BÚSQUEDA CON ARREGLOS

```
// Arreglo de objetos
const estudiantes = [
  { id: 1, nombre: "Ana", edad: 20, carrera: "Ingeniería en Sistemas" },
  { id: 2, nombre: "Luis", edad: 22, carrera: "Ingeniería Civil" },
  { id: 3, nombre: "María", edad: 21, carrera: "Ingeniería en Sistemas" },
  { id: 4, nombre: "Carlos", edad: 23, carrera: "Arquitectura" },
  { id: 5, nombre: "Elena", edad: 20, carrera: "Ingeniería en Sistemas" }
];

// 1. Buscar un estudiante por nombre (retorna el primer encontrado)
const estudianteAna = estudiantes.find(est => est.nombre === "Ana");
console.log("Buscar estudiante Ana:", estudianteAna);

// 2. Filtrar estudiantes de Ingeniería en Sistemas
const sistemas = estudiantes.filter(est => est.carrera === "Ingeniería en Sistemas");
console.log("Estudiantes de Ingeniería en Sistemas:", sistemas);

// 3. Verificar si existe un estudiante de Arquitectura
const hayArquitectura = estudiantes.some(est => est.carrera === "Arquitectura");
console.log("¿Hay estudiante de Arquitectura?:", hayArquitectura);

// 4. Verificar si todos son mayores de 18 años
const todosMayores = estudiantes.every(est => est.edad > 18);
console.log("¿Todos mayores de 18?:", todosMayores);

// 5. Obtener solo los nombres de los estudiantes (map transforma el arreglo)
const nombres = estudiantes.map(est => est.nombre);
console.log("Nombres de los estudiantes:", nombres);

// 6. Buscar por ID usando find
const estudianteId3 = estudiantes.find(est => est.id === 3);
console.log("Estudiante con ID = 3:", estudianteId3);

// 7. Filtrar por edad mayor o igual a 22
const mayores22 = estudiantes.filter(est => est.edad >= 22);
console.log("Estudiantes con edad >= 22:", mayores22);
```

1) Estructura general

```
// Arreglo de objetos
const estudiantes = [
 { id: 1, nombre: "Ana",   edad: 20, carrera: "Ingeniería en Sistemas" },
 { id: 2, nombre: "Luis",  edad: 22, carrera: "Ingeniería Civil" },
 { id: 3, nombre: "María", edad: 21, carrera: "Ingeniería en Sistemas" },
 { id: 4, nombre: "Carlos",edad: 23, carrera: "Arquitectura" },
 { id: 5, nombre: "Elena", edad: 20, carrera: "Ingeniería en Sistemas" }
];
```

- // ...: comentario (no se ejecuta).
- const: crea una **constante** (no puedes reasignar la variable estudiantes, pero **sí** modificar su contenido si fuera un objeto/array).
- []: **arreglo** (lista ordenada).
- {}: **objeto** con **propiedades** id, nombre, edad, carrera.
- Cada elemento del arreglo es un **objeto estudiante**.

2) find: busca el primer elemento que cumpla la condición

```
const estudianteAna = estudiantes.find(est => est.nombre === "Ana");
console.log("Buscar estudiante Ana:", estudianteAna);
```

- estudiantes.find(callback): recorre el arreglo y devuelve el **primer** objeto que cumpla la condición; si no hay coincidencia, devuelve undefined.
- est => est.nombre === "Ana": **función flecha** (arrow function).
  - est es el **parámetro**: cada objeto del arreglo.
  - === es **igualdad estricta** (compara valor y tipo).
- console.log(...): imprime en la consola.

Notas

- Sensible a mayúsculas/minúsculas: "Ana" ≠ "ana".
- Complejidad: O(n) (recorre hasta encontrar).

3) filter: devuelve todas las coincidencias (posiblemente 0, 1 o muchas)

```
const sistemas = estudiantes.filter(est => est.carrera === "Ingeniería en Sistemas");
console.log("Estudiantes de Ingeniería en Sistemas:", sistemas);
```

- filter crea un **nuevo array** con **todos** los elementos que cumplan la condición.
- Si nadie cumple, retorna [] (array vacío).

4) some: ¿existe al menos uno?

```
const hayArquitectura = estudiantes.some(est => est.carrera === "Arquitectura");
console.log("¿Hay estudiante de Arquitectura?:", hayArquitectura);
```

- some devuelve un **booleano** (true/false).
- Se detiene en la **primera coincidencia**.

5) every: ¿todos cumplen?

```
const todosMayores = estudiantes.every(est => est.edad > 18);
console.log("¿Todos mayores de 18?:", todosMayores);
```

- every devuelve true si **todos** cumplen la condición. Si uno no, false.

6) map: transforma cada elemento y devuelve un nuevo array con los resultados
const nombres = estudiantes.map(est => est.nombre);
console.log("Nombres de los estudiantes:", nombres);

- map no filtra, **transforma**.
- Aquí convertimos cada objeto {…} en solo su nombre → ["Ana","Luis","María","Carlos","Elena"].

7) find por ID (patrón común)
const estudianteId3 = estudiantes.find(est => est.id === 3);
console.log("Estudiante con ID = 3:", estudianteId3);

- Igual que el primer find, pero usando una clave numérica (id).
- Si no existe, el resultado será undefined (¡manejar este caso al usarlo!).

8) filter por condición numérica
const mayores22 = estudiantes.filter(est => est.edad >= 22);
console.log("Estudiantes con edad >= 22:", mayores22);

- Operador >=: **mayor o igual**.
- Devuelve todas las coincidencias.

Conceptos clave (resumen rápido)

- **const vs let:**
  - const para referencias que **no reasignas** (buena práctica por defecto).
  - let si **sí** piensas reasignar.
- **Objetos y arreglos**:
  - Objeto = colección de pares **clave–valor**.
  - Arreglo = lista **ordenada**.
- **Funciones flecha**:
  param => expresión o param => { /* bloque */ }.
  Retorno **implícito** si usas una sola expresión sin llaves.
- **Inmutabilidad "suave"**:
  Métodos como map, filter y find **no** modifican el arreglo original; retornan **nuevas** estructuras.
- **Igualdad**:
  - ===: estricta (recomendado).
  - ==: realiza conversiones implícitas (evítalo).
- **Resultados por método**:
  - find → elemento o undefined.
  - filter → array (posiblemente vacío).
  - some → true / false.
  - every → true / false.
  - map → array transformado (misma longitud que el original).