

Introducción a JavaScript

Introducción

¡Hola!  Te damos la bienvenida al cuarto encuentro del curso de *Programación web desde cero*.

Hasta ahora, hemos cubierto lo básico de **HTML** y **CSS** para crear y diseñar páginas web.

Hoy, nos adentraremos en el mundo de **JavaScript**; **un lenguaje de programación que te permite agregar interactividad y contenido dinámico a tus sitios web** a través de su sintaxis, *tipos de datos, variables y constantes, operadores, y funciones*.

Al final de esta clase, *crearás una función simple de JavaScript para mostrar una alerta en la página HTML sobre la que vienes trabajando*.

¡Empecemos! 

¿Qué es JavaScript?

Como adelantamos antes, **JavaScript** es un **lenguaje de programación utilizado para crear páginas web interactivas**. Permite crear efectos visuales y animaciones en la página, manipular contenido en tiempo real, crear formularios interactivos y mucho más.


Este lenguaje se ejecuta en el navegador web, lo que significa que no necesita ningún software adicional para funcionar. Esto lo hace extremadamente popular y accesible, ya que **se puede usar en cualquier dispositivo con acceso a un navegador web**.

Una de las características más importantes de JavaScript es su *capacidad para interactuar con HTML y CSS*, es decir, *podemos modificar y manipular elementos de una página web* a través de este lenguaje. Por ejemplo, usaremos JavaScript

cuando queramos cambiar el texto de un botón, cuando se hace clic en él o para cambiar el color de fondo de una sección de la página después de que se haya cargado.

Características de JavaScript

Para comenzar a aprender sobre JavaScript es importante conocer sus **características básicas**, por eso, en esta oportunidad te daremos un pantallazo de las mismas:

 *En este curso no vamos a estar profundizando en todas, sin embargo iremos cubriendo lo esencial para que puedas comenzar a poner en práctica este lenguaje.*

Conceptos básicos

- **Variables:** Son como "cajitas" en las que guardamos valores.
 - Palabras clave: *let, const*.
- **Tipos de datos:** Son los diferentes "ingredientes" que podemos guardar en nuestras variables.
- **Operadores:** Son símbolos que nos permiten realizar operaciones con los valores, como sumar, restar, comparar, etc.

Funciones

- **Funciones:** Son bloques de código que podemos "llamar" para ejecutar una tarea específica. Podemos crear nuestras propias funciones o usar funciones ya existentes en JavaScript.
 - Palabras clave: *function, return*.

Estructuras de control

- **Condicionales:** Nos permiten tomar decisiones en nuestro código. Ejemplo: "Si el usuario es mayor de 18 años, mostrar un mensaje".
 - Palabras clave: *if, else, else if*.
- **Bucles:** Nos permiten repetir una acción varias veces. Ejemplo: "Hacer algo 10 veces".
 - Palabras clave: *for, while, do-while*.
- **De control de excepciones:** Nos permiten manejar errores en el código. Ejemplo: Si se genera un error aparecerá el mensaje de "Se ha producido un error".
 - Palabras clave: *try, catch, finally*.

Objetos y arrays

- **Objetos:** Nos permiten agrupar variables y funciones relacionadas. Ejemplo: Un objeto que representa a una persona con sus datos y acciones.
 - Palabras clave: `{ }`, `this`.
- **Arrays:** Son listas de valores. Ejemplo: Una lista de amigos, donde cada amigo es un objeto.
 - Palabras clave: `[]`, `length`.

Manipulación del DOM

- **DOM (Document Object Model):** Es la representación de una página web en forma de objetos y sus relaciones. JavaScript nos permite modificar el DOM para cambiar cómo se ve y funciona la página web.
 - Palabras clave: `getElementById`, `querySelector`, `innerHTML`, `addEventListener`.

Sintaxis de JavaScript

Una sintaxis es la forma en la que están dispuestos y ordenados los componentes de un lenguaje. La **sintaxis de JavaScript** es similar a la de otros lenguajes de programación, pero tiene algunas particularidades propias:

- Las *sentencias simples* se encierran entre *paréntesis* `"()"`.
- Los *bloques de código* se encierran entre *llaves* `"{ }"` y se utilizan *puntos y comas* `","` para separar las sentencias.

Por ejemplo, el siguiente comando muestra la sintaxis básica de Javascript para imprimir *"Hola mundo"* en la **consola del navegador**:

```
console.log("Hola mundo");
```

Esta sería una sentencia simple por lo que está encerrada entre paréntesis. (No te preocupes ahora por comprender la composición total de la sentencia y su función, eso lo iremos viendo a medida que avancemos).

Por otro lado, podemos observar la siguiente sintaxis para este otro comando de JavaScript:

```
function sumar(num1, num2) {  
  let resultado = num1 + num2;  
  return resultado;  
}
```

En el caso de bloques de código más grandes, como **funciones** o **estructuras de control de flujo** que veremos más adelante, utilizaremos las llaves “{ }” para delimitar el bloque como mencionamos más arriba.

* Consola del navegador

La **consola del navegador** es una herramienta de desarrollo que permite ver información y mensajes relacionados con el código que se está ejecutando en una página web. Entendamos en más detalle de qué trata:

¡Manos a la obra!

1. En una nueva pestaña del navegador, hacer *clic derecho* > “Inspeccionar”.
2. Luego, ir a la pestaña de “Console” y ya estarás en la consola del navegador. También puedes hacerlo presionando F12 en la mayoría de los navegadores
3. Probar de escribir algunos comandos simples, como “Hola Mundo” (usar las comillas) o $2 + 2$.
4. Dar un “Enter” en la consola y observar lo que sucede. La consola debería ir mostrando el resultado de los comandos.
5. Ahora, ingresar el comando: `document.title`
¿Qué ocurrió? La consola puede ser utilizada para enviar comandos a la página web y obtener información útil o para realizar pruebas y depuración de código.

Para más información sobre la consola del navegador, te invitamos a ver el siguiente video:

 [Ver video](#)

Ahora bien, pasemos a comprender qué elementos pueden formar parte de estos códigos. JavaScript tiene varios **elementos fundamentales** que son esenciales para poder escribir código en este lenguaje:

- Variables y constantes
- Tipos de datos
- Operadores

A continuación, vamos a hablar un poco más sobre estos elementos.

Variables y constantes

Las **variables** y **constantes** son contenedores donde se pueden almacenar y manipular datos en Javascript.

💡 *Ten en cuenta que en la industria de la tecnología comúnmente se utiliza la palabra "variables" para referirse tanto a las variables como a las constantes.*

Declaración y asignación

Palabra reservada de una variable (puede ser: let, var o const).

```
let miVariable = "Declarar variable";
```

Data asignado a la variable

Nombre de la variable

Operador de asignación

Buenas prácticas

- 1 El nombre o identificador de la variable debe ser un nombre descriptivo que se identifique con el código.
- 2 Las variables no se pueden iniciar con números. Deben comenzar con una letra, signo de guión bajo "_" o un símbolo especial "\$".
- 3 No se pueden usar caracteres especiales ni es recomendable empezar una variable con mayúscula.
- 4 Los tipos de escritura recomendables de una variable son "lowerCamelCase" o "snake_case."
- 5 No se pueden utilizar palabras reservadas del lenguaje, como "let", "const", "var", "function", entre otras.
- 6 Las variables son case-sensitive. Por ejemplo, la variable "edad" y la variable "Edad" serían dos variables diferentes.

Variables

Las **variables** son contenedores de datos que **pueden cambiar a lo largo del tiempo**. No necesitan ser definidas con un tipo de dato específico ya que el lenguaje determina automáticamente el tipo de dato de una variable cuando se le asigna un valor.

Para entenderlo mejor, piensa en una caja donde puedes guardar diferentes cosas. En JavaScript, la caja representa una variable, y puedes guardar en ella objetos de cualquier tipo sin tener que especificar qué tipo de objeto es. Por ejemplo, puedes guardar una manzana (fruta) en la caja, y luego cambiarla por una camiseta (ropa) sin problemas. JavaScript se encarga de identificar automáticamente el tipo de objeto que hay en la caja en cada momento.

Las variables se pueden definir usando la palabra clave `"let"` o `"var"` seguida del nombre de la variable y opcionalmente un valor inicial.

```
let nombre = "Juan";
```

💡 La principal diferencia entre `"let"` y `"var"` es que las variables `"let"` tienen un ámbito de bloque, lo que significa que solo son accesibles dentro de ese bloque, mientras que las variables `"var"` tienen un ámbito de función o global, lo que quiere decir que pueden ser accedidas desde cualquier lugar dentro de la función o del ámbito global.

Se recomienda usar `let` en lugar de `var` en la mayoría de los casos, ya que puede ayudar a evitar errores y hacer que el código sea más fácil de entender. Sin embargo, es importante tener en cuenta el contexto en el que se está trabajando y elegir la palabra clave que mejor se adapte a las necesidades del proyecto en cuestión.

Constantes

Las **constantes**, por otro lado, son contenedores de valores que no cambian durante la ejecución del programa. Una vez que se ha asignado un valor a una constante, no se puede cambiar (Valor inmutable).

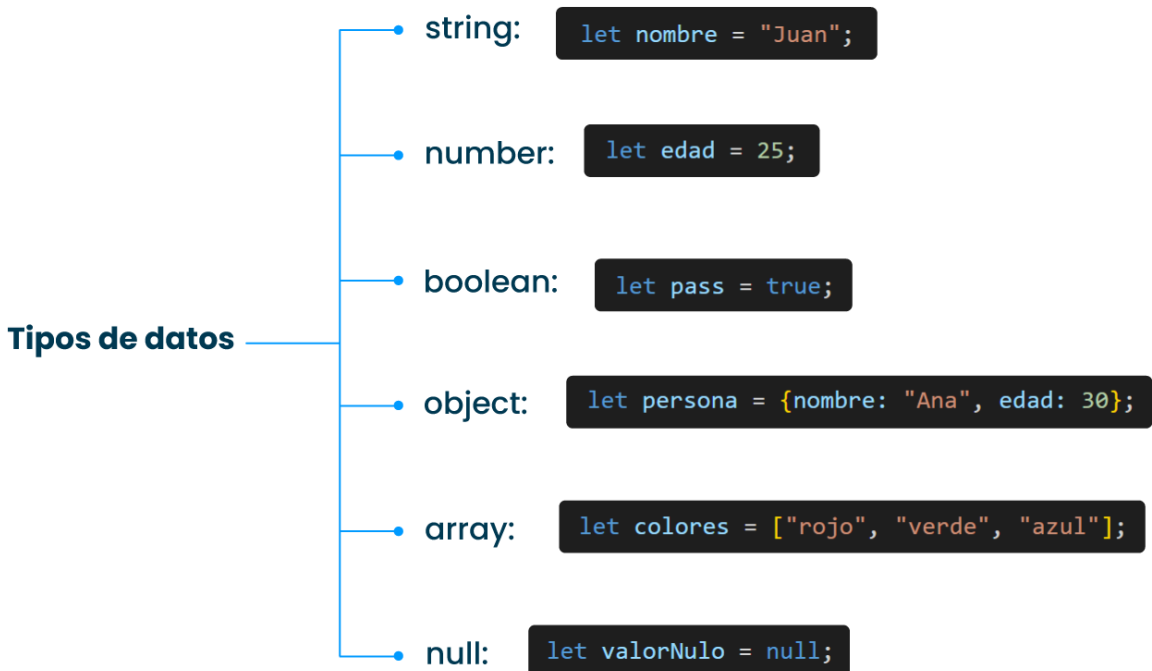
Se pueden definir utilizando la palabra clave `"const"` seguida del nombre de la constante y su valor inicial, como se muestra en el siguiente ejemplo:

```
const pi = 3.14;
```

En este ejemplo, la constante "pi" se inicializa con un valor de 3.14 y no se puede cambiar posteriormente. Si se intenta hacerlo, aparecerá un error.

Tipo de datos

JavaScript tiene varios **tipos de datos** que es importante conocer para poder manipularlos adecuadamente en el código, y comprender mejor cómo funciona este lenguaje de programación. Entre ellos se incluyen:



- **Cadena de caracteres o texto ("string"):** Representa texto y se define entre comillas simples o dobles (ambas son válidas y producen el mismo resultado).

💡 *No te preocupes si no logras entender la totalidad del código ya que lo iremos viendo a medida que avancemos. En este momento lo relevante a observar es cómo se escriben los valores de los diferentes datos.*

- **Números ("number"):** enteros y decimales.

Para representar números enteros grandes se utiliza el *tipo de dato* de **BigInt**, y se escriben con la letra "n" al final del número. Ejemplo: 9007199254740991n.

- **Booleanos ("boolean")**: "true" o "false" (verdadero o falso).
- **Objeto ("object")**: Es una colección de propiedades y valores. Cada propiedad tiene un nombre y un valor asociado, que puede ser de cualquier tipo de dato válido como números, cadenas de texto, booleanos, etc. Para crear un objeto, se utiliza la sintaxis de llaves "{}".
- **Arreglo ("array")**: Es una colección ordenada de valores (pueden ser de elementos de cualquier tipo). Cada valor en el arreglo tiene una posición o índice numérico que comienza desde cero. Para crear un arreglo, se utiliza la sintaxis de corchetes "["].
- **Null**: Representa la ausencia de valor.

Ahora, te proponemos realizar la siguiente actividad que te ayudará a comprender mejor estos tipos de datos y cómo se pueden identificar mediante el operador *"typeof"* (ya veremos qué són los operadores). Además, al ejecutarla en la consola del navegador, podrás ver los resultados de manera inmediata y experimentar con diferentes valores y tipos de datos.

¡Manos a la obra!

1. En una nueva pestaña del navegador, abrir la *consola del navegador* (puedes hacerlo presionando F12 en la mayoría de los navegadores).
2. Ingresar los siguientes comandos uno por uno y presiona enter después de cada uno:
 - `typeof 5`
 - `typeof "Hola"`
 - `typeof true`
 - `typeof {}`
3. Observar los resultados que se muestran en la consola para cada uno de los comandos.
4. Comparar los resultados y tratar de entender qué tipo de dato representa cada uno de ellos.

¿Has finalizado? Muy bien, continuemos...

Operadores

¡Muy bien! De los **elementos fundamentales** ya vimos *las variables y constantes* y los *tipos de datos*. Nos quedan por ver entonces los *operadores*.

Los **operadores** son **herramientas esenciales para manipular el valor de las variables, realizar cálculos complejos y tomar decisiones lógicas en función de comparaciones y otros tipos de condiciones.**

Existen varios tipos de operadores que se utilizan para diferentes propósitos. Algunos de ellos son:

Aritméticos	Comparación	Asignación	Lógicos
<div><div>+</div><div>Suma, Concatenar</div></div> <div><div>++</div><div>Incremento</div></div> <div><div>-</div><div>Sustracción</div></div> <div><div>--</div><div>Decremento</div></div> <div><div>*</div><div>Multiplicación</div></div> <div><div>/</div><div>División</div></div> <div><div>%</div><div>Modulo</div></div>	<div><div>></div><div>Mayor que</div></div> <div><div>>=</div><div>Mayor o igual que</div></div> <div><div><</div><div>Menor que</div></div> <div><div>=<</div><div>Menor o igual que</div></div> <div><div>==</div><div>Igual a</div></div> <div><div>===</div><div>Estrictamente igual</div></div> <div><div>!=</div><div>No igual a</div></div> <div><div>!==</div><div>Estrictamente no igual</div></div>	<div><div>=</div><div>Asignar valor a</div></div> <div><div>+=</div><div>Sumar y Asignar</div></div> <div><div>-=</div><div>Restar y Asignar</div></div> <div><div>*=</div><div>Multiplicar y Asignar</div></div> <div><div>/=</div><div>Dividir y Asignar</div></div> <div><div>%=</div><div>Módulo y Asignar</div></div>	<div><div>&&</div><div>AND</div></div> <div><div> </div><div>OR</div></div> <div><div>!</div><div>NOT</div></div>

Operadores aritméticos → Se utilizan para realizar operaciones matemáticas con valores numéricos.

Operadores de comparación → Se utilizan para comparar dos valores y devolver un resultado booleano (true o false). Ejemplos: `2 == 2` devuelve true, `5 != 5` devuelve false, o `3 >= 3` devuelve true.

Operadores de asignación → Se utilizan para asignar valores a variables y constantes. El operador más común es el operador de asignación simple `"="`. Ejemplo: `let x = 2` asigna el valor 2 a la variable x.

Operadores lógicos → Se utilizan para combinar dos o más expresiones booleanas y devolver un resultado booleano.

- **"&&" (AND):** Comprueba si dos expresiones booleanas son verdaderas. Ejemplo: `true && true` devuelve true.
- **"||" (OR):** Comprueba si al menos una de dos expresiones booleanas es verdadera. Ejemplo: `true || false` devuelve true.
- **"!" (NOT):** Se usa para negar una expresión booleana. Ejemplo: `!true` devuelve false.

Funciones

Las **funciones** en Javascript son **bloques de código que se pueden llamar en cualquier momento y que realizan una tarea específica**. Son útiles para organizar el código en bloques más pequeños y reutilizables.

Declaración

1 Declarar

- Palabra reservada "function"
- Nombre de la función
- Escribir los parámetros

// Declarar función //

```
function algunNombre(param1, param2) {
```

1

// Bloque de código //

```
let a = param1 + "ama a" + param 2;
```

```
return a;
```

```
}
```

2

2 Bloque de código

Se escribe el conjunto de instrucciones que va a realizar la función

3 Invocar función

- Llamar a la función
- Escribir los argumentos (valores)

// Invocar función //

```
algunNombre("Juan", "Carla")
```

3

Veamos cómo están conformadas las funciones:

- **Sintaxis** → Las funciones se definen utilizando la palabra clave "function", seguida del nombre de la función y los parámetros que recibe, si es que los hay. El cuerpo de la función se encierra en llaves { }.
Por ejemplo, aquí está la sintaxis básica de una función que toma dos números y los suma:

```
function sumar(num1, num2) {  
    return num1 + num2;  
}
```

- **Parámetros y argumentos** → Los *parámetros* son variables que se utilizan dentro de la función para realizar una tarea específica y los *argumentos* son

los valores reales que se pasan a la función cuando se llama.

Por ejemplo, si llamamos a la función `sumar(2, 3)`, "2" y "3" son los argumentos que se pasan a la función, y `num1` y `num2` son los parámetros que se utilizan para realizar la suma.

- **Retorno de valores** → Las funciones pueden devolver valores utilizando la palabra clave `"return"`. Los valores devueltos pueden ser de cualquier tipo de datos.

Por ejemplo, la función `"sumar"` devuelve un valor numérico, en este caso `"5"`.

```
// Definimos la función sumar con dos parámetros: num1 y num2
function sumar(num1, num2) {
    return num1 + num2;
}

// Llamamos a la función sumar con los argumentos 2 y 3
let resultado = sumar(2, 3);

// Mostramos el resultado en la consola del navegador
console.log(resultado); // Resultado: 5
```

Funciones anónimas y funciones flecha

Las *funciones anónimas* son funciones que no tienen un nombre y se pueden utilizar como expresiones. Las *funciones flecha* son una sintaxis abreviada para definir funciones anónimas y son muy útiles para escribir código más limpio y legible.

Por ejemplo, aquí vemos una función anónima que toma un número y lo duplica:

```
const duplicar = function(numero) {
    return numero * 2;
}
```

Y aquí está la misma función escrita como una función flecha:

```
const duplicar = numero => numero * 2;
```

Como los parámetros de una función generalmente se usan para realizar una tarea específica dentro de la misma, su valor no debería cambiar durante la ejecución, por lo que se recomienda usar "const" en su declaración. Sin embargo, en otros casos donde se espera que el valor cambie, se podría utilizar "let".

En resumen, las funciones en JavaScript son bloques de código que te permiten realizar tareas específicas de forma organizada y reutilizable. Con una sintaxis simple y la capacidad de aceptar parámetros y devolver valores, las funciones son una parte fundamental de cualquier programa JavaScript.

Desafío del día

Si llegaste hasta aquí, ¡Ya puedes comenzar a escribir código en Javascript! ¿Probamos? El desafío de hoy consiste en crear una función simple para mostrar una alerta en la página HTML.

1. Crear un nuevo archivo en tu editor de texto y guardarlo con una extensión ".js", como "script.js". **Este archivo contendrá nuestro código JavaScript.**

2. Vincular el archivo JavaScript externo a tu documento HTML. Abrir tu archivo "mi-primer-pagina.html" del encuentro pasado y agregar el siguiente código justo antes de la etiqueta de cierre </body>:

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

Este código indica al navegador que cargue el archivo "script.js" y ejecute el código JavaScript contenido en él.

3. Crear una función simple de JavaScript en el archivo "script.js" que mostrará una alerta cuando se llame:

```
function showAlert() { alert('¡Hola, esta es una alerta desde JavaScript!'); }
```

Ten en cuenta que en JavaScript **los paréntesis son necesarios para llamar o ejecutar una función**. Sin los paréntesis, solo se hace referencia a la función, pero no se ejecuta

Esta función, cuando se llame, mostrará una alerta del navegador con el mensaje: *"¡Hola, esta es una alerta desde JavaScript!"*.

💡 La función `showAlert()` está utilizando la **función integrada o nativa de `alert()`**. En JavaScript hay muchas funciones predefinidas que se pueden utilizar sin necesidad de definirlas.

4. Para llamar a esta función desde tu página HTML, necesitamos agregar un botón que active la función al hacer clic.

Agregar el siguiente código a tu archivo `mi-primer-pagina.html`, dentro del elemento `.container`:

```
<button onclick="showAlert()">¡Haz clic en mí!</button>
```

Este código agrega un botón con el texto *"¡Haz clic en mí!"* y un atributo `onclick`, que llama a la función `showAlert()` cuando se hace clic en el botón.

5. Guardar ambos archivos, `mi-primer-pagina.html` y `script.js`, luego actualiza tu página web en el navegador.

Ahora deberías ver el botón *"¡Haz clic en mí!"* en tu página web. Haz clic en el botón y aparecerá la alerta con el mensaje: *"¡Hola, esta es una alerta desde JavaScript!"*.

¡Desafío terminado! 🎉

Resolución del desafío

En el siguiente video te compartimos un paso a paso de cómo resolver el desafío anterior:

📺 [Resolución de Ejercicio | Introducción a JavaScript | Egg](#)

¡Felicidades! 🎊

Has creado una función simple de JavaScript y la has integrado con éxito en tu página HTML.

En el siguiente encuentro, profundizaremos en JavaScript, cubriendo *eventos y oyentes de eventos, manejo y validación de formularios*. Estos conceptos te ayudarán a agregar más interactividad y contenido dinámico a tus páginas web.

Valida tus conocimientos

Te compartimos el enlace del **material complementario** para que puedas continuar profundizando sobre el siguiente tema cuando prefieras:

- [Listado de funciones integradas de JavaScript](#)

Por último, te proponemos realizar la autoevaluación para poner a prueba los principales aprendizajes del día de hoy. ¿Vamos?

 [Realizar test](#)

Ten en cuenta que si has llegado hasta aquí, ya has cumplido con el objetivo del encuentro.

¡Hasta la próxima! 

Mapa de conceptos vistos

