JavaScript

Índice

[Conceptos básicos de JavaScript 3](#_Toc62837042)

[Historia 3](#_Toc62837043)

[Funcionamiento 4](#_Toc62837044)

[Sintaxis 4](#_Toc62837045)

[Elementos básicos del lenguaje 4](#_Toc62837046)

[Variables 4](#_Toc62837047)

[Tipo de datos 5](#_Toc62837048)

[Constantes 6](#_Toc62837049)

[Operadores 6](#_Toc62837050)

[Tabla de ejercicios 8](#_Toc62837051)

[Funciones y estructuras de control 9](#_Toc62837052)

# 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 ala 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 pose 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, per en este caso no es una comparación estricta (<=, >=).

# Tabla de ejercicios

|  |  |
| --- | --- |
| **Ejercicio 9** | Estructura |
| **Ejercicio 10** | Propiedad Input |
| **Ejercicio 11** | Formulario |
| **Ejercicio 12** | Multimedia |
| **Ejercicio 13** | Introducción a CSS |
| **Ejercicio 14** | Selector de atributos |
| **Ejercicio 15** | Herencias en CSS |
| **Ejercicio 16** | Elementos flotantes |
| **Ejercicio 17** | Formateado de párrafos y Ejercicio17css con las propiedades CSS del ejercicio |
| **Ejercicio 18** | Enlaces e imágenes en CSS y Ejercicio18css con las propiedades CSS del ejercicio |
| **Ejercicio 19** | Listas con CSS |
| **Ejercicio 20** | Menú horizontal |
| **Ejercicio 21** | Tablas |
| **Ejercicio 22** | Formulario completo y Ejercicio22css con las propiedades CSS del ejercicio |
| **Ejercicio 23** | Esquinas |
| **Ejercicio 24** | Sombras redondeadas |
| **Ejercicio 25** | Textos y columnas |
| **Ejercicio 26** | Personalización de textos |
| **Ejercicio 27** | Textos y fuentes |
| **Ejercicio 28** | Botones |

# 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