JavaScript Sets | JavaScript Tutorial In Hindi #58

Overview Q&A Downloads Announcements

JavaScript Sets | JavaScript Tutorial In Hindi #58

In today’s tutorial, we will discuss another advance topic of JavaScipt. With the introduction of ECMAScript 6, came the introduction of a new built-in JavaScript object type: the Set. So, let's start learning about this new concept.

What is set in JavaScript?

A set is a collection of unique items. In set, no element can be repeated. The set in ES6 are ordered, elements of the set can be iterated in the insertion order. The power of the set is that it can store any types of values, whether primitive or objects.

Syntax:

```
let myset=new Set([iterable]);
```

In the syntax, iterable is an object whose all elements are added to the new set created. In set, the parameter is optional. If the parameter is not specified or null is passed then a new set created is empty. It returns a new set object.

Methods of the set:-

add(): This method adds the new element with a specified *value* at the end of the Set object. Here is the syntax:

```
set1.add(x);
```

Here “x” is a value to be added to the set. And its returns set object.

delete(): This method deletes an element with the specified *value* from the Set object. Here is the syntax:

```
set1.delete(x);
```

Here “x” is a value to be deleted from the set. This method returns true if the value is successfully deleted from the set else returns false.
clear(): This method removes all the element from the set. Here is the syntax:

```
set1.clear();
```

In this method, no parameter is needed, and it returns undefined.
entries(): This method returns an iterator object which contains an array having the entries of the set, in the insertion order. Here is the syntax:

```
set1.entries();
```

This method does not need any parameter. It returns an iterator object that contains an array of [value, value] for every element of the set, in the insertion order.

has(): This method returns true if the specified *value* is present in the Set object. Here is the syntax:

```
set1.has(x);
```

Here “x” is the value to be searched in the Set. This method returns true if the value is present else it returns false.
values(): This method returns all the values from the Set in the same insertion order. Here is the syntax:

```
set1.values();
```

No parameters are needed in this method. An iterator object is returned that contains all the values of the set in the same order as they are inserted.
keys() – This method returns all the values from the Set in the insertion order. keys() is similar to the *values()* in case of Sets. Here is the syntax:

```
set1.keys();
```

No parameters are needed in this method. An iterator object returned that contains all the values of the set in the same order as they are inserted.

Example:-

Suppose have students coming, and we like to remember everyone. But repeated students should not lead to duplicates. A student must be “counted” only once. For this, we will use set.

```
let set = new Set();
let john = { name: "John" };
let harry = { name: "harry" };
let cavin = { name: "cavin" };
```

```
//  some users come multiple times
set.add(john);
set.add(harry);
set.add(cavin);
set.add(john);
set.add(harry);
// set keeps only unique values
for (let user of set) {
console.log(user.name)}
```

Code file tut58.js as described in the video

```
console.log('This is tutorial 58');

// Set stores unique values
const mySet = new Set();  // Initialize an empty set
console.log('The set looks like :',mySet);

// Adding values to this set
mySet.add('this');
mySet.add('My name');
mySet.add('this');
mySet.add('that');
mySet.add(34);
mySet.add(true);
mySet.add(false);
mySet.add('that2');
console.log('The set looks like this now:',mySet);

// Use a constructor to initialize the set
let mySet2 = new Set([1, 45, 'this', false, {a:4, b:8}, 'this']);
console.log('New set:',mySet2);


console.log(mySet.size); // Get the size of the set

console.log(mySet.has(346)); // Check whether set has 346 or not
console.log('Before removal', mySet.has('that2'));
mySet.delete('that2'); // Remove an element from the set
console.log('After removal', mySet.has('that2'));


// Iterating a set
// for(let item of mySet){
//     console.log('Item is  :', item );
// }

mySet.forEach((item)=>{
    console.log('Item is  :', item );
})

// Quiz: Can you use Array.from(mySet) to convert set into an array?
```

Previous Next