



Materia:Desarrollo Apps Moviles

Profesor: Ivan Isay Guerra Lopez

Alumno:Diego Irineo

Fecha de entrega:22/9/25

a) Desestructuración de objetos

Concepto clave: La desestructuración permite extraer propiedades de un objeto (incluyendo anidadas) directamente en variables con sintaxis compacta

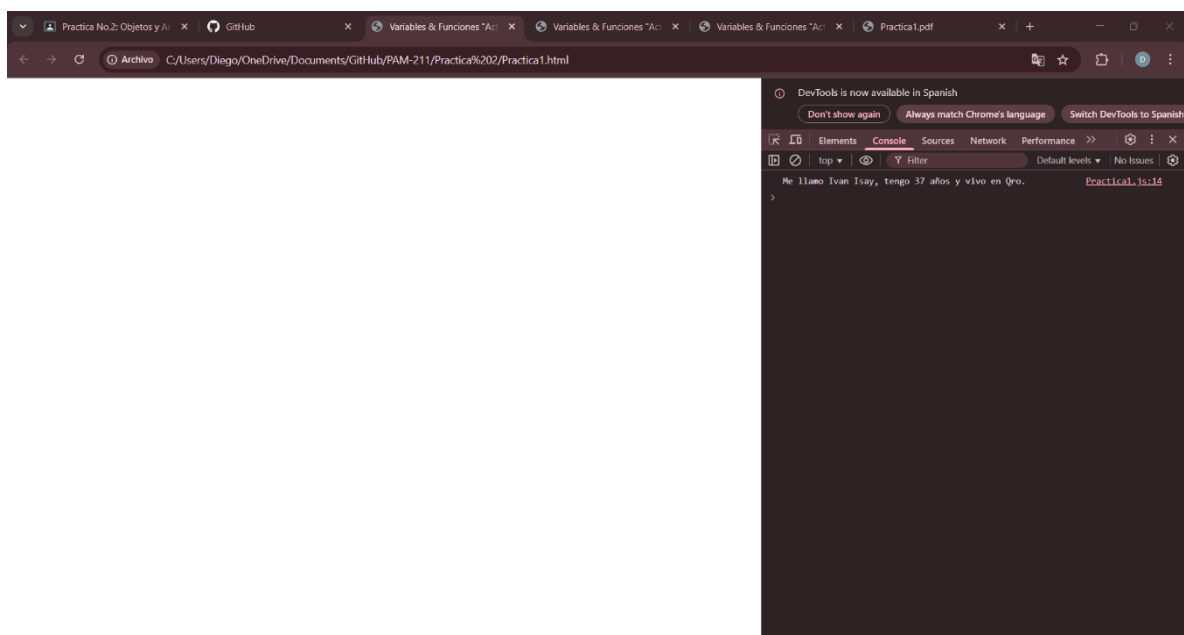
The image shows a Visual Studio Code editor window with the following details:

- Menu Bar:** File, Edit, Selection, View, Go, Run, and a search icon.
- Explorer Sidebar (Left):**
  - Root: PAM-211
  - Folder: Actividad 2
    - Folder: Practica 2
      - Practica1.html
      - Practica1.js** (selected)
      - Practica2.html
      - Practica2.js
      - Practica3.html
      - Practica3.js
    - Folder: Practica1
      - index.html
      - Practica1
      - Practica1.pdf
      - Practica2
      - Practica2.html
      - Practica3
      - Practica3.html
      - Prueba
      - .gitattributes
      - MenuPAM.java
  - OUTLINE
  - TIMELINE
  - PROJECTS
  - RUN CONFIGURATION
- Editor Window:**
  - Tab: JS Practica1.js X
  - Content:
 

```

1 // a.
2
3 const persona = {
4     nombre: "Ivan Isay",
5     edad: 37,
6     direccion: {
7         ciudad: "Qro",
8         pais: "MX"
9     }
10 };
11
12 const { nombre, edad, direccion: { ciudad } } = persona;
13 const mensaje = `Me llamo ${nombre}, tengo ${edad} años y vivo en ${ciudad}`;
14 console.log(mensaje);
15

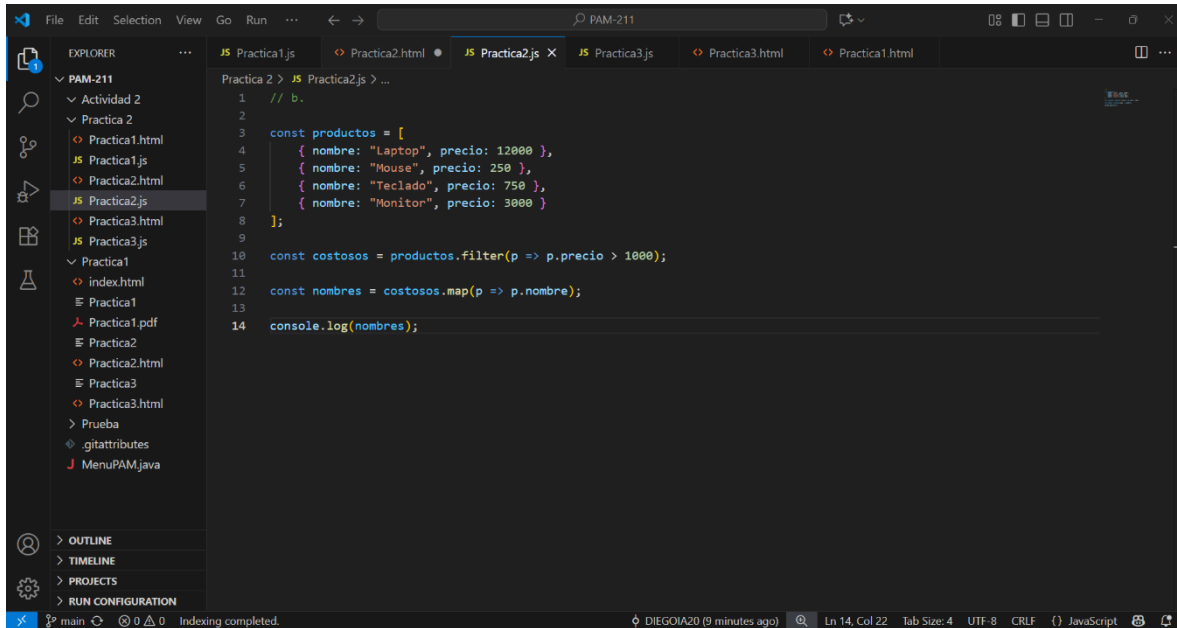
```
- Status Bar (Bottom):**
  - main
  - Ln 15, Col 1
  - Spaces: 4
  - UTF-8
  - CRLF
  - () JavaScript
  - Indexing completed.



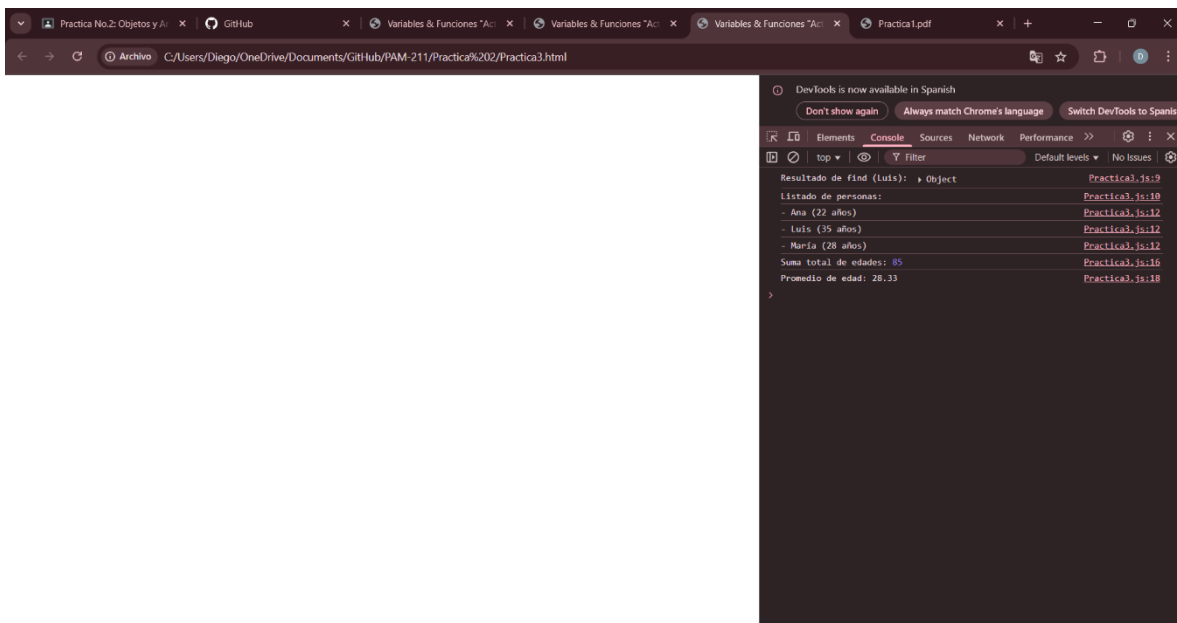
## b) Filtrado y transformación de arreglos

filter: devuelve un nuevo arreglo con los elementos que cumplen una condición

map: transforma cada elemento a otra forma (en este caso, solo el nombre)



```
1 // b.
2
3 const productos = [
4   { nombre: "Laptop", precio: 12000 },
5   { nombre: "Mouse", precio: 250 },
6   { nombre: "Teclado", precio: 750 },
7   { nombre: "Monitor", precio: 3000 }
8 ];
9
10 const costosos = productos.filter(p => p.precio > 1000);
11
12 const nombres = costosos.map(p => p.nombre);
13
14 console.log(nombres);
```



```
Resultado de find (luis): Object
listado de personas:
- Ana (22 años)
- Luis (35 años)
- María (28 años)
Suma total de edades: 85
Promedio de edad: 28.33
```

c) Búsqueda, recorrido y acumulación

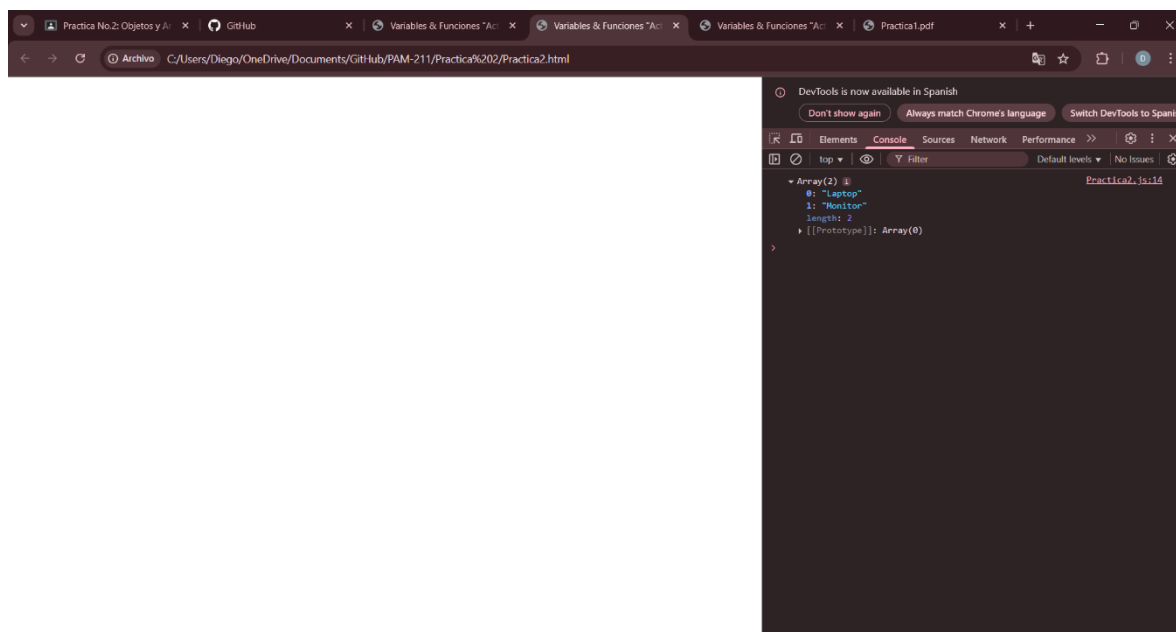
find: retorna el primer elemento que cumple la condición

forEach: ejecuta un efecto secundario por cada elemento

reduce: acumula valores y produce un único resultado

The screenshot shows an IDE with a dark theme. The Explorer panel on the left shows a project structure with folders 'PAM-211', 'Actividad 2', and 'Practica 2'. Inside 'Practica 2', there are files 'Practica1.html', 'Practica2.html', 'Practica3.html', and 'Practica3.js'. The 'Practica3.js' file is selected and its content is displayed in the main editor. The code is as follows:

```
1 // c.
2 const personas = [
3   { nombre: "Ana", edad: 22 },
4   { nombre: "Luis", edad: 35 },
5   { nombre: "Maria", edad: 28 }
6 ];
7
8 const personaLuis = personas.find(p => p.nombre === "Luis");
9 console.log("Resultado de find (Luis):", personaLuis);
10 console.log("Listado de personas:");
11 personas.forEach(({ nombre, edad }) => {
12   console.log(`- ${nombre} (${edad} años)`);
13 });
14
15 const totalEdades = personas.reduce((acum, { edad }) => acum + edad, 0);
16 console.log("Suma total de edades:", totalEdades);
17 const promedioEdad = totalEdades / personas.length;
18 console.log("Promedio de edad:", promedioEdad.toFixed(2));
```



## Conclusión

Lectura clara de objetos: Con la desestructuración (ejercicio a) simplificamos el acceso a propiedades, incluso anidadas, haciendo el código más legible y menos repetitivo

Transformación declarativa de arreglos: Con filter y map (ejercicio b) separamos en pasos lógicos (filtrar → transformar) manteniendo inmutabilidad y expresividad

Búsqueda, recorrido y agregación: Con find, forEach y reduce (ejercicio c) cubrimos patrones típicos: localizar un elemento, ejecutar efectos secundarios y combinar valores en un resultado único