Array Methods Lists...

Lists Array Methods

- concat()
- indexOf()
- join()
- lastIndexOf()
- pop()
- push()
- reverse()
- shift()
- slice()
- sort()
- splice()
- toString()
- unshift()
- valueOf()

concat()

The concat() method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array.

Syntax

```
array1.concat(arr2, arr3,arrX)
```

Example

```
const array = ['a', 'b', 'c'];
const array = ['d', 'e', 'f'];
const array = array.concat(array);
console.log(array3);
//["a", "b", "c", "d", "e", "f"]
```

indexOf()

The indexOf() method returns the first index at which a given element can be found in the array

Syntax

```
string.indexOf(searchvalue, start)
```

Example

```
const beasts = ['sahil', 'singh'];

console.log(beasts.indexOf('sahil'));
//output: 1
```

join()

The join() method returns an array as a string.
The join() method does not change the original array.

Syntax

```
array.join(separator)
```

Example

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

let text = fruits.join(" and ");
//Banana and Orange and Apple and Mango
```

lastIndexOf()

The lastIndexOf() method returns the index (position) of the last occurrence of a specified value in a string.

Syntax

```
string.lastIndexOf(searchvalue, start)
```

Example

```
let text = "Hello planet earth, you
are a great planet.";


let result=text.lastIndexOf("planet");
//36
```

pop()

The pop() method removes the last element from an array and returns that element.

Syntax

```
array.pop()
```

Example

```
const plants = [ 'kale', 'tomato'];


console.log(plants.pop());
// expected output: "tomato"
```

push()

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

Syntax

```
array.push(item1, item2, ..., itemX)
```

Example

```
const fruits = ["Banana","Orange",];
fruits.push("Kiwi", "Lemon");


// output: Banana,Orange,Kiwi,Lemon
```

reverse()

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

Syntax

```
array.reverse()
```

Example

```
const reversed = array1.reverse();

console.log(reversed);
//output: ["three", "two", "one"]
```

@code.singh @sahil singh

shift()

The **shift()** method removes the first element from an array and returns that removed element. This method changes the length of the array.

Syntax

```
array.shift()
```

Example

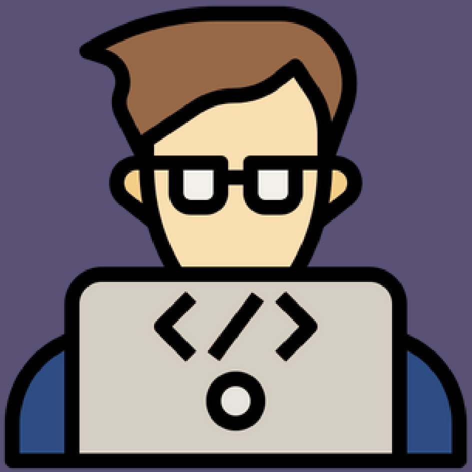
```
const fruits = ["Banana", "Orange",
"Apple", "Mango"];

fruits.shift();
//Orange,Apple,Mango
```

slice()

The slice() method returns selected elements in an array, as a new array. The slice() method selects from a given start, up to a given end.

Syntax

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



Example

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

const citrus = fruits.slice(1, 3);
//Orange,Lemon
```

sort()

The sort() sorts the elements of an array.
The sort() overwrites the original array.

Syntax

```
array.sort(compareFunction)
```

Example

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

fruits.sort();
//Apple,Banana,Mango,Orange
```

splice()

The splice() method adds and/or removes array elements.
The splice() method overwrites the original array.

Syntax

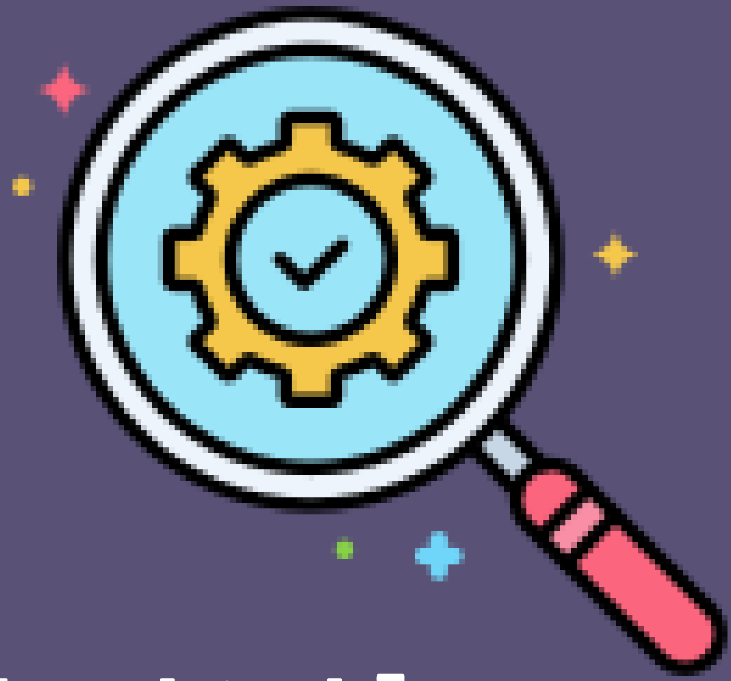
```
array.splice(index, howmany,.. itemX)
```

Example

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

fruits.splice(2, 0, "Lemon","Kiwi");
//Banana,Orange,Lemon,Kiwi,Apple,Mango
```

toString()

The toString() method returns a string representing the specified array and its elements.



Syntax

```
array.toString()
```

Example

```
const array1 = [1, 2, 'a', '1a'];

console.log(array1.toString());
// expected output: "1,2,a,1a"
```

unshift()

The **unshift()** method adds one or more elements to the beginning of an array and returns the new length of the array.

Syntax

```
array.unshift(item1, item2,..itemX)
```

Example

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

fruits.unshift("Lemon", "Pineapple");
//Lemon,Pineapple,Banana,Orange,Apple,
Mango
```

indexOf()

The indexOf() method returns the first index (position) of a specified value. The indexOf() method starts at a specified index and searches from left to right.

Syntax

```
array.indexOf(item, start)
```

Example

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

let index = fruits.indexOf("Apple");
//2
```