COLOMBIA POTENCIA DE LA VIDA

TIC

UNIVERSIDAD SERGIO ARBOLEDA

Programación
Nivel Explorador

TALENTO TECH

UNIVERSIDAD SERGIO ARBOLEDA

PREPARACIÓN SEMANA 5

INTRODUCCIÓN A JAVASCRIPT

- **Origen y Evolución**
  - Década de 1990: Netscape, un navegador web, buscaba un lenguaje de script para agregar interactividad a sus páginas.
  - Brendan Eich: Desarrolló el lenguaje en 1995 en solo 10 días. Inicialmente se llamó "LiveScript", pero se renombró a "JavaScript" por marketing.
  - Colaboración con ECMA International: Se estandarizó como ECMAScript para garantizar la compatibilidad entre navegadores.
  - Hitos Importantes en la Evolución de JavaScript:
    - 1996 - 2000: Se introduce en Netscape y Microsoft Internet Explorer. Crecimiento inicial de popularidad y funcionalidad.
    - 2005 - 2010: Aparición de bibliotecas y frameworks como jQuery, Prototype, y el crecimiento de AJAX para comunicación asíncrona.
    - 2010 - Presente: Auge de Node.js, Angular, React, Vue.js y otros frameworks de front-end. JavaScript se convierte en un lenguaje de programación omnipresente.

INTRODUCCIÓN A JAVASCRIPT

- **Importancia en el Desarrollo Web:**
  - Interactividad Dinámica: Permite la creación de efectos visuales, validación de formularios, animaciones y más, sin recargar la página.
  - Desarrollo Front-end y Back-end: Con Node.js, JavaScript ahora puede ser utilizado para el desarrollo de servidores, aplicaciones de línea de comandos y más.
  - Frameworks y Bibliotecas: La comunidad de JavaScript ha creado herramientas poderosas que simplifican el desarrollo web, facilitando la creación de aplicaciones complejas.
  - En conclusión JavaScript ha evolucionado desde un simple lenguaje de script para páginas web hasta convertirse en un lenguaje esencial en el desarrollo web moderno. Su versatilidad y capacidad para crear experiencias interactivas han contribuido enormemente a la experiencia del usuario en internet.

VARIABLES EN JAVASCRIPT

En JavaScript, las variables son contenedores para almacenar datos. La declaración de variables se puede realizar de diferentes maneras y el alcance de estas variables define dónde pueden ser accedidas dentro del código. Declaración de variables ‘var’, ‘let’, ‘const’

- var (ES5):
- Alcance: Tiene alcance de función o global.
- Reasignación: Permite reasignar valores y se puede declarar sin asignación inicial.
- Hoisting: Se "eleva" al inicio del contexto de su declaración.

Ejemplo

```
var x = 10;
var name;
```

```
function exampleVar() {
  for (var i = 0; i < 3; i++) {
    console.log(i);
  }
  console.log(i); // Imprimirá 3, ya que var no tiene alcance de bloque.
}
```

¿Qué significa que la variable se eleva al inicio del contexto de su declaración?

Cuando se usa var, la declaración de la variable se "eleva" al principio del contexto en el que está definida, pero no su inicialización (asignación de valor). Esto significa que la variable puede ser referenciada antes de ser declarada sin lanzar un error.

Ejemplo

```
console.log(x); // Resultado: undefined
var x = 5;
```

El comportamiento se interpreta realmente como:

```
var x; // La declaración de la variable es "elevada" al principio.
console.log(x); // Resultado: undefined
x = 5; // La inicialización se mantiene en su posición original.
```

❖ **let (ES6):**

- **Alcance:** Tiene alcance de bloque, lo que significa que solo es visible dentro del bloque donde se declara.
- **Reasignación:** Permite reasignar valores, pero no se puede declarar sin asignación inicial en un bloque.
- No se "eleva" (hoisting) como ‘var’.

Ejemplo

```
let y = 20;
let age = 30;
```

```
function exampleLet() {
    let j = 5;
    if (true) {
        let j = 10; // Variables separadas debido al alcance de bloque.
        console.log(j); // Imprimirá 10.
    }
    console.log(j); // Imprimirá 5.
}
```

- **const (ES6):**
  - Alcance: También tiene alcance de bloque y no puede ser reasignado.
  - Debe asignarse un valor al declararse y no se puede cambiar posteriormente.
  - El valor es inmutable, pero en el caso de objetos y arrays, las propiedades y elementos

Ejemplo

```
const z = 50;
const PI = 3.14159;
```

```
const TAX_RATE = 0.1; // Constante para una tasa impositiva.
const person = {
  name: 'John',
  age: 30
};

person.age = 31; // Modificación permitida en las propiedades del objeto.
```

Alcance de variables: Global y Local

- **Alcance Global:**
  - Las variables globales se definen fuera de cualquier función o bloque. Pueden ser accedidas y modificadas desde cualquier lugar del código, incluyendo funciones.

Ejemplo

```
var globalVar = 100;

function doSomething() {
  console.log(globalVar);
}
```

Alcance de variables: Global y Local

- **Alcance Local:**
  - Las variables locales son accesibles solo dentro de la función o bloque en la que se definen. No son accesibles desde fuera de la función o bloque.

Ejemplo

```
function doSomethingElse() {
  let localVar = 200;
  console.log(localVar);
}
```

- **Buenas practicas**

  - **Recomendaciones de Uso:** Utilizar const siempre que sea posible y solo usar let cuando sea necesaria la reasignación.
  - **Evitar var en ES6+:** En nuevos proyectos, se recomienda evitar var en favor de let y const}

TIPOS DE DATOS EN JAVASCRIPT

- **Tipos de datos primitivos:** Son los datos mas básicos y simples que se pueden representar, es decir números, strings, booleanos, null y Undefined

- **Números:** Representan valores numéricos, incluyen enteros y decimales.

Ejemplo

```
let numero = 10;
let decimal = 3.14;
```

- **Strings:** Representan secuencias de caracteres y se pueden crear con comillas simples o dobles.

Ejemplo

```
let texto = 'Hola, mundo!';
let nombre = "Juan";
```

TIPOS DE DATOS EN JAVASCRIPT

- **Booleanos:** Representan valores lógicos de verdadero (true) o falso (false). Resultan bastante útiles para evaluaciones condicionales.

Ejemplo

```
let esMayor = true;
let esMenor = false;
```

- **Null:** Representa la ausencia intencional de algún valor. Es un valor asignado a una variable que indica la ausencia de un valor significativo.

Ejemplo

```
let valorNulo = null;
```

TIPOS DE DATOS EN JAVASCRIPT

- **Undefined:** Indica que una variable ha sido declarada pero aún no se le ha asignado ningún valor. Es el valor predeterminado para variables no inicializadas.

Ejemplo

```
let valorIndefinido;
```

En consecuencia:

```
let edad = 25; // Número
let nombre = 'María'; // String
let esMayorDeEdad = true; // Booleano
let valorNulo = null; // Null
let valorIndefinido; // Undefined

console.log(typeof edad); // Resultado: number
console.log(typeof nombre); // Resultado: string
console.log(typeof esMayorDeEdad); // Resultado: boolean
console.log(typeof valorNulo); // Resultado: object (una peculiaridad en JavaScript, no es tipo de dato primitivo)
console.log(typeof valorIndefinido); // Resultado: undefined
```

TIPOS DE DATOS EN JAVASCRIPT

- **Tipos de datos no primitivos (Objetos):** Los tipos de datos no primitivos en JavaScript se refieren principalmente a los objetos. A diferencia de los tipos de datos primitivos que almacenan un solo valor, los objetos son estructuras de datos complejas que pueden almacenar colecciones de datos y funcionalidades.

Creación de Objetos

```
let persona1 = { nombre: 'Juan' };
let persona2 = persona1; // Ambas variables apuntan al mismo objeto

persona2.nombre = 'María';
console.log(persona1.nombre); // Resultado: 'María'
```

TIPOS DE DATOS EN JAVASCRIPT

- **Propiedades y Métodos**
  - **Propiedades:** Son pares clave-valor que describen las características del objeto.
  - **Métodos:** Son funciones asociadas al objeto y permiten realizar acciones.
  - **Acceso a propiedades y métodos:**

Notación de punto

```
console.log(persona.nombre); // Acceso a una propiedad
persona.edad = 31; // Modificación de una propiedad
```

Notación de Corchetes

```
console.log(persona['direccion'].ciudad); // Acceso a una propiedad usando corchetes
```

TIPOS DE DATOS EN JAVASCRIPT

- **Objetos y Referencias:** Los objetos se almacenan por referencia, lo que significa que cuando se asigna un objeto a una variable, se está asignando una referencia a la ubicación en memoria donde se guarda el objeto.

Ejemplo

```
let persona1 = { nombre: 'Juan' };
let persona2 = persona1; // Ambas variables apuntan al mismo objeto

persona2.nombre = 'María';
console.log(persona1.nombre); // Resultado: 'María'
```

TIPOS DE DATOS EN JAVASCRIPT

❖ **Tipos de Objetos Comunes**

- **Objetos Integrados:** Son aquellos predefinidos en el lenguaje y proporcionan funcionalidades útiles para manipular datos, fechas, expresiones regulares, realizar operaciones matemáticas y más, algunos de los mas comunes son:
  - **Array:** Permite almacenar y manipular listas de elementos.
  - **Funcionalidades:** Métodos como push, pop, splice, forEach, map, filter para agregar, eliminar, recorrer y manipular elementos en un array.

Ejemplo

```
let miArray = [1, 2, 3, 4, 5];

console.log(miArray.length); // Devuelve la longitud del array
miArray.push(6); // Agrega un elemento al final del array
miArray.pop(); // Elimina el último elemento del array
console.log(miArray); // Resultado: [1, 2, 3, 4, 5]
```

TIPOS DE DATOS EN JAVASCRIPT

- **Date:** Permite trabajar con fechas y horas
  - **Funcionalidades:** Crear, representar y manipular fechas. Métodos para obtener y establecer años, meses, días, horas, etc.

Ejemplo

```
let fechaActual = new Date();

console.log(fechaActual.getFullYear()); // Obtiene el año actual
console.log(fechaActual.getMonth()); // Obtiene el mes actual (0-11)
console.log(fechaActual.getDate()); // Obtiene el día del mes actual
```

TIPOS DE DATOS EN JAVASCRIPT

- **RegExp:** Utilizado para trabajar con expresiones regulares
- **Funcionalidades:** Realizar búsquedas y manipulaciones complejas de patrones en cadenas de texto utilizando métodos como *test y exec.*

Ejemplo

```
let expresion = /hello/i; // Buscar 'hello' de manera insensible a mayúsculas/minúsculas
let texto = 'Hello World!';

console.log(expresion.test(texto)); // Devuelve true si encuentra la coincidencia
console.log(expresion.exec(texto)); // Devuelve la coincidencia encontrada
```

TIPOS DE DATOS EN JAVASCRIPT

- **Math:** Proporciona funciones matemáticas
  - **Funcionalidades:** Métodos para operaciones matemáticas comunes como sqrt, pow, floor, ceil, random, entre otros.

Ejemplo

```
let numero = 4.5;

console.log(Math.sqrt(numero)); // Devuelve la raíz cuadrada
console.log(Math.floor(numero)); // Devuelve el entero más cercano hacia abajo
console.log(Math.random()); // Devuelve un número aleatorio entre 0 y 1
```

TIPOS DE DATOS EN JAVASCRIPT

- **JSON:** Manejo de datos en formato JSON(JavaScript Object Notation).
  - **Funcionalidades:** Métodos para convertir objetos JavaScript a strings JSON (JSON.stringify) y strings JSON a objetos JavaScript (JSON.parse).

Ejemplo

```
let persona = { nombre: 'Juan', edad: 25 };
let personaJSON = JSON.stringify(persona); // Convertir objeto a JSON

console.log(personaJSON); // Resultado: '{"nombre":"Juan","edad":25}'
let personaObjeto = JSON.parse(personaJSON); // Convertir JSON a objeto

console.log(personaObjeto.nombre); // Resultado: 'Juan'
```

TIPOS DE DATOS EN JAVASCRIPT

- **JSON:** Map, Set, WeakMap, WeakSet (ES6+): Estructuras de datos para almacenar colecciones de valores únicos.
  - **Funcionalidades: Map** para asociar claves y valores**, Set** para almacenar valores únicos**, WeakMap y WeakSet** para almacenar referencias débiles.

Ejemplo

```
let mapa = new Map();
mapa.set('nombre', 'Ana');
mapa.set('edad', 30);

console.log(mapa.get('nombre')); // Devuelve el valor asociado a 'nombre'
console.log(mapa.size); // Devuelve la cantidad de elementos en el mapa
```

TIPOS DE DATOS EN JAVASCRIPT

- **Promise (ED6+):** Proporciona una manera de manejar operaciones asincrónicas
  - **Funcionalidades:** Representa un valor que puede estar disponible ahora, en el futuro o nunca, permitiendo un mejor manejo de tareas asíncronas.

Ejemplo

```
let promesa = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve('Operación exitosa'); // Resuelve la promesa después de un tiempo
  }, 2000);
});

promesa.then((mensaje) => {
  console.log(mensaje); // Imprime el mensaje si la promesa se resuelve correctamente
});
```

TIPOS DE DATOS EN JAVASCRIPT

- **Intl :** Proporciona soporte para internacionalización (i18n) y localización(I10n).

  - **Funcionalidades:** Formateo de fechas, monedas, números, entre otros, basado en diferentes locales.

Ejemplo

```
let numero = 123456.789;

console.log(new Intl.NumberFormat('es-ES').format(numero)); // Formatea el número según el idioma
console.log(new Intl.DateTimeFormat('es-ES').format(new Date())); // Formatea la fecha según el idioma
```

TIPOS DE DATOS EN JAVASCRIPT

- **Objetos del Documento(DOM):** Representan elementos HTML y proporcionan métodos para interactuar con ellos en páginas web.
- **Objetos Personalizados:** Definidos por el usuario para representar estructuras de datos específicas para su aplicación.
- **Uso de Objetos en Aplicaciones:**
  - **Organización de Datos:** Los objetos permiten estructurar y organizar datos relacionados.
  - **Abstracción y Modularidad:** Facilitan la encapsulación de datos y funcionalidades.
  - **Interacción con APIs y Librerías:** Muchas APIs y librerías en JavaScript utilizan objetos para proporcionar funcionalidades complejas.

TIPOS DE DATOS EN JAVASCRIPT

Para terminar veamos un ejemplo completo

Manipulación y dinamismo de elementos HTML

```html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Manipulación de Elementos HTML</title>
</head>
<body>
  <h1 id="titulo">Texto Inicial</h1>
  <button id="boton">Cambiar Texto</button>

  <script>
    // Manipulación del texto al hacer clic en el botón
    let boton = document.getElementById('boton');
    let titulo = document.getElementById('titulo');

    boton.addEventListener('click', function() {
      titulo.textContent = 'Nuevo Texto';
    });
  </script>
</body>
</html>
```

TIPOS DE DATOS EN JAVASCRIPT

Aquí usamos JavaScript para manipular elementos HTML. Al hacer clic en el botón, el texto del elemento <h1> se actualiza dinámicamente.

- Otro ejemplo podría ser una calculadora básica, de tal modo que nuestro código seria algo así:

```javascript
// Funciones para operaciones matemáticas
function sumar(a, b) {
  return a + b;
}

function restar(a, b) {
  return a - b;
}

function multiplicar(a, b) {
  return a * b;
}

function dividir(a, b) {
  if (b === 0) {
    return "No se puede dividir por cero";
  }
  return a / b;
}
```

```javascript
// Interacción con el usuario mediante prompt
let operacion = prompt("Elige una operación: sumar, restar, multiplicar, dividir");
let numero1 = parseFloat(prompt("Ingresa el primer número"));
let numero2 = parseFloat(prompt("Ingresa el segundo número"));

let resultado;

// Realizar la operación seleccionada
switch (operacion) {
  case "sumar":
    resultado = sumar(numero1, numero2);
    break;
  case "restar":
    resultado = restar(numero1, numero2);
    break;
  case "multiplicar":
    resultado = multiplicar(numero1, numero2);
    break;
  case "dividir":
    resultado = dividir(numero1, numero2);
    break;
  default:
    resultado = "Operación no válida";
}

// Mostrar el resultado
console.log(`El resultado de la operación ${operacion} es: ${resultado}`);
```

TIPOS DE DATOS EN JAVASCRIPT

Aquí usamos funciones, interacción con el usuario a través de prompt, condicionales y un switch para realizar diferentes operaciones matemáticas.

```
const boton = document.querySelector("#boton");
boton.addEventListener("click", () => {
  console.log("¡El botón fue clickeado!");
});
```

EJERCICIO PRACTICO 1: Crear una calculadora simple que permita realizar operaciones básicas.
Pistas: Debemos crear un archivo llamado index.html, otro llamado calculator.js y el estilo llamado styles.css

¿Preguntas?