que viene representado por el símbolo + y que nos permite unir dos cadenas de texto en una.

#### Operadores lógicos

Su principal objetivo es en una estructura de control. Estos operadores trabajan directamente con tipos de datos booleanos, lo que significa que el resultado de una operación lógica únicamente podrá ser: true o false.

Los tipos de operadores lógicos que nos proporciona JavaScript son dos: *and* y *or*, que están representados por los símbolos && y || respectivamente.

#### Operadores condicionales

El resultado obtenido en un operador condicional es de tipo booleano, los operadores son:

* **Igualdad**: Se usa para comprobar si dos valores son iguales (==).
* **Distintos**: Se usa para saber si dos valores son distintos (!=).
* **Mayor** **y** **menor**: Op0eradores utilizados para comparar únicamente dos valores numéricos (<, >).
* **Mayor o igual y menor o igual**: Similar a los operadores anteriores, pero en este caso no es una comparación estricta (<=, >=).

# Tabla de ejercicios

|  |  |
| --- | --- |
| **Ejercicio 29** | Integración JavaScript en HTML |
| **Ejercicio 30** | Estructura de control |
| **Ejercicio 31** | Bucle for |
| **Ejercicio 32** | Bucle While |
| **Ejercicio 33** | Bucle Do While |
| **Ejercicio 34** | Funciones |
| **Ejercicio 35** | Retornar valor |
| **Ejercicio 36** | Formulario 1 |
| **Ejercicio 37** | Formulario 2 |

# Funciones y estructuras de control

## Estructuras de control

También conocida como estructuras de control de flujo, son un conjunto de instrucciones que, dependiendo de su funcionamiento, modificaran el flujo del código con el fin de ejecutar unas instrucciones u otras. Estructuras:

* Estructura de control if: Se usa con los operadores lógicos y de comparación para la toma de decisión. La estructura condicional if es, sin duda, la más básica de todas y la más utilizada.
* Estructura de control if-else: Es una ampliación de la anterior, consiste en agregar una nueva palabra reservada, else, que marcara el inicio de las instrucciones que se ejecutaran cuando no se cumpla la condición del if.
* Estructura de control switch: La estructura de control switch-case, el programador podrá consultar el valor de una variable de entre por ejemplo quince valores posibles, mientras que para hacer esto con un if tendríamos que implementar un nivel de anidamiento muy grande o quince veces la estructura if.

## Bucle for

La funcionalidad de esta estructura y en general de todas las estructuras iterativas es la de repetir la ejecución de una o más instrucciones un número determinado de veces. El número de iteraciones dependerá del cumplimiento de ciertas condiciones definidas en la propia estructura.

En este tipo de estructuras, también se puede hablar de bucles for anidados si dentro del bucle principal se incluye otro bucle for secundario. Se trata de un mecanismo muy útil para recorrer estructuras de datos.

## Bucle while

El bucle While ejecuta una o varias instrucciones mientras no se cumpla la condición especificada al comienzo de la estructura. A diferencia del bucle for, que utiliza principalmente variables numéricas en sus comparaciones, el bucle While utilizar tipos de datos booleanos. Son estructuras muy utilizadas para procesos de búsqueda.

## Bucle Do While

La estructura do while tiene un funcionamiento similar al del bucle while, con la única diferencia de que, en este caso, primero se ejecuta una iteración y después se evalúa la condición del bucle, mientras que, en el bucle while, primero se evaluaba la condición del bucle y, dependiendo del resultado, se ejecutaban o no las iteraciones. Esto nos asegura de que por lo menos una vez si que se van a ejecutar las instrucciones definidas en el bucle.

## Funciones

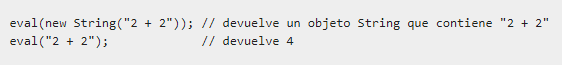
Las funciones, también conocidas como subprogramas, subrutinas o procedimientos, son bloques de código que realizan una tarea específica y que, dependiendo de su implantación, pueden retornar valores. El uso de funciones es la forma optima de desarrollar aplicaciones, dividiendo el programa principal en pequeños subprogramas que se llaman entre si hasta dar con la solución final. Las funciones pueden reutilizarse e invocarse desde cualquier punto de nuestro código tantas veces como sea necesario.

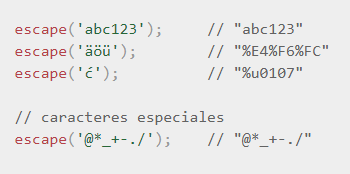
## Retornar valores

La forma que tiene JavaScript de devolver valores en una función es mediante el uso de la palabra clave ***return*** seguida de un valor o variable. Es importante resaltar que dentro de una función únicamente se puede retornar un valor y que el uso de return implica de forma automática el fin de la función.

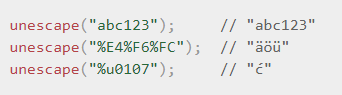
## Funciones predefinidas

JavaScript también dispone de un amplio abanico de funciones predefinidas que nos resultaran de vital importancia en el desarrollo de aplicaciones Web dinámicas y para solucionar problemas so situaciones especiales en el tratamiento de los tipos de datos. Funciones predefinidas:

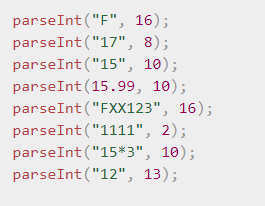
* ***Eval*** (código): Esta función recoge una cadena de texto por parámetro, la evalúa y la convierte en una instrucción válida para ser ejecutada.
* ***Escape*** (cadena): La función cadena retorna la codificación hexadecimal de la cadena de texto pasada por parámetros.



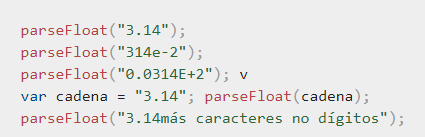
* ***Unescape*** (cadena): Su finalidad es descodificar la cadena de texto pasada por parámetro y convertirla en texto legible para el usuario.



* ***EncodeURI*** (cadena\_url): Es una función similar a escape, pero diseñada especialmente para direcciones de Internet. En este caso, se codifica una URL pasada por parámetro sustituyendo los caracteres especiales por secuencias de escape representadas en UTF-8. Su uso está destinado a codificar ciertas URL que puedan contener caracteres especiales antes ser enviadas al servidor.
* ***DecodeURI*** (cadena\_url): Es la función complementaria a EncodeURI. Su finalidad es decodificar una URL que ha sido codificada previamente con EncodeURI.
* ***encodeURIComponent*** (cadena\_url): Tiene el mismo uso y sintaxis que EncodeURI con la diferencia de que esta función también codifica los caracteres =, & y ?.
* ***decodeURIComponent*** (cadena\_url): Decodifica la URL pasada por parámetro y que fue codificada por encodeURIComponent.
* ***parseInt*** (cadena\_número, base): Convierte un numero pasado como texto en el mismo número, pero ya en formato numérico entero. EL segundo parámetro es opcional y se usa para indicar la base numérica de la conversión, si se omite se usará base decimal. Si intentamos hacer una operación aritmética con un numero en formato texto, obtendríamos una concatenación de cadenas.



* ***ParseFloat*** (cadena\_numero, base): Su funcionamiento es exactamente igual a parseInt, pero en este caso el resultado de la conversión será un dato numérico con decimales.



* ***isNaN*** (expresión): Esta función retorna false si la expresión pasada por parámetro es un número y true en caso contrario.

# Objetos y arrays

## Objetos en JavaScript

Un objeto es una entidad o elemento que encapsula una serie de servicios o datos que ofrece a través de una interfaz de paso de mensajes y que constituye la parte fundamental de la Programación Orientado a Objetos. Los objetos contienen toda la información que permite identificar una entidad de forma completa e inequívoca, al mismo tiempo que disponen de mecanismos tales como los métodos para interactuar con otros objetos.

## Propiedades y métodos

Un objeto se puede dividir básicamente en tres elementos bien diferenciados, propiedades métodos y constructores:

* Propiedades: Son las variables que describen las características principales del objeto al que representan. Pueden tomar diferentes valores dependiendo del tipo de datos al que representen. El valor de estas propiedades pueden ser constantes o variables y representan de forma única la objeto.
* Métodos: Son las funciones o acciones propias de cada objeto y cuya finalidad principal es la de modificar las propiedades de este. Internamente, un método es un mensaje que recibe el objeto para realizar una acción previamente implementada.
* Constructores: Un constructor es un método que se utiliza para crear e inicializar un objeto en tiempo de ejecución.

## Objetos predefinidos

### El objeto window

Su función es la de gestionar, administrar y proveer cualquier tipo de información acerca de la ventana que contiene la pagina principal. Propiedades del objeto window:

* Closed: Propiedad de tipo booleano que nos indica si una ventana ha sido cerrada o no.
* defaultStatus: Propiedad que define el texto que se mostrara en la barra de estado del navegador.
* History: Esta propiedad es, a su vez, otro objeto que representa los enlaces a las paginas que el usuario ha visitado con anterioridad.
* Location: Del mismo modo que ocurre con la propiedad anterior, location también es otro objeto, en este caso contiene y gestiona toda la información relevante a la URL de la Web actual.
* Name: Propiedad que contiene el nombre de la ventana actual.
* Status: Valor de la barra de estado del a ventana actual.
* Parent: Propiedad que hace referencia ala ventana padre de la ventana actual.
* Self: Propiedad u objeto que hace referencia a la propiedad ventana, es similar a usar window.
* Top: Ventana superior del navegador.

#### Métodos del objeto window

* Open (url, nombre, opciones): Método que abre una ventana emergente con la pagina Web indicada en el parámetro URL, el segundo parámetro hace referencia al nombre que le daremos a la ventana y el tercer parámetro define las diferentes opciones o configuraciones de nuestra ventana. Opciones más comunes para open:
  + Height: Define el alto de la ventana en pixeles.
  + Width: Ancho de la ventana en pixeles.
  + Scrollbars: Especifica si se muestran las barras de desplazamiento de la ventana o no.
  + Resizable: Esta opción define si a nuestra ventana emergente se le puede modificar el tamaño con el uso del ratón.
  + Status: Dependiendo del valor muestra u oculta la barra de estado de la ventana en cuestión.
  + Menubar: Especifica si se muestra o no la barra de menus.
  + Toolbar: Especifica si se muestra o no la barra de herramientas de la ventana.
* Close(): Cierra la ventana actual.
* Blur(): Insta a la ventana actual a perder el foco.
* Focus(): Al contrario que el anterior, este método hace que la ventana actual sea la que se encuentre activa. Lo que se conoce como obtener el foco.
* Alert(mensaje): Su finalidad es la de mostrar por pantalla el mensaje pasado por parámetro.
* Confirm(mensaje): Este método también muestra un mensaje modal por pantalla, pero, en esta ocasión, el usuario deberá elegir entre dos opciones Aceptar o Cancelar, y dependiendo de la opción seleccionada, este método devolverá true o false respectivamente.
* Prompt(mensaje, valor\_por\_defecto): Método que se utiliza para solicitar un dato al usuario mediante un cuadro de dialogo. La información introducida en el cuadro de texto será el valor que retorne nuestro método. El primer parámetro de entrada es el mensaje que mostrara la ventana emergente y el segundo parámetro el valor por defecto del cuadro de texto.

### El objeto document

Es el objeto encargado de administrar el contenido y lo elementos presentes en nuestra pagina Web. Este objeto, al igual que la mayoría de los objetos predefinidos en JavaScript, depende directamente del objeto window. Propiedades:

* bgColor: Especifica el color de fondo de nuestra página Web.
* fgColor: Define el color del texto de nuestra página Web.
* linkColor: Especifica el color de los enlaces o hipervínculos.
* vlinkColor, alinkColor: Define el color de los enlaces visitados y activos respectivamente.
* Images: Array o vector con todas las imágenes presentes en la página Web actual. Cada posición del array hace referencia a una imagen.
* Links: Array que contiene todos los enlaces definidos en la pagina Web. El funcionamiento y la sintaxis son exactamente igual que los de la propiedad images.
* Forms: Array con todos los formularios de nuestra página Web.
* Title: Especifica el título de la página Web.

#### Método del objeto document

* Write(texto): Muestra en pantalla el texto pasado por parámetro, este texto puede contener código HTML que será interpretado por JavaScript.
* Writeln(texto): Idéntico uso y sintaxis que la anterior propiedad, pero en este caso se agrega una línea en blanco de forma automática después de escribir el texto pasado por parámetro.

### El objeto location

Objeto dependiente de window, tiene como finalidad gestionar y consultar toda la información relativa al a URL de la página Web actual. Propiedades:

* Host: Propiedad que nos indica el nombre del servidor o dominio y el puerto donde está corriendo la página Web.
* Hostname: Propiedad idéntica al anterior, pero, en este caso, únicamente se mostrará el nombre del servidor o dominio.
* Port: Puerto del servidor Web.
* Protocol: Protocolo usado por la URL de la pagina Web, normalmente http o https.
* Href: Propiedad que especifica la URL de la pagina actual, es una propiedad de lectura/escritura. Esto significa que podremos hacer una redirección a otra Web simplemente modificando el valor de esta propiedad.
* Pathname: Propiedad que retorna la parte de la URL que hace referencia al recurso del servidor Web.

#### Métodos del objeto location

* Reload(): Fuerza la recarga de la página actual.
* Replace(URL): Carga en la página actual la URL especificada en el parámetro de entrada. Se trata de un mecanismo similar al que se consigue modificando la propiedad href, con la diferencia de que, en este caso, la entrada en el historial de paginas visitadas es reemplazada por la nueva URL.

### El objeto screen

Objeto que tiene como finalidad proporcionar información útil acerca del dispositivo de video o pantalla de usuario. El objeto screen es otra propiedad que desciende directamente del objeto window. Propiedades:

* availHeight: Alto de pantalla en pixeles disponibles para el navegador.
* availWidth: Ancho de pantalla en pixeles disponible para el navegador.
* Width: Anchura total de pixeles de la pantalla de usuario.
* Height: Altura total en pixeles de la pantalla de usuario.
* colorDepth: numero de bits por pixeles utilizados para representar la profundidad de colores de la pantalla.

### El objeto de navigator

Objeto que suministra toda la información disponible acerca del navegador que esta mostrando la página Web. Este objeto únicamente tiene implementado un meto, la mayor parte de la información útil esta presente en sus propiedades:

* appCodeName: Propiedad que especifica el nombre del código del navegador. No es una propiedad muy útil, puesto que todos los navegadores devuelven el mismo resultado.
* appName: Nombre o marca comercial del navegador.
* Language: Propiedad que retorna la cadena de texto o código representativo del lenguaje del navegador Web.
* Platform: Cadena de texto que describe el sistema operativo sobre el que esta funcionando nuestro navegador Web.
* Plugins: Propiedad que retorna un array o vector con todos los plugins instalados en nuestro navegador.
* mimeTypes: Esta propiedad retorna un array con todos los tipos de MIME que soporta nuestro navegador.
* userAgent: Devuelve una cadena de texto con información completa del navegador Web.
* cookieEnabled: Propiedad de tipo booleano que retorna true si el navegador tiene activadas las cookies y false en caso contrario.

#### Métodos del objeto navigator

* javaEnabled(): Método utilizado para determinar si el navegador en cuestión tiene activado el plugin de Java, retornara true o false dependiendo de si esta activado o no.

### El objeto History

Objeto descendiente de window que gestiona todas las direcciones de Internet o URL que hemos visitado y que se almacenan en el historial del navegador. Básicamente, lo que hace este objeto es almacenar la referencia a las paginas Web visitadas para posteriormente poder recorrer dichas paginas hacia delante o hacia atrás. Propiedades:

* length: Numero entero que representa la cantidad de entradas almacenadas actualmente en el historial.

#### Métodos del objeto history

* back(): Este método provoca que se carguen en nuestro navegador la pagina que se ha visitado justo anteriormente.
* Forward(): Al contrario que el anterior método, este nos redirecciona a la pagina siguiente en el historial de navegación.
* Go(numero): Este método nos llevara directamente a la Web posicionada en el numero indicado por el parámetro de entrada orden del historial del navegador.

### El objeto Date

Es el mecanismo que proporciona JavaScript para la manipulación de fechas y horas. Propiedades:

* No maneja fecha anterior al 1 de enero de 1970.
* El primer día de la semana es domingo y tiene como valor 1.
* Los meses del año se contabilizan del 0 al 11.
* Formato de horas: HH:MM:SS.
* Gestiona y maneja las fechas en milisegundos.

#### El constructor de Date

Se invoca de dos maneras diferentes:

* Var fecha new Date(): De esta forma hemos creado e inicializado el objeto Date con la fecha actual del sistema.
* Var fecha new Date(año, mes, día, hora, minutos, segundos): Constructor con una fecha concreta.

#### Metodos del objeto Date

* getDate(): Método que retorna el día del mes de la fecha creada en el constructor.
* getDay(): Devuelve un numero entero que representa el día de la semana de la fecha en cuestión.
* getHours(): Retorna un entero entre 0 y 23 que representa la hora de la fecha creada.
* getMinutes(): Entero de 0 a 59 que representa los minutos de la hora en cuestión.
* getSeconds(): Devuelve los segundos transcurridos en el minuto actual.
* getTime(): Método que retorna los milisegundos transcurridos desde el 1 de enero de 1970 hasta ahora.
* getMonth(): Retorna un entero entre 0 y 11 que representa el mes de la fecha creada por el constructor.
* getYear(): Año de la fecha actual.
* toLocaleString(): Retorna una cadena de texto con fecha actual y con el formato de la zona horaria especificada en el sistema.
* toGMTString(): Método que devuelve una cadena con la fecha y hora completa en formato o estándar GMT.
* setDate(diaMes): Establece el día de la semana pasada por parámetro en el objeto Date actual.
* setDate(diaSemana): Define el dia de la semana pasado por parámetros en el objeto Date actual.
* setHours(horas): Especifica la hora del dia pasado por parámetros en la fecha creada por el objeto actual.
* setMinutes(minutos): Define los minutos pasados por parámetros de la hora actual en el objeto Date.
* setMonth(mes): Define el mes pasado por parámetros en la fecha definida por el objeto Date.
* setSeconds(segundos): Define los segundos pasados por parámetros en la fecha definida por el objeto Date.
* setYear(año): Especifica el año pasado por parámetro en la fecha actual.
* setTime(milisegundos): Modifica la fecha actual con la fecha en milisegundos pasada por parámetro.

### El objeto String

Como finalidad la manipulación y administración de las cadenas de texto en JavaScript. Propiedades:

* length: Propiedad que retorna la longitud o numero de caracteres de una cadena de texto.

##### Metodos del objeto String

* indexOf(cadenaBusqueda, inicio): Método muy útil que nos devuelve la posición de la primera coincidencia de la cadena pasada por parámetro dentro de la cadena actual.
* lastIndexOf(cadena\_busqueda, inicio): En este caso el valor retornado se corresponde con la posición de la ultima ocurrencia de la cadena de búsqueda.
* Substring(inicio, fin): Este método devuelve la cadena de texto que se encuentra delimitada entre las posiciones pasadas por los parámetros.
* Link(url): Método que convierte la cadena de texto en un enlace que apuntara a la URL pasada por parámetros.
* toLowerCase(): Convierte en minúsculas la cadena de texto actual.
* toUpperCase(): Convierte en mayúsculas la cadena de texto actual.
* Big(): Método que muestra la cadena de texto con un tamaño de fuente de gran tamaño.
* Small(): Muestra la cadena de texto con un tamaño de fuente bastante pequeño.
* Blink(): Asigna a la cadena de texto un efecto de parpadeo.
* Italics(): Método que muestra el texto en cursiva.
* Strike(): Método que asigna un efecto tachado a la cadena de texto.
* Sup(): Muestral acedan de texto con un formato de superíndice.
* Sub(): Muestra la cadena de texto con un formato de subíndice.
* charAt(índice): Método que retorna el carácter situado en la posición indicada por el parámetro índice de la cadena actual.
* Fontcolor(color): Color de la cadena de texto pasada por parámetros.
* Fontsize(tamaño): Modifica el tamaño de la cadena de texto por el tamaño pasado por parámetro.

### El objeto Array

Los arrays, también conocidos como vectores, son estructuras de datos lineales que almacenan la información en posiciones continuas, pudiéndose almacenar en cada posición del array cualquier tipo de dato. Sintaxis:

* Var vector= new Array(): Esta es la forma de crearse un array vacío, sin datos y sin un tamaño determinado.
* Var vector= new Array(5): Similar al anterior, pero en este caso hemos creado un array vacío de cinco posiciones.
* Var vector= new Array (“John”, “Wayne”, 45): Crear e inicializar un array en una sola instrucción.

#### Matrices

Son estructuras de datos muy útiles para almacenar grupos de información que requieran una organización dividida en filas y columnas. Propiedades:

* Length: Propiedad que nos retorna el numero de elementos que tiene un array, es lo que se conoce también como longitud de un array.

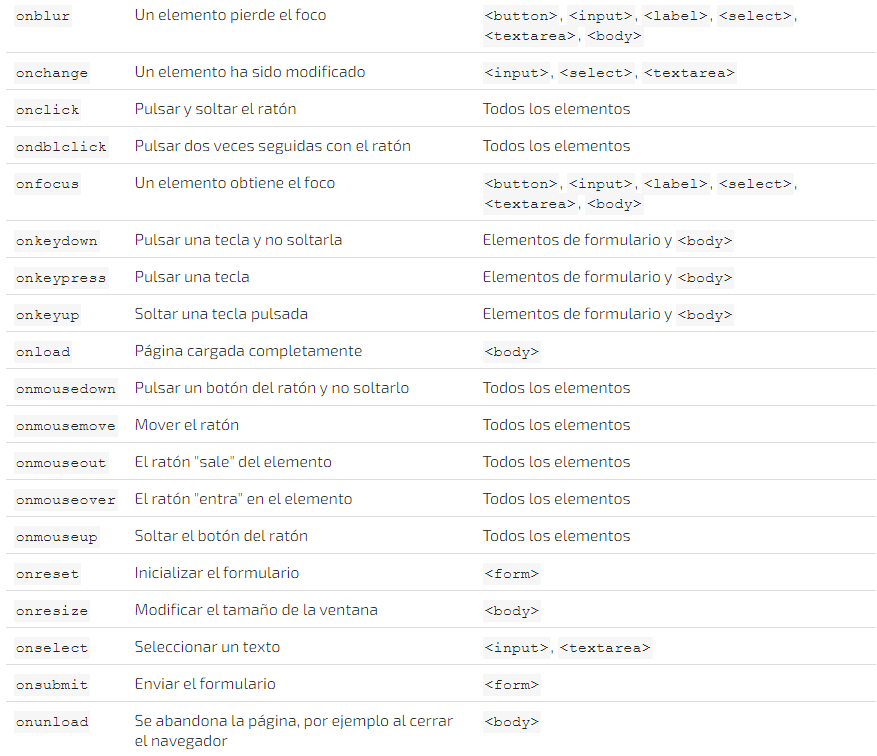
#### Metodos del objeto Array

* Concat(Array): Método que tiene como función concatenar el array actual con el array pasado por parámetro.
* Join(separador): Método que retorna una cadena de texto con el contenido de todas las posiciones de un array separadas por el carácter pasado por parámetros.
* Reserve(): Método que invierte el orden de los elementos almacenados en un vector, detal forma que el primer elemento pasaría a ser el último y así sucesivamente.
* Sort(): Este método realizar una ordenación de los datos presentes en un array.

# Eventos, integración HTML y JavaScript

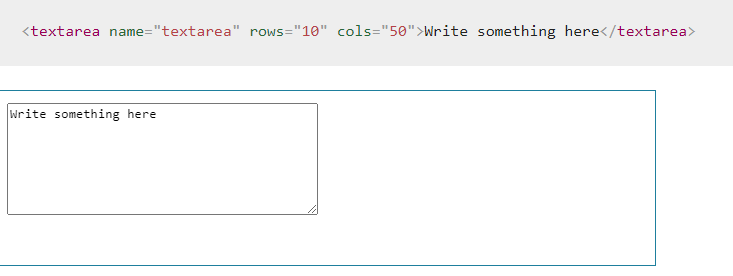
Un evento es una acción del usuario sobre un elemento HTML; es, por lo tanto, el mecanismo principal que permite a la interacción de los usuarios con los elementos de una pagina Web. Los eventos están identificados por la palabra reservada *on* seguida del nombre de la acción, así, por ejemplo, un evento *onclick* se correspondería con la acción de hacer clic con el ratón sobre cualquier elemento de la página.

## Eventos

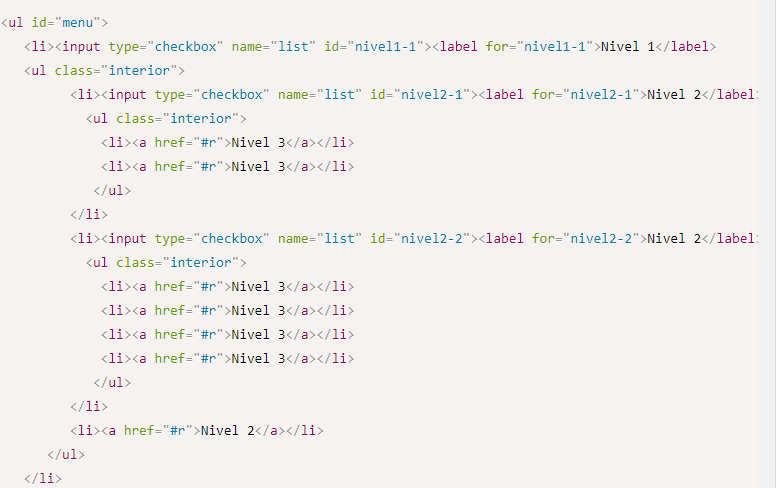


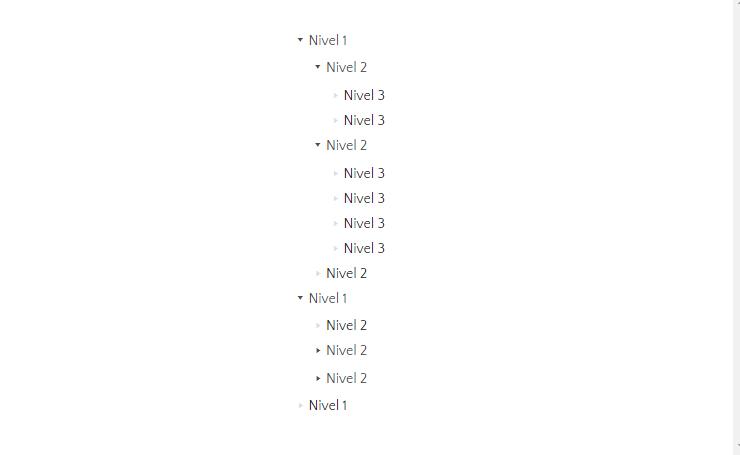
# Formularios y JavaScript

## Cuadros y áreas de texto



## Listas desplegables





## Casillas de verificación o checkbox



## Botones de opción o radio button



