2023

# JavaScript Map.

Ibrahim Here! 💪

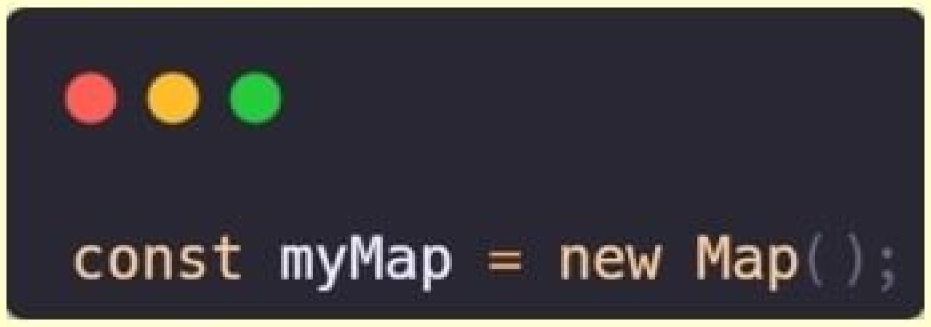
## What is a Map?

JavaScript provides a built-in data structure called a ***Map*** that allows us to store key-value pairs where both the key and the value can be of any type, including objects, functions, and primitives. In this explanation, we will discuss ***how to create a Map, how to add and remove elements from a Map, how to iterate over a Map and Map vs Object.***

## Creating a Map:

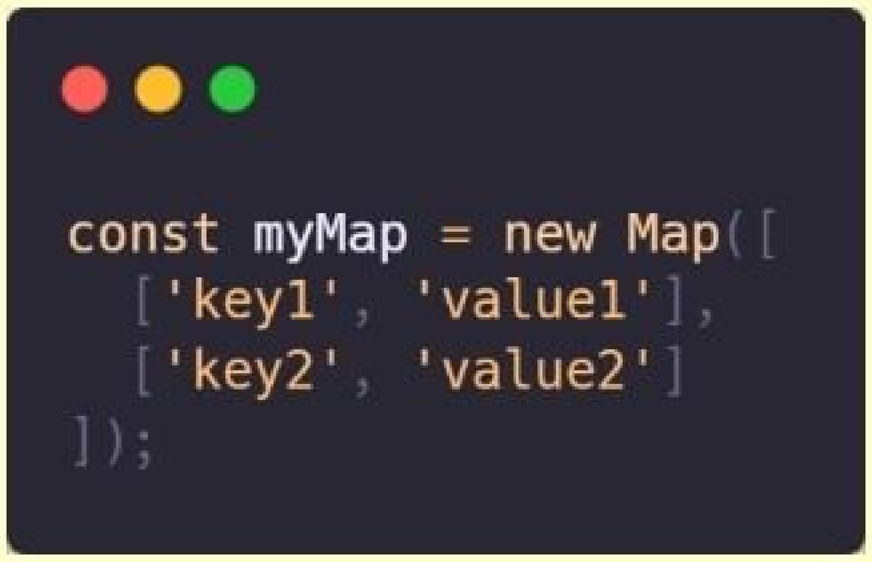
To create a ***Map***, you can use the ***Map() constructor function***.

### Example:



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

This creates an empty ***Map*** object. You can also initialize a ***Map*** with some key-value pairs by passing an array of arrays, where each sub-array contains a key-value pair:



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

In this example, we have created a ***Map*** with two key-value pairs.

## Adding and Removing Elements from a Map:

To add a new key-value pair to a ***Map***, you can use the ***set() method***.

**Example:**

```
myMap.set('key3', 'value3');
```

This will add a new key-value pair to the ***Map***. If the key already exists, the value will be updated.

To remove a key-value pair from a ***Map***, you can use the ***delete() method***.

**Example:**

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

This will remove the key-value pair with the key ***'key2'*** from the ***Map***.

## Accessing Elements in a Map:

To get the value associated with a key in a ***Map***, you can use the ***get() method***.

**Example:**

```
const value1 = myMap.get('key1');
```

This will return the value associated with the key ***'key1'***. If the key does not exist in the ***Map***, it will return undefined.

You can also check if a key exists in a ***Map*** using the ***has() method***.

**Example:**

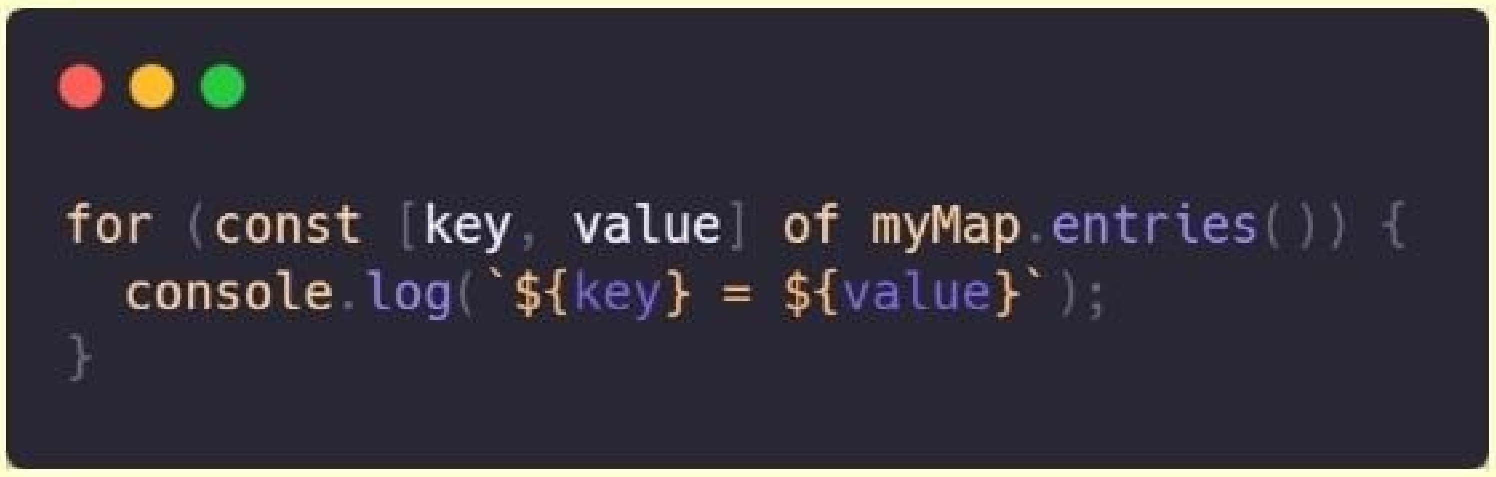
```
const hasKey = myMap.has('key1');
```

This will return true if the key ***'key1'*** exists in the ***Map***, and false otherwise.

## Iterating over a Map:

You can iterate over a ***Map*** using the ***entries(), keys()***, or ***values() methods***. The entries() method returns an iterator that contains an array of ***[key, value]*** pairs for each element in the ***Map***. The ***keys() method*** returns an iterator that contains the keys of each element in the ***Map***. The ***values() method*** returns an iterator that contains the values of each element in the ***Map***.

### Example:



```
for (const [key, value] of myMap.entries()) {
  console.log(`${key} = ${value}`);
}
```

This will log each key-value pair in the ***Map*** to the ***console***.

## JavaScript Map vs Object:

| Map | Object |
|---|---|
| Maps can contain objects and other data types as keys. | Objects can only contain strings and symbols as keys. |
| Maps can be directly iterated and their value can be accessed. | Objects can be iterated by accessing its keys. |
| The number of elements of a Map can be determined by size property. | The number of elements of an object needs to be determined manually. |
| Map performs better for programs that require the addition or removal of elements frequently. | Object does not perform well if the program requires the addition or removal of elements frequently. |

## Conclusion:

In summary, a ***Map*** is a data structure in JavaScript that stores ***key-value pairs*** where both the key and the value can be of any type. We can create a ***Map*** using the ***Map constructor, add and remove elements using the set() and delete() methods, and iterate over the elements using the forEach() method***. The ***Map*** data structure is a useful tool for ***storing*** and ***manipulating*** data in JavaScript.

# Did you like this post?

⭐ ❤️ 💪

If you're new to my profile.
I post content on coding
in general. Visit my profile.

Save for Later.