Caltech | Center for Technology & Management Education

Full Stack Java Developer

simplilearn

JavaScript

Arrays

Learning Objectives

By the end of this lesson, you will be able to:

- Define arrays
- List the different operations performed in arrays
- Learn the different parameters and syntaxes used in this element
- Describe how the concat() method is used to join two arrays
- Learn the commonly used iterators in JavaScript

Learning Objectives

By the end of this lesson, you will be able to:

- Describe the ways to test the elements of an array
- Learn how to map an array
- Understand how to use the arrow function to create a function
- List the JavaScript string methods
- Understand template literals and how they are used

A Day in the Life of a Full Stack Developer

You are working on an employee management system application. Here, all the employee data is stored in the form of a table. You store all the employee data in the form of an array. Each employee's data is an array individually as it contains multiple parameters of data.

The data is stored in different arrays for different categories. There can be a situation where you might find details about the same employee in different tables and hence in different arrays.

You are supposed to create a module where a user can get all the related data about an employee at once. You may have to concatenate the arrays.

To do so, understand how to map an array, concat() method, arrow functions, and more.

What are Arrays?

Array

An array is a special value that can hold more than one value.

```
let arr = new Array ();
```

```
let arr = [];
```

Array

Push() method adds an element at the end of an array.

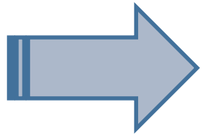
```
arrayname.push ( element );
```

Array

Push() – Example

```
<!DOCTYPE html>
<html>
<script>
    let bike = [‘yamaha', 'honda'];
    colors.push(‘bmw');
    alert(bike);
</script>

</html>
```



Output

```
Yamaha honda bmw
```

Array

The find() method is used in the array to find the element.

- This function returns the value of the first element that passes a test.
- The function for empty elements is not executed by this technique.

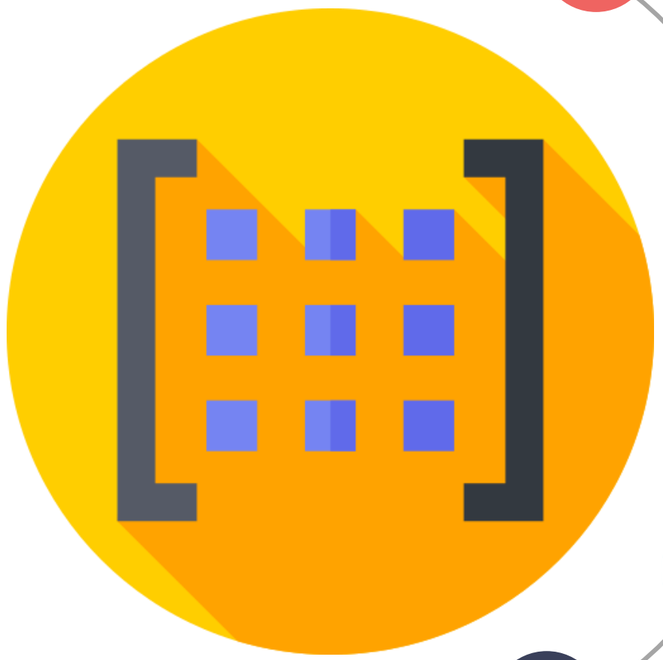
Array

Find() method – Syntax:

```
array.find(function (currentValue,index,arr), thisValue)
```

Array

The parameters are:



function()
A function to run for each array element (required)

currentValue
The value of the current element (required)

Index
The index of the current element index (optional)

arr
The array of the current element

thisValue
The value passed to the function as this value (optional)

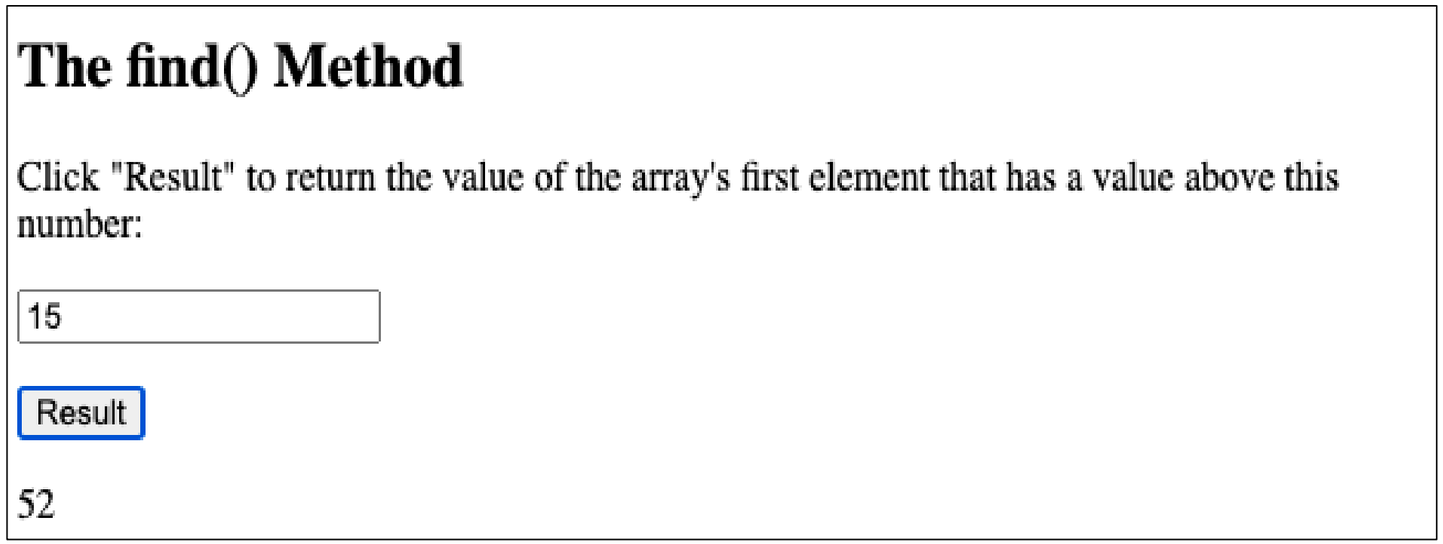
Array

The following is an example of parameters:

```
<!DOCTYPE html>
<html>
<body>
    <h2>The find() Method</h2>
    <p>Click "Result" to return the value of the array's first element that has a
value above this number:</p>
    <p><input type="number" id="numberCheck" value="15"></p>
    <button onclick="myFunction()">Result</button>
    <p id="result"></p>
    <script>
        const ages = [52, 19, 29, 44];
        function checkNumber(age) {
            return age > document.getElementById("numberCheck").value;
        }
        function myFunction() {
            document.getElementById("result").innerHTML = ages.find(checkNumber);
        }
    </script>
</body>
</html>
```

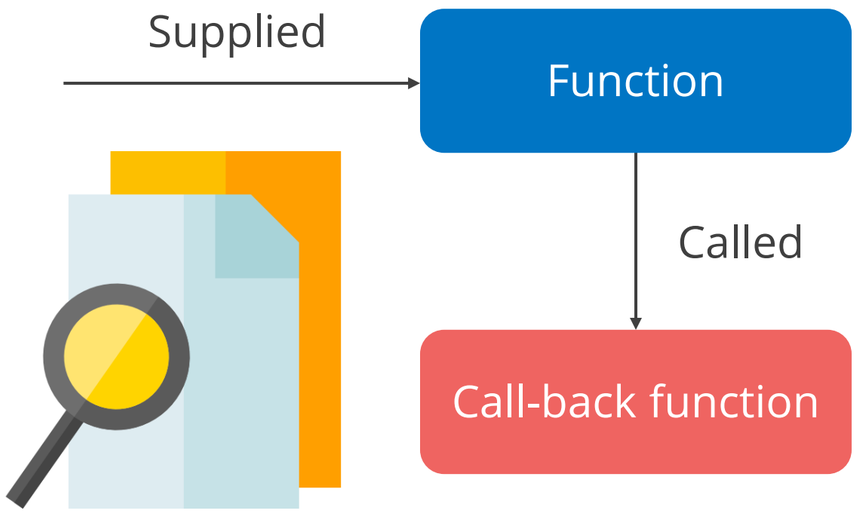
Array

Output:



Array

The find method returns the value of the first member in the array that matches the provided testing routines.



It is used to check if a given element exists in the array.

Array

Find method – Example:

```
const subjects = [
        { id:1, name: 'x' },
        { id:2, name:'y' },
      ];
const sub = subjects.find(function(sub) {
     return sub.name === 'x';
});
console.log(sub);
```

Array

The FindIndex method returns -1 if the element does not exist in the array.

FindIndexmethod – Example:

```
const subjects = [
        { id:1, name: 'x' },
        { id:2, name:'y' },
      ];
const sub = subjects.findIndex(function(sub) {
     return sub.name === 'ab';
});
console.log(sub);
```

Array

JavaScript has different ways to remove the element from the start, in the middle, and at the end.

1. Pop method
2. Shift method
3. Splice method

Array

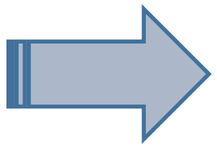
The pop method is used to delete the last element from an array.

```
pop();
```

Array

Pop method - Example

```
const numbers = [ 1,2,3,4,5,6 ];
numbers.pop();
console.log(numbers);
```



Output

```
1,2,3,4,5
```

Array

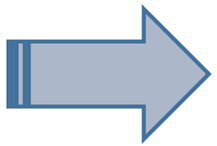
The shift method removes an element at the beginning of the array.

```
shift();
```

Array

Shift Method - Example

```
const numbers = [ 1,2,3,4,5,6 ];
numbers.shift();
console.log(numbers);
```



Output

```
2,3,4,5,6
```

Array

The splice method takes starting index as the first parameter and the number of elements to be eliminated from the starting index as a second parameter.

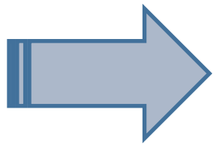
Syntax:

```
splice();
```

Array

Splice Method - Example

```
const numbers = [ 1,2,3,4,5,6 ];
numbers.splice(3,2);
console.log(numbers);
```



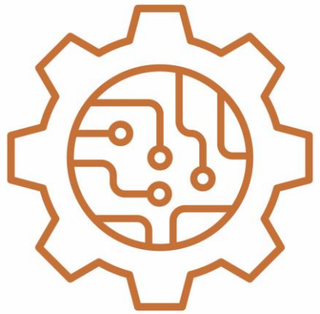
Output

```
1,2,3,6
```

Array

An array can be emptied in four different ways:

- Assigning new empty array
- Setting its length to zero
- Using the splice() method
- Using the pop() method



Array

An array can be emptied by assigning a new empty array:

```
Let a = [ 1,2,3,4,5,6,7,8,9]
a = [ ]
```

Array

An array can be emptied by setting its length to zero:

```
a.length = 0;
```


When the length property is set to zero, all the elements of the array are automatically deleted.

Array

An array can be emptied by using the splice() method:

```
a.splice(0,a.length);
```

The splice() method removes all the elements of the array and returns the removed elements as an array.

Array

Pop method removes each element of the array using the pop() method.

```
while (arr.length > 0)
{
        arr.pop();
}
```

Array

The two arrays can be combined using the Concat() method.

Concat() method – Example

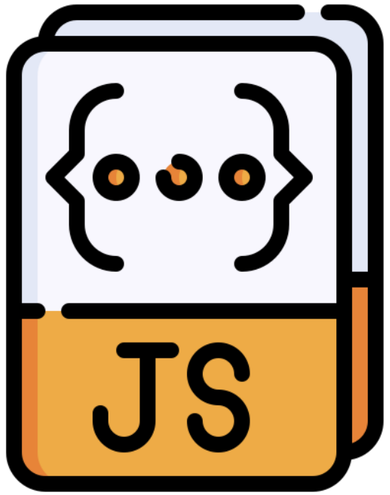
```
const a1 = [1,2,3];
const a2 = [6,7,8];
const  combined = a1.concat(a2);
console.log(combined);
```

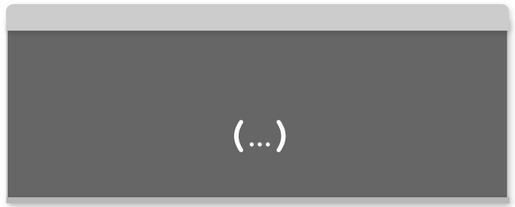
Output

```
1,2,3,6,7,8
```

Array

The spread operator is commonly used to make shallow copies of JS objects.

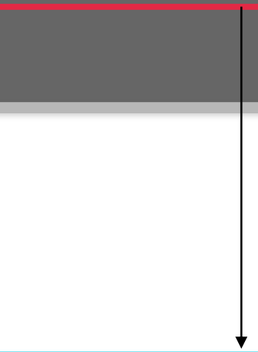




Array

Example of the spread operator:

```
const arrValue = [ 'hello', 'I', 'am', 'reet' ];
console.log(arrValue);  // [ "hello", "I", "am", "reet" ];
console.log(...arrValue);  // hello I am reet
```



console.log(...arrValue) is equivalent to console.log ('hello', 'I', 'am', 'reet');

Array

To replicate the objects into a single array, spread syntax is used:

```
const arr1 = [ one, two ];
const arr2 = [ arr1… , three, four ];
console.log (arr2);
//Output
//[ one, two, three, four ]
```

Array

In JavaScript, the objects are assigned by the reference and not by the values.

Using the '=' operator to create a new copy is not allowed.

```
Let arr1 = [ 1, 2, 3, 4, 5 ];
Let arr2 = [...arr1];
console.log(arr1); // [ 1, 2, 3, 4, 5 ]
console.log(arr2); // [ 1, 2, 3, 4, 5 ]

// append an element to the array
arr1.push(6);
  console.log(arr1); //[ 1, 2, 3, 4, 5, 6 ]
  console.log(arr2); //[1, 2, 3, 4, 5, 6 ]
```

Array

The spread operator is used with object literals to distinguish itself from the others.

```
const obj1 = { a:1, b:2 };
const obj2 = { c:3, d:4 };
// add members obj1 and obj2 to obj3
Const obj3 = {...obj1, ...obj2 };
console.log(obj3);  // { a:1 , b:2,
c:3, d:4 }
```


The rest parameter can be used to accept numerous parameters in a function call.

Array

Example:

```
Let fun = function(...args)
{
    console.log(args);
}
fun(4); // [4]
fun( 5, 6, 7, 8 ); // [ 5, 6, 7, 8 ]
```

The remainder parameter takes all four arguments when they are given.

Array

The spread operator is used to pass many arguments to the function.

Example:

```
Function sum (a,b,c ) {
      console.log( a+b+c );
}
const num = [ 2, 4 ,8, 9 ];
Sum (....num) // 14
```

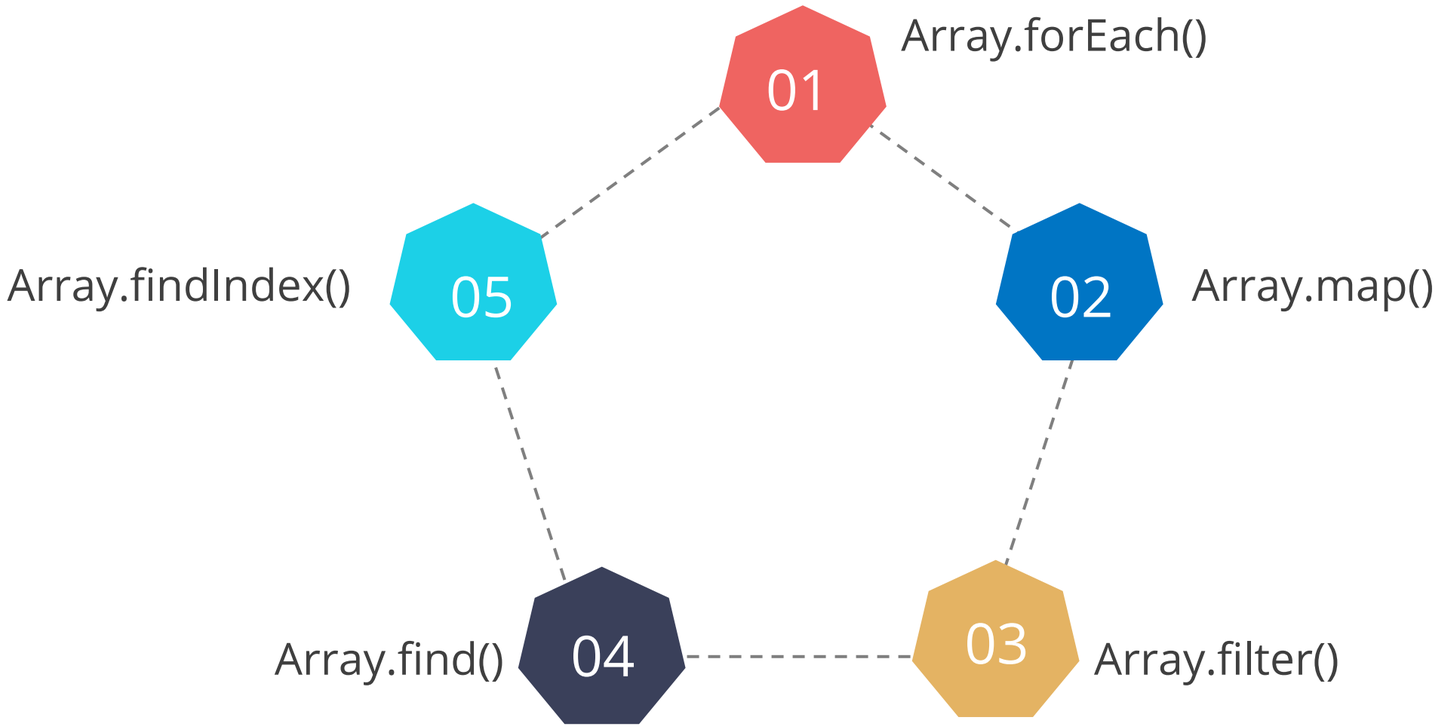
Array

The array iterator is the most used in JavaScript.

Iterators are the methods that are called to manipulate elements of the arrays and return some values.

Array

The commonly used iterators are:



Array

The concat() method is used to join two arrays in JavaScript.

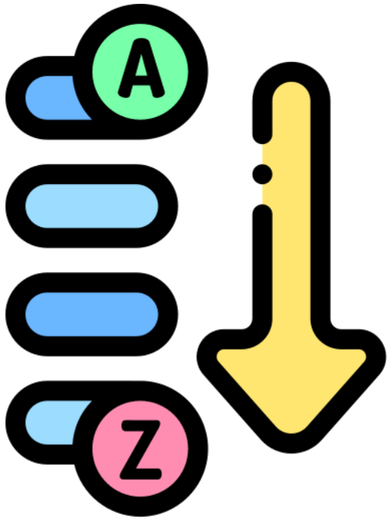
Syntax:

```
concat()
concat(value0)
contact(value0, value1)
concat(value0, value1…..valueN)
```

This function returns the new array rather than altering the current arrays.

Array

The sort() method sorts the items of an array in a specific order.

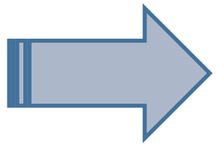



```
sort()
```

Array

Sort() method – Example:

```
Let city = [ 'Ludhiana', 'Chandigarh',
'Jalandhar', 'Patiala' ];
// sort the city array in ascending order
Let sortArray = city.sort();
console.log(sortArray);
```



Output

```
// Output: [ 'Chandigarh',
'Jalandhar', 'Ludhiana',
'Patiala' ]
```

Array

Sort() parameters syntax:

```
arr.sort(compareFunction)
```

CompareFunction (optional) is used to define a custom sort order.

Array

The array elements checking is done for a loop.

Loop through the numbers array's entries, checking whether each one is less than or equal to zero.

Array

The filter() method returns a new array containing the provided data.

It does not execute the function for the empty elements or change the original array.

```
array.filter(function (currentValue, index, arr), thisValue)
```

Array

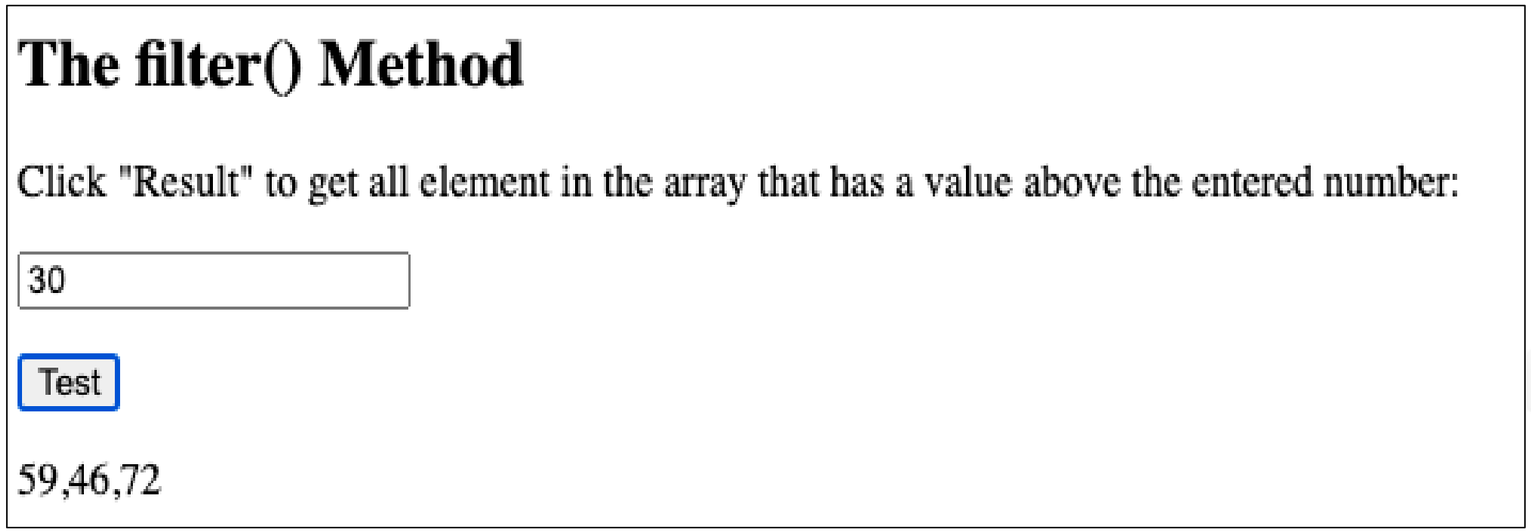
Example of the filter() method:

```
<!DOCTYPE html>
<html>

<body>
    <h2>The filter() Method</h2>
    <p>Click "Result" to get all element in the array that has a value above the entered
number:</p>
    <p><input type="number" id="numberCheck" value="30"></p>
    <button onclick="myFunction()">Test</button>
    <p id="result"></p>
    <script>
        const ages = [59, 21, 46, 72];
        function checkNumber(age) {
            return age > document.getElementById("numberCheck").value;
        }
        function myFunction() {
            document.getElementById("result").innerHTML = ages.filter(checkNumber);
        }
    </script>
</body>
</html>
```

Array

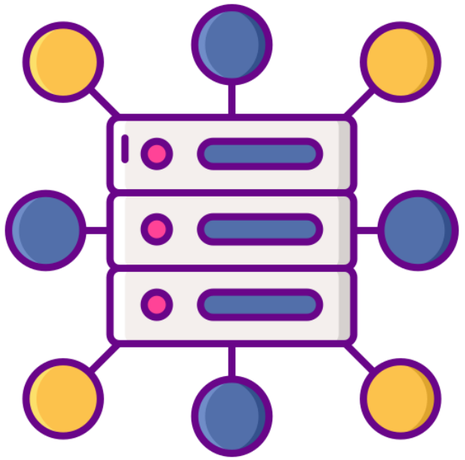
Output of the filter() method:



Array

The mapping of an array is done with the help of the map() function.

The function is not executed for empty elements, and it does not change the original value.



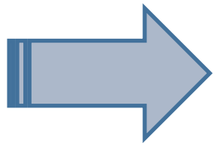
```
array.map(function(currentValue, index, arr)  , thisValue)
```

Array

Example:

```
<!DOCTYPE html>
<html>
<body>
    <p>Multiply every element in the array with
10:</p>
    <p id="result"></p>
    <script>
        const numbers = [10, 20, 30, 40];
        const arr = numbers.map(funFunction);

document.getElementById("result").innerHTML = arr;
        function funFunction(num) {
            return num * 10;
        }
    </script>
</body>
</html>
```

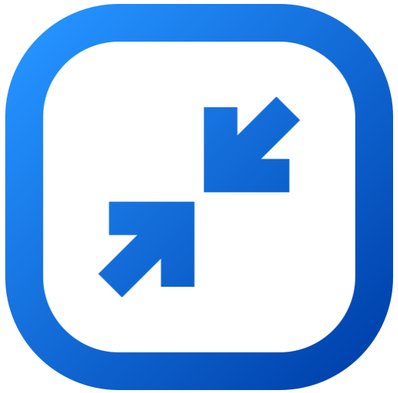


Output:

```
Multiple every element in the
array by 10:
100,200,300,400
```

Array

The reduce method is used to reduce the array to a single value.



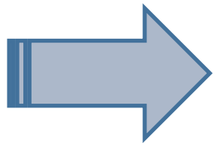
```
array.reduce(function(total, currentValue,
currentIndex, arr), initialValue)
```

Array

Example:

```
<!DOCTYPE html>
<html>
<body>
    <h2>Reduce() Method</h2>
    <p>Calculate the sum of the rounded numbers in
a given array.</p>
    <p id="result"></p>
    <script>
        const numbers = [1.6, 2.3, 5.8, 3.2];

document.getElementById("result").innerHTML =
numbers.reduce(getSum, 0);
        function getSum(total, num) {
            return total + Math.round(num);
        }
    </script>
</body>
</html>
```



Output:

```
Reduce() Method
Calculate the sum of the
rounded numbers in a given
array.
12
```

Arrow Function

Arrow Function

The arrow function is a better way of creating a function than the function's expressions.

```
Let fun = ( arg1, arg2,arg3b,..............., argN ) => expression
```

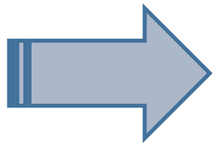
Arrow Function

Example:

```
<!DOCTYPE html>
<html>

<body>
    <script>
        let sum = (a, b) => a + b;
        alert(sum(2, 6));
    </script>
</body>

</html>
```



Output

```
This page says
8
```

Arrow Function

The multi-line arrow functions work for more complex objects like expressions or statements.

Example of multi-line arrow function:

```
let sum = (x, y) => {  // the curly brace
opens a multiline function
let add = x + y;
return add; // if we use curly braces, then
we need an explicit "return"
};
alert( sum(2, 3) ); // 5
```

Key Takeaways

- An array is a special value that can hold more than one value.
- The find method returns the value of the first member in the array that matches the provided testing routines.
- The pop method is used to delete the last element from an array.
- The spread operator is used to pass many arguments to the function.
- The reduce method is used to reduce the array to a single value.

TECHNOLOGY

Thank You

simplilearn