
Introducción a JavaScript

JavaScript es un lenguaje de programación que apareció por primera vez en 1995 como el lenguaje de secuencias de comandos (scripting) para el navegador Netscape. Desde entonces, se ha convertido en uno de los lenguajes de programación más utilizados del mundo. Si bien su objetivo inicial era agregar interactividad a los sitios web, desde entonces ha llegado a hacer casi todo, incluida la creación de aplicaciones de escritorio y API de back-end.

JavaScript está en todas partes, y a lo largo de los años, se han construido muchos frameworks usándolo como base, como jQuery, React, Vue.js y Svelte. Todo esto puede hacer que el aprendizaje de JavaScript te intimide, ya que a menudo hay muchas formas diferentes de lograr lo mismo.

En esta parte del curso, cubriremos cómo funciona JavaScript a un nivel fundamental. Eso hará que sea más fácil entender cómo funcionan frameworks como React y Vue.js. Discutiremos por qué las cosas funcionan como lo hacen en JavaScript y las diversas peculiaridades que vienen con años de desarrollo continuo en el lenguaje.

JavaScript hoy en general se divide en dos categorías principales:

- **Código del lado del cliente**, al que nos referiremos como el “front end”, que se incluye a través de archivos HTML y agrega interactividad a los sitios web. El código del lado del cliente se carga y procesa directamente en el navegador o dispositivo web de alguien. En la práctica, esto significa escribir un archivo .js o archivo HTML que contiene JavaScript y cargarlo directamente en tu navegador web.
- **Código del lado del servidor**, al que nos referiremos como “back end”, que se utiliza para escribir servidores web. Este tipo de código está escrito en servidores web y se puede usar para crear cosas como API que el usuario final nunca ve directamente. Utiliza un tiempo de ejecución para ejecutar el código en los servidores web. El tiempo de ejecución más común utilizado para esto se llama Node.js. El código del lado del servidor se carga y procesa en un servidor que está separado del dispositivo del usuario.

Todo lo que discutimos será aplicable tanto al front-end como al back end, pero nuestro enfoque estará en JavaScript front-end ya que allí es donde comenzó JavaScript. A veces tocamos JavaScript de forma back-end cuando sea necesario para que ciertos conceptos puedan entenderse más fácilmente.

Fundamentos de JavaScript

JavaScript se basa en un estándar de lenguaje llamado ECMAScript. La forma en que JavaScript debería funcionar exactamente se documenta en un estándar específico llamado ECMA-262. Dado que ECMAScript no proporciona ninguna herramienta para compilar JavaScript, cada implementación de JavaScript crea su propia versión de JavaScript. Eso incluye su navegador y compiladores de back-end como Node.js.

En su mayor parte, estas implementaciones siguen a ECMA-262, pero dado que cada implementación es realizada por diferentes equipos, puede haber algunas discrepancias menores o diferentes conjuntos de características dependiendo del navegador o implementación.

En este primer documento, cubriremos cómo puedes prepararte para comenzar a usar JavaScript, incluido cómo configurar proyectos de JavaScript cuando uses Node.js por ejemplo. En los documentos futuros, exploraremos cómo escribir el código JavaScript.

Clasificación de tipo de JavaScript

La forma en que se escribe un lenguaje generalmente nos da una idea amplia de cómo funciona. Como la mayoría de los otros lenguajes, JavaScript te permite definir “variables” para almacenar datos, y estas variables tienen tipos. “Tipo” se refiere al tipo de datos que se utilizan. Por ejemplo, un número es de tipo Number, y se conoce una mezcla de caracteres y/o números como que tiene un tipo llamado String.

Si has usado otros lenguajes, JavaScript puede parecer diferente ya que está debilitado en sus tipos (weakly typed). Eso significa que si bien otros lenguajes requieren que menciones explícitamente en el código qué tipo de datos son diferentes variables, JavaScript no. JavaScript también a menudo se conoce como **tipado dinámicamente**, lo que significa que interpretará dinámicamente qué tipo de datos se basan en el contexto en el que lo encuentra.

Para comprender esto mejor, veamos cómo se definen las variables en JavaScript. Por lo general, las definimos así:

```
let x = "Alguna cadena"  
let y = 5  
let z = false
```

Verás que no se definen tipos aquí. Por ejemplo, no tuvimos que mencionar que "Alguna cadena" era un String. JavaScript determina los tipos basados en el contexto, por lo que

tomará `X` como una cadena simplemente porque ponemos su valor en las comillas. Del mismo modo, se interpretará y dinámicamente como un tipo número, ya que carece de comillas y `z` como de tipo booleano, ya que no tiene comillas y usa la palabra clave `false`.

Esto hace que JavaScript sea bastante fácil de recoger, pero bastante difícil de dominar. La falta de una tipificación fuerte puede significar que, sin saberlo, creas errores en tu software, ya que JavaScript no siempre arrojará errores si aparecen tipos inesperados, y lo que es peor, JavaScript puede interpretar dinámicamente los tipos incorrectamente en algunos casos.

Para aplicaciones más complejas con muchos casos de prueba, los desarrolladores a menudo fomentan el uso de TypeScript en lugar de JavaScript por este motivo. TypeScript es JavaScript, pero extendido. Está firmemente escrito, lo que significa que los tipos deben mencionarse en tu código.

¿Para qué se usa JavaScript?

Como mencionamos en la introducción, JavaScript puede ser compilado y utilizado de dos maneras principales. El primero es crear experiencias interactivas con front end, justo en tu navegador. El segundo es como código de servidor de back-end. JavaScript front-end es representado por el navegador dentro de las páginas web, mientras que el código del servidor requiere un tiempo de ejecución como Node.js, que compila el código que escribes para ejecutarlo directamente en el servidor.

Cuando estás dentro del navegador, algunas de las principales cosas que JavaScript puede hacer son

- Agregar, cambiar o eliminar CSS/HTML cuando un usuario interactúa con algo.
- Creación de nuevas etiquetas HTML programáticamente.
- Seguimiento de la acción del usuario y la producción de comentarios al usuario (como mostrar una ventana emergente cuando un usuario hace clic en algo).
- Almacenar datos para el usuario localmente a través de almacenamiento local o cookies.
- Creación de experiencias de usuario de una sola página donde no se necesita refrescarse de página.

En el back end, los principales casos de uso son

- Crear rutas/puntos finales de URL en un servidor y dictar qué sucede si un usuario navega allí.

- Creación de rutas/puntos finales de URL para APIs (interfaces de programación de aplicaciones), por lo que podemos enviar y recibir datos hacia y desde el servidor.
- Creación de servidores WebSocket, con los que los usuarios pueden interactuar desde su experiencia front-end. Estos se pueden usar para hacer cosas como salas de chat.
- Comprimir páginas web pre-renderizadas para una experiencia web más rápida.
- Manipular datos enviados al servidor (a veces a través de WebSocket o API) y almacenarlos en una base de datos de back-end.

Escribiendo Javascript

JavaScript en el front end se encuentra dentro de HTML en las páginas web. Como tal, la familiaridad con HTML es bastante importante cuando trabajamos con JavaScript. Para crear tu primer archivo que contenga JavaScript, puedes comenzar haciendo un archivo .html. Por lo general, llamamos a la página de inicio de un sitio web index.html al crear sitios web, por lo que, para este ejemplo, crea un nuevo archivo HTML llamado index.html.

Los archivos .html pueden ser abiertos por cualquier navegador web, como Google Chrome. Puedes editar tu archivo HTML abriéndolo en un texto o editor de código (Notepad++ incluido) y poniendo en este estándar “boilerplate” HTML:

```
<!DOCTYPE html>
<html>
<head>
  <title>My First JavaScript</title>
</head>
<body>
<p>Hello World</p>
<script type="text/javascript">
  // This is JavaScript!
</script>
</body>
</html>
```

Nota: Puedes usar cualquier editor de texto (como Notepad++) para crear HTML e incluso JavaScript. Si bien eso funciona bien, es mejor usar un editor de código profesional. Uno de los editores de código más populares utilizados en la comunidad de desarrollo de software es VS Code. Puedes descargarlo a través de <https://code.visualstudio.com/>. Esto codificará tu JavaScript y te dará muchas otras características útiles.

En el ejemplo anterior, insertamos una etiqueta <script> dentro de nuestro cuerpo HTML. La etiqueta <script> es donde va nuestro JavaScript:

```
<script type="text/javascript">  
  // This is JavaScript!  
</script>
```

Dado que las aplicaciones JavaScript pueden ser bastante largas, puedes ver esta etiqueta `<script>` sustituida para un archivo. Esto puede ser útil ya que nos permite separar nuestro HTML y JavaScript en diferentes archivos.

Por ejemplo, si tuviéramos un archivo JavaScript separado llamado `myScript.js` almacenado en la misma carpeta que el archivo `index.html`, podríamos cargarlo en nuestro documento HTML utilizando el atributo `src` en la etiqueta de `script`:

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

También puedes ver JavaScript integrado en HTML a través de los atributos de las etiquetas HTML. Por ejemplo, JavaScript se puede colocar dentro de un botón para que algo suceda cuando un usuario hace clic en ese botón:

```
<button onclick="//JavaScript here"></button>
```

Configuración de un editor de código

A lo largo de esta parte del curso, escribiremos código utilizando el Visual Studio Code, Notepad++. Si no tienes un editor de código preferido, recomendaría encarecidamente usar VS Code, que puedes descargar a través del sitio web oficial (<https://code.visualstudio.com>).

Aunque es posible construir una carrera de software a través del uso de Notepad++, no se recomienda. Los editores de código modernos como VS Code te brindan una tonelada de características útiles, que incluyen un resaltado de código, lo que hace que sea más fácil leer lo que estás escribiendo. También vienen con características más avanzadas como terminales incorporadas, para cuando necesitas esas cosas.

Si descargas VS Code, serás recibido con una pantalla que se parece a la que se muestra en la figura 1. A partir de ahí, podrás hacer nuevos archivos y carpetas para almacenar tus proyectos.

Inicio de nuestro código

Lo divertido de escribir JavaScript front-end es que una vez que guardes tu archivo `.html`, puedes cargarlo directamente en cualquier navegador web para probarlo.

Pruébalo abriendo index.html en tu navegador web, ya sea arrastrándolo a la ventana o abriéndolo a través de Archivo ► Abrir archivo ...

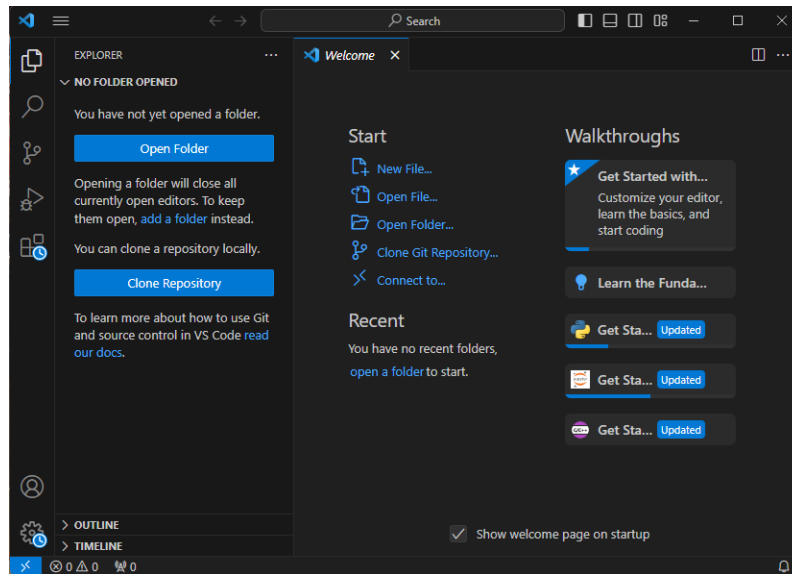


Figura 1: Después de instalar VS Code, abre la aplicación. Ve a File ► Open Folder... o hacer clic en "Open Folder" en VS Code y encontrar tu carpeta "My JavaScript".

Para ver tu código en el navegador en sí, puedes hacer clic derecho en la página web en cualquier lugar y elegir "Inspeccionar" en Google Chrome. Si Chrome no es tu navegador preferido, también existen funciones similares en otros navegadores. El uso de "Inspeccionar" también te permitirá ver información complementaria sobre lo que has creado, como lo muestra la figura 2.

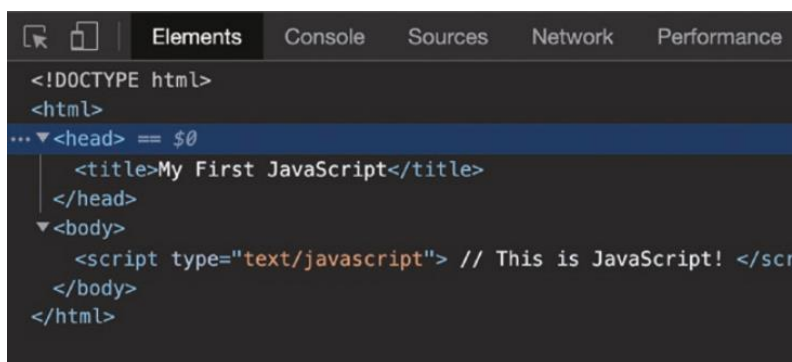


Figura 2: "Inspect" o "Inspect Element" te permite ver más información sobre tu código en el navegador web.

"Inspeccionar" o "Inspect Element" es una herramienta vital en el desarrollo web. A lo largo de este documento, utilizaremos la pestaña de la consola dentro de las herramientas de

desarrollador para probar el código y ver los resultados del código que ejecutamos. Como tal, es muy útil familiarizarse con esta pantalla ahora.

Cómo comenzar con la escritura de JavaScript de back-end

Ahora hemos discutido cómo JavaScript se puede ejecutar dentro de un navegador web como Google Chrome. Antes de ir más allá, veamos brevemente cómo ejecutamos a JavaScript back-end también, que ocasionalmente tocaremos en este curso.

JavaScript back end se ejecuta directamente en tu computadora o en un servidor web en lugar de dentro de un navegador web. Para ejecutar JavaScript así, debes usar un runtime como Node.js.

Node.js se puede descargar e instalar en tu computadora a través del sitio web oficial de Node.js (<https://nodejs.org/en/download>). Después de que se haya instalado, podrás ejecutar el código JavaScript directamente desde una ventana de terminal ejecutando tu archivo .js.

Puedes hacerlo abriendo el terminal o línea de comando en tu computadora, a la que se puede acceder en macOS a través de la aplicación Terminal o en Windows a través de la aplicación "cmd".

Ejecución del comando node te permite compilar y ejecutar archivos JavaScript. Vamos a probarlo: crea un archivo llamado index.js en un editor de código o bloc de notas, y agrega el siguiente código JavaScript antes de guardar:

```
console.log ("¡Hola mundo!")
```

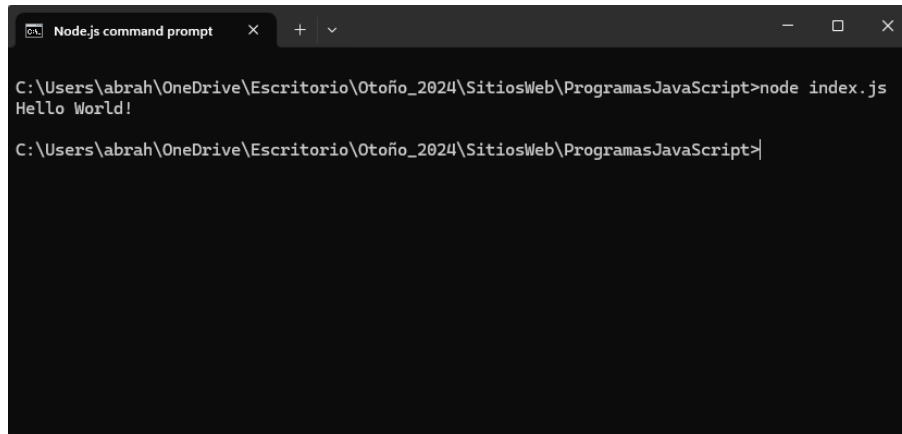
Luego puedes ejecutar este archivo utilizando el comando node en la terminal:

```
node index.js
```

Esto producirá una salida que se parece a lo que se muestra en la figura 3.

Nota: Si guardaste tu archivo index.js en otro directorio, deberás proporcionar el enlace completo del directorio. Para navegar por los directorios, usa el comando cd. Por ejemplo, si tu archivo index.js estaba en "/Users/Pepito/Documents/", ejecutarías cd /Users/Pepito/Documents/ y solo después de eso, ejecuta node index.js.

Las aplicaciones Node.js como esta se usan con frecuencia para crear APIs, que no cubriremos en este curso.



```
Node.js command prompt
C:\Users\abrah\OneDrive\Escritorio\Otoño_2024\SitiosWeb\ProgramasJavaScript>node index.js
Hello World!
C:\Users\abrah\OneDrive\Escritorio\Otoño_2024\SitiosWeb\ProgramasJavaScript>|
```

Figura 3: Ejecutar un script JavaScript desde la línea de comandos es tan simple como usar el comando "node" seguido del enlace del directorio al archivo que deseas ejecutar.

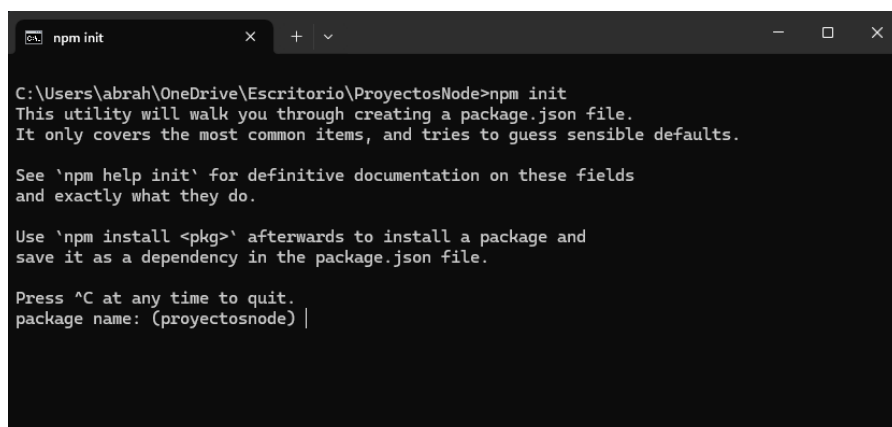
Creación de proyectos Node.js JavaScript

En el ejemplo anterior, ejecutamos un solo archivo usando Node.js, es más común, sin embargo, crear un proyecto Node.js cuando inicias algo nuevo en Node.js. Esto también se realiza a través de la terminal o el cmd. El primer paso es hacer una nueva carpeta y navegar a ella usando el comando `cd`.

En este ejemplo, creamos una carpeta llamada "ProyectosNode" en mi escritorio y lo navegamos usando el siguiente comando en la Terminal:

```
cd ~/Desktop/ProyectosNode
```

Una vez hecho esto, puedes usar el comando `npm init`, que se instala junto con Node.js, para iniciar tu proyecto. Puedes ver como se ve en la figura 4.



```
npm init
C:\Users\abrah\OneDrive\Escritorio\ProyectosNode>npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See 'npm help init' for definitive documentation on these fields
and exactly what they do.

Use 'npm install <pkg>' afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (proyectosnode) |
```

Figura 4: Cuando usas el comando `npm init`, se te pedirá que ingreses información sobre tu nuevo proyecto como se muestra anteriormente.

Todo lo que tienes que hacer ahora es escribir respuestas a cada pregunta y presionar Entrar. Por ejemplo, la primera pregunta te pedirá cómo deseas llamar a su proyecto, así que escribe el nombre de tu proyecto y presiona Entrar.

Tu carpeta ahora contendrá un archivo `paquete.json` que resume la información que proporcionaste. Dado que inicializaste tu proyecto Node.js, ahora podrás ejecutar otros comandos como `npm install` ahora, lo que te permite instalar dependencias de terceros.

Soporte de JavaScript

El software tradicional generalmente es escrito por un desarrollador y se descarga en la computadora o dispositivo de un usuario. Este es el caso con cosas como videojuegos, aplicaciones en tu teléfono o grandes aplicaciones como Adobe Photoshop.

Al escribir código en JavaScript, las cosas son muy diferentes. El software que instala el usuario es el navegador, ¡no tu sitio web! El navegador luego carga su página web dentro de él. Dado que todos tienen su propia preferencia del navegador, y no todos mantienen sus navegadores actualizados, JavaScript que funciona en un navegador a menudo no puede funcionar en otro. Por ejemplo, Firefox puede admitir una nueva función de JavaScript, pero Chrome puede no. Lo peor de esto es que realmente no puedes usar una nueva función de JavaScript front end hasta que la mayoría de los navegadores la hayan implementado.

Si viene de otros lenguajes, entonces preocuparse por el apoyo del navegador será un concepto extranjero para ti. En JavaScript, es algo real.

En los últimos tiempos, dado que la mayoría de los navegadores son “perennes” (lo que significa que se actualizan automáticamente), esto se ha convertido en un problema menos de lo que solía ser, pero a veces diferentes navegadores simplemente no están de acuerdo sobre lo que debería y no debería implementarse. Las nuevas características prometedoras pueden terminar implementadas en Just Chrome, solo Safari o simplemente Firefox.

A lo largo de esta parte del curso, solo miraremos a JavaScript con un amplio soporte de navegador, lo que significa que no necesitas preocuparte si puedes o no puedes usarlo. Sin embargo, cuando comienzas a explorar JavaScript en tu propio tiempo, y especialmente cuando se busca una funcionalidad más avanzada, es importante verificar si los navegadores lo admiten. Puedes encontrar buenas tablas de soporte de navegador en sitios web como <https://caniuse.com/> o <https://desarrollador.mozilla.org/>.

Se puede encontrar un ejemplo de una tabla de soporte del navegador en la figura 5, para la función GPU.

	🖥️					📱					
	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android
GPU 🏗️	✓	✓	🔷	✗	✗	✗	✗	✗	✗	✗	✗
	113	113	Nightly	No	No	No	No	No	No	No	No
	*	*		*							
<code>getPreferredCanvasFormat</code> 🏗️	✓	✓	🔷	✗	✗	✗	✗	✗	✗	✗	✗
	113	113	Nightly	No	No	No	No	No	No	No	No
	*	*		*							

Figura 5: No todos los navegadores admiten cada nueva función de JavaScript. En el ejemplo anterior, solo Chrome y Edge tienen soporte. Otros navegadores solo tienen apoyo parcial o ninguno. Eso significa que, si intentaras implementar esto en un sitio web, ¡solo algunos usuarios podrían usarlo!

Resumen

En este primer documento, hemos visto cómo configurar tu espacio de trabajo para comenzar a escribir código con JavaScript. Hemos discutido para qué se usa JavaScript y algunas de las trampas o diferencias entre él y otros lenguajes. Ahora que hemos cubierto los conceptos básicos, veamos cómo escribir el código JavaScript.