Introducción a Javascript

En este documento introduciremos el lenguaje de programación javascript, para la realización de ejercicios de la unidad 04, del módulo de lenguaje de marcas.

Módulo 1: Introducción a JavaScript

¿Qué es JavaScript?
JavaScript es un lenguaje de programación que permite agregar interactividad a las páginas web.
Ejemplo: Un script simple que muestra un mensaje en la consola.
Javascript: script.js

```
console.log("¡Hola, mundo!");
```

Incorporación de JavaScript en HTML
Se puede insertar JavaScript directamente en HTML con <script> o enlazar un archivo externo.
Ejemplo: HTML con un script de JavaScript interno.
Html: index.html

```
<script>
  console.log("Este es un script interno");
</script>
```

Variables y Tipos de Datos
Las variables pueden almacenar diferentes tipos de datos, como números o cadenas.
Ejemplo: Declaración de una variable y asignación de un valor.

Javascript: script.js

```
let mensaje = "¡Aprendiendo JavaScript!";
console.log(mensaje);

// Diferencia var y let
function pruebaVar() {
  var x = 1;
  if (true) {
    var x = 2;  // misma variable
    console.log(x);  // 2
  }
```

```
  console.log(x);  // 2
}

function pruebaLet() {
  let y = 1;
  if (true) {
    let y = 2;  // diferente variable
    console.log(y);  // 2
  }
  console.log(y);  // 1
}
```

Módulo 2: Estructuras Básicas de Control

Operadores y Expresiones
Los operadores permiten realizar operaciones matemáticas o comparar valores.
Ejemplo: Suma de dos números.
Javascript: script.js

```
let suma = 5 + 3;
console.log(suma); // Muestra 8
```

Estructuras Condicionales

if y else permiten ejecutar código basado en condiciones.
Ejemplo: Comprobación de una condición.
Javascript: script.js

```
let numero = 10;
if (numero > 5) {
  console.log("El número es mayor que 5");
}
```

Funciones
Las funciones son bloques de código reutilizables.
Ejemplo: Función que muestra un mensaje.
Javascript: script.js

```
function mostrarMensaje() {
  console.log("Mensaje desde una función");
}
mostrarMensaje();
```

Módulo 3: Selección y manipulación del DOM

¿Qué es el DOM?
El DOM es una representación del HTML de una página web que permite su manipulación.
Ejemplo: Cambiar el contenido de un elemento.
Javascript: script.js

```
// Suponiendo que existe un elemento con id="demo" en el HTML
document.getElementById("demo").textContent = "¡Hola DOM!";
```

Selección de Elementos del DOM
document.getElementById() selecciona un elemento del HTML por su ID.
Ejemplo: Obtención de un valor de un campo de entrada.
Javascript: script.js

```
// Suponiendo que existe un campo de entrada con id="emailInput"
let email = document.getElementById('emailInput').value;
```

- Uso de document.querySelector y document.querySelectorAll
  Estos métodos seleccionan el primer elemento o todos los elementos que coincidan con un selector CSS respectivamente.
  Ejemplo: Selección de elementos.
  Javascript: script.js

  ```
  // Seleccionar el primer botón
  let boton = document.querySelector("button");

  // Seleccionar todos los elementos con la clase 'ejemplo'
  let elementos = document.querySelectorAll(".ejemplo");
  ```

Manipulación de Elementos del DOM
Cambiar el contenido, estilo y atributos de los elementos seleccionados.
Ejemplo: Cambiar el contenido de un elemento.
Javascript: script.js

```
// Cambiar el texto de un elemento
document.querySelector("#demo").textContent = "Nuevo texto";
```

Manipulación del Contenido Interno con innerHTML
innerHTML se usa para obtener o establecer el contenido HTML de un elemento.
Ejemplo: Cambio del contenido HTML de un elemento.

Javascript: script.js

```
// Cambiar el HTML de un elemento
document.getElementById("demo").innerHTML = "<b>Nuevo contenido</b>";
```

Eventos
Los eventos son acciones que ocurren en la página, como clics o pulsaciones de teclas.
Ejemplo: Ejecutar una función cuando se hace clic en un botón.
Javascript: script.js

```
// Suponiendo que existe un botón con id="miBoton"
document.getElementById('miBoton').addEventListener('click', function() {
  console.log("Botón clickeado");
});
// Otra forma
document.querySelector("#miBoton").addEventListener('click', function() {
  console.log("Botón clickeado");
});
```

Módulo 4: Asincronía y Peticiones de Red

Asincronía en JavaScript
JavaScript puede realizar tareas sin bloquear la ejecución del script.
Ejemplo: Uso de setTimeout.
Javascript: script.js

```
setTimeout(function() {
  console.log("Mensaje después de 2 segundos");
}, 2000);
```

Introducción a AJAX y Fetch API
fetch permite realizar peticiones HTTP para obtener datos de servidores.
Ejemplo: Obtener datos de un URL.
Javascript: script.js

```
fetch('https://api.example.com/datos')
  .then(response => response.json())
  .then(data => console.log(data));

// Otro formato
fetch('ruta/a/archivo.xml')
  .then(response => response.text())
  .then(data => {
    // Procesar los datos aquí
```

```
});
```

Promesas
Las promesas son objetos que representan la finalización o el fracaso de una operación asíncrona.
Ejemplo: Creación de una promesa.
Javascript: script.js

```
let miPromesa = new Promise((resolve, reject) => {
  // Código asíncrono aquí
});
```

Módulo 6: Trabajo con Arrays y Objetos

Introducción a Arrays
Los arrays almacenan múltiples valores en una sola variable.
Ejemplo: Crear un array y acceder a sus elementos.
Javascript: script.js

```
let frutas = ["Manzana", "Banana", "Cereza"];
console.log(frutas[1]); // Muestra "Banana"
```

Introducción a Objetos
Los objetos almacenan datos en forma de pares clave-valor.
Ejemplo: Crear un objeto y acceder a sus propiedades.
Javascript: script.js

```
let persona = {
  nombre: "Ana",
  edad: 25
};
console.log(persona.nombre); // Muestra "Ana"
```

Recorrido de Arrays y Objetos
Iterar sobre los elementos de un array o las propiedades de un objeto.
Ejemplo: Uso de forEach en un array.
Javascript: script.js

```
frutas.forEach(function(fruta) {
  console.log(fruta);
});
```

Módulo 7: Manejo de Errores y Depuración

Manejo de Errores
Capturar y manejar errores para prevenir fallos en el código.
Ejemplo: Uso de try y catch.
Javascript: script.js

```
try {
  // Intentar ejecutar código que puede fallar
  let resultado = posibleError();
} catch (error) {
  console.log(error);
}
```

Depuración en JavaScript
Uso de console.log.
Herramientas de depuración en navegadores y extensiones de editores.

Módulo 8: Fundamentos Avanzados

Closures y Ámbito de Variables
Closures permiten acceder a variables exteriores desde una función interna.
Ejemplo: Creación de un closure.
Javascript: script.js

```
function crearSaludo(saludo) {
  return function(nombre) {
    console.log(saludo, nombre);
  };
}
let saludaHola = crearSaludo("Hola");
saludaHola("Pedro"); // Muestra "Hola Pedro"
```

Patrones de Diseño Básicos
Estrategias reutilizables para resolver problemas comunes de programación.
Ejemplo: Patrón Módulo.
Javascript: script.js

```
let contador = (function() {
  let valor = 0;
  return {
    incrementar: function() { valor++; },
    obtenerValor: function() { return valor; }
  };
})();
```

```
contador.incrementar();
console.log(contador.obtenerValor()); // Muestra 1
```

Uso del Signo Dólar $ en JavaScript

El signo $ no es una característica propia de JavaScript puro, sino que suele asociarse con bibliotecas como jQuery.

En algunos scripts, $ se utiliza como alias para ciertas funciones, como una forma más corta de document.querySelector o document.querySelectorAll.

Ejemplo de uso típico en jQuery:

Javascript: script.js

```
//Ejemplo de uso como alias en JavaScript puro:
// Definir $ como un alias de document.querySelector
const $ = document.querySelector;
$("#demo").textContent = "Usando $ como alias";

// En jQuery, seleccionar elementos con clase 'boton' y agregar un evento de clic
$(".boton").click(function() {
  console.log("Botón clickeado");
});
```

ES6 y Más Allá

Nuevas características y mejoras en JavaScript.

Ejemplo: Uso de arrow functions y template literals.

Javascript: script.js

```
let suma = (a, b) => a + b;
console.log(`La suma es: ${suma(2, 3)}`); // Muestra "La suma es: 5"
```