Coding Bootcamp

JavaScript

Arrays

Learning Objectives

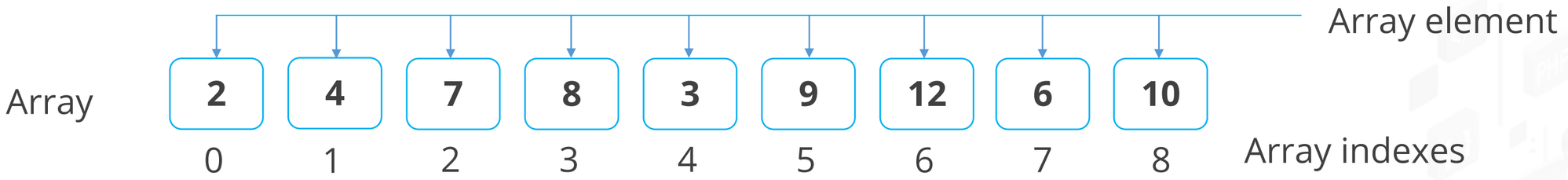
By the end of this lesson, you will be able to:

- Utilize arrays to store and manipulate multiple values in JavaScript
- Apply the find method to search for specific elements within an array based on defined criteria
- Implement the pop method to remove elements from the end of an array as needed
- Employ the spread operator to efficiently transmit multiple arguments to functions
- Analyze the process of reducing an array and its implications on program efficiency

Arrays

Array

An array is a special data structure that can hold multiple values.



This syntax uses the Array constructor to create a new array.

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

This syntax uses an array literal notation to create a new array.

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

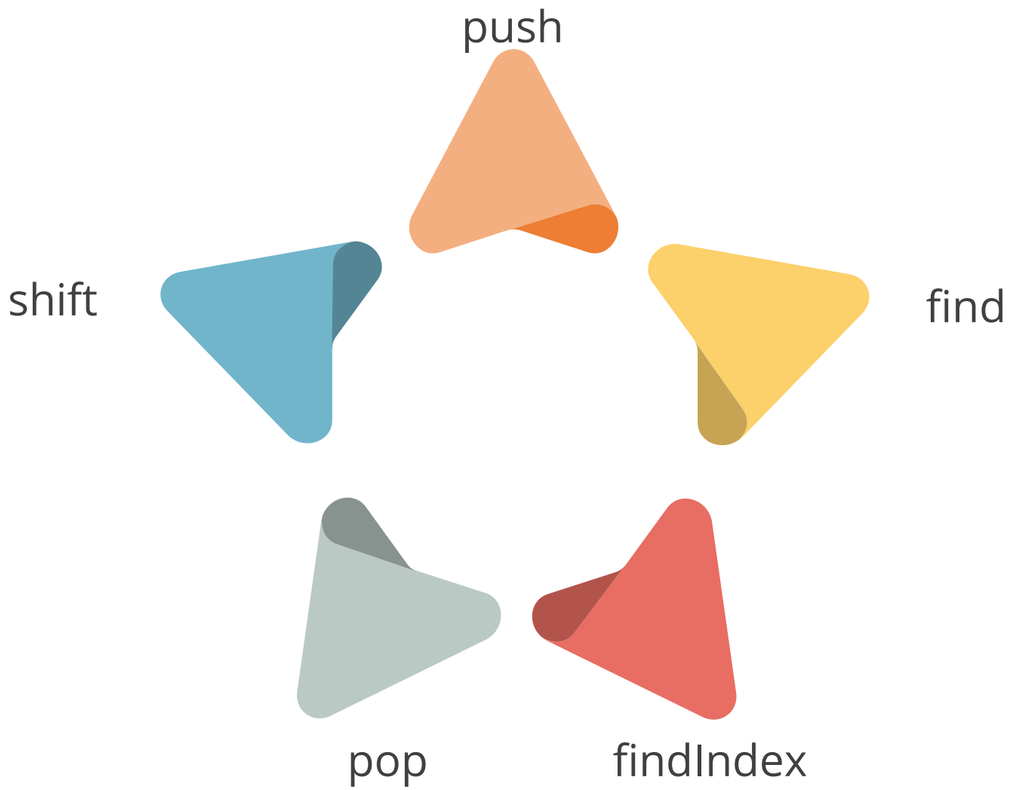
Parameters

It is defined in a function's parentheses and serves as local variables that store input values for the function's logic.

function()	A function to run for each array element
currentValue	The value of the current element
Index	The index of the current element index
arr	The array of the current element
thisValue	The value passed to the function as this value

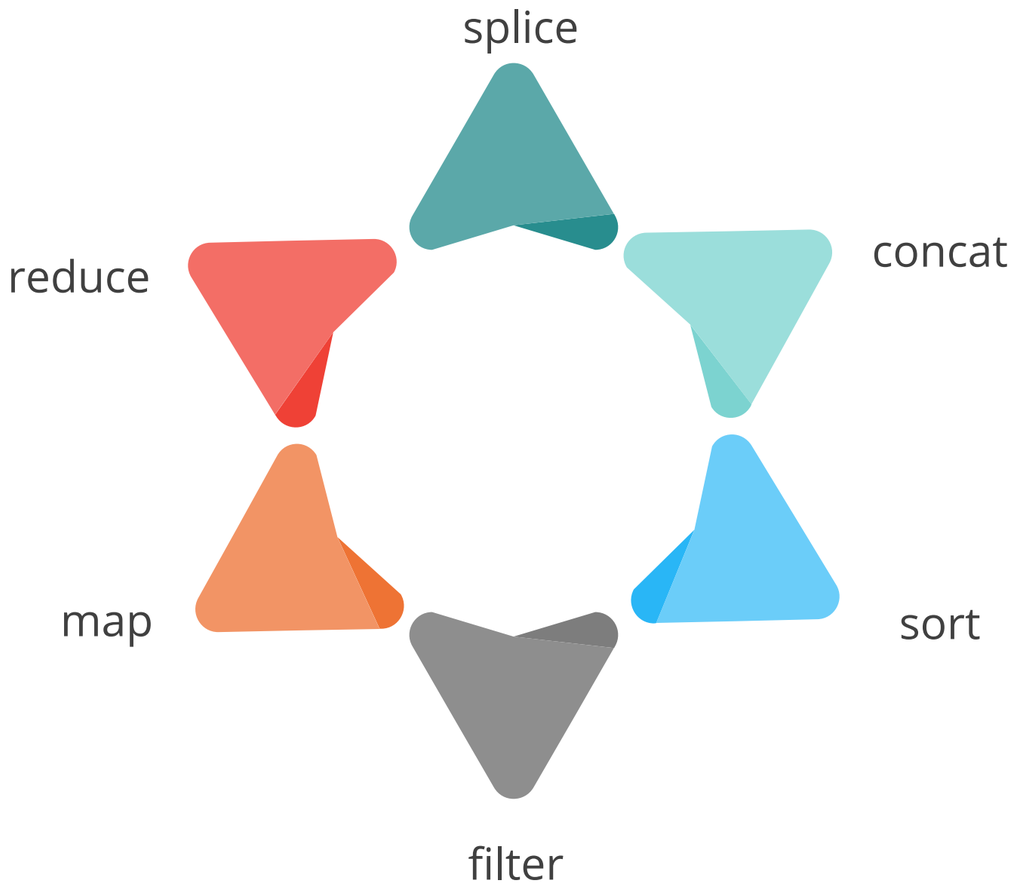
Methods

Some of the commonly used array methods for manipulation and iteration are:



Methods

Some of the commonly used array methods for manipulation and iteration are:



Push

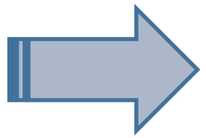
It adds an element at the end of an array.

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

Push() – Example:

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

</html>
```



```
Yamaha honda bmw
```

Find

It is used in the array to find the element.

This function returns the value of the first element that passes a test.

This function does not execute for empty elements.

Syntax:

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

Example:

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

FindIndex

It is used to find the index of the first element in an array that satisfies a provided testing function.

It returns the index of the first element that matches the condition; if no elements match, it returns -1.

Syntax:

```
array.findIndex(callback(element[, index[, array]])[, thisArg])
```

Example:

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

Pop

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

```
pop();
```

Pop method - Example

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

Output

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

Shift

It removes an element at the beginning of the array.

```
shift();
```

Shift Method - Example

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

Output

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

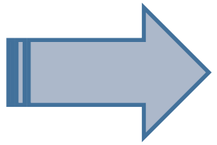
Splice

It takes the start index and the number of elements to remove.

```
splice();
```

Splice Method - Example

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



Output

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

Ways to Empty an Array

An array can be emptied in four different ways:

Assigning a new empty array

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

Setting its length to zero

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

Using the splice() method

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

Using the pop() method

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

Spread Operator

It is commonly used to make shallow copies of JS objects.

```
(...)
```

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');

Spread Operator

It is used with object literals to distinguish itself from the others.

Syntax:

```
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.

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.

Concat

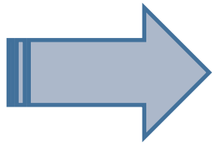
The concat() method is used to join two arrays in JavaScript.

Syntax:

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

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
```

Sort

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

Syntax

```
sort()
```

Parameter syntax

```
arr.sort(compare
Function)
```

Sort() method – Example:

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

Output

```
[ 'Chandigarh', 'Jalandhar',
'Ludhiana', 'Patiala' ]
```

Filter

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)
```

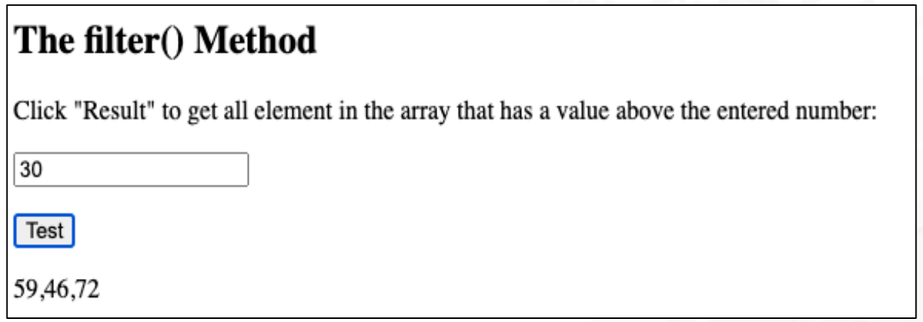
Filter: Example

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>
```

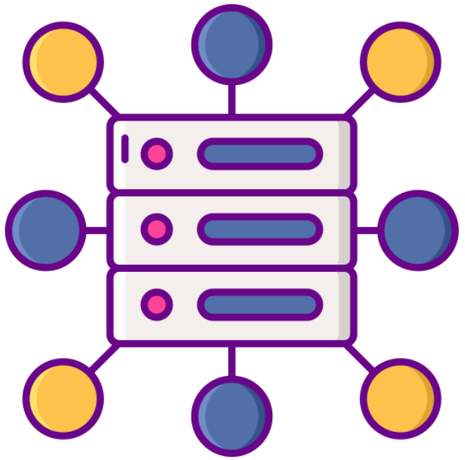
Output of the filter() method:



Map

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.



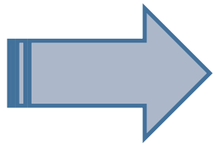
```
array.map(function(currentValue, index, arr)  , thisValue)
```

Map: Example

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>
```

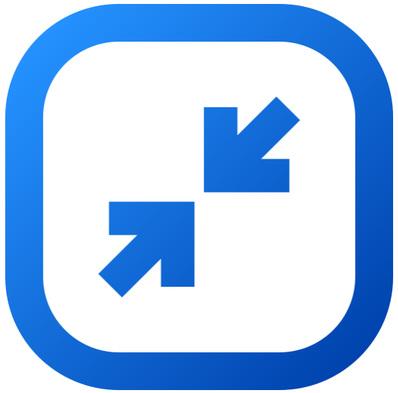


Output

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

Reduce

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



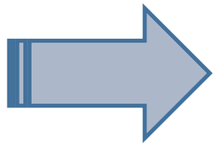
```
array.reduce(function(total, currentValue,
currentIndex, arr), initialValue)
```

Reduce: Example

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

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

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

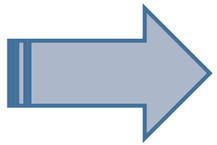
Arrow Function: Example

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
```

Creating a Shopping Cart List of Product Objects

Problem Statement:

You have been asked to demonstrate how to create a shopping cart using JavaScript arrays, add dishes to the cart, and display the contents of the cart.

Outcome:

By completing this task, you will be able to create a shopping cart using JavaScript arrays, add dishes to the cart, and display the contents of the cart. This involves creating a menu and a shopping cart, then implementing logic to add dishes to the cart.

Note: Refer to the demo document for detailed steps:
01_Creating_a_Shopping_Cart_with_a_List_of_Products

ASSISTED PRACTICE

Assisted Practice: Guidelines

Steps to be followed are:

1. Creating a menu and shopping cart
2. Implementing logic to add dishes to the cart

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