UD3 - Resumen

1. Arrays

🔷 Creación de Arrays

Arrays vacíos:

```
let array1 = new Array();
let array2 = Array();
let array3 = [];
```

Arrays con elementos:

```
// Con constructor
let array1 = new Array(2);       // [empty × 2]
let array2 = new Array(3, 4);    // [3, 4]
let array3 = new Array("Luis");  // ["Luis"]

// Con corchetes
let array4 = [2];                // [2]
let array5 = [1, 3];             // [1, 3]
let array6 = ["Luis"];           // ["Luis"]
```

🔷 Modificación de Valores

```
let edades = [10, 20, 30];
edades[0] = 11;   // [11, 20, 30]
edades[2] = 998;  // [11, 20, 998]
```

🔷 Mostrar Arrays

```
// En navegador
document.write(edades[0]);
```

```
// En consola (Ctrl + Shift + i)
console.log(edades[1]);
```

⚠️ Elementos Vacíos

```
let procesadores = ["Intel", "AMD"];    // [0:"Intel", 1:"AMD"]
let procesadores2 = [,"Intel","AMD"];   // [0:empty, 1:"Intel", 2:"AMD"]
```

2. Arrays Bidimensionales

🔷 Creación

```
// Array bidimensional de 2 filas × 3 columnas
let tablaNotas = [[], []];
```

🔷 Asignación de Valores

```
tablaNotas[0][0] = 1;
tablaNotas[0][1] = 2;
tablaNotas[0][2] = 3;
tablaNotas[1][0] = 4;
tablaNotas[1][1] = 5;
tablaNotas[1][2] = 6;
```

🔷 Recorrido (Bucles Anidados)

```
for (let i = 0; i < tablaNotas.length; i++) {
    for (let j = 0; j < tablaNotas[i].length; j++) {
        console.log(`Posición [${i}][${j}]: ${tablaNotas[i][j]}`);
    }
}
```

3. Recorrido de Arrays

🔷 Propiedad `length`

```
let precios = [10, 20, 30, 40];
console.log(precios.length);  // 4
```

🔷 Bucle FOR

```
for (let i = 0; i < precios.length; i++) {
    console.log(`El precio ${i} es: ${precios[i]}`);
}
```

Característica: Imprime los valores vacíos

🔷 Bucle FOR IN

```
for (let elemento in precios) {
    console.log(`El precio ${elemento} es: ${precios[elemento]}`);
}
```

Ventajas:

- No necesita inicializar contador
- No necesita controlar tamaño del array
- No imprime valores vacíos

🔷 Bucle FOR OF

```
for (let precio of precios) {
    console.log(precio);
}
```

Ventajas:

- Simplificado, automático
  Desventajas:
- No se conocen los índices
- Imprime valores vacíos

4. Añadir Elementos a Arrays

🔷 Método 1: Por índice

```
let elementos = ["a", 7, true];
elementos[3] = 22.45;
// Resultado: ["a", 7, true, 22.45]
```

🔷 Método 2: `push()` (al final)

```
elementos.push("xyz");
// Resultado: ["a", 7, true, 22.45, "xyz"]
```

🔷 Método 3: `unshift()` (al principio)

```
elementos.unshift("primero");
// Resultado: ["primero", "a", 7, true, 22.45, "xyz"]
```

Nota: `unshift()` y `push()` permiten añadir varios valores a la vez.

5. Eliminación de Elementos

🔷 Método 1: `shift()` y `pop()`

```
let elementos = ["a", 7, true, 90.54, "Lucía", 12];

// shift() - elimina el primer elemento
let primero = elementos.shift();  // "a"

// pop() - elimina el último elemento
let ultimo = elementos.pop();     // 12

// Si no hay elementos, devuelve undefined
```

🔷 Método 2: Modificar longitud

```
elementos.length = 2;
// Resultado: ["a", 7] (elimina los demás)
```

🔷 Método 3: `splice()`

```
let elementos = ["a", 7, true, 90.54, "Lucía", 12];
let eliminados = elementos.splice(3, 2);

// elementos → ["a", 7, true, 12]
// eliminados → [90.54, "Lucía"]
```

Explicación: `splice(3, 2)` elimina 2 elementos empezando desde la posición 3.

6. Concatenación de Arrays

🔷 Método `concat()`

```
let array1 = [1, 2, 3];
let array2 = [4, 5, 6];
let nuevoArray = array1.concat(array2);

// nuevoArray → [1, 2, 3, 4, 5, 6]
// array1 y array2 permanecen sin cambios
```

7. Copia de Arrays

🔷 Método `slice()`

```
let elementos = ["a", 7, true, 90.54, "Lucía", 12];

// Copia completa
let copiaCompleta = elementos.slice();

// Copia parcial
let copiaParcial = elementos.slice(2, 4);  // [true, 90.54]
```

Parámetros:

- **Primero:** índice inicial (incluido)
- **Segundo:** índice final (no incluido)

8. Búsqueda en Arrays

🔷 indexOf()

```
let frutas = ["manzana", "naranja", "plátano", "manzana"];
let posicion = frutas.indexOf("naranja");  // 1
let noEncontrado = frutas.indexOf("uva");  // -1
```

🔷 lastIndexOf()

```
let posicion = frutas.lastIndexOf("manzana");  // 3 (última aparición)
```

🔷 includes()

```
let existe = frutas.includes("plátano");  // true
let noExiste = frutas.includes("uva");    // false
```

9. Método join()

🔷 Unir elementos en cadena

```
let frutas = ["manzana", "naranja", "plátano"];

let resultado1 = frutas.join();       // "manzana,naranja,plátano"
let resultado2 = frutas.join("-");    // "manzana-naranja-plátano"
let resultado3 = frutas.join(" | ");  // "manzana | naranja | plátano"
```

10. Ordenación de Arrays

🔷 sort()

```
let numeros = [40, 100, 1, 5, 25];
numeros.sort();  // [1, 100, 25, 40, 5] (orden Unicode)
```

```
let palabras = ["Zorro", "árbol", "Manzana"];
palabras.sort(); // ["Manzana", "Zorro", "árbol"] (mayúsculas primero)
```

🔷 reverse()

```
let numeros = [1, 2, 3, 4, 5];
numeros.reverse();  // [5, 4, 3, 2, 1]
```

---

11. Conjuntos (Sets)

🔷 Creación de Conjuntos

```
// Conjunto vacío
let conjunto = new Set();

// Conjunto con datos
let conjunto1 = new Set([34, "Girasol", 25.9]);
let conjunto2 = new Set("cadena");  // {"c", "a", "d", "e", "n", "a"}
```

🔷 Recorrido con FOR...OF

```
let conjunto = new Set(['primero', "segundo", "tercero", "primero"]);

for (let elemento of conjunto) {
    console.log(elemento);
}
// Resultado: "primero", "segundo", "tercero" (sin duplicados)
```

🔷 Añadir Elementos

```
let conjunto = new Set();
conjunto.add(1);
conjunto.add(2).add(3);  // Múltiples añadidos
```

🔷 Eliminar Elementos

```
// Eliminar elemento específico
conjunto.delete(2);

// Vaciar completamente
conjunto.clear();
```

🔷 Tamaño del Conjunto

```
let conjunto = new Set().add(1).add(1).add(2).add(9);
console.log(conjunto.size);  // 3 (los duplicados se eliminan)
```

🔷 Búsqueda en Conjuntos

```
let conjunto = new Set().add(1).add(2).add(9);

if (conjunto.has(9)) {
    console.log("Encontrado");
}
```

🔷 Conversión a Array

```
let conjunto = new Set().add(1).add(1).add(2).add(9);
const array = [...conjunto];  // [1, 2, 9]
```

🔷 Unión de Conjuntos

```
let array1 = [1, 2, 3];
let array2 = [3, 4, 5];
let conjunto = new Set([...array1, ...array2]);  // {1, 2, 3, 4, 5}
```

---

12. Mapas

🔷 Creación de Mapas

```
// Mapa vacío
let mapa = new Map();

// Mapa con datos iniciales
const frutas = new Map([
    ["A", "Manzana"],
    ["B", "Plátano"],
    ["C", "Naranja"],
    ["D", "Uva"]
]);
```

🔷 Recorrido de Mapas

```
let telefonos = new Map([
    [123456789, "Juan"],
    [987654321, "María"]
]);

// Recorrido completo
for (let persona of telefonos) {
    console.log(persona);  // [123456789, "Juan"], [987654321, "María"]
}

// Clave y valor por separado
for (let [telefono, persona] of telefonos) {
    console.log(`El teléfono de ${persona} es: ${telefono}`);
}

// Solo claves
for (let telefono of telefonos.keys()) {
    console.log(telefono);
}

// Solo valores
for (let persona of telefonos.values()) {
```

```
    console.log(persona);
}
```

🔷 Añadir y Eliminar Elementos

```
// Añadir elemento
telefonos.set(615885225, "Elena");

// Añadir múltiples (si clave existe, se sobreescribe)
telefonos.set(111111111, "Carlos").set(222222222, "Ana");

// Eliminar elemento
telefonos.delete(123456789);
```

🔷 Búsqueda en Mapas

```
let telefonos = new Map([[123456789, "Juan"]]);

// Verificar existencia
if (telefonos.has(123456789)) {
    console.log("Encontrado");
} else {
    console.log("No está");
}

// Obtener valor
console.log(telefonos.get(123456789));  // "Juan"
```

🔷 Conversión a Array

```
let telefonos = new Map([[123456789, "Juan"], [987654321, "María"]]);
let array = [...telefonos];  // [[123456789, "Juan"], [987654321, "María"]]
```

→ Resumen de Métodos

Arrays:

- **Añadir:** `push()`, `unshift()`
- **Eliminar:** `pop()`, `shift()`, `splice()`
- **Buscar:** `indexOf()`, `lastIndexOf()`, `includes()`
- **Transformar:** `concat()`, `slice()`, `join()`

Conjuntos:

- **Añadir:** `add()`
- **Eliminar:** `delete()`, `clear()`
- **Buscar:** `has()`
- **Información:** `size`

Mapas:

- **Añadir:** `set()`
- **Eliminar:** `delete()`
- **Buscar:** `has()`, `get()`
- **Recorrer:** `keys()`, `values()`