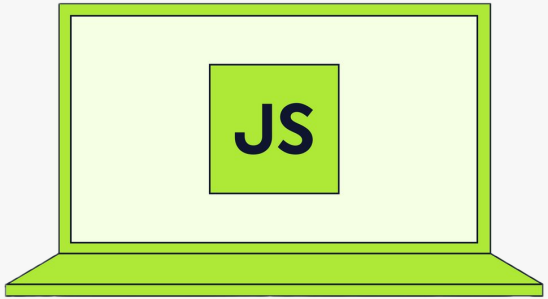
Newton School of Technology

The Complete Javascript Course



@newtonschool

Lecture 6: Array and Array Methods

-Vishal Sharma

JS

Table of Contents

Newton School of Technology

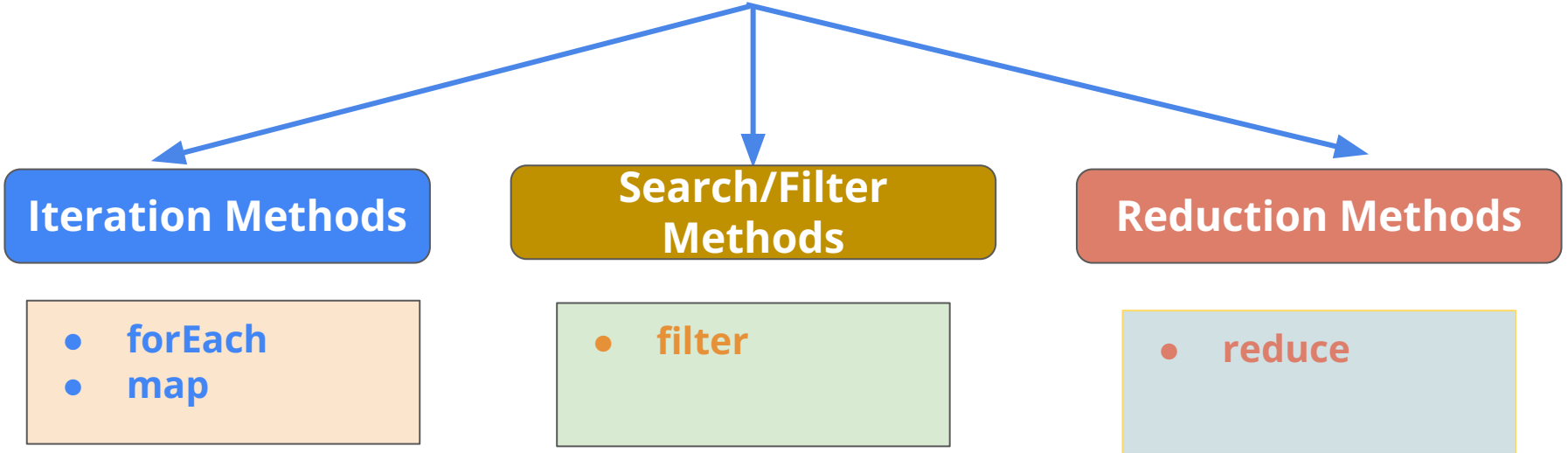
Newton School of Technology

Array Traversal and Manipulation Methods

What are these methods??

Newton School of Technology

We need these advanced array methods because they provide a **concise, readable, and efficient way** to work with arrays. We can broadly classify them as:



Pre-discussed: Loops in real life

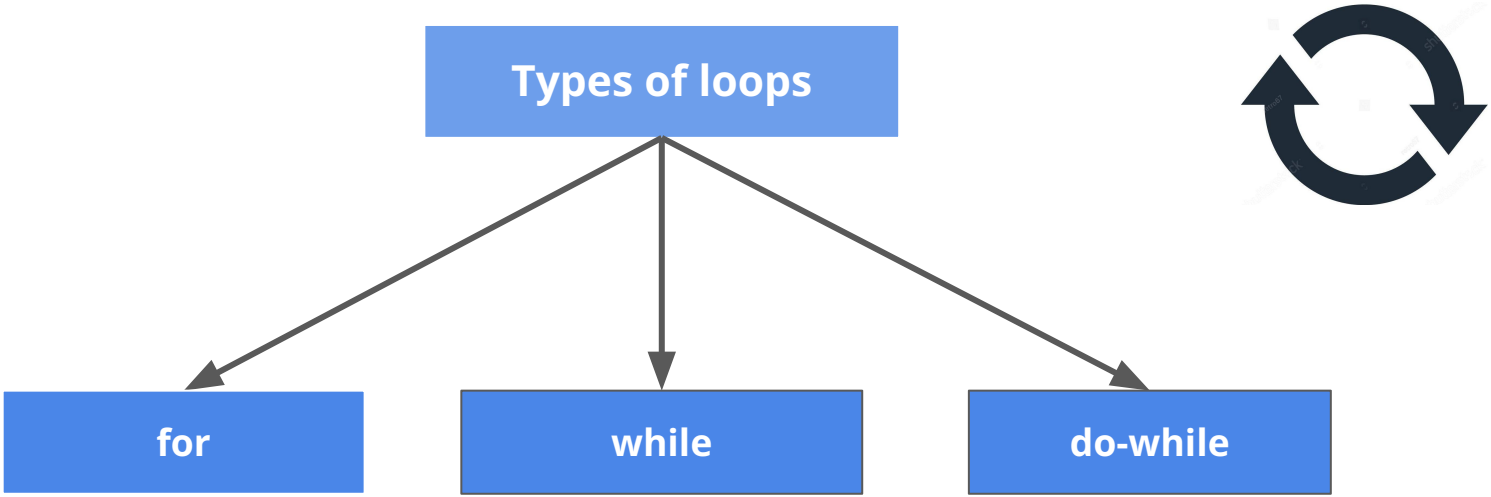
Newton School of Technology

Every day we follow certain routine which we repeat all over the week, like brushing your teeth, taking shower, etc.

Pre-discussed: Loops in Programming

Newton School of Technology

Just like we repeat several tasks in daily like, we need some tasks to be repeated in programming and we repeat those tasks with the help of loops.



Newton School of Technology

Why do we need newer methods?

Need for newer methods

Newton School of Technology

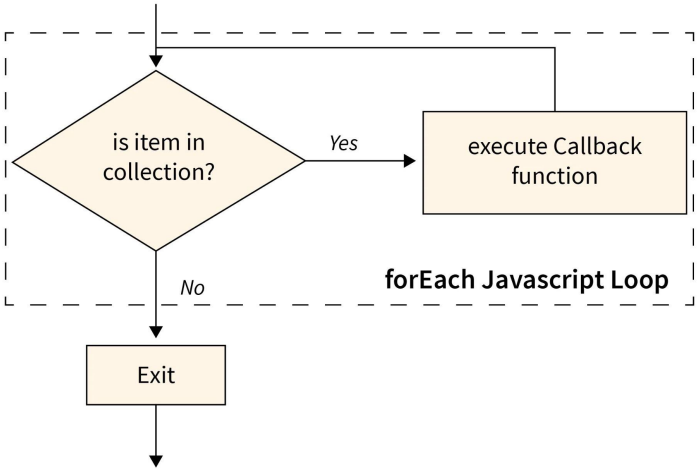
We need newer iteration methods to simplify code, enhance readability, and enable efficient operations like forEach, mapping, filtering and reducing arrays.

Let’s talk about forEach and map methods

Iteration Methods: forEach

Newton School of Technology

The forEach() method executes a provided function once for each array element but does not create a new array.



You need not to write test condition, neither to initialize nor to increment/decrement anything

forEach syntax

Newton School of Technology

Let’s have a look at its syntax:-

Syntax:

```
originalArray.forEach((element, index, array) => {
   // Action logic here
});
```

Parameters:

- **element:** The current element being processed.
- **index:** The index of the current element.
- **array:** The original array being traversed.

Can you observe that there is a function passed to forEach method? What kind of function it is??

forEach example

Newton School of Technology

Let’s learn to write forEach method, here we will declare and assign an array and iterate over it using forEach method. Let’s begin:-

```
let arr = [1, 2, 3, 4];

// Iterating using forEach
arr.forEach((num) => {
    console.log("Number: ", num);
    // Logs each number in the array
})

// Output:
// Number: 1
// Number: 2
// Number: 3
// Number: 4
```

Can you appreciate how easy it is compared to traditional javascript loops?

forEach comparison with for loop

Newton School of Technology

Let’s compare it with for loop of javascript, which one is easier to write?

```javascript
let arr = [1, 2, 3, 4];

// Iterating using forEach
arr.forEach((num) => {
    console.log("Number: ", num);
    // Logs each number in the array
})

// Output:
// Number: 1
// Number: 2
// Number: 3
// Number: 4
```

forEach

```javascript
let arr = [1, 2, 3, 4];

// Iterating using for loop
for (let i = 0; i < arr.length; i++) {
    console.log("Number: ", arr[i]);
    // Logs each number in the array
}

// Output:
// Number: 1
// Number: 2
// Number: 3
// Number: 4
```

for loop

When to use what: for and forEach

Newton School of Technology

Let’s compare it with the for loop in JavaScript; which one is easier to write??

for loop	forEach
When you want greater flexibility, as you can decide start index, test condition, etc.	forEach is utilized for its simplicity and readability, providing a straightforward way to iterate over array
You can conditionally break or skip code based on condition using break and continue. Hence optimize the performance	You cannot use the break and continue statement and hence can’t optimize the performance.

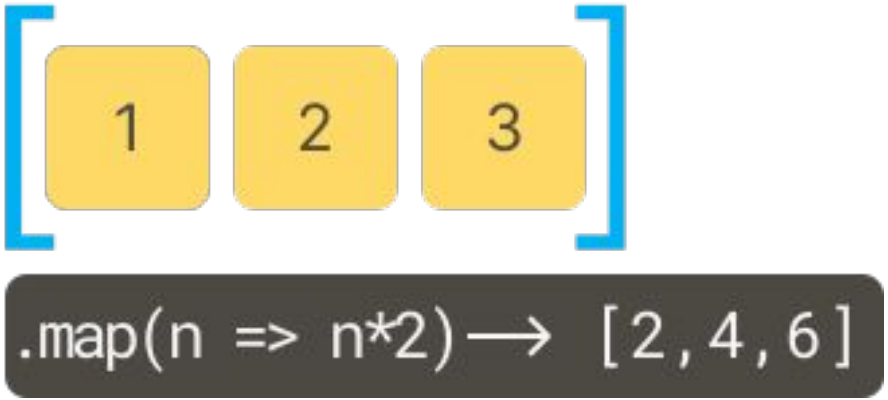
Practice Question

Store 10 student names in an array and use a forEach loop to print their names one by one.

Iteration Methods: What is map()

Newton School of Technology

map is used to iterate over an array and creates a new array with the results, while forEach simply iterates without returning anything.



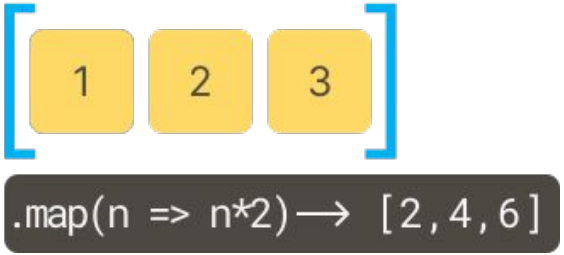
Here we are iterating over an array and returning a new array with values doubled.

map() Example 1

Newton School of Technology

Let’s write code for implementing it:-

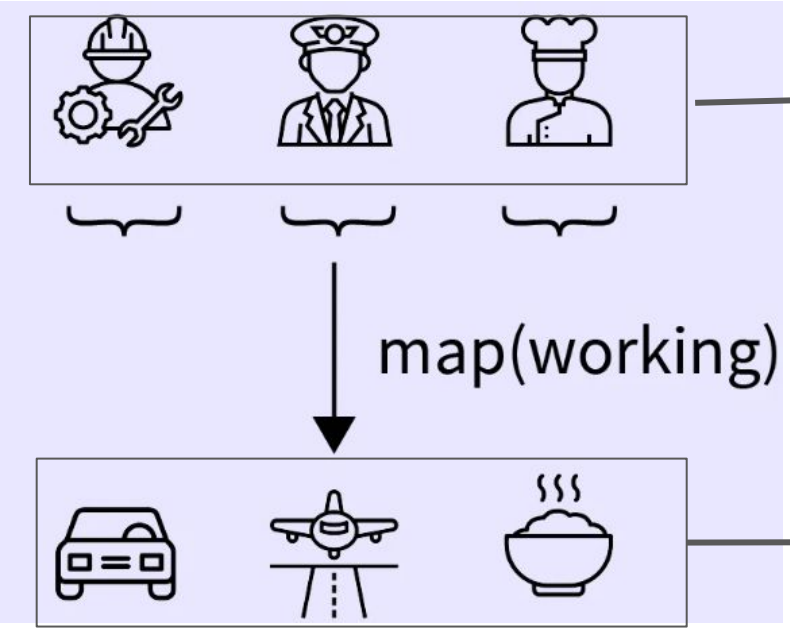
```
// Original array
let numbers = [1, 2, 3];

// Using map to create a new array with doubled values
let doubledNumbers = numbers.map(num => num * 2);

// Printing the result
console.log(doubledNumbers); // Output: [2, 4, 6]
```



map() example 2

Newton School of Technology

You are provided with a workers array and you need to iterate over that array and return an array listing their workplaces.



The map function iterates over the peoples array and returns an array of their workplaces.

map() example 2

Newton School of Technology

Let’s write code example of illustration in previous slide:-

```
// Array of people with their details
let people = [
    { name: "Ram", designation: "Mechanic", workplace: "Garage" },
    { name: "Shyam", designation: "Pilot", workplace: "Airport" },
    { name: "Aryan", designation: "Chef", workplace: "Restaurant" }
];

// Extracting workplaces using map
let workplaces = people.map(person => person.workplace);

// Printing the workplace array
console.log(workplaces);
// Output: ["Garage", "Airport", "Restaurant"]
```

Here we have iterated over people array and returned an array which lists their workplaces and stored in a variable workplaces.

Practice Question

Use map to convert an array of strings to their uppercase versions.

Difference: forEach and map method

Newton School of Technology

Let’s observe the difference between the two:-

forEach	map
Return value: **undefined**	Return value: **newArray will be created** based on your callback function
Original Array: **not modified**	Original Array: **not modified**
newArray is not created after the end of a method call	newArray is not created after the end of a method call

Newton School of Technology

Higher Order functions

Higher order functions: map()

Newton School of Technology

In previous illustrations we have used .forEach() and .map() method, these functions are taking a callback function as input. These type of functions are called higher order functions.

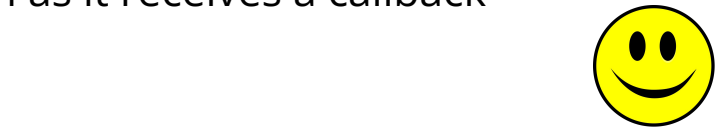
```
1  const arr = [1, 2, 3, 4, 5];
2
3  const printItem = function(item) {
4      console.log("Item: ", item);
5  }
6
7  arr.map(printItem);
```

Here we passed a callback function to map which is higher order function

Higher order functions: forEach()

Newton School of Technology

Similarly forEach() is also a higher order function as it receives a callback function as an argument.



```
1  const arr = [1, 2, 3, 4, 5];
2
3  const printItem = function(item) {
4      console.log("Item: ", item);
5  }
6
7  forEach(printItem);
```

Similarly we would have a look into more higher order functions in upcoming slides.

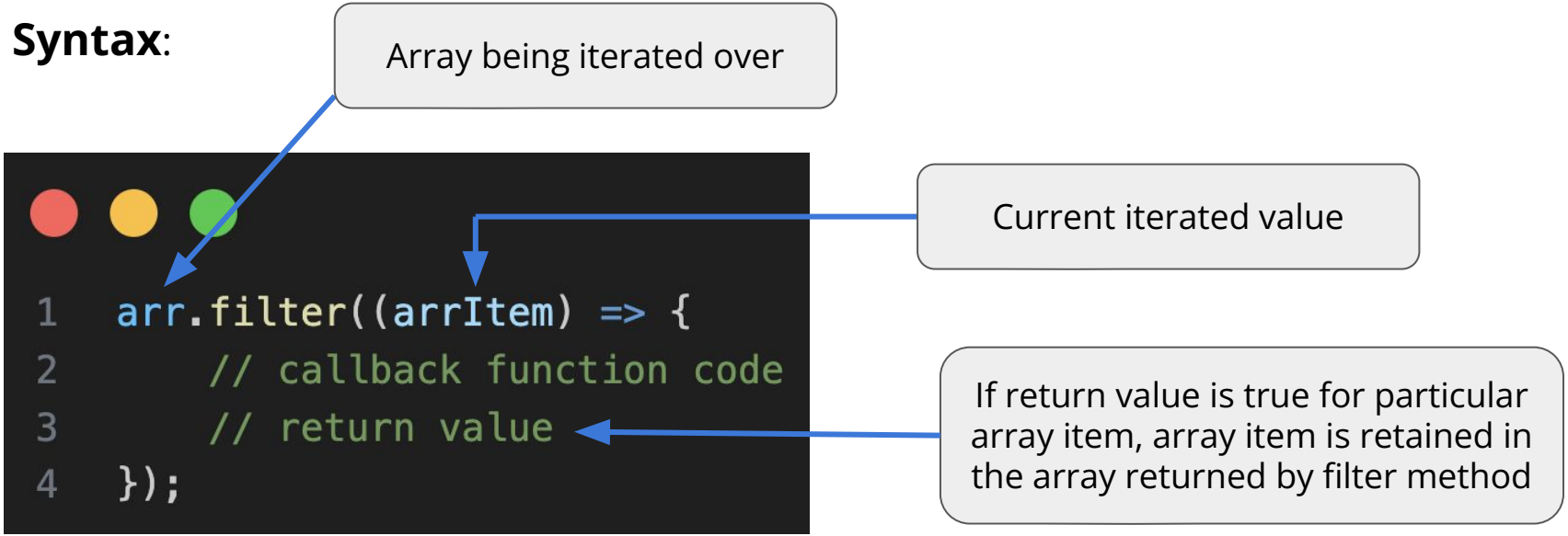
Callback function passed to higher order function forEach()

Search/Filter Methods: filter()

Newton School of Technology

The filter method returns a new array with values from the original array that satisfy the condition in the callback function.

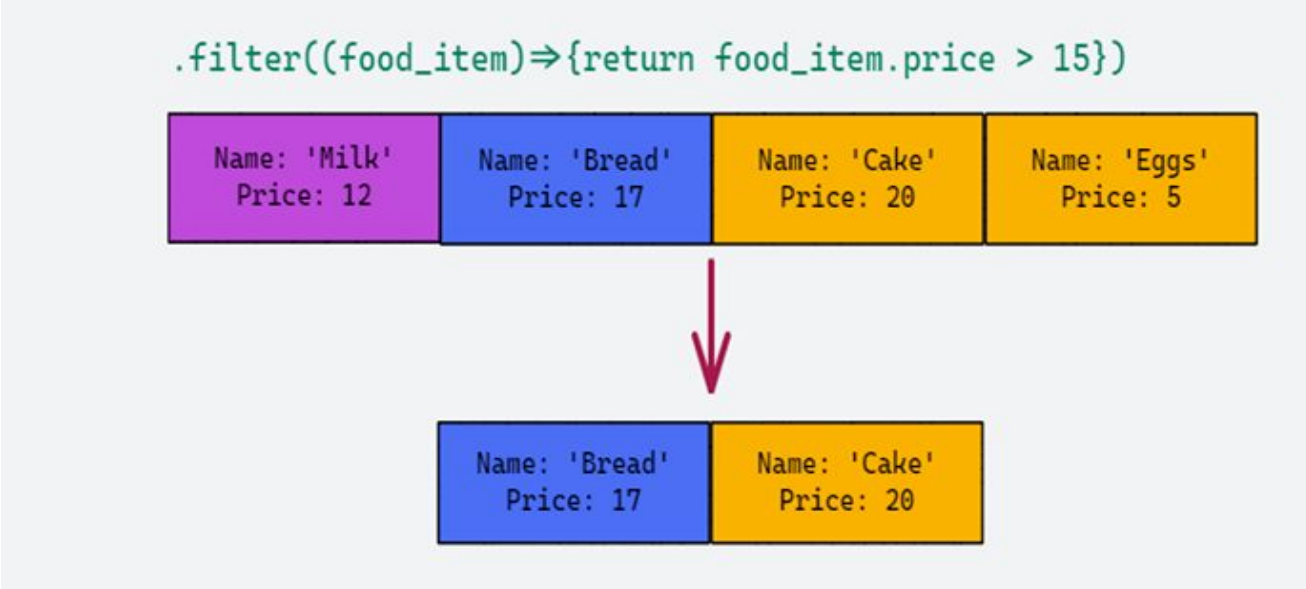
Syntax:



Search/Filter Methods: filter()

Newton School of Technology

Let's say you have an array containing food items and it need to be iterated and return just items having price over 15. Here is diagrammatic representation of the solution:-



Here only those items got filtered out whose price was more than 15.

filter() example

Newton School of Technology

Let’s write code for example in the previous slide:-

```
let foodItems = [
    { name: "milk", price: 12 },
    { name: "bread", price: 17 },
    { name: "cake", price: 20 },
    { name: "eggs", price: 5 }
];

let filteredItems = foodItems.filter(function(foodItem){
    return foodItem.price > 15;
});

console.log(filteredItems);
// Output:
// [ { name: 'bread', price: 17 },
//   { name: 'cake', price: 20 } ]
```

Here bread and cake is filtered out.

Practice Question

Write a function using a filter to return all elements in an array greater than a given value.

Reduction Methods: What is reduce()

Newton School of Technology

The reduce method transforms an array into a single value by applying a function to each element, using an accumulator to store the result. After processing all elements, it returns the final value.

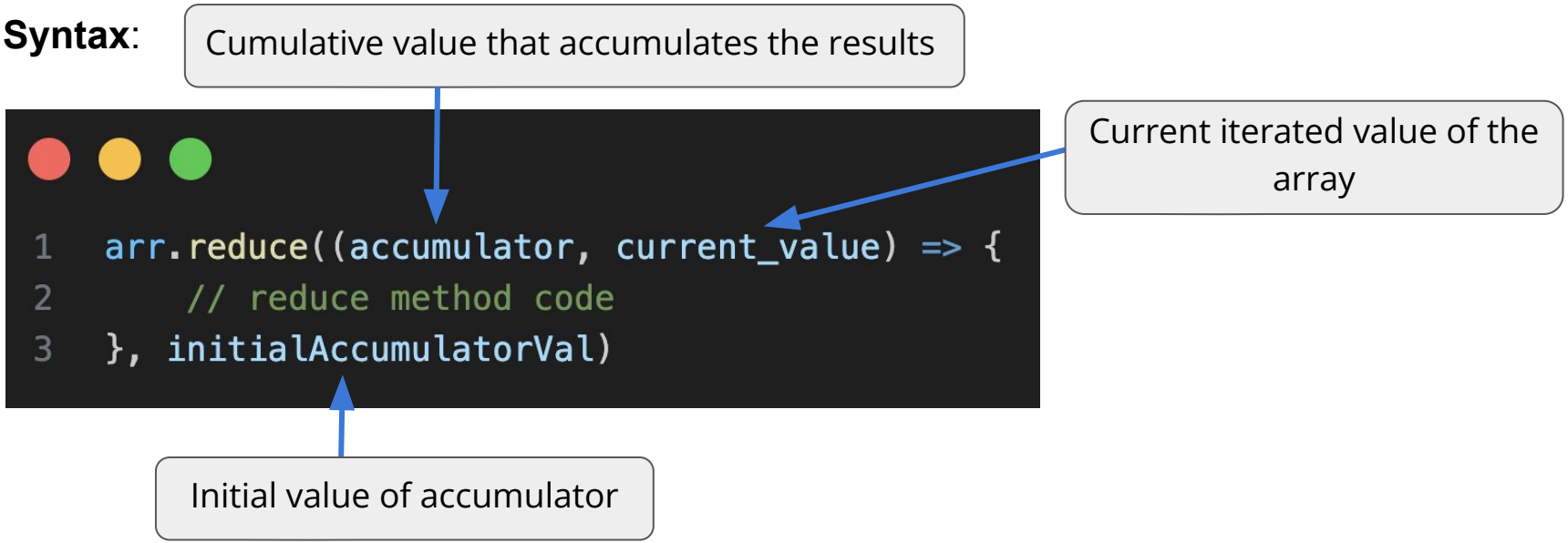
Just like mixing ingredients to make a dish, the reduce method combines values to produce a single result.

reduce(): syntax

Newton School of Technology

The `reduce` method iterates through an array and applies a callback function to accumulate all elements into a single value, such as a sum, product, array, or object.

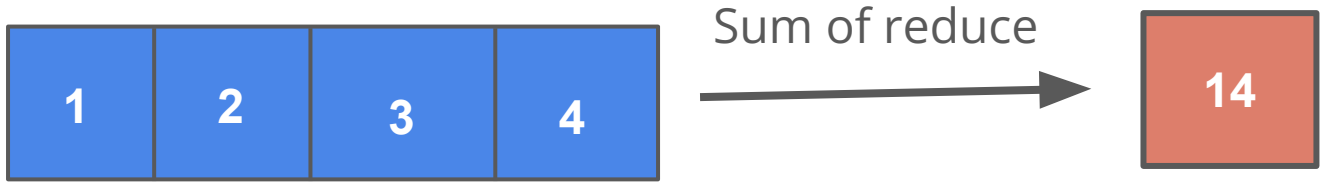
Syntax:



reduce() example 1

Newton School of Technology

Here we were provided with an array, we can use reduce() method to calculate the sum and return the value 14.



reduce() example 1

Newton School of Technology

Let’s implement that example:-

```
1  const numbers = [1, 2, 3, 4];
2
3  // Using reduce to calculate the sum
4  const sum = numbers.reduce((acc, num) => acc + num, 0);
5
6  console.log(sum); // Output: 10
```

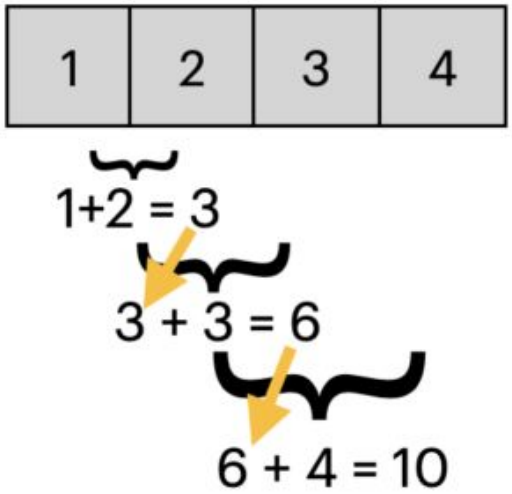
reducer function accepts these arguments:-

1. Accumulator(acc): cumulative value that accumulates the results
2. Current Value(num): current value being processed

reduce() example 1

Newton School of Technology

The reduce method iterates over the array, adding each value to an accumulator, starting from 0, and returns the final sum after processing all elements. In this case, the sum of [1, 2, 3, 4] is 10.

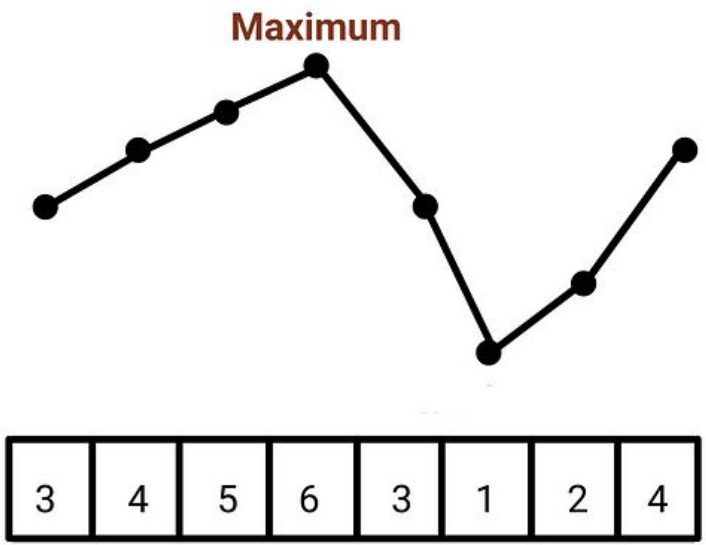


The accumulator starts at zero, adding values at each step, and the final sum is calculated at the end.

reduce() example 2

Newton School of Technology

Here given an array find out maximum value by scanning the array using reduce method.



You might be tempted to use forEach or map method. Wait...wait... there is even better way i.e. reduce()

reduce() example 2

Newton School of Technology

Let’s write the javascript code using reduce().

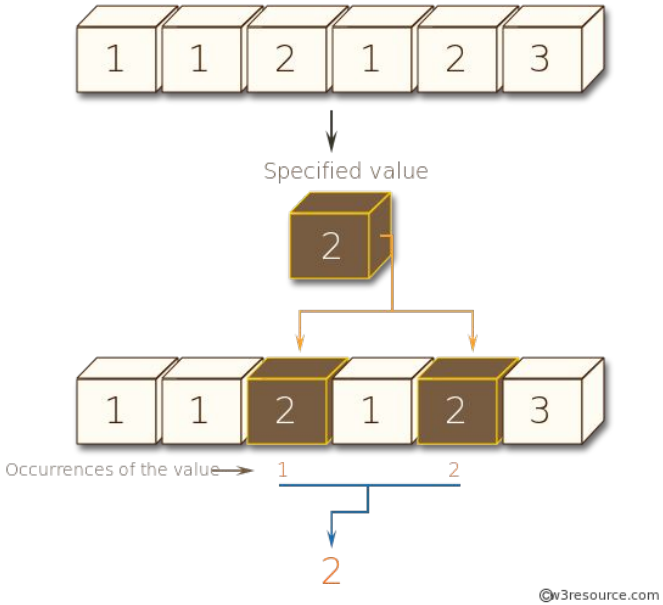
```
1  const numbers = [3, 4, 5, 6, 3, 1, 2, 4];
2
3  const maxValue = numbers.reduce((max, current) => {
4    return current > max ? current : max;
5  }, numbers[0]); // Initialize with the first element of the array
6
7  console.log(`The maximum value is: ${maxValue}`);
8  // Output: The maximum value is: 6
```

Observe how crisp and clear the code is with reduce().

reduce() example 3

Newton School of Technology

Provided an array of integers find out how many times integer value 2 got repeated using reduce method.



Do this one by yourself..

reduce() example 3

Newton School of Technology

Let’s have a look at the solution:-

```
1  const numbers = [1, 1, 2, 1, 2, 3];
2
3  const count = numbers.reduce((total, current) => {
4    return current === 2 ? total + 1 : total;
5  }, 0); // Initialize count to 0
6
7  console.log(`The number 2 appears ${count} times.`);
8  // Output: The number 2 appears 2 times.
```

Were you able to do it by yourself??

Practice Question

Find the first string in the array having more than 5 characters.

In Class Questions

Newton School of Technology

Thanks for watching!