Map y Set

- **Map** es un diccionario clave-valor donde **cualquier tipo** puede ser usado como clave
  - Es la mayor diferencia con los arrays asociativos, donde las claves solo pueden ser cadenas de texto
  - Con cualquier tipo nos referimos no sólo a cadenas, números… sino incluso objetos o funciones
- **Set** permite almacenar valores **únicos** de cualquier tipo, es decir no pueden estar duplicados

Objeto **Map**: Propiedades y métodos

- PROPIEDADES
- MÉTODOS

Size	Número de valores en el mapa

Map([conjunto])	Constructor. Acepta un conjunto de pares-valor
set(key,value)	Añade nueva pareja clave-valor
get(key)	Obtiene el valor asociado a una clave
delete(key)	Borra una pareja clave-valor mediante la clave
has(key)	Comprueba si hay determinada clave en el mapa
values()	Devuelve los valores del mapa
keys()	Devuelve las claves del mapa
entries()	Devuelve un conjunto de matrices [key,value]
clear()	Elimina todos los valores del mapa

Objeto **Map**: Ejemplo

```
let mapa = new Map();
mapa.set('1', 'str1'); // un string como clave
mapa.set(1, 'num1'); // un número como clave
mapa.set(true, 'bool1'); // un booleano como clave

// Map mantiene el tipo de dato en las claves, por lo
que estas dos son diferentes:
alert( mapa.get(1) ); // 'num1’
alert( mapa.get('1') ); // 'str1’
alert( mapa.size ); // 3
```

- Recorrido:

```
for(var [clave, valor] of mapa) {
    console.log(clave + " = " +valor);
}
```

```
1 = str1
1 = num1
true = bool1
```

Objeto `Set`: Propiedades y métodos

- PROPIEDADES

`size`	Número de valores en el mapa

- MÉTODOS

`add(element)`	Añade un nuevo valor
`delete(element)`	Borra un valor
`has(element)`	Comprueba si hay un elemento en el conjunto
`values()`	Devuelve un objeto iterable con cada uno de los valores del conjunto.
`clear()`	Elimina todos los valores del conjunto

RETO EXTRA
Evitar duplicados

- Dados los siguientes arrays:

```
let array = [100, 23, 23, 23, 23, 67, 45];
let outputArray = [];
```

- Haz que en outputArray estén los mismos números, pero evitando duplicados. Al final del proceso, en este ejemplo, el array contendrá:

```
outputArray = [100, 23, 67, 45]
```

- Será necesario hacerlo de dos formas:
  - Una "manual" usando bucles para recorrer el array original, viendo si cada número ya lo tenemos en el de salida o no.
  - Otra buscando en internet información sobre el método from de Array y el objeto Set.