JavaScript TUTORIAL

JavaScript Array.from()

Summary: in this tutorial, you will learn about the JavaScript `Array.from()` method that creates a new array from an array-like or iterable object.

Introduction to JavaScript Array.from() method

To create an array from an array-like object in ES5, you iterate over all array elements and add each of them to an intermediate array like this:

```
function arrayFromArgs() {
  var results = [];
  for (var i = 0; i < arguments.length; i++) {
    results.push(arguments[i]);
  }
  return results;
}

var fruits = arrayFromArgs('Apple', 'Orange', 'Banana');
console.log(fruits);
```

Output:

```
[ 'Apple', 'Orange', 'Banana' ]
```

To make it more concise, you can use the `slice()` method of the `Array.prototype` as follows:

```
function arrayFromArgs() {
  return Array.prototype.slice.call(arguments);
}
```

```
var fruits = arrayFromArgs('Apple', 'Orange', 'Banana');
console.log(fruits);
```

ES6 introduces the `Array.from()` method that creates a new instance of the `Array` from an array-like or iterable object. The following illustrates the syntax of the `Array.from()` method:

```
Array.from(target [, mapFn[, thisArg]])
```

In this syntax:

- `target` is an array-like or iterable object you want to convert to an array.
- `mapFn` is the map function to call on every element of the array.
- `thisArg` is the `this` value inside the `mapFn` function.

The `Array.from()` returns a new instance of `Array` that contains all elements of the `target` object.

JavaScript Array.from() method examples

Let's take some examples of using the `Array.from()` method.

1) Creating an array from an array-like object

The following example uses the `Array.from()` method to create a new array from the `arguments` object of a function:

```
function arrayFromArgs() {
    return Array.from(arguments);
}

console.log(arrayFromArgs(1, 'A'));
```

Output:

```
[ 1, 'A' ]
```

In this example, we create an array from the arguments of the `arrayFromArgs()` function and return it.

2) Using JavaScript Array.from() method with a mapping function

The `Array.from()` method accepts a callback function that allows you to execute the mapping function on every element of the array that is being created. For example:

```
function addOne() {
    return Array.from(arguments, x => x + 1);
}
console.log(addOne(1, 2, 3));
```

Output:

```
[ 2, 3, 4 ]
```

In this example, we increased each argument of the `addOne()` function by one and add the result to the new array.

3) Using Array.from() method with the this value

If the mapping function belongs to an object, you can optionally pass the third argument to the `Array.from()` method. You can reference the object inside the function as the `this` value. For example:

```
let doubler = {
  factor: 2,
  double(x) {
    return x * this.factor;
  },
};
let scores = [5, 6, 7];
let newScores = Array.from(scores, doubler.double, doubler);
console.log(newScores);
```

Output:

```
[ 10, 12, 14 ]
```

4) Creating an array from an iterable object

Since the `Array.from()` method also works on an iterable object, you can use it to create an array from any object that has a `[symbol.iterator]` property. For example:

```
let even = {
  *[Symbol.iterator]() {
    for (let i = 0; i < 10; i += 2) {
      yield i;
    }
  },
};
let evenNumbers = Array.from(even);
console.log(evenNumbers);
```

Output:

```
[0, 2, 4, 6, 8]
```

In this example:

- First, define the `even` object with the `[System.iterator]` that returns even numbers from 0 to 10.
- Then, use the `Array.from()` method to create a new array of even numbers from the `even` object.

Summary

- Use the JavaScript `Array.from()` method to create an array from an array-like or iterable object.