Array Methods

Concat:

The Concat () methods creates a new array by joining (concatenating) existing arrays.

Syntax:

array. Concat (value1, value2, ..., value)

Examples:

Const arr1= [‘Apple’, ‘Kiwi’, ‘Watermelon’, ‘Pineapple’]

Const arr2= [1,2,3,4,5,6,7]

Let Concat array=arr1.concat(arr2);

Console.log (Concat array)

Output

['Apple', 'Kiwi', 'Watermelon', 'pineapple', 1, 2, 3, 4, 5, 6, 7]

Every:

The Every () method is used primarily with arrays. It tests whether all elements in the array satisfy the provided testing function. It returns a boolean value (true or false).

Syntax:

array. every (function (current Value, index, array), this Value)

Examples:

```
let words = ["apple", "banana", "orange", "grape"];

let all Fruits words = words. Every (word => word. Length > 3);

console.log (all Fruits words);
```

Output

All fruits words are longer than 3 characters? true

Fill:

The **Fill ()** method is used primarily with arrays in JavaScript. It fills all the elements of an array from a start index to an end index with a static value.

Syntax:

```
array. Fill (value, start, end)
```

Examples:

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

array. Fill (0, 2, 4);

console.log(array);
```

Output

[1,2,3,4,5]

[1, 2, 0, 0, 5]

Find:

The **Find ()** method is used primarily with arrays in JavaScript. It returns the value of the first element in the array that satisfies the provided testing function.

Syntax:

array. Find (function (current Value, index, array), this ary)

Examples:

```
let numbers = [1, 3, 5, 8, 10];

let first Even = numbers. Find (num => num % 2 === 0);

console.log (first Even);
```

Output

First even number: 8

FindIndex

The **FindIndex()** method is used primarily with arrays in JavaScript.It returns the index of the first element in the array that satisfies the provided testing function. If no elements satisfy the function, -1 is returned.

Syntax:

array.findIndex(function(currentValue, index, array), this ary)

Examples:

```
let numbers = [1, 3, 5, 8, 10];

let index = numbers.findIndex(num => num % 2 === 0);

console.log( index);
```

Output:

Index of first even number: 3

Flat

The **Flat()** method is used primarily with arrays in JavaScript. It flattens nested arrays recursively up to a specified depth. it flattens nested arrays with a depth of 1.

Syntax:

```
array.flat(depth)
```

Examples:

```
const myArr = [[1,2],[3,4],[5,6]];
const newArr = myArr.flat();

console.log(newarr)
```

output:

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

Includes

The **Includes()** method is used primarily with arrays in JavaScript. It checks whether an array includes a certain value, returning true or false as appropriate

Syntax

```
array.includes(searchElement, fromIndex)
```

examples:

```
let array = ['apple', 'banana', 'cherry', 'date'];
let includesCherry = array.includes('cherry', 2);
console.log(includesCherry);
```

Output:

Array includes 'cherry' from index 2: true

Indexof

The **IndexOf()** method is used primarily with arrays in JavaScript. It returns the first index at which a given element can be found in the array, or -1 if it is not present.

Syntax:

```
indexOf(searchValue, fromIndex)
```

Examples:

```
let array = ['apple', 'banana', 'cherry', 'date'];
let indexOfCherry = array.indexOf('cherry', 2);
console.log(indexOfCherry);
```

Output:

Index of 'cherry' starting from index 2: 2

Join

The **Join()** method converts all elements of an array into a string and concatenates them, optionally separating

each element with a specified separator string. If no separator is provided, a comma is used by default.

Syntax:

array.join(separator)

Examples:

let arrayfruits = ['apple', 'banana', 'cherry'];

let result = arrayfruits.join(“and”);

console.log(result);

Output: "apple and banana and cherry"

Lastindexof

The **Lastindexof()** method of string values searches this string and returns the index of the last occurrence of the specified substring. It takes an optional starting position and returns the last occurrence of the specified substring at an index less than or equal to the specified number.

Syntax:

array.lastIndexOf(searchElement, fromIndex)

Examples:

let array = ['apple', 'banana', 'cherry', 'date', 'banana'];

let lastIndexof = array.lastIndexOf('banana', 3);

console.log(lastIndexof);

Output:

Last index of 'banana' from index 3: -1

Pop

The **pop()** method is used with arrays in remove the last element from an array and return that element.

Syntax:

array.pop()

Examples:

const fruit = ['Kiwi', 'Apple', 'Orange', 'Mango'];

const remove fruits = fruit. Pop ();

console.log (remove fruits);

output:

Mango

Push

The **push()** is an array method that adds one or more elements to the end of an array and returns the new length of the array.

Syntax:

Array.push()

Example:

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

Output: [1, 2, 3, 4]

Reverse

The reverse() method reverses the order of the elements in an array.

Syntax:

```
array.reverse()
```

Example

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

Output: [4,3,2,1]

Unshift

The unshift() method adds new elements to the beginning of an array.

Syntax:

```
Array.unshift()
```

Example:

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
const added= fruits.unshift("Lemon", "Pineapple");
console.log(added);
output:
["Lemon", "Pineapple", "Banana", "Orange", "Apple", "Mango"];
```

shift

The shift() method removes the first item of an array.

Syntax:

```
Array.shift()
```

Example

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
const remove= fruits.shift();
console.log(remove);
output:
[ "Orange", "Apple", "Mango"]
```

Slice

The slice() method returns a shallow copy of a portion of an array into a new array object selected from start to end

Syntax:

```
array.slice(start, end)
```

Example:

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

const citrus = fruits.slice(1, 3);

console.log()

output:

[ 2,3 ]
```

Some

The some() method checks if any array elements pass a test provided as a callback function, returning true if any do and false if none do.

Syntax:

```
arr.some(callback(element,index,array),thisAry);
```

Example:

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

const GreaterThanTen = numbers.some(number =>
number > 10);

console.log(GreaterThanTen);

 Output: false
```

Sort

The Sort() method arranges the elements of an array in place and returns the sorted array.

Syntax

```
arr.sort();
```

Example:

```
let sorting = ["yellow", "blue", "green"]
console.log(sorting.sort());
```

output:

```
['blue', 'green', 'yellow']
```

Splice

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

Syntax:

```
Array.splice( )
```

Example:

```
const colors = ["white", "Green","Yellow","Red"];
colors.splice(1, 2, "Blue");
console.log(colors);
```

Output:

```
['white', 'Blue', 'Red']
```

Tostring

The Tostring()method converts array in to string, and returns the results.

Syntax:

```
array.toString()
```

Example:

```
const fruits = ["Banana", "Orange", "Apple",
"Mango"];

let text = fruits.toString();

console.log( );
```

output:

```
[ Banana,Orange,Apple,Mango ]
```

Filter

The filter() is an array method that creates a new array with all elements that pass the test implemented by the provided function.

Syntax:

```
const newArray = array.filter(callback(element,
index, array), thisArg)
```

Example:

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

const evenNumbers = numbers.filter(number =>
number % 2 === 0);

console.log(evenNumbers);
```

Output:

```
[2, 4]
```

Reduce

The reduce() method is used to "reduce" an array of values down to a single value. It does this by iterating over each element in the array and applying a callback function (also known as a "reducer" function) to each one. The reducer function takes the accumulated result from the previous iteration and the current element as arguments, and returns a new accumulated result.

syntax:

```
array.reduce(callback(accumulator, currentValue, index, array), initialValue);
```

Example:

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

const sum = numbers.reduce((abc,cab)=>{

return abc+cab },0);
```

Output : [15]

Map

The map( ) method it creates a new array by applying a provided function to each element of the original array.

Syntax:

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

Example:

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

const doubled = numbers.map(num => num * 2);

console.log(doubled);
```

Output: [2, 4, 6, 8, 10]

Foreach

The forEach( ) method that allows you to iterate over an array and perform an operation on each element of the array.

Syntax:

```
array.forEach(function(currentValue, index, array) {

});
```

Example:

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

numbers.forEach(function(numbered){

  console.log(numbered * 2);

});
```

Output:

[2 ,6,10,16,20]

String Methods

Concat

The Concat () method defines that join the two or more strings into one string.

Syntax:

```
string.Concat();
```

Example:

```
let str1 = "hello";

let str2 = "world";

let str3 = str1.concat(str2);

console.log(str3);
```

output:

HelloWorld

Includes

The includes() string method defines to check whether is it Boolean (True/ False).

Syntax:

```
String.includes()
```

Example:

```
let str = 'Hello World';

console.log(str.includes('World'));
```

output:

True

IndexOf

The Indexof() method defines returns the position of the first occurrence of specified character in a string.

Syntax:

1.String.indexof(string str)

2. string.indexof(string str, int from index)

Example:

```
let word = "A,B,C,D"
console.log(word.indexOf('D', 4));
```

output: 6

LastIndexOf

The lastindexof() method defines that it returns the position of the last occurrence of specified a character into a string.

Syntax:

String. Lastindexof(search values, from index)

Example:

```
let str = "hello world hello";
```

```
console.log(str.lastIndexOf("hello"));
```

Output: 12

PadEnd

The PadEnd() methods a string with another string until it reaches a given length.

Syntax:

```
Str.padEnd(Length,string)
```

Example:

```
let name = "john";
console.log(name.padEnd(8, '**'));
```

output: john****

PadStart

The pad start () methods a string from the start.

Syntax:

```
Str.Padstart(length,string)
```

Example:

```
let colrs = "Red";
console.log(colrs.padStart(6, '--'));
```

output:

[---Red]

Repeat

The repeat () methods returns a string with a number of copies of a string.

Syntax:

Str.repeat(count)

Examples:

let str4 = 'Hello';

console.log(str1.repeat(3));

output:

HelloHelloHello

Replace

The replace() method returns a new string with the values replaced.

Syntax:

String.replace(search value, new value)

Example:

Let sentence= 'i can read the sentence';

Console.log(sentence.replace('i', 'I');

Output:

I can read the sentence

Search

The Search() method matches a string against a regular epression.

Syntax:

String.search(regexp)

Example:

Let sentence='she is my friend.'

Console.log(sentence.search('ismy')

Output: -1

Slice

The slice() method returns the extracted part in a new string.

Syntax:

String.slice(start, end)

Example:

Let first='hello world!';

Let result=first.slice(0,5)

Console.log(result)

Output: Hello

Split

The split() method split a string into an array of substring.

Syntax:

String.split(separator, limit)

Example:

```
Const text='How was your day'
Const result=text.split('w')
Console.log(result)
```

Output:

```
[ 'Ho', ' ' , 'as your day' ]
```

startsWith

the startsWith() method returns Boolean if a string starts with a specified string.

Syntax:

String.startWith()

Example:

```
Let world='HelloWorldHellowelcome'
Let result=world.startswith('Hello',1)
```

Output: -1

Substr

The substr() method begins at a specified position, and returns a specified number of characters.

Syntax:

String.substr(start, length)

Example:

```
Let text ='Hello World'
Let result=text.substr(1,4)
Console.log(result)
```

Output: ello

Substring

The substring() method remove the characters between two position,from a string and returns the string.

Sytnax:

```
String.substring(start, end)
```

Example:

```
Let text='Hello world'
Let result=text.substring(1,4)
Console.log(result)
```

Output:

Ell

toLowerCase

the toLowerCase() method converts the string into lowercase letter.

Syntax:

String.tolowercase()

Example:

```
Let text ='HELLO WORLD!'
Let result=text.tolowercase();
Console.log(result)
```

Output:

Hello world!

ToUpperCase

the toUpperCase() method converts the string into uppercase letter.

Syntax:

String.touppercase()

Example:

```
Let text ='hello world'
Let result=text.touppercase();
Console.log(result)
```

Output:

HELLO WORLD!

Trim

The trim() method removes the whitespace from the both sides of a string.

Syntax:

String.trim()

Example:

```
Let text= 'Quite please'
Let result=text.trim()
Console.log(result)
```

Output:

Quite please

trimEnd (or trimRight)

The trimend() method removes the whitespaces from the end of a string.

Syntax:

String.trimend()

Example:

```
Let message=' welcome to the world'
Let result=message.trimend()
Console.log(result)
```

Output:

'welcome to the world '

trimStart (or trimLeft)

The trimstart()method removes the whitespace from the starting/beginning of a string.

Syntax:

String.trimstart()

Example:

```
Let message='welcome to the world'
Let result= message.trimstart()
Console.log(result)
```

Output:

' welcome to the world'

charAt

The charAt() method retruns the character a specified position in a string.

Syntax:

String.charAt(i) [Index]

Example:

```
Let sentence= 'the quick brown fox'
```

```
Let result=sentence.charAt
Console.log(sentence(10));
Output:   b
```

charCodeAt

The charCodeAt () method returns the Unicode of the character at a specified position in string.

Syntax:

```
stringcharcodeAt()
```

Example:

```
 Let str= 'the quick brown fox'
Let result= str.charCodeAt((10))
Console.log(result)
Output:
98  (Unicode for letter B)
```