

Creating a Set

You can create a set using the `Set` constructor:

```
let mySet = new Set();
```

You can also initialize a set with values by passing an iterable (like an array):

```
let mySet = new Set([1, 2, 3, 4, 5]);  
console.log(mySet); // Output: Set { 1, 2, 3, 4, 5 }
```

Set Methods and Properties

1. `add()`

Adds a new element to the set.

```
mySet.add(6);  
console.log(mySet); // Output: Set { 1, 2, 3, 4, 5, 6 }
```

2. `has()`

Checks if a value is in the set.

```
console.log(mySet.has(3)); // Output: true  
console.log(mySet.has(10)); // Output: false
```

3. `delete()`

Removes a specified value from the set.

```
mySet.delete(4);  
console.log(mySet); // Output: Set { 1, 2, 3, 5, 6 }
```

4. `clear()`

Removes all elements from the set.

```
mySet.clear();
```

```
console.log(mySet); // Output: Set {}
```

5. size

Returns the number of elements in the set.

```
let mySet = new Set([1, 2, 3]);  
console.log(mySet.size); // Output: 3
```

Iterating Over a Set

Sets are iterable, meaning you can loop through the elements:

Using **for...of** loop:

```
for (let item of mySet) {  
    console.log(item);  
}  
// Output:  
// 1  
// 2  
// 3
```

Removing Duplicates:

```
let setA = new Set([1, 2, 3,4,2,4,5,7]);  
console.log(setA);
```

Two Sum:

Given an array of integers and a target sum, find if there exist two distinct numbers in the array that add up to the target. Return **true** if such a pair exists, otherwise return **false**.

Input: nums = [2, 7, 11, 15], target = 9

Output: true

Explanation: 2 + 7 = 9, so a valid pair exists.

```
function twoSum(nums, target) {  
    let seen = new Set();  
  
    for (var i=0;i<nums.length;i++) {  
        var b=nums[i];
```

```

    let a= target - b;

    // Check if the complement exists in the set
    if (seen.has(a)) {
        return true; // Pair found
    }

    // Add the current number to the set
    seen.add(nums[i]);
}

return false; // No pair found
}

// Example usage
let nums = [2, 7, 11, 15];
let target = 9;

console.log(twoSum(nums, target)); // Output: true

```

Union:

```

let setA = new Set([1, 2, 3]);
let setB = new Set([3, 4, 5]);
let union = [];

for(let value of setA){
    setA.add(value);
    union.push(value);
}
for(let value of setB){
    if(!setA.has(value)){
        union.push(value);
    }
}

console.log(union); // Output: Set { 1, 2, 3, 4, 5 }

```

Intersection:

```
let setA = new Set([1, 2, 3]);
let setB = new Set([3, 4, 5]);
let intersection = [];
for(let value of setA){
    if(setB.has(value)){
        intersection.push(value);
    }
}

console.log(intersection);
```

HashMaps:

Creating a Map

You can create a **Map** using the **Map** constructor:

```
let myMap = new Map();
```

You can also initialize a **Map** with key-value pairs:

```
let myMap = new Map([
    ['key1', 'value1'],
    ['key2', 'value2'],
]);
```

Map Methods and Properties

set(key, value) Adds or updates an element in the **Map** with the specified key and value.

```
myMap.set('key3', 'value3');
console.log(myMap); // Output: Map(3) { 'key1' => 'value1', 'key2' => 'value2', 'key3' => 'value3' }
```

1. **get(key)** Retrieves the value associated with the given key.

```
console.log(myMap.get('key1')); // Output: 'value1'
```

```
console.log(myMap.get('key4')); // Output: undefined (if the key doesn't exist)
```

2. **has(key)** Returns `true` if the `Map` contains the specified key, otherwise `false`.

```
console.log(myMap.has('key2')); // Output: true
```

```
console.log(myMap.has('key4')); // Output: false
```

3. **delete(key)** Removes the element with the specified key from the `Map`.

```
myMap.delete('key2');
```

```
console.log(myMap); // Output: Map(2) { 'key1' => 'value1', 'key3' => 'value3' }
```

- 4.

clear() Removes all elements from the `Map`.

```
myMap.clear();
```

```
console.log(myMap); // Output: Map(0) {}
```

- 5.

size Returns the number of key-value pairs in the `Map`.

```
let myMap = new Map([[ 'a', 1 ], [ 'b', 2 ]]);
```

6. `console.log(myMap.size);` // Output: 2

Iterating over maps:

```
myMap.forEach((value, key) => {  
    console.log(key, value);  
});
```

```
// Output:
```

```
// key1 value1
```

```
// key3 value3
```

Calculating Frequencies of Each element in array:

```
function countFrequencies(arr) {
```

```
let frequencyMap = new Map();

for (let num of arr) {
    if (frequencyMap.has(num)) {
        frequencyMap.set(num, frequencyMap.get(num) + 1);
    } else {
        frequencyMap.set(num, 1);
    }
}

return frequencyMap;
}
```

// Example usage

```
let arr = [1, 2, 2, 3, 3, 3, 4];
let freqMap = countFrequencies(arr);
```

```
freqMap.forEach((value, key) => {
    console.log(key, '=>', value);
});
```

// Output:

```
// 1 => 1
```

```
// 2 => 2
```

```
// 3 => 3
```

```
// 4 => 1
```

Homework: Majority Element