Un bloque de código se presenta de la siguiente manera:

html

CopiarEditar

<html>

<script type="text/javascript">

alert("¡Hola!");

</script>

</html>

Cualquier entrada o salida en la línea de comandos se escribe así:

javascript

CopiarEditar

console.log("Hola mundo!")

**Negrita**: Indica un término nuevo, una palabra importante o palabras que ves en pantalla, por ejemplo, en menús o cuadros de diálogo, también aparecen en el texto de esta forma. Por ejemplo:

"Si haces clic derecho y seleccionas **Inspeccionar** en sistemas macOS, verás que aparece una pantalla similar a la de la siguiente captura."

**Ponte en contacto**

Siempre agradecemos los comentarios de nuestros lectores.

* **Comentarios generales**: Envía un correo a feedback@packtpub.com e indica el título del libro en el asunto de tu mensaje.
* **Preguntas**: Si tienes dudas sobre algún aspecto de este libro, escríbenos a questions@packtpub.com.

**Erratas**

Aunque hemos puesto todo nuestro empeño en asegurar la precisión del contenido, pueden ocurrir errores. Si encuentras alguno en este libro, te agradeceríamos que nos lo reportaras.  
Por favor, visita:  
🔗 <http://www.packtpub.com/submit-errata>,  
selecciona tu libro, haz clic en el enlace del **Formulario de envío de erratas** y proporciona los detalles.

⚠️ **Advertencias o notas importantes** aparecen de esta forma.

💡 **Consejos útiles** aparecen así.

**Piratería**

Si encuentras copias ilegales de nuestras obras en cualquier formato en Internet, te agradeceríamos que nos informaras de la dirección o nombre del sitio web donde se encuentran.  
Por favor, contáctanos en copyright@packtpub.com con un enlace al material.

**Si estás interesado en convertirte en autor**

Si dominas un tema y estás interesado en escribir o colaborar en un libro, por favor visita:  
🔗 <http://authors.packtpub.com>

**Comparte tus opiniones**

Una vez que hayas leído *JavaScript de Principiante a Profesional*, ¡nos encantaría conocer tu opinión!  
Haz clic [aquí](https://www.amazon.com/dp/1800562521) para ir directamente a la página de reseñas en Amazon de este libro y compartir tus comentarios.

Tu reseña es importante para nosotros y para la comunidad tecnológica, y nos ayudará a asegurarnos de ofrecer contenido de excelente calidad.

**Comenzando con JavaScript**

¡Parece que has decidido empezar a aprender JavaScript. ¡Excelente elección!\*\*  
JavaScript es un lenguaje de programación que puede usarse tanto del lado del servidor como del lado del cliente en las aplicaciones.

El **lado del servidor** de una aplicación es la lógica del backend, que normalmente se ejecuta en computadoras ubicadas en centros de datos y que interactúa con la base de datos.  
El **lado del cliente**, por otro lado, es lo que se ejecuta en el dispositivo del usuario, y en el caso de JavaScript, suele ser el navegador web.

Es muy probable que ya hayas usado funcionalidades escritas en JavaScript. Si alguna vez has utilizado un navegador web como **Chrome**, **Firefox**, **Safari** o **Edge**, entonces definitivamente lo has hecho.  
**JavaScript está en todas partes en la web**.

Por ejemplo:

* Si entras en una página web y te aparece un mensaje pidiéndote aceptar cookies y al hacer clic en "Aceptar" el mensaje desaparece... eso es JavaScript en acción.
* Si navegas por un sitio web y se abre un submenú cuando pasas el mouse... más JavaScript.
* ¿Filtraste productos en una tienda en línea? También es JavaScript.
* ¿Te apareció un chat automático después de unos segundos en una web? ¡Lo adivinaste! Es JavaScript.

Prácticamente **cualquier interacción** que tenemos con una página web se debe a JavaScript:  
Los botones que haces clic, las tarjetas de cumpleaños que creas, los cálculos que realizas…  
Cualquier cosa que requiera algo más que una página estática, necesita JavaScript.

**En este capítulo, cubriremos los siguientes temas:**

* ¿Por qué deberías aprender JavaScript?
* Configuración de tu entorno
* ¿Cómo entiende el navegador el lenguaje JavaScript?

**Uso de la consola del navegador**

* Agregar JavaScript a una página web
* Escribir código en JavaScript

**¿Por qué deberías aprender JavaScript?**

Hay muchas razones por las que deberías querer aprender JavaScript. JavaScript se originó en **1995**, y a menudo se considera **el lenguaje de programación más utilizado**. Esto se debe a que JavaScript es el lenguaje que los navegadores web entienden y soportan de forma nativa.  
Si tienes un navegador web y un editor de texto, **ya tienes todo lo necesario para interpretarlo instalado en tu computadora**.  
Eso sí, existen entornos de desarrollo mejores, y hablaremos de ellos más adelante en este capítulo.

JavaScript es un excelente lenguaje de programación para principiantes, y la mayoría de los desarrolladores de software avanzados conocen al menos algo de JavaScript, ya que seguramente lo han utilizado en algún momento.  
Es una **gran elección para principiantes** por varias razones. La primera es que puedes comenzar a crear aplicaciones realmente geniales **mucho antes de lo que imaginas**.  
Para cuando llegues al **Capítulo 5 (Bucles)**, ya serás capaz de escribir scripts bastante complejos que interactúan con los usuarios. Y al final del libro, podrás crear **páginas web dinámicas** capaces de hacer todo tipo de cosas.

JavaScript se puede usar para escribir **muchos tipos diferentes de aplicaciones y scripts**.  
Por supuesto, se usa para programar en el navegador web, pero también se puede usar en la **lógica interna** de una aplicación (como la comunicación con una base de datos), así como para crear **juegos, scripts de automatización** y muchas otras cosas.

Además, JavaScript permite **diferentes estilos de programación**, es decir, distintas formas de estructurar y escribir código. La forma en la que programes dependerá del propósito de tu script.  
Si nunca antes has programado, es posible que estos conceptos aún no te resulten claros —y no es necesario que lo sean en este momento—, pero vale la pena saber que JavaScript se puede utilizar en paradigmas de programación como:

* **Orientación a objetos (semi o completa)**
* **Programación funcional**
* **Programación estructurada o procedimental**

Una vez que domines los conceptos básicos de JavaScript, tendrás acceso a **una gran variedad de bibliotecas y frameworks** que te harán la vida como desarrollador mucho más fácil, permitiéndote hacer más en menos tiempo.  
Algunos ejemplos muy populares incluyen:

* **React**
* **Vue.js**
* **jQuery**
* **Angular**
* **Node.js**

No te preocupes por aprenderlos ahora; solo considéralos como un **extra que vendrá más adelante**. Algunos de ellos los veremos brevemente al final de este libro.

Finalmente, mencionaremos la **comunidad de JavaScript**. JavaScript es un lenguaje de programación muy popular y muchas personas lo usan.  
Especialmente si eres principiante, no habrá problema para el que no puedas encontrar una solución en internet.

La comunidad de JavaScript es enorme. El popular foro **Stack Overflow** contiene mucha ayuda para todo tipo de problemas de programación y tiene una sección enorme dedicada a JavaScript.  
Te encontrarás visitando esa página frecuentemente mientras buscas soluciones, consejos y trucos.

Si JavaScript es tu primer lenguaje de programación, eres nuevo en toda la comunidad de software y te espera una experiencia muy gratificante.  
Los desarrolladores de software, sin importar el lenguaje que usen, **aman ayudarse entre sí**. Hay foros y tutoriales en línea donde puedes encontrar respuestas para casi todas tus preguntas.  
Como principiante, puede que al principio te cueste entender todas las respuestas, pero **no te rindas**, sigue intentando y aprendiendo, y pronto lo entenderás.

**Configuración de tu entorno**

Existen muchas formas de configurar un entorno para programar en JavaScript. Para empezar, probablemente tu computadora ya tenga lo mínimo necesario para escribir código en JavaScript.  
Sin embargo, te recomendamos hacer tu vida un poco más fácil usando un **IDE**.

**Entorno de Desarrollo Integrado (IDE)**

Un **Entorno de Desarrollo Integrado (IDE)** es una aplicación especial que se usa para escribir, ejecutar y depurar código.  
Puedes abrirlo como cualquier otro programa. Por ejemplo, para escribir un documento de texto abres un programa, seleccionas el archivo correcto y comienzas a escribir. Programar es algo similar: abres el IDE y escribes tu código.

Si quieres ejecutar el código, el IDE normalmente tiene un botón especial para eso. Al presionarlo, el código se ejecuta desde dentro del IDE.  
En el caso de JavaScript, a veces tendrás que abrir el navegador manualmente para ver el resultado.

Un IDE hace más que eso: generalmente tiene **resaltado de sintaxis**, lo que significa que ciertos elementos del código aparecen en colores diferentes, ayudándote a detectar errores fácilmente.  
Otra gran característica es la **sugerencia automática**, donde el editor te ayuda mostrando opciones de código que puedes usar en el lugar donde estás escribiendo. Esto se llama a menudo **autocompletado**.

Muchos IDEs tienen plugins especiales para facilitar el trabajo con otras herramientas y añadir funciones adicionales, por ejemplo, una recarga automática en el navegador.

Hay muchos IDEs disponibles y difieren en lo que ofrecen.  
A lo largo del libro usaremos **Visual Studio Code**, pero eso es solo una preferencia personal.  
Otros IDEs populares al momento de escribir este libro incluyen **Atom**, **Sublime Text** y **WebStorm**.

Hay muchos IDEs y siguen apareciendo nuevos, por lo que es probable que el más popular en el momento en que leas esto no esté en esta lista.  
Existen muchas otras opciones. Puedes hacer una búsqueda rápida en la web de IDEs para JavaScript. Hay algunas cosas a las que debes prestar atención al seleccionar un IDE: asegúrate de que soporte resaltado de sintaxis, depuración (debugging) y autocompletado para JavaScript.

**Navegador web**

También necesitarás un navegador web. La mayoría de los navegadores funcionan perfectamente para esto, pero es mejor no usar **Internet Explorer**, que no soporta las últimas características de JavaScript.  
Dos buenas opciones serían **Chrome** y **Firefox**, ya que soportan las últimas funciones de JavaScript y cuentan con plugins muy útiles.

**Herramientas extra**

Hay muchas cosas adicionales que puedes usar mientras programas, por ejemplo, plugins para el navegador que te ayudarán a depurar o harán que el código sea más fácil de visualizar.  
No necesitas ninguno de estos en este momento, pero mantén la mente abierta cuando te topes con una herramienta que otros recomienden con entusiasmo.

**Editor en línea**

Puede que no tengas acceso a una computadora, quizás solo una tableta, o que no puedas instalar nada en tu laptop.  
Existen excelentes editores en línea para esos casos. No mencionamos ninguno en particular porque están evolucionando rápidamente y probablemente estén obsoletos cuando leas esto.  
Pero si buscas en la web “online JavaScript IDE”, encontrarás muchas opciones en las que puedes comenzar a programar JavaScript y ejecutar el código con solo pulsar un botón.

**¿Cómo entiende el navegador JavaScript?**

JavaScript es un lenguaje **interpretado**, lo que significa que la computadora lo entiende mientras lo ejecuta.  
Algunos lenguajes se procesan antes de ejecutarse, esto se llama **compilación**, pero JavaScript no es así.  
La computadora simplemente interpreta JavaScript en tiempo real.

El “motor” que entiende JavaScript se llamará aquí **intérprete**.

Una página web no está compuesta solo por JavaScript. Las páginas web están escritas en tres lenguajes: **HTML**, **CSS** y **JavaScript**.

* **HTML** determina **qué hay en la página**; es decir, contiene el **contenido**.  
  Si hay un párrafo en la página, el HTML contendrá un elemento de párrafo.  
  Si hay un encabezado, fue HTML quien lo agregó, y así sucesivamente.  
  HTML está formado por elementos, también llamados **etiquetas**. Estas especifican **qué hay en la página**.

Aquí tienes un pequeño ejemplo que creará una página web con el texto “Hello world”:

<html>

<body>

Hello world!

</body>

</html>

En el **Capítulo 9: El Modelo de Objetos del Documento (DOM)**, tendrás una pequeña introducción a HTML, así que **no te preocupes si nunca lo has visto antes**.

* **CSS** se encarga del **diseño o estilo** de la página web.  
  Por ejemplo, si el color del texto es azul, eso se logra con CSS.  
  El tamaño de la fuente, el tipo de letra y la posición en la página también se definen con CSS.
* **JavaScript** es la **pieza final del rompecabezas**.  
  Define **qué puede hacer** la página web y **cómo puede interactuar** con el usuario o con el backend (la parte lógica del servidor).

**ECMAScript**

Cuando empieces a trabajar con JavaScript, tarde o temprano te encontrarás con el término **ECMAScript**.  
Este es el **estándar o especificación oficial** del lenguaje JavaScript.

El estándar actual es **ECMAScript 6 (también conocido como ES6)**.  
Los navegadores usan esta especificación para implementar y soportar JavaScript (además de otros elementos como el **DOM**, que veremos más adelante).  
JavaScript tiene muchas implementaciones que pueden variar ligeramente, pero ECMAScript se considera la **base común** que todas esas implementaciones deben incluir.

**Uso de la consola del navegador**

Puede que ya lo hayas visto, o no, pero los navegadores web tienen una opción incorporada para ver el código que hace posible la página que estás visitando.

Si presionas **F12** en una computadora con Windows mientras estás en el navegador, o haces clic derecho y seleccionas **Inspeccionar** en sistemas macOS, aparecerá una pantalla como la que se muestra en la siguiente imagen:

Puede que funcione de forma ligeramente diferente según el navegador que estés usando en tu computadora, pero hacer **clic derecho** y seleccionar **Inspeccionar** generalmente **funciona bien** para abrir la consola del navegador:

Esta captura de pantalla contiene varias **pestañas en la parte superior**.  
Actualmente estamos viendo la pestaña de **Elementos**, que contiene todo el **HTML y CSS** (¿lo recuerdas?).  
Si haces clic en la pestaña **Consola** (Console), encontrarás en la parte inferior del panel un espacio donde puedes **insertar código directamente**.

Es posible que veas algunos **mensajes de advertencia o errores** en esta pestaña.  
Esto es bastante común, y **no debes preocuparte** si la página sigue funcionando correctamente.

La **consola** es usada por los desarrolladores para **registrar lo que está ocurriendo** y para realizar tareas de **depuración (debugging)**.  
La depuración consiste en **encontrar el problema cuando una aplicación no se comporta como se espera**.  
La consola te da información útil sobre lo que está pasando, especialmente si registras mensajes relevantes.

De hecho, este es el **primer comando** que vamos a aprender:

console.log("Hello world!");

Si haces clic en la pestaña de consola, introduces el código JavaScript anterior y luego presionas **Enter**, verás la salida del código justo allí.  
Se verá como en la siguiente captura de pantalla:

A lo largo del libro, trabajarás mucho con la instrucción console.log() en tu código para **probar fragmentos de código** y **ver los resultados**.  
También existen otros métodos de consola, como:

* console.table(), que **crea una tabla** cuando los datos ingresados pueden presentarse de forma tabular.
* console.error(), que registrará los datos ingresados pero con un **estilo que resalta que se trata de un error**.

**📝 Ejercicio práctico 1.1**

**Trabajando con la consola:**

1. Abre la consola del navegador, escribe 4 + 10 y presiona **Enter**.  
   ¿Qué ves como respuesta?
2. Usa la sintaxis de console.log() y coloca un valor entre los paréntesis redondos.  
   Intenta ingresar tu nombre entre comillas (esto es para indicar que es una **cadena de texto**, lo explicaremos más a fondo en el siguiente capítulo).

Ejemplo:

console.log("TuNombre");

**Agregar JavaScript a una página web**

Hay **dos formas** de vincular JavaScript a una página web:

1. **Escribir el código JavaScript directamente dentro del HTML**, usando dos etiquetas <script>.
2. **Vincular un archivo externo de JavaScript** al archivo HTML usando la etiqueta <script> dentro del <head> de la página HTML.

**Directamente en HTML**

Aquí tienes un ejemplo de cómo escribir una página web muy simple que mostrará una **ventana emergente (popup)** con el mensaje *“Hi there!”*:

<html>

<script type="text/javascript">

alert("Hi there!");

</script>

</html>

Si guardas esto como un archivo con extensión .html y lo abres en tu navegador, obtendrás algo como la siguiente captura de pantalla:

**Figura 1.3:** JavaScript hizo que apareciera este cuadro emergente con el texto "Hi there!"

El comando alert crea una **ventana emergente** que muestra un mensaje.  
Ese mensaje se especifica entre los paréntesis que siguen al comando alert.

Actualmente, tenemos el contenido directamente dentro de nuestras etiquetas <html>, lo cual **no es una buena práctica**.

Deberíamos crear **dos elementos principales dentro de <html>**:

* <head>: donde se escribe la **metainformación** y donde, más adelante, también se conectan archivos externos.
* <body>: donde va el **contenido de la página web**.

Además, debemos informar al navegador qué tipo de documento estamos escribiendo, usando la **declaración <!DOCTYPE>**.

Como estamos escribiendo JavaScript dentro de un archivo HTML, debemos usar:

html

CopiarEditar

<!DOCTYPE html>

Aquí tienes un ejemplo con las buenas prácticas aplicadas:

¿Quieres que te lo escriba completamente ya con <!DOCTYPE>, <head> y <body>?

<!DOCTYPE html> <html>

<head>

<title>This goes in the tab of your browser</title> </head>

<body>

The content of the webpage <script>

console.log("Hi there!"); </script>

</body>

</html>

Este ejemplo de página web mostrará lo siguiente:  
**El contenido de la página web**.

Si miras en la consola del navegador, ¡encontrarás una sorpresa!  
También ha ejecutado el JavaScript y registra el mensaje **Hi there!** en la consola.

**📝 Ejercicio práctico 1.2**

**JavaScript en una página HTML:**

1. Abre tu editor de código y crea un archivo HTML.
2. Dentro de tu archivo HTML, configura las etiquetas básicas: <!DOCTYPE>, <html>, <head> y <body>. Luego agrega las etiquetas <script>.
3. Coloca algo de código JavaScript dentro de las etiquetas <script>.  
   Puedes usar, por ejemplo:

console.log("¡Hola mundo!");

**Vincular un archivo externo a nuestra página web**

También puedes **vincular un archivo JavaScript externo** a tu archivo HTML.  
Esto es considerado una **mejor práctica**, ya que permite organizar mejor el código y **evitar páginas HTML demasiado largas** debido a bloques de JavaScript.

Además, con esta práctica puedes **reutilizar el mismo archivo JavaScript** en varias páginas de tu sitio web sin tener que copiar y pegar el código.

Por ejemplo, si tienes el mismo script en 10 páginas distintas y necesitas hacer un cambio, solo tendrás que modificar **un solo archivo**, en lugar de cambiarlo en cada una de las páginas.

Primero vamos a crear un archivo JavaScript por separado.  
Estos archivos tienen la extensión **.js**.  
Voy a llamarlo **ch1\_alert.js**.

Este será el contenido de nuestro archivo:

alert("¡Saludando desde un archivo diferente!");

Luego, vamos a crear un archivo HTML separado (nuevamente, usando la extensión **.html**).  
Y le daremos este contenido:

<html>

<script type="text/javascript" src="ch1\_alert.js"></script>

</html>

Asegúrate de colocar los archivos en la **misma ubicación**, o de **especificar la ruta** hacia el archivo JavaScript en tu HTML.  
Los nombres distinguen entre mayúsculas y minúsculas, por lo que deben coincidir **exactamente**.

**🔍 Tienes dos opciones para establecer la ruta al archivo .js:**

**✅ Ruta absoluta**

Una **ruta absoluta** empieza desde la raíz del sistema de archivos:

* En Linux/macOS: /
* En Windows: normalmente C:/

Este tipo de ruta funciona bien **en tu máquina local**, pero tiene una desventaja:  
Si más adelante mueves tu carpeta del proyecto a un servidor, **la ruta absoluta dejará de funcionar**.

**✅ Ruta relativa (más recomendada)**

Una **ruta relativa** indica cómo llegar al archivo **desde la ubicación del archivo actual**.

* Si el archivo .js está en la **misma carpeta**, solo debes escribir su nombre:

<script src="ch1\_alert.js"></script>

* Si está en una carpeta llamada example **dentro** de tu carpeta actual:

<script src="example/ch1\_alert.js"></script>

* Si está en una carpeta superior:

<script src="../ch1\_alert.js"></script>

Si abres el archivo HTML en tu navegador, deberías ver lo siguiente:

**Figura 1.4:** Ventana emergente creada por JavaScript en un archivo diferente.

**📝 Ejercicio práctico 1.3**

**Vincular un archivo JavaScript externo (.js):**

1. Crea un archivo JavaScript (.js) y añade una línea con alert() o console.log().  
   Por ejemplo:

console.log("Hola desde un archivo JS externo!");

1. Crea un archivo HTML que use ese archivo .js usando la etiqueta <script src="...">.
2. Abre tu archivo HTML en el navegador y asegúrate de que se ejecuta el script.

**Ejercicio práctico 1.3 (continuación)**

1. **Crea un archivo separado** llamado app con la extensión .js.
2. Dentro del archivo .js, añade algo de código JavaScript.  
   Por ejemplo:

alert("Hola desde app.js!");

console.log("Este mensaje viene desde app.js");

1. **Vincula el archivo .js** dentro del archivo HTML que creaste en el ejercicio práctico 1.2:

<!DOCTYPE html>

<html>

<head>

<title>Práctica JS</title>

</head>

<body>

<h1>Hola mundo</h1>

<script src="app.js"></script>

</body>

</html>

1. Abre el archivo HTML en tu navegador y verifica que el código JavaScript se ejecuta correctamente:
   * Verás una alerta con el mensaje.
   * Podrás ver el mensaje en la consola del navegador.

**Escribir código JavaScript**

Ya tenemos mucho contexto, pero… ¿cómo se **escribe realmente código JavaScript**?  
Hay algunos aspectos importantes a tener en cuenta:

* Cómo **formatear** el código.
* Usar el **nivel de indentación correcto**.
* Utilizar **puntos y coma (;)** adecuadamente.
* Agregar **comentarios** en el código.

**🧱 Formato del código**

El código debe estar bien formateado.  
Si tienes un archivo largo con muchas líneas y no sigues algunas reglas básicas de formato, será difícil entender lo que has escrito, incluso para ti mismo.

**🧩 Reglas básicas de formato**

Las dos más importantes por ahora son:

1. **Indentaciones**
2. **Puntos y coma**

Hay también convenciones de nombres (naming conventions), pero las abordaremos más adelante conforme vayan apareciendo.

**🔄 Indentación y espacios en blanco**

Cuando escribes código, muchas veces una línea **pertenece a un bloque de código** (el código entre llaves { ... }).  
Si ese es el caso, se recomienda **indentar** (es decir, agregar espacios o tabulaciones al comienzo de la línea) para hacer visible la jerarquía del código.

Esto ayuda mucho a entender qué instrucciones están dentro de un bloque, y dónde empieza uno nuevo.

Aquí tienes un ejemplo que **no necesitas entender aún**, pero que muestra la **diferencia entre código con y sin formato**.

**❌ Sin saltos de línea ni indentación:**

let status = "new"; let scared = true; if (status === "new") { console.

log("Welcome to JavaScript!"); if (scared) { console.log("Don't worry

you will be fine!"); } else { console.log("You're brave! You are going

to do great!"); } } else { console.log("Welcome back, I knew you'd like

it!"); }

Es difícil de leer, ¿verdad?

**✅ Con indentación y formato adecuado:**

let status = "new";

let scared = true;

if (status === "new") {

console.log("Welcome to JavaScript!");

if (scared) {

console.log("Don't worry, you will be fine!");

} else {

console.log("You're brave! You are going to do great!");

}

} else {

console.log("Welcome back, I knew you'd like it!");

}

Mucho más claro y legible.

**Con saltos de línea pero sin indentación:**

let status = "new";

let scared = true;

if (status === "new") {

console.log("Welcome to JavaScript!");

if (scared) {

console.log("Don't worry you will be fine!");

} else {

console.log("You're brave! You are going to do great!");

}

} else {

console.log("Welcome back, I knew you'd like it!");

}

**📄 Con saltos de línea y con indentación adecuada:**

let status = "new";

let scared = true;

if (status === "new") {

console.log("Welcome to JavaScript!");

if (scared) {

console.log("Don't worry you will be fine!");

} else {

console.log("You're brave! You are going to do great!");

}

} else {

console.log("Welcome back, I knew you'd like it!");

}

**✅ ¿Por qué es importante esto?**

Como puedes ver, ahora es **mucho más fácil ver dónde terminan los bloques de código**.  
Gracias a la indentación, puedes identificar claramente qué instrucciones pertenecen al bloque if, cuáles están dentro del else, y así sucesivamente.

En el ejemplo **sin indentaciones**, tendrías que contar manualmente las llaves {} para saber cuándo termina cada bloque. Aunque **la indentación no es obligatoria para que el código funcione**, es una **muy buena práctica**.

Te lo agradecerás más adelante cuando tu código crezca.

**🔚 Puntos y coma (;)**

Después de **cada instrucción**, debes añadir un **punto y coma** al final de la línea.  
JavaScript es bastante tolerante y puede funcionar aunque te olvides de algunos, pero **lo ideal es adquirir desde el inicio el hábito de ponerlos**.

🔸 **Sí se usan puntos y coma:**

let name = "Chris";

console.log(name);

🔸 **No se usan puntos y coma:**

* Después de declarar un bloque como if, for, while, function, etc.
* Justo antes o después de una llave {}.

❌ Incorrecto:

if (true);

{

console.log("Esto no funcionará como esperas");

}

✅ Correcto:

if (true) {

console.log("Esto sí funciona");

}

**🗨️ Comentarios en el código**

Con los **comentarios**, puedes indicarle al intérprete de JavaScript que **ignore ciertas líneas** del archivo. Estas líneas **no se ejecutarán** al correr el código.  
Es muy útil poder evitar que ciertas partes del código se ejecuten, y esto puede hacerse por varias razones:

1. ❌ **No quieres ejecutar** una parte del código mientras pruebas el script, así que la comentas para que el intérprete la ignore.
2. 🧾 **Metadata**: Añadir contexto sobre el archivo, como el autor, fecha, o una breve descripción de lo que hace el código.
3. 💡 **Explicar partes del código** o por qué tomaste una decisión específica al programar.

**✍️ Hay dos maneras de escribir comentarios:**

**1. Comentarios de una sola línea:**

// Soy un comentario de una sola línea

// console.log("Este código no se ejecutará porque es un comentario");

También puedes hacer un **comentario al final de una línea de código**:

let x = 5; // Este valor representa la edad

**2. Comentarios multilínea:**

/\* Soy un comentario de varias líneas.

Todo lo que esté entre estos símbolos será ignorado.

console.log("Este código tampoco se ejecutará");

\*/

**🧪 Ejercicio práctico 1.4 – Añadir comentarios**

1. Añade una nueva instrucción en tu archivo JavaScript donde **asignes un valor a una variable**.  
   Como aún no hemos cubierto variables en profundidad, puedes usar esta línea:

let a = 10; // Se asigna el valor 10 a la variable 'a'

1. Después, **muestra el valor en la consola** con console.log() y **añade un comentario** indicando qué hará esa instrucción:

console.log(a); // Esto mostrará el valor de la variable 'a' en la consola

**✍️ Continuación del ejercicio práctico 1.4 — Comentarios multilínea al final**

Al final de tu código JavaScript, añade un **comentario multilínea** que explique brevemente para qué sirve el archivo, algo así como:

/\*

Este archivo contiene código de ejemplo para mostrar:

- Cómo usar alertas y prompts

- Cómo agregar comentarios

- Cómo imprimir en la consola

- Cómo generar números aleatorios

\*/

**💬 Usar prompt() para pedir datos al usuario**

El comando prompt() es muy parecido a alert(), pero en vez de solo mostrar un mensaje, **permite al usuario introducir texto**.

Por ejemplo, cambia la línea de alert en tu archivo Hi.html por:

<script type="text/javascript">

prompt("Hi! How are you?");

</script>

Cuando abras el archivo en tu navegador y recargues la página, verás una ventana emergente con un cuadro para escribir texto, como este:

Este valor que escriba el usuario será devuelto al script y podrás almacenarlo en una variable para usarlo después (pronto aprenderás a manejar variables).

**🎲 Generar números aleatorios con Math.random()**

Para hacer ejercicios divertidos, JavaScript tiene un comando para generar números aleatorios.

Aunque no entiendas bien ahora cómo funciona, recuerda esta instrucción:

Math.random();

Este comando devuelve un número decimal aleatorio entre 0 (inclusive) y 1 (exclusivo).

Podemos hacerlo en la consola y ver el resultado aparecer si lo registramos:

js

CopiarEditar

console.log(Math.random());

Este número será un decimal entre 0 y 1. Si queremos un número entre 0 y 100, podemos multiplicarlo por 100, así:

console.log(Math.random() \* 100);

Si no queremos que el resultado sea decimal, podemos usar la función Math.floor, que redondea hacia abajo al entero más cercano:

console.log(Math.floor(Math.random() \* 100));

No te preocupes si aún no lo entiendes completamente. Esto se explicará con más detalle más adelante en el libro. En el Capítulo 8, Métodos Incorporados de JavaScript, hablaremos sobre los métodos incorporados con más profundidad. Hasta entonces, solo confía en que esto genera un número aleatorio entre 0 y 100.

**Proyecto del capítulo**  
Creando un archivo HTML y un archivo JavaScript vinculado  
Crea un archivo HTML y un archivo JavaScript separado. Luego, conecta el archivo JavaScript desde el archivo HTML.

1. En el archivo JavaScript, muestra tu nombre en la consola y agrega un comentario multilínea en tu código.
2. Intenta comentar el mensaje de la consola en tu archivo JavaScript para que no aparezca nada en la consola.

**Autoevaluación**

1. ¿Cuál es la sintaxis en HTML para agregar un archivo externo de JavaScript?
2. ¿Puedes ejecutar JavaScript desde un archivo con extensión .js directamente en tu navegador?
3. ¿Cómo se escribe un comentario multilínea en JavaScript?
4. ¿Cuál es la mejor manera de evitar que una línea de código se ejecute pero que puedas conservarla mientras haces debugging?

**Resumen**  
¡Muy bien hecho! Has empezado con JavaScript. En este capítulo hemos hablado de mucho contexto que necesitarás conocer antes de comenzar a programar en JavaScript. Vimos que podemos usar JavaScript para muchos propósitos, y uno de los casos más populares es la web. Los navegadores pueden trabajar con JavaScript porque tienen una parte especial llamada intérprete, que procesa JavaScript.

Vimos que tenemos varias opciones para escribir JavaScript en nuestra computadora. Necesitaremos un IDE, un programa que nos permita escribir y ejecutar nuestro código.

Agregar JavaScript a una página web puede hacerse de varias maneras. Vimos cómo incluirlo dentro del elemento <script> y cómo agregar un archivo JavaScript separado a una página.

Terminamos este capítulo con algunas notas importantes sobre cómo escribir código bien estructurado, legible y fácil de mantener, que esté bien documentado con comentarios. También vimos que podemos escribir en la consola usando el método console.log() y pedir datos al usuario con prompt(). Por último, vimos que podemos generar números aleatorios con la función Math.random().

A continuación, veremos los tipos de datos básicos de JavaScript y los operadores que podemos usar para manipularlos.

**Elementos esenciales de JavaScript**  
En este capítulo, trataremos algunos de los componentes fundamentales de JavaScript: **variables y operadores**. Comenzaremos con las variables, qué son y qué diferentes tipos de datos existen. Necesitamos estos bloques básicos para almacenar y trabajar con valores variables en nuestros scripts, lo que los hace dinámicos.

Una vez que hayamos cubierto las variables, estaremos listos para abordar los operadores. En esta etapa se discutirán los operadores **aritméticos, de asignación, condicionales y lógicos**. Necesitamos los operadores para modificar nuestras variables o para obtener información sobre ellas. De esta manera, podemos realizar cálculos básicos basados en factores como la entrada del usuario.

A lo largo del camino, cubriremos los siguientes temas:  
• Variables  
• Tipos de datos primitivos  
• Análisis y modificación de tipos de datos  
• Operadores

**Nota:** las respuestas a los ejercicios, proyectos y cuestionarios de autoevaluación se pueden encontrar en el **Apéndice**.