JS JavaScript
TUTORIAL

Array.prototype.reduce()

Summary: in this tutorial, you will learn how to use the JavaScript Array `reduce()` method to reduce an array to a value.

Introduction to the JavaScript Array reduce() method

The Array `reduce()` method iterate over an array and reduce their elements to a single value.

The `reduce()` method executes a function for each value in the array from left to right and store the return value of the function in a accumulator.

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

```
array.reduce(callbackFn,initialValue)
```

The `reduce()` method takes two arguments:

- `callbackFn` is a function to execute on each element of the `array`. This function is often referred to as a *reducer* function.
- `initialValue` is the value to use a the first argument to the first call of the `callbackFn`. The `initialValue` is optional. If you omit it, the `reduce()` method will use the first element of the `array` as the initial accumulator (more on accumulator this shortly).

The `callbackFn` function has the following syntax:

```
function callbackFn(accumulator, currentValue, index, array)
```

The `callbackFn` function takes four arguments:

- `accumulator` is the accumulated value previously returned in the last call of the `callbackFn`, or `initialValue`, if provided.

- `currentValue` is the current element being processed.
- `index` (optional) is the index of the current element being processed in the array.
- `array` (optional) is the array that call the `reduce()` method.

The following table illustrates the logic when the `reduce()` method executes the `callbackFn` function for the first time according to the `initialValue` argument:

`initialValue`	**`accumulator`**	**`currentValue`**
Provided	`initialValue`	`array[0]`
Not Provided	`array[0]`	`array[1]`

JavaScript Array reduce() method examples

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

1) Basic Array reduce() method example

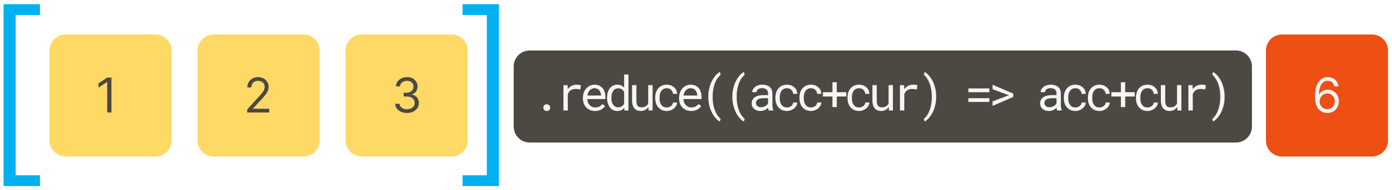
The following example uses the `reduce()` method to calculate the total of elements in the `numbers` array:

```
const numbers = [1, 2, 3];
const total = numbers.reduce(
  (accumulator, currentValue) => accumulator + currentValue
);

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

Output:

```
{ total: 6 }
```



How it works.

First, define a array that stores three numbers 1, 2, and 3:

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

Second, call the `reduce()` method of the `numbers` array to calculate the total of numbers in the array:

```
const total = numbers.reduce(
  (accumulator, currentValue) => accumulator + currentValue
);
```

Third, display the total in the console:

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

2) Using reduce() method with an initial value

Suppose that you have the following `shoppingCart` array of product objects:

```
const cart = [
  {
    product: 'phone',
    qty: 1,
    price: 500,
  },
  {
    product: 'Screen Protector',
    qty: 1,
    price: 10,
  },
  {
    product: 'Memory Card',
    qty: 2,
    price: 20,
  },
];

const reducer = (acc, item) => acc + item.qty * item.price;
const total = cart.reduce(reducer, 0);

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

Output:

```
{ total: 550 }
```

How it works.

First, define a cart that stores three items:

```
const cart = [
  {
    product: 'phone',
    qty: 1,
    price: 500,
  },
  {
    product: 'Screen Protector',
    qty: 1,
    price: 10,
  },
  {
    product: 'Memory Card',
    qty: 2,
    price: 20,
  },
];
```

Each item has three properties name, qty, and price.

Second, define a `reducer()` function that calculate the total amount of the items in the cart:

```
const reducer = (acc, item) => acc + item.qty * item.price;
```

For each item, we calculate qty * price and add its result to the accumulator.

Third, calculate the total of items by calling the reduce() method with the `reducer()` function and initial value of zero:

```
const total = cart.reduce(reducer, 0);
```

Note that if we don’t pass 0 as an initial value, the reduce() method will take the first element of the cart array as the initial value.

But the first elemnent is an object, we cannot add it to the accumulator which is a number. Hence, it will return unexpected value.

Finally, display the result to the console:

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

Summary

- Use the JavaScript Array `reduce()` method to reduce an array to a value.