Material descargable

Programación Web Desde Cero

egg

Javascript

¡Hola! 👋

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.

¡Comencemos!🚀

¿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 profundizar en todas, sin embargo, iremos cubriendo lo esencial para que puedas 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 "elementos" 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.*

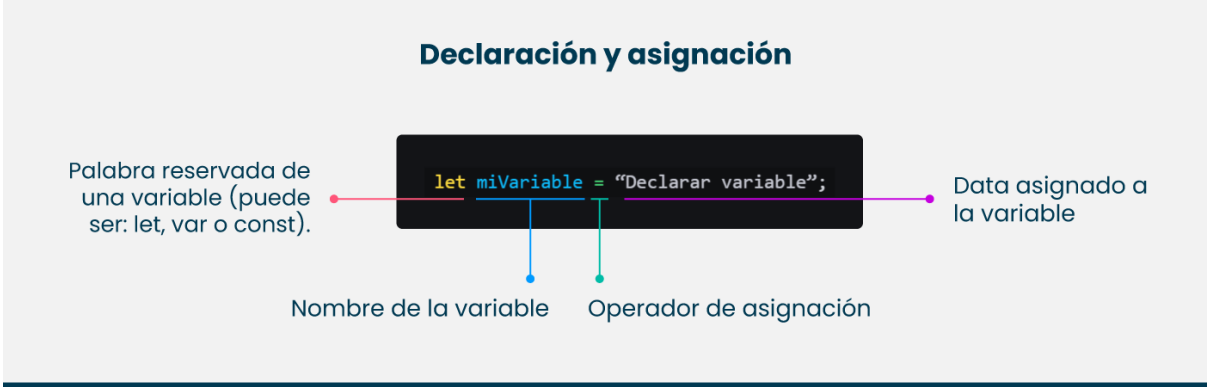
Consola del navegador

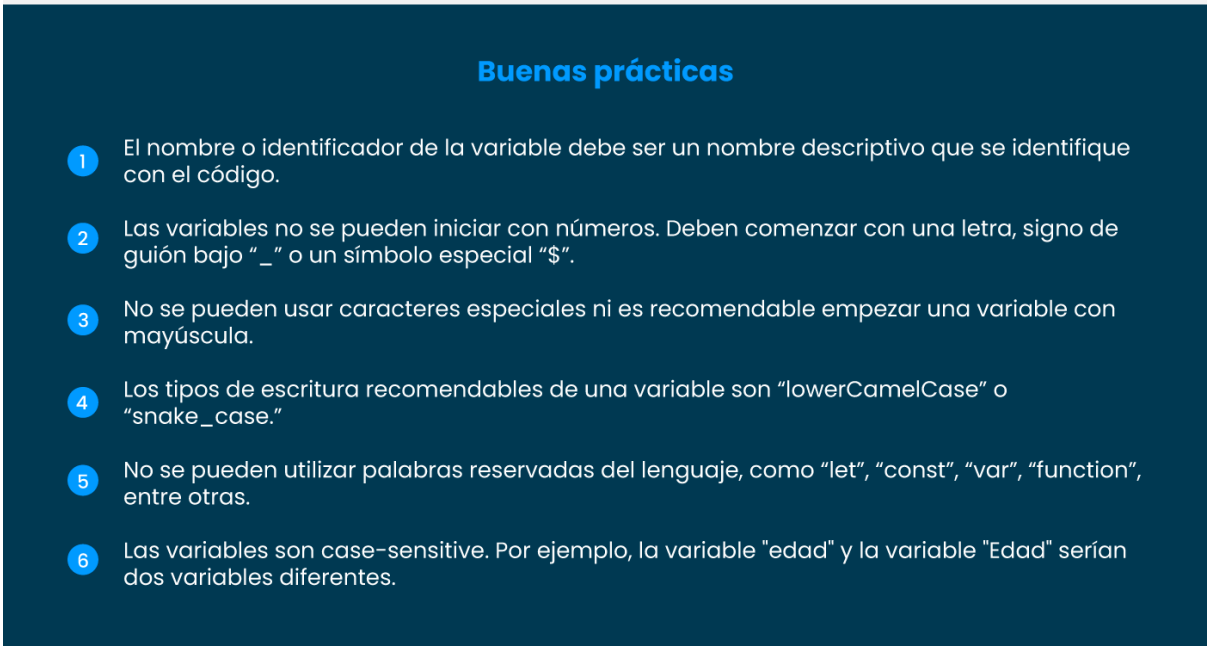
La consola del navegador es una herramienta de desarrollo que permite ver información y mensajes relacionados con el código que se está ejecutando en una página web.

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.*





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).

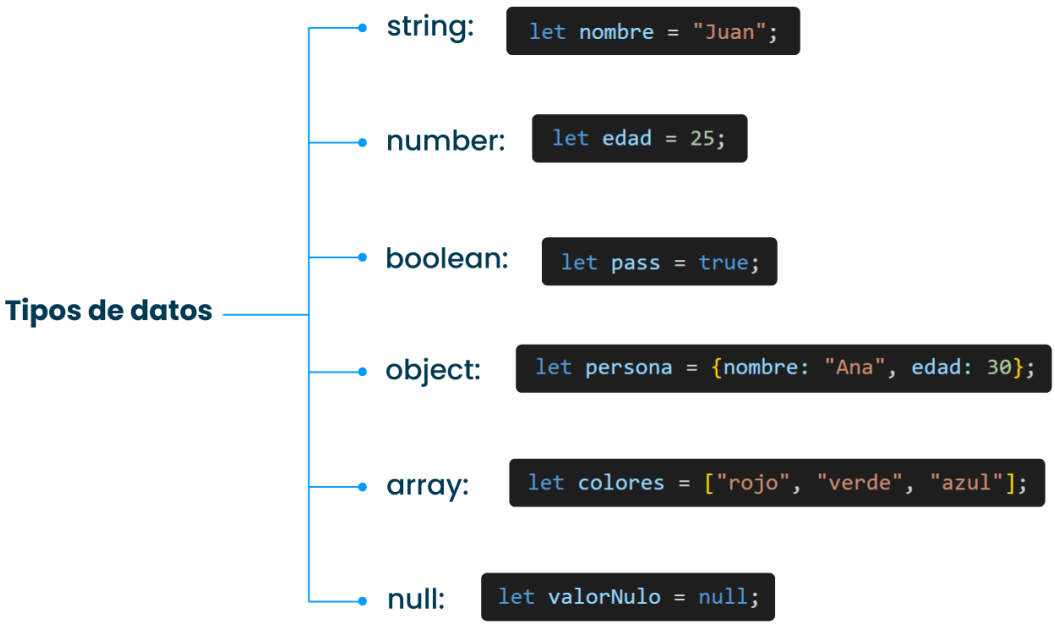
Se pueden definir utilizando la palabra clave *"const"* seguida del nombre de la constante y su valor inicial, como se muestra en el siguiente ejemplo:

```
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.

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:

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.

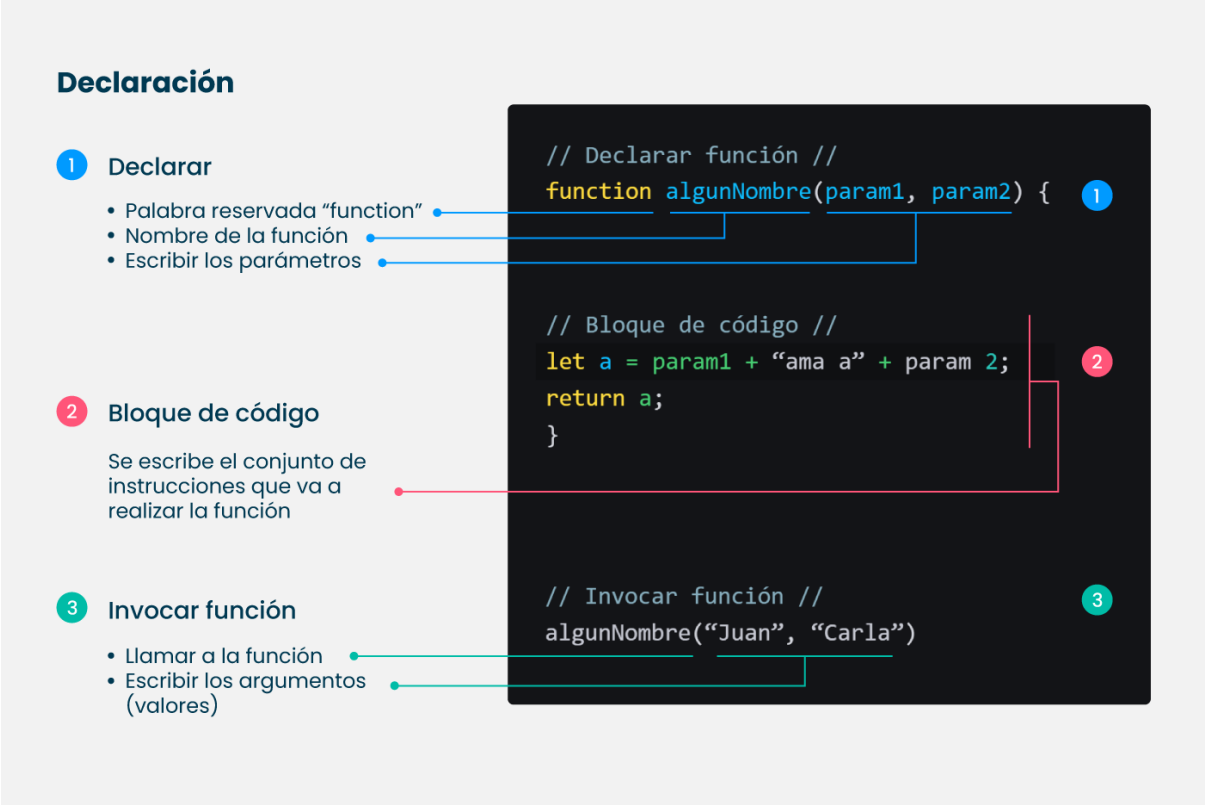
Operadores de asignación → Se utilizan para asignar valores a variables y constantes. El operador más común es el operador de asignación simple "=". Ejemplo: let x = 2 asigna el valor 2 a la variable x.

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.



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 flechas

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.

Estructuras de control de flujo

Las ***estructuras de control de flujo*** son utilizadas para controlar el flujo de ejecución del programa y tomar decisiones basadas en ciertas condiciones.

Algunos ejemplos de *estructuras de control de flujo* incluyen:

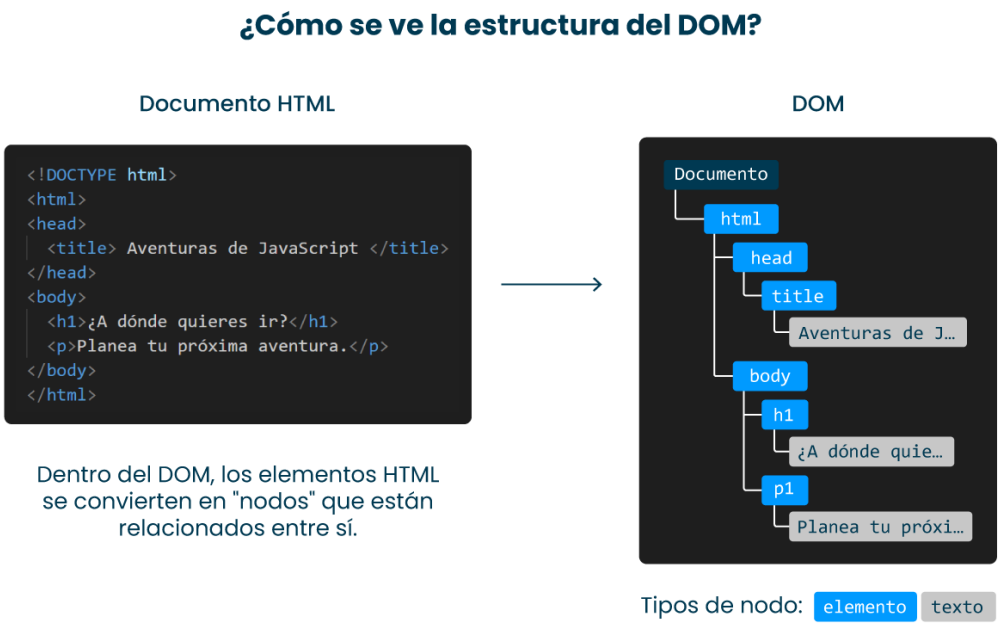
- **Estructuras condicionales:** if, else, else if → Se usan para ejecutar diferentes bloques de código dependiendo de si una condición es verdadera o falsa.
- **Estructuras de bucles:** for, while, do-while → Se usan para repetir un bloque de código hasta que se cumpla una condición.
- **Estructuras de control de excepciones:** try, catch, finally → Se usan para manejar errores en el código.

Las estructuras de control de flujo **se pueden anidar unas dentro de otras, lo que permite crear programas complejos y avanzados**. También es importante tener en cuenta la eficiencia del código y evitar anidaciones excesivas o redundantes que puedan afectar el rendimiento del programa.

Manipulación del DOM

El ***DOM*** ("Document Object Model" o "Modelo de Objetos del Documento") es una representación de la estructura de un documento HTML que se organiza como un árbol de objetos ("nodos"), donde cada objeto representa una parte del documento, como un elemento o un atributo.

En otras palabras, es una interfaz de programación de aplicaciones (API) que proporciona una forma estándar para que los programadores accedan y manipulen los elementos HTML de un documento web.



JavaScript se utiliza para acceder a los objetos del DOM y manipularlos, lo que permite actualizar dinámicamente el contenido y la estructura de una página web en tiempo real.

La manipulación del DOM puede incluir la creación, eliminación, modificación y movimiento de elementos y atributos en una página web.

💡 *Es importante tener en cuenta que la manipulación del DOM debe hacerse detalladamente y no en exceso, ya que puede afectar el rendimiento de la página.*

Encontrando los elementos HTML

Como mencionamos JavaScript se puede usar para manipular elementos HTML, pero para hacerlo, debemos encontrarlos primero.

Se puede realizar la búsqueda a partir de las siguientes características:

- Encontrando elementos HTML por su **id**

```
const elemento = document.getElementById( "id" );
```

- Encontrando elementos HTML por su **nombre de etiqueta**

```
const paragraphs = document.getElementsByTagName( "p" );
```

- Encontrando elementos HTML por su **nombre de clase**

```
const myClasses = document.getElementsByClassName( "miClase" );
```

Propiedades y métodos del DOM

En el DOM tanto los métodos como las propiedades son utilizados para manipular y acceder a los *elementos HTML* en una página web.

Las ***propiedades del DOM*** son valores que pueden ser leídos o modificados directamente. Algunos ejemplos de propiedades incluyen *.innerText*, *.innerHTML*, *.value*, *.src*, *.href*, *.id*, *.className*, *.style*, entre otras. Estas pueden ser accedidas utilizando la notación de punto o la notación de corchetes.

Por otro lado, los ***métodos del DOM*** son funciones que se utilizan para realizar una acción en un elemento HTML como agregar un nuevo elemento, eliminar un elemento existente, cambiar su estilo, etc. Algunos ejemplos de métodos incluyen

createElement(), appendChild(), removeChild(), setAttribute(), addEventListener(), entre otros.

Eventos

Los ***eventos*** son *acciones que ocurren en la página web o en el navegador*, como hacer clic en un botón, pasar el mouse sobre una imagen y escribir en un campo de texto, entre otros.

Estos eventos pueden ser detectados y manejados a través de JavaScript para realizar acciones específicas. Es decir, **son una herramienta poderosa para interactuar con el usuario y mejorar su experiencia en la página web.**

Tipos de eventos

En JavaScript hay una gran cantidad de eventos disponibles. Algunos de los más comunes son:

- Eventos de ratón: Se activan cuando el usuario interactúa con el mouse, como hacer clic, mover el mouse, presionar y soltar el botón del mouse, etc.
- Eventos de teclado: Se activan cuando el usuario interactúa con el teclado, como presionar una tecla, soltarla, mantenerla presionada, etc.
- Eventos de formulario: Se activan cuando el usuario interactúa con un formulario; al hacer clic en un botón de envío y cambiar el valor de un campo de formulario, entre otros.
- Eventos de ventana: Se activan cuando el usuario interactúa con la ventana del navegador, como cambiar el tamaño de la ventana, minimizarla o maximizarla y cerrarla, etc.

El ***manejo de eventos*** es una parte importante de la programación en Javascript que permite a los desarrolladores crear interacciones dinámicas y personalizadas en sus páginas o aplicaciones web.

A continuación, lo veremos en más detalle...

Manejo de eventos

Para manejar un evento en JavaScript se utiliza un oyente de eventos (“event listener”).

Un ***oyente de eventos*** es una función que se encarga de "escuchar" estos eventos específicos y ejecutar ciertas acciones en función de lo ocurrido. Algunos ejemplos podrían ser:

- Cambio de contenido al hacer clic en una pestaña.
- Una validación al enviar un formulario.
- Expandir y colapsar elementos al hacer clic en un botón.
- Cargar datos adicionales al llegar al final de una página (scroll infinito).
- Cambiar el estilo de la barra de navegación al desplazarse.

El método *“addEventListener()”* es el que se utiliza para agregar un *oyente de eventos* a un elemento HTML. Este método toma dos argumentos:

- **Tipo de evento** que se desea escuchar.
- **Función** que se ejecutará cuando se active el evento.

Resumen

¡Felicidades!

Esperamos que esta información te haya sido de utilidad y que te lleves para repasar:

- Qué es Javascript
- Cómo es sus sintaxis básica
- Cuáles son sus componentes principales