■ Ficha 0 – Introducción a JavaScript

■ 1. ¿Qué es JavaScript?

JavaScript es un lenguaje **interpretado**, **dinámico** y **multiplataforma**, utilizado tanto en el **navegador** (frontend) como en el **servidor** con Node.js (backend). Es **sensible a mayúsculas y minúsculas** (*case sensitive*): nombre ≠ Nombre.

Frontend ↔ Backend (flujo simplificado)

```
// Navegador (cliente)          // Servidor (Node + Express)
fetch('http://localhost:3000/usuarios')  app.get('/usuarios', (req, res) => {
  .then(r => r.json())                     res.json(usuarios);
  .then(data => console.log(data));      });
```

■ 2. Sintaxis básica y buenas prácticas

• Termina sentencias con ; (opcional pero recomendable). • Usa **let** para variables mutables y **const** para constantes. • Evita var. • Usa sangrías y llaves {} claras. • Comenta tu código.

```
// Comentarios
// una línea
/* varias
   líneas */

const PI = 3.1416;       // const para valores fijos
let contador = 0;        // let para variables que cambian
```

■ 3. Tipos de datos y operadores

Tipo	Ejemplo	Notas
String	"Hola"	Texto
Number	42, 3.14	Enteros y decimales
Boolean	true / false	
Array	[1, 2, 3]	Lista ordenada
Object	{ nombre: 'Ana' }	Pares clave-valor
Undefined	undefined	Sin asignar
Null	null	Vacío intencionado
Symbol / BigInt	Symbol('id'), 10n	Avanzado

```
// Operadores
let a = 10, b = 5;
a + b; a - b; a * b; a / b; a % b;

// Comparación
5 == "5"   // true  (solo valor, ¡cuidado!)
5 === "5"  // false (valor y TIPO)

// Lógicos y ternario
true && false; true || false; !true;
const edad = 20;
const mensaje = edad >= 18 ? "Adulto" : "Menor";
```

■ 4. Estructuras de control

```
if (edad >= 18) {
  console.log("Eres mayor de edad");
} else {
  console.log("Eres menor de edad");
}

for (let i = 0; i < 3; i++) { console.log("i =", i); }

let j = 0;
while (j < 2) { console.log("j =", j); j++; }

// Recorriendo arrays
const nombres = ["Ana", "Juan", "Nora"];
nombres.forEach(n => console.log(n));
```

■ 5. Funciones y ámbito

```
// Declaración clásica
function saludar(nombre) {
  return "Hola " + nombre;
}

// Función flecha (moderna)
const saludar2 = (nombre) => `Hola ${nombre}`;

// Ámbito de variables
let x = 1;
function ejemplo() {
  let x = 2;           // diferente a la x exterior
  return x;
}
```

■ 6. Objetos y arrays

```
const persona = { nombre: "Ana", edad: 30 };
console.log(persona.nombre); // "Ana"

const usuarios = [
  { id: 1, nombre: "María", edad: 30 },
  { id: 2, nombre: "Juan",  edad: 25 }
];

// Métodos útiles
usuarios.push({ id: 3, nombre: "Nora", edad: 27 });
const mayores = usuarios.filter(u => u.edad >= 26);
const soloNombres = usuarios.map(u => u.nombre);
const juan = usuarios.find(u => u.id === 2);
```

■ 7. Desestructuración y JSON

```
// Desestructuración de objetos
const usuario = { nombre: "Lucía", email: "lucia@mail.com", edad: 22 };
```

```
const { nombre, email, edad } = usuario;

// JSON ↔ Objeto
const texto = JSON.stringify(usuario);  // objeto → JSON
const copia = JSON.parse(texto);        // JSON → objeto
```

Nota:

JSON usa comillas dobles para claves y valores cuando está en texto. En el código, los objetos JavaScript no las necesitan para las claves simples.

■ 8. Node.js y Express (concepto y ejemplo)

Node.js ejecuta JavaScript en el servidor. Express es un framework minimalista para crear APIs. Usa **app.use(express.json())** para traducir el JSON que envía el cliente a req.body.

```
const express = require('express');
const app = express();
const PORT = 3000;

// Traductor JSON → req.body
app.use(express.json());

// GET: listar usuarios
const usuarios = [
  { id: 1, nombre: "María", email: "maria@mail.com", edad: 30 },
  { id: 2, nombre: "Juan",  email: "juan@mail.com",  edad: 25 }
];

app.get('/usuarios', (req, res) => {
  res.json(usuarios);
});

// GET por id
app.get('/usuarios/:id', (req, res) => {
  const id = Number(req.params.id);
  const u = usuarios.find(x => x.id === id);
  if (!u) return res.status(404).json({ error: "No encontrado" });
  res.json(u);
});

app.listen(PORT, () => console.log(`Servidor en http://localhost:${PORT}`));
```

■ 9. Buenas prácticas para el reto

• Valida datos antes de guardar. • Devuelve códigos HTTP coherentes (200/201/400/404). • Nombra bien las variables. • Comenta lo justo. • Prueba con Thunder Client. • Guarda tu colección de peticiones.

■ 10. Mini chuleta de sintaxis

```
// Variables
let x = 1; const y = 2;

// Función flecha
```

```
const sumar = (a, b) => a + b;

// Desestructuración
const { id, nombre } = { id: 1, nombre: "Ana" };

// Array → map / filter / find
arr.map(f); arr.filter(f); arr.find(f);

// Express JSON
app.use(express.json());
```

■ 11. Comandos esenciales (Node.js y Express)

Comando	Descripción
npm init -y	Crear package.json rápido
npm install express	Instalar Express
node app.js	Ejecutar el servidor
npm install nodemon -D	Instalar nodemon (desarrollo)
npx nodemon app.js	Arrancar con reinicio automático

Consejo clave:

> “**express.json()** actúa como un traductor: convierte el JSON que envía el cliente en un objeto JavaScript accesible en req.body”.