1. Fundamentos del Lenguaje 🛠️

Descripción

Abarca los elementos estructurales básicos del código: cómo se almacenan y se manipulan los datos (**variables y tipos**), y cómo se dirige el flujo de ejecución del programa (**estructuras de control y funciones**).

Ejemplo en JavaScript Puro

JavaScript

```
// Variables y Tipos de Datos
const nombre = 'Juan';      // String (const: no cambia)
let edad = 25;              // Number (let: puede cambiar)
const esMayorDeEdad = true;  // Boolean
let notas = [8, 9, 7];      // Array

// Estructuras de Control (If/Else)
if (edad >= 18) {
  console.log(nombre + ' puede votar.');
} else {
  console.log(nombre + ' aún no puede votar.');
}

// Bucle (For)
for (let i = 0; i < notas.length; i++) {
  console.log('Nota ' + (i + 1) + ': ' + notas[i]);
}

// Función Flecha
const saludar = (n) => {
```

```
  return `Hola, ${n}.`;
};

console.log(saludar(nombre));
```

Explicación

Estos son los bloques de construcción. Las **variables** almacenan cualquier información que necesite el programa. Las **estructuras de control** (como if y for) permiten que el código tome decisiones y repita tareas. Las **funciones** organizan el código en bloques reutilizables.

---

2. JavaScript Asíncrono (async/await) ⏱️

Descripción

Es el mecanismo utilizado para manejar operaciones que no terminan inmediatamente (como llamadas a bases de datos o peticiones a una API). En lugar de congelar el programa, el código asíncrono ejecuta la tarea en segundo plano y continúa con otras instrucciones, esperando la respuesta en un momento posterior usando **Promesas** y las palabras clave **async/await**.

Ejemplo en JavaScript Puro

JavaScript

```
// Simulación de una llamada a una API que tarda 2 segundos
function obtenerDatosDeUsuario() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve({ id: 101, nombre: 'Alice', email: 'alice@ejemplo.com' });
    }, 2000); // 2000 milisegundos = 2 segundos
```

```
  });
}

// La función que usa la operación lenta debe ser declarada como 'async'
async function mostrarDatos() {
  console.log('1. Iniciando la obtención de datos...');

  // 'await' pausa esta función hasta que la Promesa se resuelva (después de 2s)
  const datos = await obtenerDatosDeUsuario();

  console.log('2. Datos obtenidos satisfactoriamente.');
  console.log('3. Usuario:', datos.nombre);
}

mostrarDatos();
```

Explicación

La palabra clave **async** convierte una función en asíncrona, permitiéndole usar **await** en su interior. **await** se coloca delante de una función que devuelve una Promise y detiene temporalmente la ejecución *dentro de esa función async* hasta que la promesa se resuelva (éxito) o se rechace (error). Esto hace que el código asíncrono se vea y se lea como si fuera síncrono.

---

3. Módulos y Entorno Node.js 🧩

Descripción

Los **módulos** permiten dividir una aplicación grande en archivos pequeños e independientes. El **entorno Node.js** utiliza el sistema de módulos para cargar dependencias (como librerías) y compartir código entre archivos usando las sintaxis export (para hacer disponible el código) e import (para usar el código en otro archivo).

Ejemplo en JavaScript Puro

Archivo: calculadora.js (Módulo Exportado)

JavaScript

```
// Exporta la función para que otros archivos puedan usarla
export function sumar(a, b) {
  return a + b;
}

// Exporta una constante
export const PI = 3.14159;
```

Archivo: app.js (Módulo Importador)

JavaScript

```
// Importa las funciones y constantes que se necesitan
import { sumar, PI } from './calculadora.js';

const resultado = sumar(10, 5);
console.log('Suma:', resultado); // Salida: Suma: 15
console.log('Valor de PI:', PI);
```

Explicación

Este sistema evita colisiones de nombres y promueve la reutilización. **export** define qué partes del archivo son públicas. **import** le dice a Node.js dónde buscar el código que se necesita (./calculadora.js) y qué partes usar (sumar, PI).

---

4. Estructuras de Datos Comunes (Arrays y Objetos) 🧱

Descripción

Son los contenedores de datos más importantes en JavaScript. Los **Objetos** almacenan colecciones de datos mediante pares **clave-valor** (propiedades). Los **Arrays** almacenan listas ordenadas de valores.

Ejemplo en JavaScript Puro

JavaScript

```
// Objeto
const libro = {
  titulo: 'Cien años de soledad',
  autor: 'Gabriel García Márquez',
  paginas: 496,
  genero: 'Realismo mágico'
};

// Acceder a las propiedades del Objeto
console.log(`Título: ${libro.titulo}`); // Notación de punto

// Array
const frutas = ['manzana', 'banana', 'kiwi', 'mango'];
```

```
// Acceder a elementos del Array
console.log(`La primera fruta es: ${frutas[0]}`);

// Manipulación de Arrays (método .map)
const frutasMayusculas = frutas.map(fruta => fruta.toUpperCase());
console.log(frutasMayusculas);
// Salida: ['MANZANA', 'BANANA', 'KIWI', 'MANGO']
```

Explicación

Los **Objetos** son ideales para representar entidades (como un libro, una persona o una configuración). Los **Arrays** son esenciales cuando se trabaja con listas o colecciones de datos, y sus métodos integrados (map, filter, forEach) son herramientas poderosas para transformar y procesar grandes conjuntos de información.