# Introducción

Este texto presenta JavaScript y analiza algunos de sus conceptos fundamentales.

## ¿Qué debes conocer previamente?

Esta guía presume que tienes los siguientes antecedentes básicos:

- **Comprensión general de Internet** y la (World Wide Web).
- **Buen conocimiento práctico** del lenguaje de marcado de hipertexto (HTML).
- **Alguna experiencia en programación**. Si eres nuevo en la programación, prueba uno de los tutoriales vinculados en la página principal sobre JavaScript.

## Dónde encontrar información sobre JavaScript

La documentación de JavaScript en MDN incluye lo siguiente:

- Aprende desarrollo web proporciona información para principiantes e introduce conceptos básicos de programación e Internet.
- La Guía de JavaScript (esta guía) proporciona una descripción general sobre el lenguaje JavaScript y sus objetos.
- La Referencia de JavaScript proporciona material de referencia detallado para JavaScript.

Si eres nuevo en JavaScript, comienza con los artículos en el área de aprendizaje y la Guía de JavaScript. Una vez que tengas una firme comprensión de los fundamentos, puedes usar la Referencia de JavaScript para obtener más detalles sobre objetos y declaraciones individuales.

## ¿Qué es JavaScript?

JavaScript es un **lenguaje de programación multiplataforma orientado a objetos** que se utiliza para hacer que las páginas web sean interactivas (p. ej., Que tienen animaciones complejas, botones en los que se puede hacer clic, menús emergentes, etc.). También hay versiones de JavaScript de **lado del servidor** más avanzadas, como **Node.js**, que te permiten agregar más funcionalidad a un sitio web que

simplemente descargar archivos (como la colaboración en tiempo real entre varias computadoras).

JavaScript contiene una biblioteca estándar de objetos, como **Array, Date y Math**, y un conjunto básico de elementos del lenguaje como **operadores, estructuras de control y declaraciones**. El núcleo de JavaScript se puede extender para una variedad de propósitos completándolo con objetos adicionales; por ejemplo:

- *JavaScript de lado del cliente* extiende el núcleo del lenguaje al proporcionar objetos para controlar un navegador y su *Modelo de objetos de documento* (DOM por *Document Object Model*). Por ejemplo, las extensiones de lado del cliente permiten que una aplicación coloque elementos en un formulario HTML y responda a eventos del usuario, como clics del mouse, formularios para ingreso de datos y navegación de páginas.
- *JavaScript de lado del servidor* amplía el núcleo del lenguaje al proporcionar objetos relevantes para ejecutar JavaScript en un servidor. Por ejemplo, las extensiones de lado del servidor permiten que una aplicación se comunique con una base de datos, brinde continuidad de información de una invocación a otra de la aplicación o realice manipulación de archivos en un servidor.

Esto significa que, en el navegador, JavaScript puede cambiar la apariencia de la página web (DOM). Y, del mismo modo, el JavaScript de **Node.js** en el servidor puede responder a solicitudes personalizadas desde el código escrito en el navegador.

## JavaScript y Java

JavaScript y Java son similares en algunos aspectos, pero fundamentalmente diferentes en otros. El lenguaje JavaScript se parece a Java, pero no tiene el ti**pado estático ni la fuerte verificación de tipos de Java**. JavaScript sigue la mayoría de la sintaxis de las expresiones de Java, convenciones de nomenclatura y construcciones de control de flujo básicas, razón por la cual se cambió el nombre de LiveScript a JavaScript.

A diferencia del sistema de clases en **tiempo de compilación de Java** creado por declaraciones, JavaScript admite un sistema de **tiempo de**

**ejecución** basado en una pequeña cantidad de tipos de datos que representan valores numéricos, booleanos y de cadena.

JavaScript tiene un **modelo de objetos basado en prototipos** en lugar del **modelo de objetos basado en clases más común**. El modelo basado en prototipos proporciona herencia dinámica; es decir, lo que se hereda puede variar en objetos individuales. JavaScript también admite funciones sin requisitos declarativos especiales. Las funciones pueden ser propiedades de objetos, ejecutándose como métodos débilmente tipados.

JavaScript es un lenguaje de forma muy libre en comparación con Java. No es necesario declarar todas las variables, clases y métodos. No tienes que preocuparte por si los métodos son públicos, privados o protegidos, y no tienes que implementar interfaces. Las variables, los parámetros y los tipos de retorno de función no se tipifican explícitamente.

Java es un lenguaje de programación basado en clases diseñado para una ejecución rápida y con seguridad de tipos. La seguridad de tipos significa, por ejemplo, que no puedes convertir un entero de Java en una referencia de objeto o acceder a la memoria privada corrompiendo el código de bytes de Java. El modelo basado en clases de Java significa que los programas constan exclusivamente de clases y sus métodos. La herencia de clases de Java y la tipificación fuerte generalmente requieren jerarquías de objetos estrechamente acopladas. Estos requisitos hacen que la programación Java sea más compleja que la programación JavaScript.

Por el contrario, JavaScript desciende en espíritu de una línea de lenguajes más pequeños de tipado dinámico como HyperTalk y dBASE. Estos lenguajes de «*scripting*» ofrecen herramientas de programación a una audiencia mucho más amplia debido a su sintaxis más sencilla, funcionalidad especializada incorporada y requisitos mínimos para la creación de objetos.

| **JavaScript** | **Java** |
| --- | --- |
| Orientado a objetos. No hay distinción entre tipos de objetos. La herencia se realiza a través del mecanismo de prototipo, y las propiedades y | Basado en clases. Los objetos se dividen en clases e instancias con toda la herencia a través de la jerarquía de clases. Las clases y las instancias no |

| **JavaScript** | **Java** |
| --- | --- |
| métodos se pueden agregar a cualquier objeto de forma dinámica. | pueden tener propiedades o métodos agregados dinámicamente. |
| Los tipos de datos de las variables no se declaran (tipado dinámico, tipado flexible). | Los tipos de datos de las variables se deben declarar (tipado estático, fuertemente tipado). |
| No se puede escribir automáticamente en el disco duro. | Puede escribir automáticamente en el disco duro. |

## JavaScript y la especificación ECMAScript

JavaScript está estandarizado en **Ecma International**, la asociación europea para estandarizar los sistemas de información y comunicación (ECMA antes era un acrónimo para la Asociación Europea de Fabricantes de Computadoras) para ofrecer un lenguaje de programación internacional estandarizado basado en JavaScript. Esta versión estandarizada de JavaScript, denominada ECMAScript, se comporta de la misma manera en todas las aplicaciones que admiten el estándar. Las empresas pueden utilizar el lenguaje estándar abierto para desarrollar su implementación de JavaScript. El estándar ECMAScript está documentado en la especificación ECMA-262.

El estándar ECMA-262 también está aprobado por ISO (Organización Internacional de Normalización) como ISO-16262. También puedes encontrar la especificación en el sitio web de Ecma International. La especificación ECMAScript no describe el modelo de objetos de documento (DOM), que está estandarizado por el World Wide Web Consortium (W3C) y/o WHATWG (Grupo de trabajo de tecnología de aplicaciones de hipertexto web). El DOM define la forma en que los objetos de documentos HTML se exponen a tu «script». Para tener una mejor idea de las diferentes tecnologías que se utilizan al programar con JavaScript, consulta el artículo Descripción de las Tecnologías JavaScript.

JavaScript es un lenguaje de programación, mientras que ECMAScript es un estándar que define un conjunto de reglas para escribir código JavaScript y asegurar que este funcione de la misma manera en todos los sistemas

## Documentación de JavaScript versus especificación de ECMAScript

La especificación ECMAScript es un conjunto de requisitos para implementar ECMAScript. Es útil si deseas implementar funciones del lenguaje compatibles con los estándares en tu implementación o motor ECMAScript (como SpiderMonkey en Firefox o V8 en Chrome).

El documento **ECMAScript *no* está destinado a ayudar a los programadores** de scripts. **Utiliza la documentación de JavaScript para obtener información al escribir tus scripts**.

La especificación ECMAScript utiliza terminología y sintaxis que puede resultar desconocida para un programador de JavaScript. Aunque la descripción del lenguaje puede diferir en ECMAScript, el lenguaje en sí sigue siendo el mismo. JavaScript admite todas las funciones descritas en la especificación ECMAScript.

La documentación de JavaScript describe aspectos del lenguaje que son apropiados para un programador de JavaScript.

## Cómo empezar con JavaScript

Comenzar con JavaScript es fácil: todo lo que necesitas es un navegador web moderno. Esta guía incluye algunas funciones de JavaScript que solo están disponibles actualmente en las últimas versiones de Firefox, por lo que se recomienda utilizar la versión más reciente de Firefox.

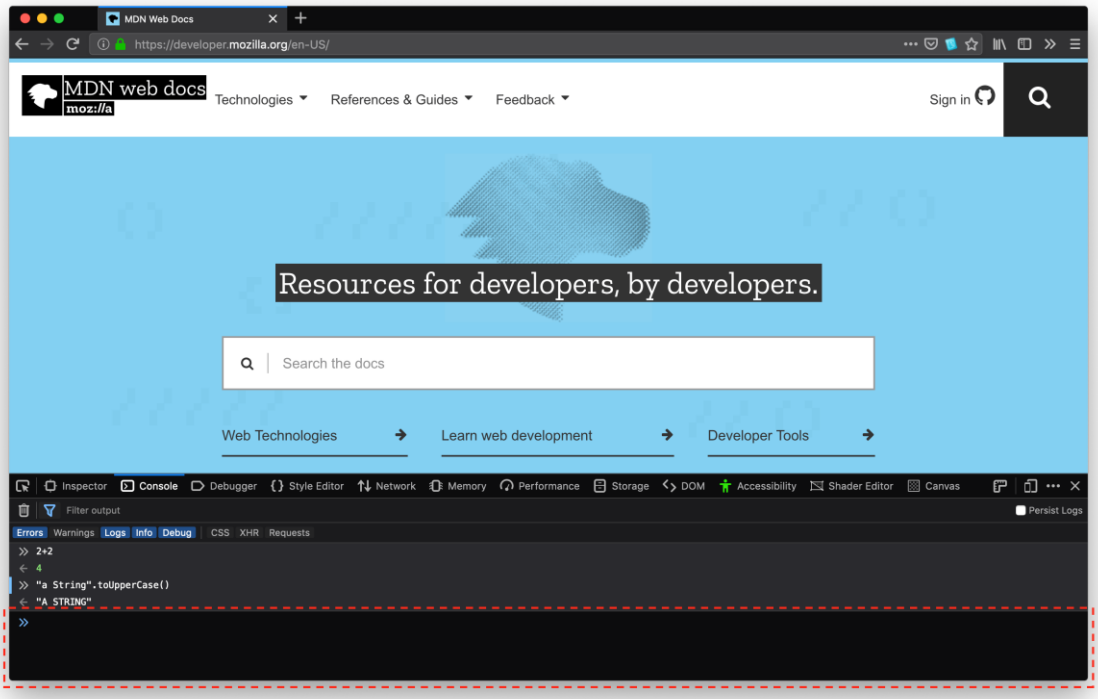
La herramienta *Consola web* integrada en Firefox es útil para experimentar con JavaScript; Puedes usarla en dos modos: modo de entrada unilínea y modo de entrada multilínea.

## Entrada unilínea en la consola web

La Consola web te muestra información sobre la página web cargada actualmente, y también incluye un intérprete de JavaScript que puedes usar para ejecutar expresiones de JavaScript en la página actual.

Para abrir la Consola web (Ctrl+Mayús+I en Windows y Linux o Cmd-Opción-K en Mac), abre el menú **Herramientas** en Firefox y selecciona "**Desarrollador ▶ Consola web**".

La consola web aparece en la parte inferior de la ventana del navegador. En la parte inferior de la consola hay una línea de entrada que puedes usar para ingresar JavaScript, y la salida aparece en el panel de arriba:




La consola funciona exactamente de la misma manera que eval: devuelve la última expresión ingresada. En aras de la simplicidad, te puedes imaginar que cada vez que ingresas algo en la consola, en realidad estás rodeado por console.log alrededor de eval, así:

JSCopy to Clipboard

```
function greetMe(tuNombre) {
  alert("Hola " + tuNombre);
}
console.log(eval("3 + 5"));
```

**Entrada multilínea en la consola web**

El modo de entrada unilínea de la consola web es ideal para realizar pruebas rápidas de expresiones JavaScript, pero aunque puedes ejecutar varias líneas, no es muy conveniente para eso. Para JavaScript más complejo, puedes utilizar el modo de entrada multilínea.

## Hola mundo

Para comenzar a escribir JavaScript, abre la Consola web en modo multilínea y escribe tu primer código "Hola mundo" en JavaScript:

JSCopy to Clipboard

```
(function () {
  "use strict";
  /* Inicio de tu código */
  function greetMe(tuNombre) {
    alert("Hola " + tuNombre);
  }

  greetMe("Mundo");
  /* Fin de tu código */
})();
```

Presiona Cmd+Intro o Ctrl+Intro (o haz clic en el botón **Ejecutar**), ¡para ver cómo se desarrolla en tu navegador!

En las siguientes páginas, esta guía te presenta la sintaxis de JavaScript y las características del lenguaje, de modo que puedas escribir aplicaciones más complejas.

Pero por ahora, recuerda incluir siempre el (function() { "use strict"; antes de tu código, y agrega })(); al final de tu código. Aprenderás qué significa IIFE , pero por ahora puedes pensar que hacen lo siguiente:

1. Mejoran enormemente el rendimiento.
2. Evitan la semántica estúpida en JavaScript que hace tropezar a los principiantes.

3. Evitan que los fragmentos de código ejecutados en la consola interactúen entre sí (por ejemplo, que algo creado en una ejecución de consola se utilice para una ejecución de consola diferente).

## Obtener ejemplos de código de developer.mozilla

Los ejemplos de código que encontrará en el Área de aprendizaje están todos disponibles en GitHub. Si desea copiarlos todos en su computadora, la forma más fácil es descargar un ZIP de la última rama del código principal.

Si prefiere copiar el repositorio de una manera más flexible que permita actualizaciones automáticas, puede seguir las instrucciones a continuación:

1. Instalar Git en su máquina. Este es el software del sistema de control de versiones subyacente sobre el que funciona GitHub.
2. Abra el símbolo del sistema (Windows) o terminal (Linux, macOS) de su computadora.
3. Para copiar el repositorio del área de aprendizaje en una carpeta llamada área de aprendizaje en la ubicación actual a la que apunta su símbolo del sistema/terminal, usa el siguiente comando:

   BASHCopy to Clipboard

   git clone https://github.com/mdn/learning-area
4. Ahora puedes ingresar al directorio y encontrar los archivos que buscas (ya sea usando su Finder/Explorador de archivos o el comando cd) .

Puedes actualizar el repositorio learning-area con cualquier cambio realizado en la versión principal en GitHub con los siguientes pasos:

1. En el símbolo del sistema/terminal, ve al directorio learning-area usando cd. Por ejemplo, si estuviera en el directorio principal:

   BASHCopy to Clipboard

   cd learning-area

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Introduction

2. Actualice el repositorio usando el siguiente comando:

   BASHCopy to Clipboard

   git pull

https://www.youtube.com/watch?v=bQciY0LFQvo&list=PL2Z95CSZ1N4HXvLWg8oL4IpyJx27HafcD&index=4