Arrays

Arrays are an essential data structure in JavaScript that allow you to store multiple values in a single variable. An array is an ordered collection of elements, and each element can be accessed by its index, starting from 0. JavaScript arrays can hold values of different data types, such as numbers, strings, objects, or even other arrays.

Creating an Array:

You can create an array using square brackets [] and separating the elements with commas.

```
// Empty Array
const emptyArray = [];

// Array with elements
const numbers = [1, 2, 3, 4, 5];
const fruits = ['apple', 'banana', 'orange'];
const mixedArray = [1, 'hello', true, { name: 'John' }];
```

Array Method

The push() method

It is a built-in array method that adds one or more elements to the end of an array. It is commonly used to append new elements to the existing array.

```
const fruits = ['apple', 'banana'];

fruits.push('orange');
console.log(fruits); // Output: ["apple", "banana", "orange"]

fruits.push('grape', 'watermelon');
console.log(fruits); // Output: ["apple", "banana", "orange",
"grape", "watermelon"]
```

The pop() method

It is a built-in array method that removes the last element from an array and returns that element. It effectively "pops" the last element off the array. This method modifies the original array and reduces its length by one.

```
const numbers = [1, 2, 3, 4, 5];

const lastElement = numbers.pop();
console.log(lastElement); // Output: 5
console.log(numbers); // Output: [1, 2, 3, 4]
```

The toString()

The toString() method is used to convert an array (or any other object) to a string representation

```
const fruits = ['apple', 'banana', 'orange'];
const numbers = [1, 2, 3, 4, 5];

const fruitsAsString = fruits.toString();
const numbersAsString = numbers.toString();

console.log(fruitsAsString); // Output: "apple,banana,orange"
console.log(numbersAsString); // Output: "1,2,3,4,5"
```

split() method

Split is used to split a string into an array of substrings based on a specified separator. It takes the separator as an argument, and the string is split into an array wherever the separator occurs. If no separator is provided, the entire string becomes the only element of the resulting array.

```
const fruitsString = "apple,banana,orange";
const numbersString = "1-2-3-4-5";

const fruitsArray = fruitsString.split(',');
console.log(fruitsArray); // Output: ["apple", "banana",
"orange"]

const numbersArray = numbersString.split('-');
console.log(numbersArray); // Output: ["1", "2", "3", "4", "5"]
```

join() method:

The join() method is used to join all the elements of an array into a single string. It takes an optional separator as an argument, which specifies how the elements should be separated in the resulting string. If no separator is provided, the elements are joined with a comma by default.

```
const fruits = ['apple', 'banana', 'orange'];
const numbers = [1, 2, 3, 4, 5];

const fruitsString = fruits.join(); // Joins with default
separator (comma)
console.log(fruitsString); // Output: "apple,banana,orange"

const numbersString = numbers.join('-'); // Joins with a custom
separator (hyphen)
console.log(numbersString); // Output: "1-2-3-4-5"
```

The shift() method

It is a built-in array method that removes the first element from an array and returns that element. It effectively "shifts" all remaining elements in the array one position to the left. This method modifies the original array and reduces its length by one.

```
const fruits = ['apple', 'banana', 'orange'];

const firstFruit = fruits.shift();
console.log(firstFruit); // Output: "apple"
console.log(fruits); // Output: ["banana", "orange"]
```

Unshift() method

In JavaScript, the unshift() method is used to add one or more elements to the beginning of an array. It modifies the original array and returns the new length of the array.

```
let fruits = ['apple', 'banana', 'orange'];
console.log(fruits); // Output: ['apple', 'banana',
'orange']

fruits.unshift('mango');
console.log(fruits); // Output: ['mango', 'apple',
'banana', 'orange']

fruits.unshift('pear', 'grape');
console.log(fruits); // Output: ['pear', 'grape',
'mango', 'apple', 'banana', 'orange']
```

Slice:

In JavaScript, the slice() method is used to extract a portion of an array and return it as a new array. It doesn't modify the original array, but instead, it creates a shallow copy of the selected elements.

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

const sliced1 = numbers.slice(2, 6);
console.log(sliced1); // Output: [3, 4, 5, 6]

const sliced2 = numbers.slice(4);
console.log(sliced2); // Output: [5, 6, 7, 8, 9, 10]

const sliced3 = numbers.slice(-3);
console.log(sliced3); // Output: [8, 9, 10]

const sliced4 = numbers.slice(1, -2);
console.log(sliced4); // Output: [2, 3, 4, 5, 6, 7, 8]

const original = ['a', 'b', 'c', 'd', 'e'];
const shallowCopy = original.slice();
console.log(shallowCopy); // Output: ['a', 'b', 'c',
'd', 'e']
```

Splice()

In JavaScript, the splice() method is used to change the contents of an array by removing or replacing existing elements and/or adding new elements. It modifies the original array in place and returns an array containing the removed elements.

```
const numbers = [1, 2, 3, 4, 5];

// Removing elements
const removed1 = numbers.splice(2, 2);
console.log(removed1); // Output: [3, 4]
console.log(numbers); // Output: [1, 2, 5]

// Removing and replacing elements
const removed2 = numbers.splice(1, 1, 'a', 'b', 'c');
console.log(removed2); // Output: [2]
console.log(numbers); // Output: [1, 'a', 'b', 'c', 5]

// Adding elements without removing any
numbers.splice(2, 0, 'x', 'y');
console.log(numbers); // Output: [1, 'a', 'x', 'y',
'b', 'c', 5]
```

Every()

In JavaScript, the every() method is used to check if all elements in an array satisfy a given condition. It tests the array elements against a provided callback function and returns true if all elements pass the test, otherwise it returns false.

```
const numbers = [1, 2, 3, 4, 5];

// Check if all elements are greater than 0
const result1 = numbers.every(function (element) {
  return element > 0;
});
console.log(result1); // Output: true

// Check if all elements are even
const result2 = numbers.every(function (element) {
  return element % 2 === 0;
});
console.log(result2); // Output: false
```

find()

In JavaScript, the find() method is used to retrieve the first element in an array that satisfies a given condition. It iterates over the array elements and returns the first element for which the provided callback function returns true. If no element satisfies the condition, undefined is returned.

```
const numbers = [1, 2, 3, 4, 5];

// Find the first even number
const result1 = numbers.find(function (element) {
  return element % 2 === 0;
});
console.log(result1); // Output: 2

// Find the first number greater than 3
const result2 = numbers.find(function (element) {
  return element > 3;
});
console.log(result2); // Output: 4

// Find the first element that satisfies a custom
condition
const result3 = numbers.find(function (element) {
  return element % 2 === 0 && element > 2;
});
console.log(result3); // Output: 4
```

delete()

In JavaScript, the find() method is used to retrieve the first element in an array that satisfies a given condition. It iterates over the array elements and returns the first element for which the provided callback function returns true. If no element satisfies the condition, undefined is returned.

```
const numbers = [1, 2, 3, 4, 5];

// Find the first even number
const result1 = numbers.find(function (element) {
  return element % 2 === 0;
});
console.log(result1); // Output: 2

// Find the first number greater than 3
const result2 = numbers.find(function (element) {
  return element > 3;
});
console.log(result2); // Output: 4

// Find the first element that satisfies a custom
condition
const result3 = numbers.find(function (element) {
  return element % 2 === 0 && element > 2;
});
console.log(result3); // Output: 4
```

Sort()

The sort() method is a built-in array method that is used to sort the elements of an array in place and return the sorted array. By default, the sort() method sorts elements alphabetically as strings. However, this default sorting behavior may not be appropriate for all scenarios, especially when sorting numbers or custom objects.

```
const fruits = ['banana', 'apple', 'orange', 'grape'];

fruits.sort();
console.log(fruits); // Output: ["apple", "banana", "grape",
"orange"]
```

Sorting Numbers:

By default, the sort() method sorts elements as strings, which may lead to unexpected results when sorting numbers. To correctly sort numbers, you need to provide a compare function (also known as a comparator) as an argument to the sort() method.

```
const numbers = [4, 1, 8, 3, 6];

numbers.sort((a, b) => a - b);
console.log(numbers); // Output: [1, 3, 4, 6, 8]
```