Apuntes sobre Arrays (Arreglos)

Un array es una estructura de datos que guarda una colección ordenada de elementos. Cada elemento puede ser de cualquier tipo de dato (números, strings, objetos, otros arrays) y se accede a ellos por un índice numérico que empieza en 0. Los arrays en JavaScript son dinámicos, su tamaño puede cambiar después de su creación.

1. Declaración y Tipos:

- **Literal de array:** `const miArray = [elemento1, elemento2, elemento3];`
- **Constructor Array():** `const miArray = new Array(elemento1, elemento2, elemento3);` o `const miArray = new Array(longitud);` (Si se da un solo número, crea un array de esa longitud, pero sin elementos inicializados).
- **Arrays tipados (TypedArrays):** Ofrecen control preciso sobre el tipo de dato, mejorando el rendimiento. Ej: `Int8Array`, `Uint16Array`, `Float32Array`.

2. Acceso a Elementos:

Se accede usando el índice entre corchetes:

```
let primerElemento = miArray[0];
let ultimoElemento = miArray[miArray.length - 1];
```

Acceder a un índice fuera de rango devuelve `undefined`.

3. Métodos Comunes:

- **`push(...)`:** Agrega elementos al final.
- **`pop()`:** Elimina y devuelve el último elemento.
- **`unshift(...)`:** Agrega elementos al principio.
- **`shift()`:** Elimina y devuelve el primer elemento.
- **`splice(inicio, cantidadAEliminar, ...elementos)`:** Modifica el array eliminando o reemplazando elementos.
- **`length`:** Propiedad, devuelve la cantidad de elementos.
- **`forEach(callback)`:** Itera sobre cada elemento.
- **`map(callback)`:** Crea un nuevo array aplicando una función a cada elemento.
- **`filter(callback)`:** Crea un nuevo array con los elementos que cumplen una condición.
- **`reduce(callback, valorInicial)`:** Reduce el array a un solo valor.
- **`concat(...)`:** Concatena arrays o valores.
- **`slice(inicio, fin)`:** Crea un nuevo array con una porción del original.
- **`indexOf(elemento, desde)`:** Devuelve el primer índice del elemento.
- **`includes(elemento, desde)`:** Devuelve `true` si el array contiene el elemento.
- **`reverse()`:** Invierte el orden de los elementos.

- **`sort(funcionDeComparacion)`:** Ordena los elementos.
- **`find(callback)`:** Devuelve el primer elemento que cumple una condición.
- **`findIndex(callback)`:** Devuelve el índice del primer elemento que cumple una condición.

4. Ejemplos:

```
const numeros = [1, 2, 3];
numeros.push(4); // numeros = [1, 2, 3, 4]

const frutas = ["manzana", "banana"];
frutas.unshift("uva"); // frutas = ['uva', 'manzana', 'banana']

const cuadrados = numeros.map((numero) => numero * numero); //
    cuadrados = [1, 4, 9, 16]
```

5. Iteración con bucles:

```
for (let i = 0; i < miArray.length; i++) {
  console.log(miArray[i]);
}

for (const elemento of miArray) {
  console.log(elemento);
}
```

6. Consideraciones de rendimiento:

`push()` y `pop()` son más eficientes que `shift()` y `unshift()`. Si necesitas insertar/eliminar al principio frecuentemente, considera usar otra estructura de datos (ej: lista doblemente enlazada).