Material descargable

egg

Teoría **JavaScript**

¿Qué es JavaScript?

JavaScript es un lenguaje de programación utilizado principalmente para hacer que las páginas web sean interactivas. Permite agregar funcionalidades a un sitio web, como botones que responden cuando se hace clic en ellos, formularios que validan la información ingresada por los usuarios y elementos que se actualizan automáticamente sin necesidad de recargar la página.

JavaScript se ejecuta en el navegador web y es muy popular debido a su facilidad de uso y versatilidad. Con JavaScript, los desarrolladores pueden crear experiencias dinámicas en línea para mejorar la interacción y la usabilidad de los sitios web.

Variables

En la programación, las variables desempeñan un papel fundamental, ya que nos permiten almacenar y manipular datos de manera eficiente. **Una variable en JavaScript es como un contenedor que guarda un valor específico**.. Las variables permiten almacenar datos temporalmente en la memoria de la computadora para su posterior uso.

Al definir una variable en JavaScript, utilizamos la palabra clave var, let o const, seguida por un nombre que le asignamos a esa variable. También podemos asignarle un valor inicial opcional. Por ejemplo:

```
var edad;
let nombre;
const PI = 3.1416;
```

La variable nos proporciona flexibilidad en nuestro código, ya que podemos cambiar y actualizar su valor a medida que el programa se ejecuta. Esto es

especialmente útil cuando necesitamos almacenar y trabajar con datos que pueden cambiar en diferentes etapas de un programa.

Tipos de variables

var: var se utilizaba anteriormente para declarar variables, pero desde la introducción de let y const, se ha vuelto menos común. Aún se encuentra en código heredado y puede tener comportamientos diferentes en relación con el ámbito (scope) de las variables. En la mayoría de los casos, es preferible utilizar let o const en su lugar.

let: let se utiliza para declarar variables que pueden ser reasignadas posteriormente. Es una buena elección cuando necesitas una variable cuyo valor puede cambiar durante la ejecución del programa. La variable declarada con let tiene un alcance de bloque, lo que significa que solo está disponible dentro del bloque donde se declara.

const: const se utiliza para declarar variables que tienen un valor constante e inmutable. Una vez que se le asigna un valor, no se puede reasignar. Es una buena elección cuando necesitas una variable cuyo valor no debe cambiar, y evitar reasignaciones accidentales.. Al igual que let, const también tiene un alcance de bloque.

Cualquiera de ellas puede a su vez, almacenar diferentes tipos de datos:

- **Variables numéricas:**

  Las variables numéricas almacenan valores numéricos, como enteros o decimales. Podemos realizar operaciones aritméticas básicas con ellas, como suma, resta, multiplicación y división. Por ejemplo:

```
let edad = 25;
let altura = 1.75;

let suma = edad + altura;
let resta = edad - altura;
let multiplicacion = edad * altura;
let division = edad / altura;
```

- **Variables de cadena de texto (string):**

  Las variables de tipo string almacenan texto. Podemos concatenar cadenas de texto utilizando el operador +. También podemos utilizar métodos de cadenas para manipular y obtener información sobre ellas. Por ejemplo:

```
let nombre = "Juan";
let apellido = "Pérez";

let nombreCompleto = nombre + " " + apellido;
let longitudNombre = nombre.length;
let mayusculas = apellido.toUpperCase();
```

- **Variables booleanas:**

  Las variables booleanas almacenan valores de verdad, que pueden ser true (verdadero) o false (falso). Podemos utilizar operadores lógicos como && (y), || (o) y ! (negación) para realizar operaciones booleanas. Por ejemplo:

```
let esMayorDeEdad = true;
let tieneLicencia = false;

let puedeConducir = esMayorDeEdad && tieneLicencia;
let noPuedeVotar = !esMayorDeEdad || tieneLicencia;
```

Nomenclaturas de Variables

Las convenciones de nomenclatura son reglas o pautas que se siguen al nombrar variables en un lenguaje de programación para mejorar la legibilidad del código y facilitar su comprensión. En JavaScript, algunas de las convenciones de nomenclatura más comunes son las siguientes:

- **Camel Case:** Es una convención en la que cada palabra en el nombre de la variable comienza con mayúscula, excepto la primera. Por ejemplo:

```
let miVariable;
let sumaTotal;
```

- **Pascal Case:** Similar a Camel Case, pero la primera letra también está en mayúscula. A menudo se utiliza para nombrar clases o constructores en JavaScript.

```
function MiFuncion() {
  // código aquí
}
```

- **Snake Case:** Las palabras están en minúsculas y se separan con guiones bajos.

```
let mi_variable;
let suma_total;
```

- **Kebab Case:** Similar a Snake Case, pero se usan guiones en lugar de guiones bajos.

```
let mi-variable;
let suma-total;
```

- **Prefijos y sufijos descriptivos:** Agregar prefijos o sufijos descriptivos a las variables puede mejorar la claridad del código. Por ejemplo:

```
let nombreUsuario;
let contadorElementos;
```

- **Evitar nombres genéricos:** Trata de evitar nombres genéricos como temp, dato, o valor. Utiliza nombres descriptivos que indiquen el propósito o contenido de la variable.
- **Evitar abreviaturas confusas:** Si bien algunas abreviaturas son comunes y comprensibles, evita aquellas que puedan ser confusas o mal interpretadas.
- **Consistencia:** Mantén consistencia en el estilo de nomenclatura a lo largo de tu código y sigue las convenciones establecidas en tu equipo o proyecto.

Tipos de datos en JavaScript

Los tipos de datos JavaScript se dividen en dos grupos: tipos primitivos y tipos objeto.

Primitivos	
Dato	**Descripción**
Numérico	Números, ya sea enteros o reales.
String	Cadenas de texto.
Boolean	Valores lógicos como true o false.
null	Cuando un dato no existe.
undefined	Cuando no se le asigna un valor a la variable.

Los tipos objeto tienen sus propias subdivisiones

Objetos	
Tipo De Objeto	**Descripción**
Tipo predefinido de JavaScript	Date: fechas
Tipo predefinido de JavaScript	RegExp: expresiones regulares
Tipo predefinido de JavaScript	Error: datos de erros
Tipos definidos por el programador / usuario	Funciones Simples y Clases
Arrays	Serie de elementos o formación tipo vector o matriz. Lo consideramos un objeto especial

Enlace

Enlazar el código JavaScript , permite que se ejecute en el contexto del navegador web cuando se carga o interactúa con el archivo HTML. Además, es recomendable colocar los enlaces externos e internos dentro de la sección <body> al final del archivo HTML para asegurarse de que los elementos HTML se hayan cargado antes de que el código JavaScript intente manipularlos.

Recuerda que al enlazar JavaScript con HTML, es necesario asegurarse de utilizar la sintaxis correcta y verificar la ruta o ubicación de los archivos JavaScript para garantizar que se carguen y ejecuten correctamente en el contexto del archivo HTML.

Existen tres formas principales de enlazar JavaScript con HTML:

1. **Enlace externo:**

Esta forma implica crear un archivo JavaScript separado con extensión ".js" y enlazarlo en el archivo HTML utilizando la etiqueta <script>.

El enlace externo se realiza mediante el atributo src, el cual especifica la ruta o URL del archivo JavaScript externo.

Por ejemplo: <script src="script.js"></script>.

Al utilizar esta forma, es importante asegurarse de que la ruta del archivo JavaScript sea correcta y esté accesible desde el archivo HTML.

2. **Enlace interno**:

Esta forma implica escribir el código JavaScript directamente dentro del archivo HTML utilizando la etiqueta <script>.

El código JavaScript puede colocarse en la sección <head> o en la sección <body> del archivo HTML.

```
<script>
  // Código JavaScript aquí
</script>
```

3. **Eventos en línea:**

Esta forma implica utilizar atributos de eventos en línea en elementos HTML para asociar directamente código JavaScript.

Los atributos de eventos en línea, como onclick, onload, onmouseover, etc., se utilizan para especificar el código JavaScript que se ejecutará cuando ocurra un evento determinado.

Por ejemplo: <button onclick="alert('¡Hiciste clic en el botón!')">Haz clic</button>.

Esta forma es útil para agregar interactividad rápida y sencilla a elementos HTML específicos.

Entrada y salida de datos

Una consola web es una herramienta que se utiliza principalmente para registrar información asociada a una página web como: solicitudes de red, JavaScript, errores de seguridad, advertencias, CSS, etc. Esta, nos permite interactuar con una página web ejecutando una expresión JavaScript en el contenido de la página. En JavaScript, Console, es un objeto que proporciona acceso a la consola de depuración del navegador. Podemos abrir una consola en el navegador web Chrome, usando: Ctrl + Shift + J para Windows/Linux y Option + Command ⌘ + J para Mac.

El objeto "console" nos proporciona varios métodos. A continuación te presentamos algunos de ellos y una breve descripción sobre su uso:

Método	Descripción	Ejemplo

console.log()	Se utiliza principalmente para registrar (imprimir) la salida en la consola. Podemos poner cualquier tipo dentro del log (), ya sea una cadena, matriz, objeto, booleano, etc.	console.log("abc");
		console.log(123);
		console.log([1,2,3,4]);

console.error()
 	
console.warn
	Error: se utiliza para registrar mensajes de error en la consola.
	Warn: se usa para registrar mensajes de advertencia
		console.error("Mensaje de error");
		
		console.warn("Mensaje de advertencia");

console.clear()	Se usa para limpiar la consola.	console.clear();

console.time()
 	
console.timeEnd()
	Siempre que queramos saber la cantidad de tiempo empleado por un bloque o una función, podemos hacer uso de los métodos time() y timeEnd().	console.time('abc');
		
		console.timeEnd('abc');

console.table()	Este método nos permite generar una tabla dentro de una consola. La entrada debe ser una matriz o un objeto que se mostrará como una tabla.	console.table({'a':„ 'b':2});

console.count()	Este método se usa para contar el número que la función alcanza con este método de conteo.	for(let i=0;i<5;i++){
		console.count(i);
		}

Objeto window

El objeto window de JavaScript nos sirve para controlar la ventana del navegador. Es el objeto principal en la jerarquía y contiene las propiedades y métodos para controlar la ventana del navegador. Tiene algunos métodos como:

Salida de datos:

Para mostrar información al usuario, podemos utilizar el método console.log(). Este método imprime un mensaje o valor en la consola del navegador. Por ejemplo:

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

También podemos utilizar el método alert() para mostrar mensajes emergentes en una ventana de diálogo. Por ejemplo:

```
alert("¡Bienvenido!");
```

Además, podemos manipular el contenido de una página web utilizando el método document.write(). Este método es útil cuando deseamos mostrar información directamente en el cuerpo del documento HTML. Por ejemplo:

```
document.write("El resultado es: " + resultado);
```

Entrada de datos:

Para obtener datos del usuario, podemos utilizar el método prompt(). Este método muestra un cuadro de diálogo que permite al usuario ingresar un valor. El valor ingresado por el usuario se almacena en una variable. Por ejemplo:

```
var nombre = prompt("Ingrese su nombre:");
```

También podemos utilizar formularios HTML para obtener datos del usuario. Los datos ingresados en los campos de un formulario se pueden recuperar utilizando JavaScript. Para ello, podemos acceder a los elementos del formulario mediante su identificador o mediante métodos como getElementById().

Por ejemplo:

HTML:

```
<input type="text" id="nombreInput">
<button onclick="obtenerNombre()">Enviar</button>
```

JavaScript

```
function obtenerNombre() {
  var nombre = document.getElementById("nombreInput").value;
  console.log("Hola, " + nombre + "!");
}
```

Typeof

La función *typeof* se utilizada para obtener el tipo de dato que tiene una variable.

```
console.log(typeof 42);

    // expected output: "number"

    console.log(typeof 'blubber');

    // expected output: "string"

    console.log(typeof true);

    // expected output: "boolean"
```