The Arrays Object

The Array object lets you store multiple values in a single variable. It stores a fixed-size sequential collection of elements of the same type. An array is used to store a collection of data, but it is often more useful to think of an array as a collection of variables of the same type.

There are 3 ways to construct array in JavaScript

1. **By array literal**
2. **By creating instance of Array directly (using new keyword)**
3. **By using an Array constructor (using new keyword)**

1) Array literal

The syntax of creating array using array literal is given below:

var arrayname=[value1,value2.....valueN];

As you can see, values are contained inside [ ] and separated by , (comma).

Let's see the simple example of creating and using array in JavaScript.

```
<script>
var emp=["Sonoo","Vimal","Ratan"];
```

```
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br/>");
}
</script>
```

The .length property returns the length of an array.

Output of the above example

```
Sonoo
Vimal
Ratan
```

2) Array directly (new keyword)

The syntax of creating array directly is given below:

```
var arrayname=new Array();
```

Here, new keyword is used to create instance of array.

Let's see the example of creating array directly.

```
<script>
var i;
var emp = new Array();
emp[0] = "Arun";
emp[1] = "Varun";
emp[2] = "John";
```

```
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Output of the above example

```
Arun
Varun
John
```

3) Array constructor (new keyword)

Here, you need to create instance of array by passing arguments in constructor so that we don't have to provide value explicitly.

The example of creating object by array constructor is given below.

```
<script>
var emp=new Array("Jai","Vijay","Smith");
for (i=0;i<emp.length;i++){
document.write(emp[i] + "<br>");
}
</script>
```

Output of the above example

```
Jai
Vijay
Smith
```

Array length Property

The array length property in JavaScript is used to set or return the number of elements in an array.

Syntax:

It is used to set the array length.

array.length = number

It returns the length of the array.

array.length

Return Value:

It returns a numerical value, denoting the number of elements in the array object

Below is an example of the Array length property.

Example:

This example returns the length of the array.

```
function array() {
   let colors = ["green", "blue", "red",
      "yellow", "black", "white"];
```

```
    console.log(colors.length);
}
array();
```

Output
6

More example codes for the above property are as follows:

Example 1:

- Javascript

```
let arr = new Array("Geeks", "for", "Geeks");
console.log("arr.length:" + arr.length);
```

Output:
arr.length:3

Example 2:

- Javascript

```
let arr = new Array(5, 10, 15);
console.log("arr.length:" + arr.length);
```

Output:
arr.length:3

Array Methods

concat()

The JavaScript Array concat() Method is used to merge two or more arrays together. This method does not alter the original arrays passed as arguments but instead, returns a new Array.

Syntax:

let newArray1 = oldArray.concat()

let newArray2 = oldArray.concat(value0)

let newArray3 = oldArray.concat(value0,value1)

.....

let newArray = oldArray.concat(value1 , [value2, [...,[valueN]]])

Parameters: The parameters of this method are the arrays or the values that need to be added to the given array. The number of arguments to this method depends upon the number of arrays or values to be merged.

Return value: This method returns a newly created array that is created after merging all the arrays passed to the method as arguments.

Example 1

Below is an example of the Array concat() method to join three arrays.

```
// JavaScript code for concat() method
function func() {
    let num1 = [11, 12, 13],
        num2 = [14, 15, 16],
        num3 = [17, 18, 19];
    console.log(num1.concat(num2, num3));
}
func();
```

Output

[11, 12, 13, 14, 15,16, 17, 18, 19]

Example 2

In this example, the method concat() concatenates all the arguments passed to the method with the given array into one array which it returns as the answer.

```
// JavaScript code for concat() method
function func() {
    let alpha = ["a", "b", "c"];
    console.log(alpha.concat(1, [2, 3]));
}
func();
```

Output

['a', 'b', 'c', 1, 2, 3]

Example 3:

In this example, the method concat() concatenates both arrays into one array which it returns as the answer.

```
// JavaScript code for concat() method
function func() {
    let num1 = [[23]];
    let num2 = [89, [67]];
    console.log(num1.concat(num2));
}
func();
```

Output

[[23], 89, [67]]

fill()

Method fills a given range of array elements with the given value. This method is used to manipulate the existing array according to our needs.

Syntax:

arr.fill(value,start,end)

Parameters:

- Value: Value to be filled.
- Start: Start index (included) and its default value is 0.
- End: End index (excluded) and its default index is this.length.

Return value: It returns the modified array.

Example 1:

In this example, we will fill an array with 0 starting from an index at 2 till the end index of 4.

```
// input array contain some elements.
let array = [1, 2, 3, 4];

// Here array.fill function fills with 0
// from position 2 till position 3.
console.log(array.fill(0, 2, 4));
```

Output:

Array [1, 2, 0, 0]

Example 2:

In this example, we will fill the array with 9 starting from an index at 2 till the end index of the array.

```
// input array contain some elements.
let array = [1, 2, 3, 4, 5, 6];
```

```
// Here array.fill function fill with
// 9 from position 2 till end.
console.log(array.fill(9, 2));
```

Output:

Array [1, 2, 9, 9, 9, 9]

Example 3:

In this example, we will fill the array completely with 6. This is done by not giving the start and end index to the fill method.

```
// input array contain some elements.
let array = [1, 2, 3, 4];

// Here array.fill function fill with
// 6 from position 0 till end.
console.log(array.fill(6));
```

Output:

Array [6, 6, 6, 6]

filter()

The JavaScript Array filter() Method is used to create a new array from a given array consisting of only those elements from the given array which satisfy a condition set by the argument method.

Syntax:

array.filter(callback(element, index, arr), thisValue)

Parameters: This method accepts five parameters as mentioned above and described below:

- callback: This parameter holds the function to be called for each element of the array.
- element: The parameter holds the value of the elements being processed currently.
- index: This parameter is optional, it holds the index of the current element in the array starting from 0.
- arr: This parameter is optional, it holds the complete array on which Array.every is called.
- thisValue: This parameter is optional, it holds the context to be passed as this is to be used while executing the callback function. If the context is passed, it will be used like this for each invocation of the callback function, otherwise undefined is used as default.

Return value: This method returns a new array consisting of only those elements that satisfied the condition of the arg_function.

Example 1:

In this example, the method filter() creates a new array consisting of only those elements that satisfy the condition checked by canVote() function.

```
 // JavaScript to illustrate findIndex() method
function canVote(age) {
    return age >= 18;
}

function func() {
    let filtered = [24, 33, 16, 40].filter(canVote);
    console.log(filtered);
}
func();
```

Output:

[24,33,40]

Example 2:

In this example, the method filter() creates a new array consisting of only those elements that satisfy the condition checked by isPositive() function.

```
function isPositive(value) {
    return value > 0;
}
```

```
let filtered = [112, 52, 0, -1, 944].filter(isPositive);
console.log(filtered);
```

Output:

[112,52,944]

Example 3:

In this example, the method filter() creates a new array consisting of only those elements that satisfy the condition checked by isEven() function.

```
function isEven(value) {
   return value % 2 == 0;
}

let filtered = [11, 98, 31, 23, 944].filter(isEven);
console.log(filtered);
```

Output:

[98,944]

find()

The Javascript arr.find() method in Javascript is used to get the value of the first element in the array that satisfies the

provided condition. It checks all the elements of the array and whichever the first element satisfies the condition is going to print. This function will not work function having the empty array elements and also does not change the original array.

Syntax:

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

Parameters: This method accepts 5 parameters as mentioned above and described below:

- function: It is the function of the array that works on each element.
- currentValue: This parameter holds the current element.
- index: It is an optional parameter that holds the index of the current element.
- arr: It is an optional parameter that holds the array object to which the current element belongs to.
- thisValue: This parameter is optional. If a value is to be passed to the function to be used as its “this” value else the value “undefined” will be passed as its “this” value.

Return value: It returns the array element value if any of the elements in the array satisfy the condition, otherwise it returns undefined.

We will understand the concept of the Javascript Array find() method through the examples.

Example 1:

The below example illustrates the JavaScript Array find() method to find a positive number.

```
// Input array contain some elements.
let array = [-10, -0.20, 0.30, -40, -50];

// Method (return element > 0).
let found = array.find(function (element) {
   return element > 0;
});

// Printing desired values.
console.log(found);
```

Output:

0.3

Example 2:

Here, the arr.find() method in JavaScript returns the value of the first element in the array that satisfies the provided testing method.

```
// Input array contain some elements.
let array = [10, 20, 30, 40, 50];

// Method (return element > 10).
let found = array.find(function (element) {
   return element > 20;
});
```

```
// Printing desired values.
console.log(found);
```

Output:

30

Example 3:

In this example, whenever we need to get the value of the first element in the array that satisfies the provided testing method at that time we use arr.find() method in JavaScript.

```
// Input array contain some elements.
let array = [2, 7, 8, 9];

// Provided testing method (return element > 4).
let found = array.find(function (element) {
    return element > 4;
});

// Printing desired values.
console.log(found);
```

Output:

7

flat()

The Javascript arr.flat() method was introduced in ES2019. It is used to flatten an array, to reduce the nesting of an array. The flat() method is heavily used in the functional programming paradigm of JavaScript. Before the flat() method was introduced in JavaScript, various libraries such as underscore.js were primarily used.
Syntax:
arr.flat([depth])

Parameters: This method accepts a single parameter as mentioned above and described below:

- depth: It specifies, how deep the nested array should be flattened. The default value is 1 if no depth value is passed as you guess it is an optional parameter.

Return value: It returns an array i.e. depth levels flat than the original array, it removes nesting according to the depth levels.
Below is an example of the Array flat() method.

Example 1

In this example, we will see the basic implementation of the Array flat() method.

```
// Creating multilevel array
const numbers = [['1', '2'], ['3', '4',
```

```
            ['5',['6'], '7']]];

const flatNumbers= numbers.flat(Infinity);
console.log(flatNumbers);
```

Output

['1', '2', '3','4', '5', '6','7']

Example 2

The following code snippet shows, how the Array flat() method works.

```
let nestedArray = [1, [2, 3], [[]],
            [4, [5]], 6];

let zeroFlat = nestedArray.flat(0);

console.log(
`Zero levels flattened array: ${zeroFlat}`);
// 1 is the default value even
// if no parameters are passed
let oneFlat = nestedArray.flat(1);
console.log(
`One level flattened array: ${oneFlat}`);

let twoFlat = nestedArray.flat(2);
```

```
console.log(
`Two level flattened array: ${twoFlat}`);

// No effect when depth is 3 or
// more since array is already
// flattened completely.
let threeFlat = nestedArray.flat(3);
console.log(
`Three levels flattened array: ${threeFlat}`);
```

Output

Zero levels flattened array: 1,2,3,,4,5,6

One level flattened array: 1,2,3,,4,5,6

Two level flattened array: 1,2,3,4,5,6

Three levels flattened array: 1,2,3,4,5,6

Note: For depth greater than 2, the array remains the same, since it is already flattened completely.

Example 3

We can also remove empty slots or empty values in an array by using the flat() method.

```
let arr = [1, 2, 3, , 4];
let newArr = arr.flat();
console.log(newArr);
```

Output

[1, 2, 3, 4]

forEach()

The arr.forEach() method calls the provided function once for each element of the array. The provided function may perform any kind of operation on the elements of the given array.

Syntax:

array.forEach(callback(element, index, arr), thisValue)

Parameters: This method accepts five parameters as mentioned above and described below:

- callback: This parameter holds the function to be called for each element of the array.
- element: The parameter holds the value of the elements being processed currently.
- index: This parameter is optional, it holds the index of the current value element in the array starting from 0.
- array: This parameter is optional, it holds the complete array on which Array.every is called.
- thisArg: This parameter is optional, it holds the context to be passed as this is to be used while executing the callback function. If the context is passed, it will be used like this for each invocation of the callback function, otherwise undefined is used as default.

Return value: The return value of this method is always undefined. This method may or may not change the original array provided as it depends upon the functionality of the argument function.

Example 1

In this example, the Array.forEach() method is used to copy every element from one array to another.

```
// JavaScript to illustrate forEach() method
function func() {

    // Original array
    const items = [12, 24, 36];
    const copy = [];
    items.forEach(function (item) {
        copy.push(item + item+2);
    });
    console.log(copy);
}
func();
```

Output

[26, 50, 74]

Example 2:

In this example, the method forEach() calculates the square of every element of the array.

```
// JavaScript to illustrate forEach() method
function func() {

   // Original array
   const items = [1, 29, 47];
   const copy = [];
   items.forEach(function (item) {
      copy.push(item * item);
   });
   console.log(copy);
}
func();
```

Output

[1, 841, 2209]

includes()

The Javascript array.includes() method is used to know whether a particular element is present in the array or not and accordingly, it returns true or false i.e, if the element is present, then it returns true otherwise false.

Syntax:

array.includes(searchElement, start)

Parameters: This method accepts two parameters as mentioned above and described below:

- searchElement: This parameter holds the element which will be searched.
- start: This parameter is optional and it holds the starting point of the array, where to begin the search the default value is 0.

Return Value: It returns a Boolean value i.e., either True or False.

Below is an example of the Array includes() method.

Example:

In this example, we will see if a value is included in an array or not using the Array includes() method.

```
let name = ['gfg', 'cse', 'geeks', 'portal'];

a = name.includes('gfg')
// Printing result of includes()
console.log(a);
```

Output:

true

The below examples illustrate the Array includes() method in JavaScript:

Example 1:

In this example, the method will search for element 2 in that array.

```
// Taking input as an array A
// having some elements.
let A = [1, 2, 3, 4, 5];

// includes() method is called to
// test whether the searching element
// is present in given array or not.
a = A.includes(2)

// Printing result of includes().
console.log(a);
```

Output:

True

Example 2:

In this example, the method will search for the element 'cat' in that array and returns false as 'cat' is not present in the array.

```
// Taking input as an array A
// having some elements.
let name = ['gfg', 'cse', 'geeks', 'portal'];

// includes() method is called to
// test whether the searching element
```

```
// is present in given array or not.
a = name.includes('cat')

// Printing result of includes()
console.log(a);
```

Output:

false

Array indexOf()

The JavaScript Array indexOf() Method is used to find the index of the first occurrence of the search element provided as the argument to the method. This method always compares the search element to the element present in the array using strict equality. Therefore, when the search element is NaN then it returns -1 because NaN values are never compared as equal.

Syntax:

array.indexOf(element, start)

Parameters: This method accepts two parameters as mentioned above and described below:

- element: This parameter holds the element whose index will be returned.
- start: This parameter is optional and it holds the starting point of the array, where to begin the search the default value is 0.

Return value: This method returns the index of the first occurrence of the element. If the element cannot be found in the array, then this method returns -1.

Example 1:

Below is an example of the Array indexOf() method.

```
let name = ['gfg', 'cse', 'geeks', 'portal'];
a = name.indexOf('gfg')
// Printing result of method
console.log(a)
```

Output

0

Example 2:

In this example, the method will be searched for element 2 in that array and return that element index.

```
// Taking input as an array A
// having some elements.
let A = [1, 2, 3, 4, 5];

// indexOf() method is called to
// test whether the searching element
// is present in given array or not.
a = A.indexOf(2)
```

```
// Printing result of method.
console.log(a);
```

Output

1

Example 3:

In this example, the method will be searched for element 9 in that array if not found then return -1.

```
// Taking input as an array A
// having some elements.
let name = ['gfg', 'cse', 'geeks', 'portal'];

// indexOf() method is called to
// test whether the searching element
// is present in given array or not.
a = name.indexOf('cat')

// Printing result of method
console.log(a);
```

Output

-1

Array join()

The JavaScript Array join() Method is used to join the elements of an array into a string. The elements of the string will be separated by a specified separator and its default value is a comma(,).

Syntax:

array.join(separator)

Parameters: This method accepts a single parameter as mentioned above and described below:

- separator: It is Optional i.e, it can be either used as a parameter or not. Its default value is a comma(,).
-

Example 1:

In this example, the function join() joins together the elements of the array into a string using '|'.

```
function func() {
    let a = [1, 2, 3, 4, 5, 6];
    console.log(a.join('|'));
}
func();
```

Output:

1|2|3|4|5|6

Example 2:

In this example, the function join() joins together the elements of the array into a string using ‘, ‘ since it is the default value.

```
let a = [1, 2, 3, 4, 5, 6];
console.log(a.join());
```

Output

1,2,3,4,5,6

Example 3:

In this example, the function join() joins together the elements of the array into a string using ‘ ‘ (empty string).

```
let a = [1, 2, 3, 4, 5, 6];
console.log(a.join(''));
```

Output:

123456

Array lastIndexOf()

The JavaScript Array lastIndexOf() Method is used to find the index of the last occurrence of the search element provided as the argument to the function.

Syntax:

array.lastIndexOf(element, start)

Parameters: This method accepts two parameters as mentioned above and described below:

- element: This parameter holds the element whose index will be returned.
- start: This parameter is optional and it holds the starting point of the array, where to begin the search the default value is 0.

Return value: This method returns the index of the last occurrence of the element. If the element cannot be found in the array, then this method returns -1.

Below is an example of the Array lastIndexOf() method.

Example:

```
let name = ['gfg', 'cse', 'geeks', 'portal'];

a = name.lastIndexOf('gfg')


// Printing result of method

console.log(a);
```

Output:

0

Example 1

In this example, the method will be searched for element 2 in that array and return that element index.

```
// Taking input as an array A
// having some elements.
let A = [1, 2, 3, 4, 5];

// lastIndexOf() method is called to
// test whether the searching element
// is present in given array or not.
a = A.lastIndexOf(2)

// Printing result of method.
console.log(a);
```

Output:

1

Example 2:

In this example, the method will search for the element 9 in that array if not found then return -1.

```
// Taking input as an array A
// having some elements.
let name = ['gfg', 'cse', 'geeks', 'portal'];
```

```
// lastIndexOf() method is called to
// test whether the searching element
// is present in given array or not.
a = name.lastIndexOf('cat')

// Printing result of method
console.log(a);
```

Output:

-1

Array pop()

The JavaScript Array pop() Method is used to remove the last element of the array and also returns the removed element. This function decreases the length of the array.

Syntax:

arr.pop()

Parameters: This method does not accept any parameter. Return value This method returns the removed element array. If the array is empty, then this function returns undefined.

Below is an example of the Array pop() method.

Example 1:

```
function func() {
    let arr = ['GFG', 'gfg', 'g4g', 'GeeksforGeeks'];

    // Popping the last element from the array
    console.log(arr.pop());
}
func();
```

Output:

GeeksforGeeks

Example 2:

In this example, the pop() method removes the last element from the array, which is 4, and returns it.

```
function func() {
    let arr = [34, 234, 567, 4];

    // Popping the last element from the array
```

```
    let popped = arr.pop();
    console.log(popped);
    console.log(arr);
}
func();
```

Output:

4

34,234,567

Example 3

In this example, the function pop() tries to extract the last element of the array but since the array is empty therefore it returns undefined as the answer.

```
function func() {
    let arr = [];

    // popping the last element
    let popped = arr.pop();
    console.log(popped);
}
```

```
func();
```

Output:

undefined

Array push()

The JavaScript Array push() Method is used to add one or more values to the end of the array. This method changes the length of the array by the number of elements added to the array.

Syntax:

arr.push(element0, element1, ... , elementN)

Parameters: This method contains as many numbers of parameters as the number of elements to be inserted into the array.
Return value: This method returns the new length of the array after inserting the arguments into the array.
Below is an example of the Array push() method.

Example 1:

```
function func() {
    let arr = ['GFG', 'gfg', 'g4g'];

    // Pushing the element into the array
```

```
    arr.push('GeeksforGeeks');
    console.log(arr);

}
func();
```

Output:

GFG,gfg,g4g,GeeksforGeeks

Example 2:

In this example, the function push() adds the numbers to the end of the array.

```
function func() {
    // Original array
    let arr = [34, 234, 567, 4];

    // Pushing the elements
    console.log(arr.push(23, 45, 56));
    console.log(arr);
}
func();
```

Output:

7

34,234,567,4,23,45,56

Example 3:

In this example, the function push() adds the objects to the end of the array.

```
function func() {
    // Original array
    let arr = [34, 234, 567, 4];

    // Pushing the elements
    console.log(arr.push('jacob', true, 23.45));
    console.log(arr);
}
func();
```

Output:

7

34,234,567,4,jacob,true,23.45

Array reverse()

JavaScript Array.reverse() Method is used for the in-place reversal of the array. This method reverses the order of the array. The first element of the array becomes the last element and the last one becomes the first. It mutates the original array. We can also use the Array.toReversed() which returns the new array with the elements in the reverse order.

Syntax:

arr.reverse()

Parameters: This method does not accept any parameter

Return value: This method returns the reference of the reversed original array.

Example 1:

The below examples illustrate the JavaScript Array reverse() Method

```
function func() {
    // Original Array
    let arr = ['Portal', 'Science',
        'Computer', 'GeeksforGeeks'];
    console.log(arr);

    // Reversed array
    let new_arr = arr.reverse();
    console.log(new_arr);
}
func();
```

Output

['Portal', 'Science', 'Computer', 'GeeksforGeeks']

['GeeksforGeeks', 'Computer', 'Science', 'Portal']

Example 2:

In this example, the reverse() method reverses the sequence of the array elements of arr.

```
function func() {

    // Original Array
    let arr = [34, 234, 567, 4];
    console.log(arr);

    // Reversed array
    let new_arr = arr.reverse();
    console.log(new_arr);
}
func();
```

Output

[34, 234, 567, 4]

[4, 567, 234, 34]

Array shift()

The JavaScript Array shift() Method removes the first element of the array thus reducing the size of the original array by 1.

Syntax:

arr.shift()

Parameters: This method does not accept any parameter.
Return value: This function returns the removed first element of the array. If the array is empty then this function returns undefined.
Note: This function can also be used with other javascript objects that behave like the array.
Below is an example of the Array shift() method.

Example 1:

In this example, the shift() method removes the first string element of the array, therefore it returns GFG.

```
function func() {
    // Original array
    let array = ["GFG", "Geeks", "for", "Geeks"];

    // Checking for condition in array
    let value = array.shift();

    console.log(value);
    console.log(array);
}
func();
```

Output:

GFG

Geeks, for, Geeks

Example 2:

In this example, the shift() method removes the first element of the array, therefore it returns 34.

```
function func() {

    // Original array
    let array = [34, 234, 567, 4];

    // Checking for condition in array
    let value = array.shift();

    console.log(value);
    console.log(array);
}
func();
```

Output:

34,234,567,4

Example 3:

In this example, the shift() method tries to remove the first element of the array, but the array is empty, therefore it returns undefined.

```
function func() {

    // Original array
    let array = [];

    // Checking for condition in array
    let value = array.shift();

    console.log(value);
    console.log(array);
}
func();
```

Output:

undefined

Array slice()

The Javascript arr.slice() method returns a new array containing a portion of the array on which it is implemented. The original remains unchanged.

Syntax:

arr.slice(begin, end)

Parameters: This method accepts two parameters as mentioned above and described below:

- begin: This parameter defines the starting index from where the portion is to be extracted. If this argument is missing then the method takes begin as 0 as it is the default start value.
- end: This parameter is the index up to which the portion is to be extracted (excluding the end index). If this argument is not defined then the array till the end is extracted as it is the default end value If the end value is greater than the length of the array, then the end value changes to the length of the array.

Return value: This method returns a new array containing some portion of the original array.

Below is an example of the Array slice() method.

Example 1:

In this example, the slice() method extracts the array from the given array starting from index 2 and including all the elements less than index 4.

```
function func() {
    // Original Array
    let arr = [23, 56, 87, 32, 75, 13];
    // Extracted array
    let new_arr = arr.slice(2, 4);
    console.log(arr);
```

```
    console.log(new_arr);
}
func();
```

Output:

[23,56,87,32,75,13]

[87,32]

Example 2

In this example, the slice() method extracts the entire array from the given string and returns it as the answer, Since no arguments were passed to it.

```
function func() {
    //Original Array
    let arr = [23, 56, 87, 32, 75, 13];
    //Extracted array
    let new_arr = arr.slice();
    console.log(arr);
    console.log(new_arr);
}
func();
```

Output:

[23,56,87,32,75,13]

[23,56,87,32,75,13]

Example 2:

In this example, the slice() method extracts the array starting from index 2 till the end of the array and returns it as the answer.

```
function func() {
    //Original Array
    let arr = [23, 56, 87, 32, 75, 13];
    //Extracted array
    let new_arr = arr.slice(2);
    console.log(arr);
    console.log(new_arr);
}
func();
```

Output:

[23,56,87,32,75,13]

[87,32,75,13]

Array sort()

JavaScript Array.sort() Method is used to sort the array in place in a given order according to the compare() function. If the method is omitted then the array is sorted in ascending order.

Syntax:

arr.sort(compareFunction)

Parameters: This method accepts a single parameter as mentioned above and described below:

- compareFunction: This parameter is used to sort the elements according to different attributes and in a different order.
- compareFunction(a,b) < 0
- compareFunction(a,b) > 0
- compareFunction(a,b) = 0

Return value: This method returns the reference of the sorted original array.

Example 1:

```
// JavaScript to illustrate sort() function
function func() {

    // Original string
    let arr = ["Geeks", "for", "Geeks"]

    console.log(arr);
    // Sorting the array
    console.log(arr.sort());
}
func();
```

Output:

Geeks,for,Geeks

Geeks,Geeks,for

Example 2

In this example, the sort() method arranges the elements of the array in ascending order.

```
// JavaScript to illustrate sort() function
function func() {
    //Original string
    let arr = [2, 5, 8, 1, 4]

    //Sorting the array
    console.log(arr.sort());
    console.log(arr);
}
func();
```

Output:

1,2,4,5,8

1,2,4,5,8

Example 3:

In this example, the sort() method the elements of the array are sorted according to the function applied to each element.

```
// JavaScript to illustrate sort() function
```

```
function func() {

   // Original array
   let arr = [2, 5, 8, 1, 4];
   console.log(arr.sort(function (a, b) {
      return a + 2 * b;
   }));
   console.log(arr);
}
func();
```

Output:

2,5,8,1,4

2,5,8,1,4

Example 3:

In this example, we use the sort() method on the array of numbers & observe some unexpected behavior.

```
let numbers = [20, 5.2, -120, 100, 30, 0]
console.log(numbers.sort())
```

Output:

-120,0,100,20,30,5.2

Array splice()

The JavaScript Array splice() Method is an inbuilt method in JavaScript that is used to modify the contents of an array by removing the existing elements and/or by adding new elements.

Syntax:

Array.splice(index, remove_count, item_list)

Parameter: This method accepts many parameters some of which are described below:

- index: It is a required parameter. This parameter is the index from which the modification of the array starts (with the origin at 0). This can be negative also, which begins after many elements counting from the end.
- remove_count: The number of elements to be removed from the starting index.
- items_list: The list of new items separated by a comma operator that is to be inserted from the starting index.

Return Value: While it mutates the original array in place, still it returns the list of removed items. In case there is no removed array it returns an empty array.

Below are examples of the Array splice() method:

Example 1:

```
let webDvlop = ["HTML", "CSS", "JS", "Bootstrap"];

console.log(webDvlop);

// Add 'React_Native' and 'Php' after removing 'JS'.
let removed = webDvlop.splice(2, 1, 'PHP',
'React_Native')

console.log(webDvlop);
console.log(removed);

// No Removing only Insertion from 2nd
// index from the ending
webDvlop.splice(-2, 0, 'React')
console.log(webDvlop)
```

Output:

HTML,CSS,JS,Bootstrap

HTML,CSS,PHP,React_Native,Bootstrap

JS

HTML,CSS,PHP,React,React_Native,Bootstrap

Example 2:

Here is another example of the Array splice() method.

```
let languages = ['C++', 'Java', 'Html', 'Python', 'C'];

console.log(languages);

// Add 'Julia' and 'Php' after removing 'Html'.
let removed = languages.splice(2, 1, 'Julia', 'Php')

console.log(languages);
console.log(removed);

// No Removing only Insertion from 2nd index from the ending
languages.splice(-2, 0, 'Pascal')
console.log(languages)
```

Output:

C++,Java,Html,Python,C

C++,Java,Julia,Php,Python,C

Html

C++,Java,Julia,Php,Pascal,Python,C

Array toString()

The JavaScript Array toString() Method returns the string representation of the array elements

Syntax:

arr.toString()

Parameters: This method does not accept any parameter.

Return value:

- The method returns the string representation of the array elements.
- If the array is empty, then it returns an empty string.
- The original array is not changed by this method

Below is an example of the Array toString() method.

Example 1:

```
// JavaScript to illustrate toString() method
function func() {

    // Original array
    let arr = ["Geeks", "for", "Geeks"];

    // Creating a string
    let str = arr.toString();
    console.log(str);
}
func();
```

Output:

Geeks,for,Geeks

Example 2:

In this example, the toString() method creates a string consisting of array elements.

```
// JavaScript to illustrate toString() method
function func() {
    // Original array
    let arr = [2, 5, 8, 1, 4];

    // Creating a string
    let str = arr.toString();
    console.log(str);
}
func()
```

Output:

2,5,8,1,4

Array unshift()

The JavaScript Array unshift() Method is used to add one or more elements to the beginning of the given array. This

function increases the length of the existing array by the number of elements added to the array.

Syntax:

array.unshift(element1, element2, ..., elementX)

Parameters: This method accepts a single parameter.

- element: This parameter element is to be added at the beginning of the array.

Return value: This function returns the new length of the array after inserting the arguments at the beginning of the array.

Example 1:

Below is an example of the Array unshift() method.

```
function func() {
    // Original array
    let array = ["GFG", "Geeks", "for", "Geeks"];

    // Checking for condition in array
    let value = array.unshift("GeeksforGeeks");
    console.log(value);
    console.log(array);
}
func();
```

Output

5

['GeeksforGeeks', 'GFG', 'Geeks', 'for', 'Geeks']

Example 2:

In this example, the function unshift() adds 28 and 65 to the front of the array.

```
function func() {
    let arr = [23, 76, 19, 94];
    // Adding elements to the front of the array
    console.log(arr.unshift(28, 65));
    console.log(arr);
}
func();
```

Output

6

[28, 65, 23, 76, 19, 94]

Example 3: In this example, the unshift() method tries to add the element of the array, but the array is empty therefore it adds the value in the empty array.

```
function func() {
    let arr = [];
    console.log(arr.unshift(1));
}
```

```
func();
```

Output

1