Array.prototype.every()

Summary: in this tutorial, you will learn how to use the JavaScript Array `every()` method to check whether all elements in an array pass a test.

Introduction to JavaScript Array every() method

The `every()` method that allows you to check if every element of an array passes a test provided by a function.

Here's the syntax of the `every()` method:

```
array.every(callbackFn, thisArg)
```

The `every()` method accepts two named arguments: `callbackFn` and `thisArg` .

The `callbackFn` is a function that tests each element of the array. It has has the following form:

```
function callback(currentElement, index, array){
   //...
}
```

The `callback()` function takes three arguments:

- `currentElement` is the current element in the array being processed.
- `index` is the index of the `currentElement` .
- `array` is the array that calls the `every()` method.

The `currentElement` is required whereas the `index` and `array` are optional.

The `callbackFn` returns a *truthy* value indicating that the element passes the test, and a *falsy* value otherwise.

The `thisArg` is an optional argument that can used as `this` value inside the `callbackFn`.

The `every()` method executes the `callback()` function on every element in the array until it finds the one that causes the `callback()` return a falsy value.

In other words, the `every()` will stop calling the `callback()` function and return `false` if it encounters an element that causes `callback()` to return a falsy value.

JavaScript Array every method examples

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

Basic Array every() method examples

The following example uses the `every()` to check if every element of the numbers array is greater than zero:

```
const numbers = [1, 3, 5];
const result = numbers.every((n) => n > 0);

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

Output:

```
{ result: true }
```

Since all the numbers in the `numbers` array are greater than zero, the `every()` method returns `true`.

The following example uses the `every()` method to test if all the numbers in the `numbers` array elements are even:

```
let numbers = [2, 4, 5];
let result = numbers.every((n) => n % 2 == 0);

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

Output:

```
{ result: false }
```

The numbers array has an odd number (`5`), so the `every()` method returns `false` .

Using the thisArg argument

The following example tests whether all elements in the `numbers` array is in the range specified by the `min` and `max` of the `range` object.

```
const numbers = [2, 4, 7];

const range = {
  min: 0,
  max: 10,
};

const isInRange = (n) => n >= this.min && n <= this.max;

const result = numbers.every(isInRange, range);

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

Output:

```
{ result: true }
```

How it works.

First, define an array of numbers:

```
const numbers = [2, 4, 7];
```

Next, define an object `range` that has two properties `min` and `max` :

```
const range = {
  min: 0,
  max: 10,
};
```

Then, define the `isInRange` method that checks if a number is in the range of `min` or `max`:

```
const isInRange = (n) => n >= this.min && n <= this.max;
```

After that, execute the `isInRange` method on every element of the `numbers` array using the `every()` method:

```
const result = numbers.every(isInRange, range);
```

We pass the `range` object to the `every()` method as the second argument. Inside the `isInRage()` function, we reference the `range` object using the `this` keyword.

Finally, display the result to the console:

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

Working with empty arrays

If you call the `every()` method on an empty array, the method will always return `true` for any condition. For example:

```
const gtZero = [].every((n) => n > 0);
console.log({ gtZero });

const ltZero = [].every((n) => n < 0);
console.log({ ltZero });
```

Output:

```
gtZero: true
ltZero: true
```

Summary

- Use the Array `every()` method to test whether all elements in an array pass the test provided by a function.