# Arrays and its built in methods

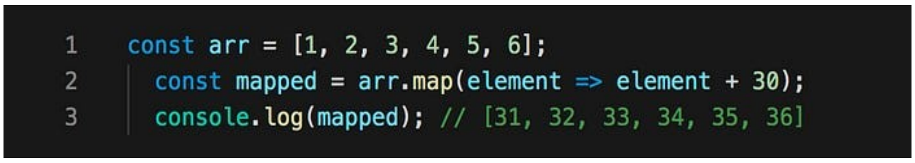
Wednesday, 11 October 2023 12:30 PM

- An array is a data structure that contains list of elements which store multiple values in a single variable. The strength of JavaScript arrays lies in the array methods.

- Array methods are functions built-in to JavaScript that we can apply to our arrays — Each method has a unique function that performs a change or calculation to our array and saves us from writing common functions from scratch.

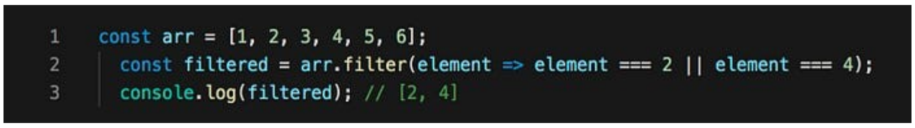
## JavaScript Array Methods are:

**1. map( )**
This method creates a new array with the results of calling a provided function on every element in this array.



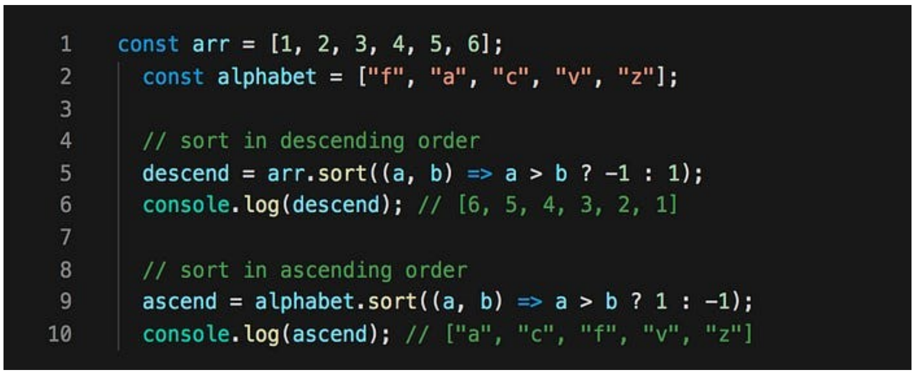
```javascript
const arr = [1, 2, 3, 4, 5, 6];
  const mapped = arr.map(element => element + 30);
  console.log(mapped); // [31, 32, 33, 34, 35, 36]
```

2. filter( )
This method creates a new array with only elements that passes the condition inside the provided function.



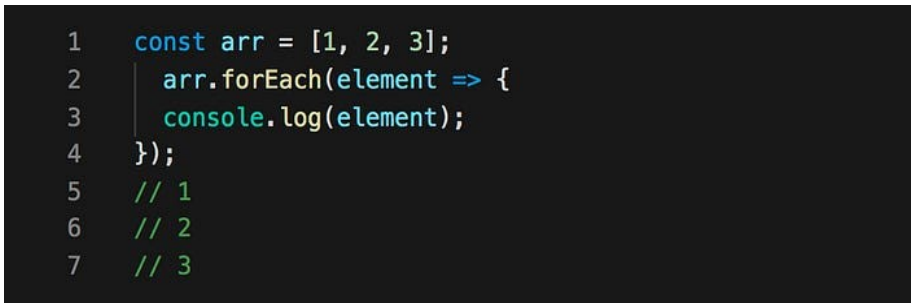
```javascript
const arr = [1, 2, 3, 4, 5, 6];
  const filtered = arr.filter(element => element === 2 || element === 4);
  console.log(filtered); // [2, 4]
```

3. sort( )
This method is used to arrange/sort array's elements either in ascending or descending order.



```javascript
const arr = [1, 2, 3, 4, 5, 6];
  const alphabet = ["f", "a", "c", "v", "z"];

  // sort in descending order
  descend = arr.sort((a, b) => a > b ? -1 : 1);
  console.log(descend); // [6, 5, 4, 3, 2, 1]

  // sort in ascending order
  ascend = alphabet.sort((a, b) => a > b ? 1 : -1);
  console.log(ascend); // ["a", "c", "f", "v", "z"]
```

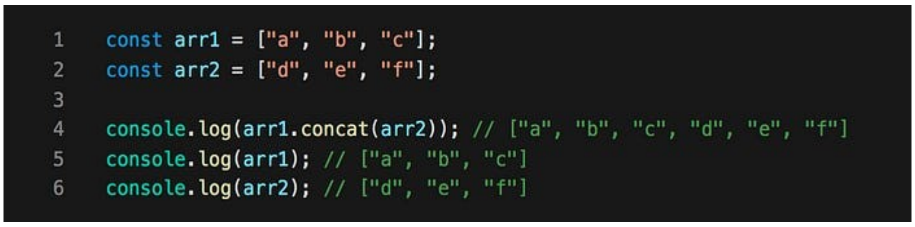
4. forEach( )
This method helps to loop over array by executing a provided callback function for each element in an array.



```javascript
const arr = [1, 2, 3];
  arr.forEach(element => {
  console.log(element);
});
// 1
// 2
// 3
```

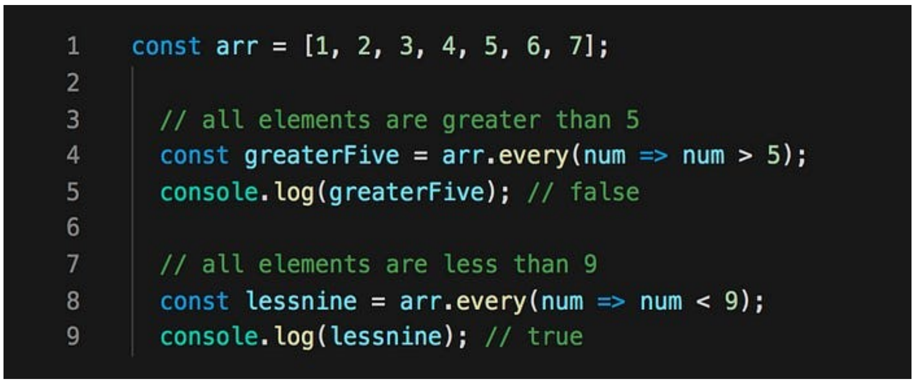
5. **concat( )**

This method is used to merge two or more arrays and returns a new array, without changing the existing arrays.



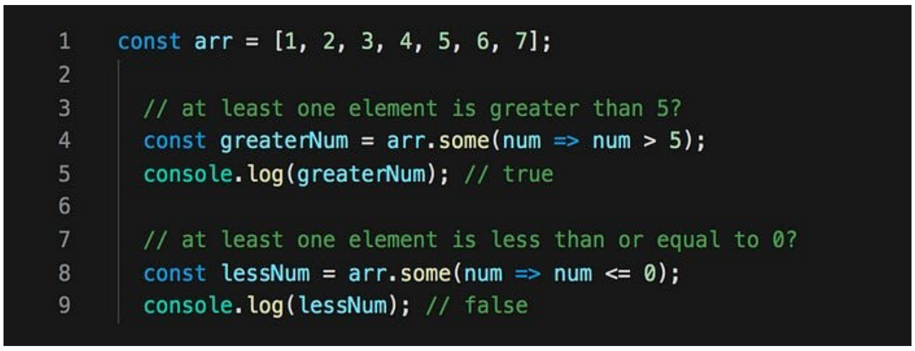
```
const arr1 = ["a", "b", "c"];
const arr2 = ["d", "e", "f"];

console.log(arr1.concat(arr2)); // ["a", "b", "c", "d", "e", "f"]
console.log(arr1); // ["a", "b", "c"]
console.log(arr2); // ["d", "e", "f"]
```

6. every( )
This method checks every element in the array that passes the condition, returning true or false as appropriate.



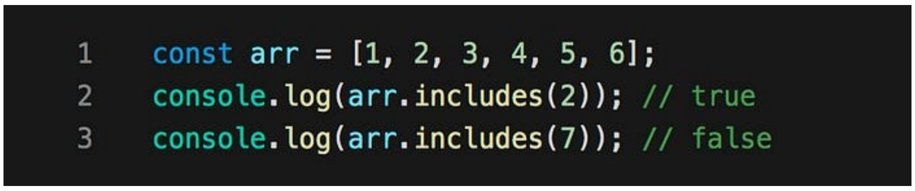
```
const arr = [1, 2, 3, 4, 5, 6, 7];

  // all elements are greater than 5
  const greaterFive = arr.every(num => num > 5);
  console.log(greaterFive); // false

  // all elements are less than 9
  const lessnine = arr.every(num => num < 9);
  console.log(lessnine); // true
```

7. some( )
This method checks if at least one element in the array that passes the condition, returning true or false as appropriate.



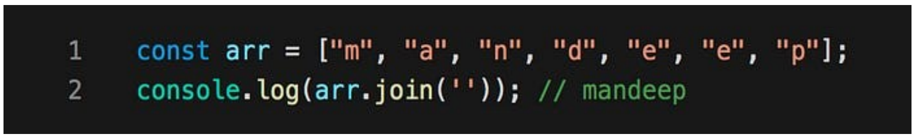
```
const arr = [1, 2, 3, 4, 5, 6, 7];

  // at least one element is greater than 5?
  const greaterNum = arr.some(num => num > 5);
  console.log(greaterNum); // true

  // at least one element is less than or equal to 0?
  const lessNum = arr.some(num => num <= 0);
  console.log(lessNum); // false
```

8. **includes( )**
This method checks if an array includes the element that passes the condition, returning true or false as appropriate.



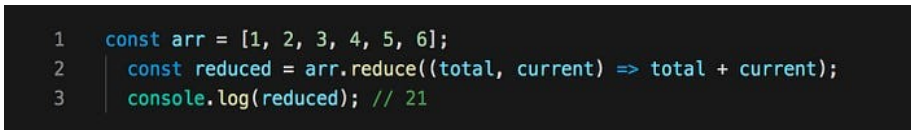
```
const arr = [1, 2, 3, 4, 5, 6];
console.log(arr.includes(2)); // true
console.log(arr.includes(7)); // false
```

9. join( )
This method returns a new string by concatenating all of the array's elements separated by the specified separator.



```
const arr = ["m", "a", "n", "d", "e", "e", "p"];
console.log(arr.join('')); // mandeep
```

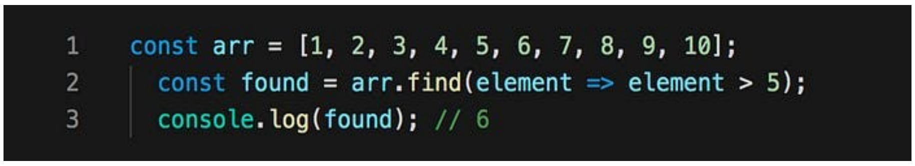
10. reduce( )
This method applies a function against an accumulator and each element in the array to reduce it to a single value.



```
const arr = [1, 2, 3, 4, 5, 6];
  const reduced = arr.reduce((total, current) => total + current);
  console.log(reduced); // 21
```

11. find( )

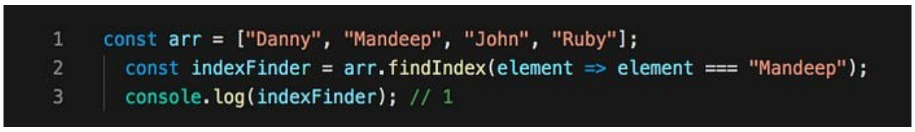
This method returns the value of the first element in an array that pass the test in a testing function.



```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
  const found = arr.find(element => element > 5);
  console.log(found); // 6
```

12. findIndex( )

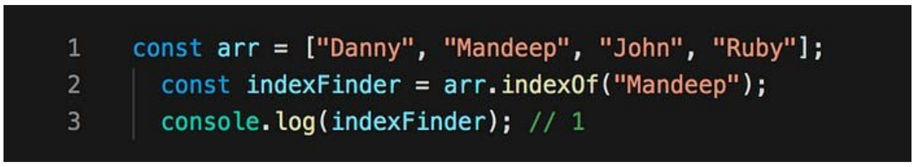
This method returns the index of the first element in an array that pass the test in a testing function.



```
const arr = ["Danny", "Mandeep", "John", "Ruby"];
  const indexFinder = arr.findIndex(element => element === "Mandeep");
  console.log(indexFinder); // 1
```

13. indexOf( )

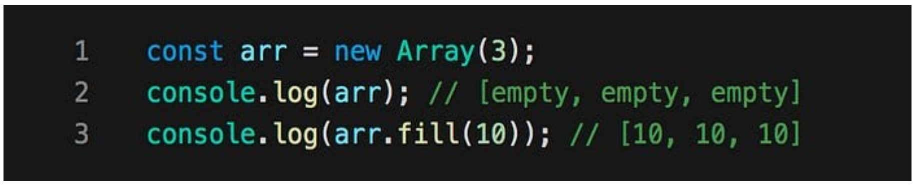
This method returns the index of the first occurrence of the specified element in the array, or -1 if it is not found.



```
const arr = ["Danny", "Mandeep", "John", "Ruby"];
  const indexFinder = arr.indexOf("Mandeep");
  console.log(indexFinder); // 1
```

14. fill( )

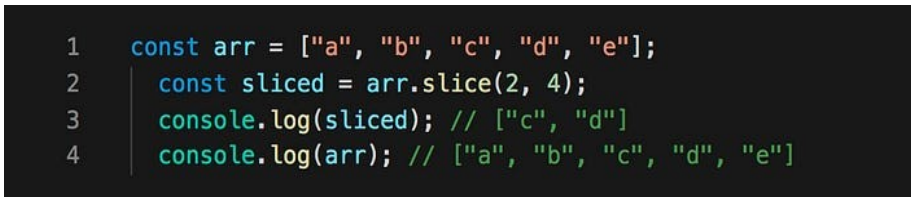
This method fills the elements in an array with a static value and returns the modified array.



```
const arr = new Array(3);
console.log(arr); // [empty, empty, empty]
console.log(arr.fill(10)); // [10, 10, 10]
```

15. slice( )

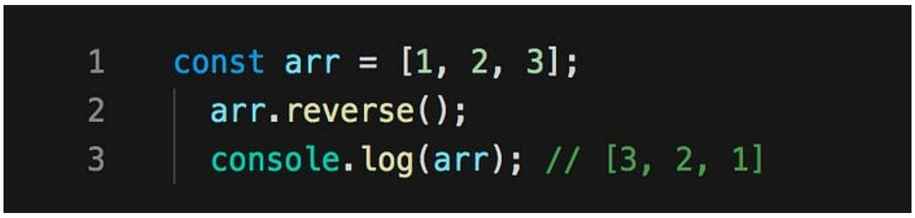
This method returns a new array with specified start to end elements.



```
const arr = ["a", "b", "c", "d", "e"];
  const sliced = arr.slice(2, 4);
  console.log(sliced); // ["c", "d"]
  console.log(arr); // ["a", "b", "c", "d", "e"]
```

16. reverse( )

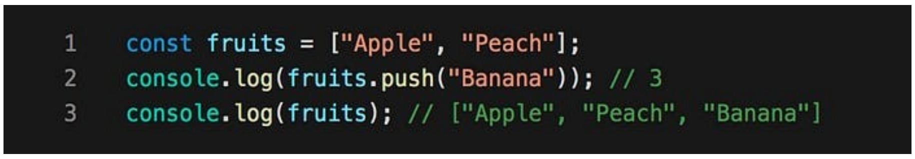
This method reverses an array in place. Element at last index will be first and element at 0 index will be last.



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

17. **push( ) --> adds in last**

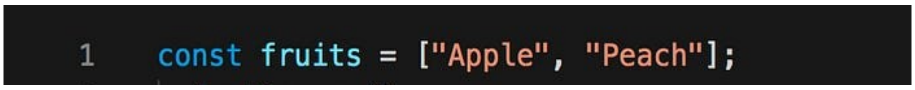
This method adds one or more elements to the end of array and returns the new length of the array.



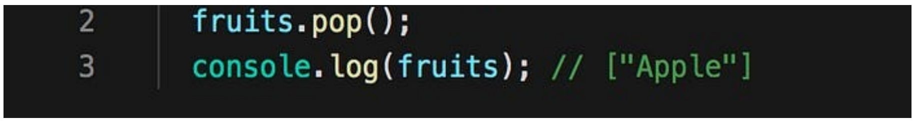
```
const fruits = ["Apple", "Peach"];
console.log(fruits.push("Banana")); // 3
console.log(fruits); // ["Apple", "Peach", "Banana"]
```

18. **pop( )** --> removes from last

This method removes the last element from the end of array and returns that element.



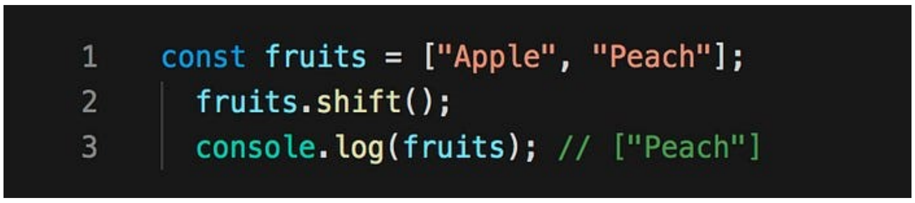
```
const fruits = ["Apple", "Peach"];
```



```
fruits.pop();
console.log(fruits); // ["Apple"]
```

19. **shift( ) --> removes from first**

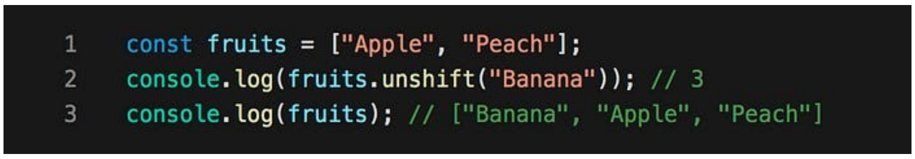
This method removes the first element from an array and returns that element.



```
const fruits = ["Apple", "Peach"];
  fruits.shift();
  console.log(fruits); // ["Peach"]
```

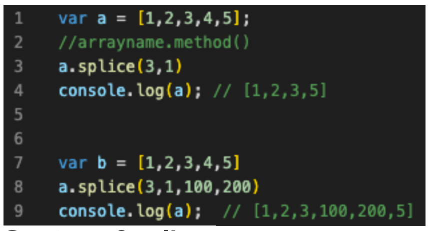
20. **unshift( ) --> adds in front**

This method adds one or more elements to the beginning of an array and returns the new length of the array.



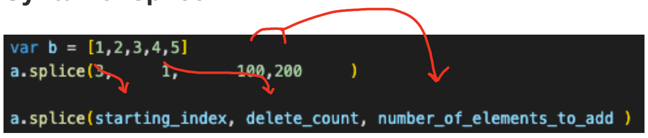
```
const fruits = ["Apple", "Peach"];
console.log(fruits.unshift("Banana")); // 3
console.log(fruits); // ["Banana", "Apple", "Peach"]
```

21. **Splice() --> adds element anywhere in between the array**



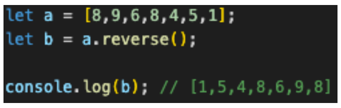
```
var a = [1,2,3,4,5];
//arrayname.method()
a.splice(3,1)
console.log(a); // [1,2,3,5]


var b = [1,2,3,4,5]
a.splice(3,1,100,200)
console.log(a);  // [1,2,3,100,200,5]
```

**Syntax of splice:**



```
var b = [1,2,3,4,5]
a.splice(3,      1,       100,200     )

a.splice(starting_index, delete_count, number_of_elements_to_add )
```

22. **Reverse - reverse the array**



```
let a = [8,9,6,8,4,5,1];
let b = a.reverse();

console.log(b); // [1,5,4,8,6,9,8]
```