JavaScript TUTORIAL

Array.prototype.toReversed()

Summary: in this tutorial, you'll learn how to use the JavaScript Array `toReversed()` method to reverse the order of elements in an array and return a new array with the elements in reversed order.

Introduction to the JavaScript Array toReversed() method

The `toReversed()` method reverses the order of elements in an array and returns a **new array** with the elements in reversed order.

Unlike the `reversed()` method that reverses the elements of the array in place, the `toReversed()` method does not modify the original array. Instead, it creates a new array with the elements in the original array in the reversed order.

Here's the basic syntax of the `toReversed()` method:

```
Array.prototype.toReversed()
```

The `toReversed()` method takes no parameters and returns a new array containing the elements in reversed order.

When you call the `toReversed()` method on a sparse array, it treats empty slots as if they have the value `undefined`.

This method is **generic**, meaning that you can call it on a non-array object that has a `length` property and integer-keyed properties.

JavaScript Array toReversed() method examples

Let's explore some examples of using the JavaScript array `toReverse()` method.

1) Using Array toReversed() method on string arrays

The following example uses the `toReverse()` method to reverse an array of strings:

```
const colors = ['red','green','blue'];
const reversedColors = colors.toReversed();

console.log(colors);
console.log(reversedColors);
```

Output:

```
['red','green','blue']
['blue', 'green', 'red']
```

2) Reversing arrays of numbers

The following example uses the `toReversed()` method to reverse the order of numbers in an array:

```
const scores = [1, 3, 5, 7];
const reversedScores = scores.toReversed();

console.log(scores);
console.log(reversedScores);
```

Output:

```
[1, 3, 5, 7]
[7, 5, 3, 1]
```

3) Reversing arrays of objects

The following example uses the `toReversed()` method to reverse the order of objects in an array:

```
const contacts = [{name: 'John'}, {name: 'Alice'}, {name: 'Bob'}];
const reversedContacts = contacts.toReversed();
```

```
console.log(contacts);
console.log(reversedContacts);
```

Output:

```
[{name: 'John'}, {name: 'Alice'}, {name: 'Bob'}]
[{name: 'Bob'}, {name: 'Alice'}, {name: 'John'}]
```

4) Reversing sparse arrays

When you call the `toReversed()` method on a sparse array, the result array remains sparse. The `toReversed()` method copies empty slots over their respective indices as empty slots:

```
const scores = [1,,7,5];
const reversedScores = scores.toReversed();

console.log(scores);
console.log(reversedScores);
```

Output:

```
[1,, 7, 5]
[5, 7, undefined, 1]
```

5) Using toReversed() method with non-array objects

The following example shows how call the `toReversed()` method on an object that has the `length` property and integer-keyed properties:

```
const arrayLike = {
  length: 3,
  unrelated: "bar",
  2: 2,
  3: 3, // ignored because the length is 3
};

const result = Array.prototype.reverse.call(arrayLike);
```

```
console.log(result);
```

Output:

```
{0: 2, 3: 3, length: 3, unrelated: 'bar'}
```

In this example, the `toReversed()` method does the following:

- First, access the `length` property of the object.
- Second, iterate through each property with an integer key ranging from `0` to `length / 2`.
- Third, swap the values at corresponding indices on both ends. Additionally, remove any destination property without a corresponding source property.
- Finally, return a new object with the elements (or properties) in reversed order.

Summary

- Use the JavaScript array `toReversed()` method to reverse the order of elements within an array and return a new reversed array.