Array.prototype.forEach()

Summary: in this tutorial, you will learn how to use the JavaScript Array `forEach()` method to execute a function on every element in an array.

Introduction to JavaScript Array forEach() method

Typically, when you want to execute a function on every element of an array, you use a for loop statement.

For example, the following code shows every element of an array to the console:

```
let ranks = ['A', 'B', 'C'];
for (let i = 0; i < ranks.length; i++) {
  console.log(ranks[i]);
}
```

Output:

```
A
B
C
```

JavaScript `Array` provides the `forEach()` method that allows you to run a function on every element.

The following code uses the `forEach()` method that is equivalent to the code above:

```
let ranks = ['A', 'B', 'C'];

ranks.forEach(function (e) {
```

```
  console.log(e);
});
```

Output:

```
A
B
C
```

The `forEach()` method iterates over elements in an array and executes a predefined function once per element.

The following illustrates the syntax of the `forEach()` method.

```
Array.forEach(callbackFn [, thisArg]);
```

The `forEach()` method takes two arguments:

1) callbackFn

The `forEach()` method executes the `callbackFn` function on every element. The `callbackFn` function accepts the following arguments:

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

The `index` and `array` are optional.

2) thisArg

The `thisArg` is a value to use as `this` inside the `callbackFn`

One limitation of the `forEach()` method compared to the `for` loop is that you cannot use the break or continue statement to control the loop.

To terminate the loop in the `forEach()` method, you must throw an exception inside the `callback` function.

> Note that the `forEach()` function returns `undefined` therefore it is not chainable like other iterative array methods: `filter()`, `map()`, `some()`, `every()`, and `sort()`.

JavaScript Array forEach() method examples

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

Basic JavaScript Array forEach() method example

The following example uses the `forEach()` method to log each number in an array to the console:

```
const numbers = [1, 2, 3];

numbers.forEach((n) => {
  console.log(n);
});
```

Output:

```
1
2
3
```

Modifying array elements

To modify array elements while iterating, you can use the second and third arguments of the callback function.

For example, the following iterate over the elements of the `numbers` array and double each element:

```
const numbers = [1, 2, 3];

numbers.forEach((n, index, array) => {
  array[index] = n * 2;
});

console.log(numbers);
```

Output:

```
[ 2, 4, 6 ]
```

Using the thisArg argument example

The following example shows how to use the `forEach()` method with the `thisArg` argument:

```
class Counter {
  constructor() {
    this.count = 0;
  }
  increase() {
    this.count++;
  }
  current() {
    return this.count;
  }
  reset() {
    this.count = 0;
  }
}

const counter = new Counter();
const numbers = [1, 2, 3];

let sum = 0;
numbers.forEach(function (n) {
  sum += n;
  this.increase();
```

```
}, counter);

console.log({ sum });
console.log({ counter: counter.current() });
```

Output:

```
{ sum: 6 }
{ counter: 3 }
```

How it works.

First, define a `Counter` class:

```
class Counter {
  constructor() {
    this.count = 0;
  }
  increase() {
    this.count++;
  }
  current() {
    return this.count;
  }
  reset() {
    this.count = 0;
  }
}
```

Next, create a new `Counter` object:

```
const counter = new Counter();
```

Then, define an array of three numbers:

```
const numbers = [1, 2, 3];
```

After that, call the `forEach()` method on the `numbers` array:

```
let sum = 0;
numbers.forEach(function (n) {
  sum += n;
  this.increase();
}, counter);
```

In the callback function, add the element to the `sum` variable and call the `increase()` method of the `counter` object.

Notice that the `counter` object is referred to as `this` inside the callback function.

Finally, log the value of the `sum` and the current counter in the console:

```
console.log({ sum });
console.log({ counter: counter.current() });
```

Summary

- Use the JavaScript Array `forEach()` method to execute a callback on every element of an array.