**Aplicación de Tecnologías Web**

Tema n.° 4

JavaScript

**Índice**

**Pág.**

[1.4. Introducción a JavaScript 3](#_Toc161301992)

[1.4.1. Breve historia de JavaScript 3](#_Toc161301995)

[1.4.2. IDE y herramientas 4](#_Toc161301996)

[1.4.3. Conceptos básicos de JavaScript 6](#_Toc161301997)

[1.4.4. Declaración de variables y constantes 10](#_Toc161301998)

[1.4.5. Expresiones básicas 13](#_Toc161301999)

[Recursos complementarios 15](#_Toc161302000)

[Referencias 15](#_Toc161302001)

[Autoevaluación 16](#_Toc161302002)



# Introducción a JavaScript



El lenguaje de programación por excelencia para la programación en el lado cliente es JavaScript. Prácticamente todos los navegadores modernos y más utilizados tienen soporte para este lenguaje.

En este tema realizaremos una introducción a este lenguaje y veremos los siguientes aspectos:

* **Historia.** Una breve introducción a la historia de este lenguaje desde su concepción hasta nuestros días.
* **IDE (*Integrated Development Environment*) y herramientas.** Exploraremos cuáles van a ser las herramientas que usaremos para desarrollar código en JavaScript.
* **Conceptos básicos.** Se estudiarán la base de este lenguaje, su sintaxis y los tipos de datos que podemos utilizar con él. Como verás, comparte similitudes con otros lenguajes de programación muy conocidos, pero tiene algunas peculiaridades que señalaremos.
* **Estructuras de control e iterativas.** Estudiaremos cuáles son las sentencias en JavaScript que nos permiten crear lógicas más complejas, en función de ciertas condiciones, incluyendo las iteraciones (bucles).

## Breve historia de JavaScript

Una mañana de mayo de 1995, un directivo de Netscape se acercó a un joven, Brendan Eich, y le dijo algo como esto:

“En menos de dos semanas lanzamos nuestro siguiente navegador: el Netscape Navigator 2.0. Queremos que incorpore un intérprete para un lenguaje de programación que pueda interactuar con las páginas HTML que el usuario visualice. Por cierto, le pondremos algún nombre relacionado con Java, ya que este es ahora mismo el lenguaje más famoso y se usa cada vez más en navegadores. Nos vamos a aliar con Sun Microsystems (quien detenta la marca Java) para intentar frenar a nuestro principal competidor: Microsoft (y su navegador Internet Explorer)”

El joven Eich ya sabía lo que tenía que hacer: sacar un nuevo lenguaje de programación de la chistera en un muy corto espacio de tiempo y hacer que se pareciera algo a Java para poder venderlo como tal a la comunidad de desarrolladores.

Inicialmente llamó a su creación Mocha (por el café moca) y luego lo cambió a LiveScript, sin embargo, fue obligado a incluir la palabra Java por motivos exclusivamente comerciales. Sun Microsystems (creador de Java) y Netscape tenían intereses en potenciar el JavaScript y querían que su nombre tuviera gancho. Justo en ese momento, Java era el lenguaje de programación más popular.

“A pesar de compartir ciertas similitudes sintácticas, Java y JavaScript son lenguajes completamente distintos y no están relacionados”

## IDE y herramientas

Para trabajar con JavaScript se recomienda hacer uso de las siguientes herramientas, aunque cada alumno puede elegir otras si así lo desea.

**Tabla 1**

IDE's y herramientas para trabajar con JavaScript

|  |  |
| --- | --- |
| Entornos de Desarrollo Integrados (IDEs) | * Visual Studio Code (VS Code): Un IDE liviano y altamente configurable desarrollado por Microsoft. Ofrece una amplia variedad de extensiones para facilitar el desarrollo web, depuración integrada, soporte para Git y una interfaz de usuario amigable. * Sublime Text: Un editor de texto simple y potente que es muy popular entre los desarrolladores web. Aunque es más ligero que algunos IDEs, ofrece muchas características útiles y es altamente personalizable mediante complementos. * Atom: Desarrollado por GitHub, Atom es un editor de texto gratuito y de código abierto que es fácil de personalizar y extender. Viene con un gestor de paquetes integrado para instalar extensiones y temas. * WebStorm: Desarrollado por JetBrains, es un IDE específico para JavaScript, TypeScript y Node.js. Ofrece herramientas avanzadas de refactorización, depuración y soporte para frameworks populares como React, Angular y Vue.js. |
| Herramientas de Gestión de Paquetes | * npm (Node Package Manager): La herramienta de gestión de paquetes por defecto para Node.js. Permite instalar, compartir y gestionar dependencias de proyectos fácilmente. * Yarn: Una alternativa a npm que se centra en la velocidad, la confiabilidad y la seguridad. Es compatible con el registro npm y proporciona características adicionales, como la instalación de paquetes offline. |
| Frameworks y Bibliotecas | * React.js: Una biblioteca de JavaScript para construir interfaces de usuario, desarrollada por Facebook. Es especialmente eficaz para construir aplicaciones de una sola página (SPA). * Angular: Un marco de desarrollo web mantenido por Google. Proporciona una estructura sólida para construir aplicaciones web escalables y dinámicas. * Vue.js: Un framework progresivo para construir interfaces de usuario. Es fácil de integrar en proyectos existentes y ofrece una curva de aprendizaje suave. |
| Herramientas de Pruebas | * Jest: Un marco de pruebas desarrollado por Facebook. Es fácil de configurar y tiene un buen rendimiento, lo que lo hace adecuado para pruebas unitarias y de integración. * Mocha: Un marco de pruebas versátil que se ejecuta en Node.js y el navegador. Es extensible y permite utilizar diferentes bibliotecas de aserciones y bibliotecas de prueba. |

## Conceptos básicos de JavaScript

En esta sección se estudiará la sintaxis básica de JavaScript. La cual abarca los siguientes aspectos:

* Tipos de datos primitivos.
* Declaración de variables y constantes.
* Expresiones y sentencias.

***Tipos de datos primitivos***

JavaScript es un lenguaje dinámico, esto implica que no es necesario especificar el tipo de los valores (si es un número, una palabra, una lista de cosas, etc.), sino que JavaScript los averigua de manera automática y, además, estos pueden cambiar sin ningún problema. Por ejemplo, es posible especificar que un valor vale inicialmente 1 (un número) y luego «hola» (una cadena de texto).

**Figura 1**

Variable cuyo tipo de dato cambia dinámicamente



JavaScript discernirá automáticamente que esa variable a es primero de tipo número (*number*) y luego de tipo cadena de texto (*string*). El ejemplo anterior produciría un error gravísimo en otros lenguajes tipados como Java o C++. En estos lenguajes, una vez que una variable ha sido especificada con un valor (de manera implícita o explicita), ya no hay vuelta atrás: **no puede cambiar de tipo.**

En los lenguajes tipados, si un valor nace con un tipo, ya no es posible cambiarlo. Esto busca que el código sea más seguro y menos propenso a errores. JavaScript es un lenguaje no tipado, es decir, dinámico.

Los **tipos de datos básicos** (también llamados primitivos) en JavaScript son:

**Figura 2**

Tipos de datos primitivos en JavaScript.

**Cadenas de texto:** Una cadena de texto no es más que una **secuencia de caracteres.** En muchos lenguajes de programación, una cadena de texto no es más que un vector o un *array* de letras. Sin embargo, en JavaScript, una cadena de texto es un tipo de **dato básico.**

Para definir una cadena de texto, simplemente tenemos que poner letras entre comillas (dobles o simples):

**Figura 3**

Ejemplo de cadenas de texto



A esta forma de definir cadenas de texto se la llama definición literal. Estrictamente, para crear una cadena de texto, deberíamos usar una función constructora junto con la cláusula new. Ejemplo:

**Figura 4**

Construcción de cadenas con new



A diferencia de otros lenguajes de programación, los tipos de datos primitivos tienen métodos. Los métodos son conjuntos de instrucciones (funciones) que se aplican sobre el dato

**Figura 5**

Métodos sobre cadenas



**Números:** Los números en JavaScript pueden ser enteros (4, 1, 0, -98, etc.) o en coma flotante (1.21). Al igual que las cadenas de texto, los números también cuentan con métodos para obtener otros resultados derivados.

La diferencia es que, en vez de usar un único punto, tenemos que usar dos para esquivar el punto de la notación decimal:

**Figura 6**

Notación exponencial 1.



Cuando un número no es computable, JavaScript, como hacen otros lenguajes, cuenta con la palabra reservada NaN (not a number). Por ejemplo, no es posible computar la multiplicación de un número por una cadena de texto y esta operación devolvería NaN. Ejemplo:

**Figura 7**

Operación lógica NaN



Es posible convertir cadenas de texto en números con la función Number():

**Figura 8**

Función Number.



**Booleanos:** Los valores booleanos solo pueden ser verdaderos o falsos (true o false). Además, las siguientes expresiones se consideran falsas (lo mismo que false) en JavaScript:

**Figura 9**

Valores falsos para un booleano.



## Declaración de variables y constantes

Las variables se usan en programación para referirse a un **dato almacenado en memoria.** Las variables han de declararse antes de usarse. Con la aparición de la versión 2015 de JavaScript, también conocida como ES6, las variables deben declararse con las palabras clave let o const.

* Con **let**, una variable puede cambiar de valor.
* Cuando usamos **const**, puede decirse que en este caso se trata de una \*constante\* en lugar de una variable. Su valor no cambiará y será inmutable.

También, al declarar una variable como constante debe quedar «inicializada» (debe recibir un valor) en esa misma línea.

Veamos varios ejemplos:

**Figura 10**

Declaración de variables y constantes



La afirmación anterior tiene una excepción: **los objetos.** Es decir, aunque una variable se declare como const, si hace referencia a un objeto, sí que es posible mutar ese objeto. Ejemplo:

**Figura 11**

Modificando objetos aún con const.



El ámbito es un término muy relevante, es casi cualquier entorno de programación. Se refiere a dónde es posible hacer referencia a una variable (o constante). Empleando un término matemático, se trata del dominio de definición de un valor. Existen siempre dos tipos de ámbitos principales en cualquier programa JavaScript: global y local.

***Ámbito global***

Indistintamente de la variable declarada fuera de cualquier bloque se dice que pertenece al ámbito global y es accesible desde cualquier lugar de nuestro programa.

El ámbito global es único (para un programa). Parece, de entrada, algo estupendo, pero no siempre es la mejor idea. Lo ideal es solo declarar variables lo más cerca posible (en términos de ámbito) de dónde van a ser utilizadas. Existen muchas razones para esta práctica: desde la limpieza y legibilidad del código a la seguridad.

***Ámbito local***

Lo primero que tenemos que decir del ámbito local es que, en realidad, existen tantos ámbitos locales como bloques hayamos definido. En cada bloque tenemos un ámbito local. Por descontado, cada bloque define su ámbito local y una variable definida ahí solo es visible dentro del bloque. Si el nombre de una variable se declara fuera y dentro de un bloque, el valor que cuenta para el bloque es el definido en este último.

**Figura 12**

Ámbitos local y global



Cuando asignamos un valor a una variable, esta asignación puede ser **directa o indirecta,** también conocida como **por referencia.** Por defecto, si asignamos un valor primitivo, la asignación es directa, es decir, al ejecutar la instrucción.

**Figura 13**

Asignación por valor.



En cambio, por referencia varias variables pueden referirse al mismo objeto.

**Figura 14**

Asignación por referencia.



## Expresiones básicas

Haciendo uso de todos los elementos que se han presentado hasta ahora, hemos construido algunas expresiones o sentencias en lenguaje JavaScript. Un tipo de expresión muy común son las comparaciones. Este tipo de sentencias constan de un operador y de dos operandos.

Para realizar comparaciones de igualdad, en otros lenguajes se suele utilizar el operador de igualdad (==), que en JavaScript devolverá true si los operandos son iguales, pero no se realiza ninguna comprobación de tipo. Para una comparación más exhaustiva que tenga en cuenta el tipo de los operandos, se recomienda utilizar e l operador de igualdad estricta (===) que, además de tener en cuenta el valor, tiene en cuenta el tipo. En JavaScript podemos encontrar además de estos, otros operadores muy similares a los que hay en cualquier lenguaje de programación.

**Figura 15.**

Operadores de comparación.



Además, también es posible combinar sentencias como las anteriores mediante operadores lógicos como pueden ser || (OR lógico) y && (AND lógico). El comportamiento de estos operadores es idéntico al que podemos observar en otros lenguajes de programación.

**Figura 16**

Usando varios operadores lógicos.



# Recursos complementarios

* Documentación de JavaScript

<https://developer.mozilla.org/es/docs/Web/JavaScript>

# Referencias

Brown, A. R., & Wilson, S. (2013). *JavaScript Testing with Jasmine: JavaScript Behavior-Driven Development.* O'Reilly Media.

Crockford, D. (2008)*. JavaScript: The Good Parts*. O'Reilly Media.

Duckett, J. (2011). *JavaScript & jQuery*. John Wiley & Sons Books.

Flanagan, D. (2011). *JavaScript: The Definitive Guide: Activate Your Web Pages (6th ed.).* O'Reilly Media.

Freeman, A., & Robson, S. (2012). *Head First JavaScript Programming: A Brain-Friendly Guide.* O'Reilly Media.

McPeak, J., Kuksenok, K., & Robinson, C. (2012). *Learning Node: Moving to the Server-Side*. O'Reilly Media.

Osmani, A. (2013). *Developing Backbone.js Applications*. O'Reilly Media.

Resig, J., Bibeault, B., & Marconcini, J. (2013). *Secrets of the JavaScript Ninja*. Manning Publications.

# Autoevaluación

1. **¿Cuál de las siguientes palabras clave se utiliza para declarar una variable en JavaScript?**

new

let

declare

var

1. **¿Qué tipo de variable se declara sin la palabra clave "var", "let" o "const" y se asigna automáticamente a la variable global?**

Local variable

Const variable

Implicit variable

Implicit global variable

1. **¿Cuál de los siguientes no es un tipo de dato primitivo en JavaScript?**

String

Array

Boolean

Number

1. **¿Cómo se verifica el tipo de dato de una variable en JavaScript?**

typeOf

checkType

typeof

variableType

1. **¿Cuál es la diferencia entre "undefined" y "null" en JavaScript?**

Son iguales

"Undefined" es un tipo de dato, mientras que "null" es un valor asignado explícitamente para indicar la ausencia de valor

"Null" es un tipo de dato, mientras que "undefined" es un valor asignado explícitamente para indicar la ausencia de valor

Ambos representan un valor indefinido

1. **¿Cuál es la forma correcta de declarar una constante en JavaScript?**

const myConstant = 10;

let myConstant = 10;

var myConstant = 10;

constant myConstant = 10;

1. **¿Qué método se utiliza para convertir una cadena a un número en JavaScript?**

parseInt()

toNumber()

convertToNumber()

stringToNumber()

1. **¿Cuál de las siguientes declaraciones es correcta para comparar tanto el valor como el tipo de dos variables?**

x == y

x === y

x = y

x !== y

1. **¿Cuál es la diferencia entre "let" y "const" al declarar variables en JavaScript?**

"let" no permite la reasignación, mientras que "const" sí lo permite

"const" no permite la reasignación, mientras que "let" sí lo permite

Ambos permiten la reasignación

Ninguna de las anteriores

1. **¿Cuál es el resultado de la expresión 5 + "5" en JavaScript?**

10

"55"

55

Error