Arrays Manipulations

Basics of Arrays:

An array is a collection of indexed elements, represented by "[ ]", with indexing starting at 0.

Ex: 1

```
<script>
let arr1 = [10,20,30,40,50];

document.write(arr1[0],`<br>`);
document.write(arr1[1],`<br>`);
document.write(arr1[2],`<br>`);
document.write(arr1[3],`<br>`);
document.write(arr1[4],`<br>`);
document.write(arr1[5],`<br>`);
//undefined
document.write(arr1[1],`<br>`);
//undefined
</script>
```

Ex: 2

```
<script>

let arr2 =
[`HTML`,`CSS`,`JAVASCRIPT`,`React`,`Node`]
;
document.write(arr2[0],`<br>`);
document.write(arr2[1],`<br>`);
document.write(arr2[2],`<br>`);
document.write(arr2[3],`<br>`);
document.write(arr2[4],`<br>`);

</script>
```

Arrays Manipulations Methods:

1) at(): at() is a method, used to get both positive index and negative index in arrays. at() method introduced in ES6 version.

```
<script>
let arr3 = [10,20,30,40,50];
document.write(arr3.at(0),arr3.at(-5), `<br>`);
document.write(arr3.at(1),arr3.at(-4), `<br>`);
document.write(arr3.at(2),arr3.at(-3), `<br>`);
document.write(arr3.at(3),arr3.at(-2), `<br>`);
document.write(arr3.at(4),arr3.at(-1), `<br>`);

</script>
```

2) map(): map() is a method used to manipulate each and every element in array. map() method introduced in ES6 version.

```
<script>

let arr4 = [10,20,30,40,50];
let new_arr = arr4.map((element,index)=>{
   return element*100;
  });
console.log(new_arr);

</script>
```

3) filter(): filter() is a method used to apply the conditions on array of elements. filter() method introduced in ES6 version.

```
<script>

let arr6 = [10,20,30,40,50];
let res1 =
arr6.filter((element,index)=>{
  return element>=30;
   });
console.log(res1);

</script>
```

Q) Write a program that manipulates an array using map() and filter().

a)

```
<script>


let arr8 = [10,20,30,40,50];

  let new_arr1 =
arr8.map((element,index)=>{
   return element*10;
   }).filter((element,index)=>{
     return element>=400;
    });
    console.log(new_arr1);

</script>
```

4) reduce(): reduce() is a method used to find the sum of an array of elements. Introduced in ES6 version.

```
<script>
let arr9 = [10,20,30,40,50];
let res3 =
arr9.reduce((firstElement,nextElement)=>{
 return firstElement+nextElement;
  });
 document.write(res3);       //150

</script>
```

Q) What is accumulator? How it works?

- a) The term "accumulator" is commonly associated with the reduce() method, which is used to process an array and reduce it to a single value.
- The accumulator is a parameter in the callback function you provide to reduce().
- It stores the result of the previous iterations and is passed along to each subsequent iteration.

Ex:

```
<script>

let arr10 = [1,2,3,4,5];
let res4 =
arr10.reduce((firstElement,nextElement)=>{
return firstElement+nextElement;
},85);
document.write(res4);     //100

</script>
```

Q) Write a program that manipulates an array using map(), filter() and reduce().

```
<script>

let arr11 = [10,20,30,40,50];
let res5 = arr11.map((element,index)=>{
 return element*100;
 }).filter((element,index)=>{
 return element<=3000;
 }).reduce((firstElement,nextElement)=>{
 return firstElement+nextElement;
 },4000);
 document.write(res5);

//10000

</script>
```

5) includes(): It is used to check the particular element is available or not. Introduced in ES6 version.
includes() is the Boolean function and it will return true or false.

```
<script>
let arr12 = [10,20,30,40,50];
document.write(arr12.includes(30));
//true
```

6) find(): It is used to check the particular element is available or not. Introduced in ES6 version.
find() is the number function and it will return number.

```
<script>
let res6 = arr12.find((element,index)=>{
  return element == 30;
 });
document.write(res6);      //30

</script>
```

7) every(): every() method is used to test if every element satisfies the condition in an array then it will return true otherwise it will return false. If the array is empty, every() returns true by default. Introduced in ES6 version.

```
<script>

let arr13 = [10,20,30,40,50];
let res7 = arr13.every((element,index)=>{
return element<100;
});
document.write(res7);  //true

let res8 = arr13.every((element,index)=>{
return element<50;
});
document.write(res8);   //false


</script>
```

8) splice():

The splice() method in ES6 (ECMAScript 2015) is a powerful tool for manipulating arrays. It allows you to remove, or replace elements in an array at a specified index.

```
<script>
let arr1 =
[10,20,30,40,50,60,70,80,90,100];
//delete 40,50,60
arr1.splice(3,3);
console.log(arr1); //[10, 20, 30, 70, 80, 90,
100]

//delete 10
arr1.splice(0,1);
console.log(arr1); //[20, 30, 70, 80, 90,
100]
//delete 100
arr1.splice(5,1);
console.log(arr1); //[20, 30, 70, 80, 90]
//add 40, 50, 60
arr1.splice(2,0,40,50,60);
console.log(arr1);//[20, 30, 40, 50, 60,
70, 80, 90]
//delete 20 and add 10,20
arr1.splice(0,1,10,20);
console.log(arr1);//[10, 20, 30, 40, 50,
60, 70, 80, 90]
//add 100 after 90
arr1.splice(9,0,100);
console.log(arr1); //[10, 20, 30, 40, 50,
60, 70, 80, 90, 100]
</script>
```

9) findIndex():

The findIndex() method in ES6 (ECMAScript 2015) is used to find the index of the first element in an array that satisfies a provided testing function. If no elements satisfy the testing function, it returns `-1`.

```
let arr2 =
[10,20,30,40,50,60,70,80,90,100];
let x = arr2.findIndex((element,index)=>{
return element == 50;
});
document.write(x);          //4

let y = arr2.findIndex((element,index)=>{
return element == 1000;
});
document.write(y);          //-1
```

10) some():

In ES6 (ECMAScript 2015) and later versions of JavaScript, the some() method is used to test whether at least one element in an array passes the test implemented by the provided function. If atleast one element satisfies the condition then it will return true. Otherwise, it returns false.

```
let arr3 = [10,20,30,40,50];
            let p =
arr3.some((element,index)=>{
 return element<=10;
 });
 document.write(p);         // true

let q = arr3.some((element,index)=>{
return element<10;
 });
document.write(q);          //false
```

11) reverse():

In ES6 (ECMAScript 2015), the reverse() method is a built-in function used to reverse the order of elements in an array. When you call reverse() on an array, it modifies the original array by reversing the order of its elements in place. It also returns the reversed array.

```
let arr4 = [10,20,30,40,50];
let new_arr = arr4.reverse();
    console.log(new_arr);
   //[50, 40, 30, 20, 10]
```

(or)

```
let str = "hello";

let reversedStr = str.split('').reverse().join('');
console.log(reversedStr); // Output: "olleh"
```

12) toString():

In ES6 (ECMAScript 2015), the toString() method is a built-in function used to convert an array into a string. This method returns a string representing the elements of the array, separated by commas.

```
let fruits = ['apple', 'banana', 'cherry'];

let result = fruits.toString();

console.log(result);

// Output: "apple,banana,cherry"
```

13) Array.form(): Array.from() is a method in ES6 (ECMAScript 2015) that creates a new, shallow-copied array instance from an array-like or iterable object. This method is particularly useful for converting objects that aren't arrays, such as NodeLists or strings, into true arrays.

```
let str = "hello";

let arr = Array.from(str);

console.log(arr);  // Output: ['h', 'e', 'l', 'l', 'o']
```

Q) Write a program that manipulates reverse an array and convert that array into string and string to array.
a)

```
console.log(

 Array.from("Hello").reverse().toString().
replaceAll(",","")
);       //olleH
```

14) join(): The join() method is used to concatenate all the elements of an array into a single string. The elements are separated by a specified separator, which can be a string or character of your choice. If no separator is specified, the elements are separated by commas by default.

```
console.log(
Array.from("CSS").reverse().join("")
);            //SSC
(or)
let characters = ['a', 'b', 'c'];
let result = characters.join('');
 console.log(result); // Output: "abc"
```

15) slice(): The slice() method is used to create a shallow copy of a portion of an array into a new array object. The selected portion of the array is defined by specifying a start and an end index. The original array is not modified. (Choose the elements in array).

```
let arr5 =
[10,20,30,40,50,60,70,80,90,100];
console.log(arr5.slice(3,6));
//[40, 50, 60]
console.log(arr5.slice(0,1));
//[10]
console.log(arr5.slice(8));
//[90, 100]
```

16) flat(): It is used to convert multi dimensional array to single dimensional array. Introduced in ES2019 (ES10).

```
let arr6 = [[[[[[[[1,2,3]]]]]]]];
let r = arr6.flat(Infinity);
console.log(r); //[1, 2, 3]
```

17) flatMap(): The flatMap() method is a combination of the map() and flat() methods. It first maps each element using a mapping function and then flattens the result into a new array. This flattening is done to a depth of 1.

```
let arr7 = [1,2,3];
let s = ["one","two","three"];
let new_arr2 =
arr7.flatMap((element,index)=>{
 return [element,s[index]];
 });
 console.log(new_arr2);
//[1, 'one', 2, 'two', 3, 'three']
```

18) forEach(): In ES6 (ECMAScript 2015), the forEach() method is a built-in array method used to execute a provided function once for each element in an array. It allows you to iterate over the elements in an array without explicitly using a loop structure like for or while.

```
var arr1 = [1000,2000,3000,4000,5000];
arr1.forEach((element,index)=>{
 document.write(element,`.......`,index
,`<br>`);
 });
```

19) for....of(): In ES6 (ECMAScript 2015), the for...of loop is a new way to iterate over iterable objects, such as arrays, strings, maps, sets, and more. It's particularly useful for iterating over the elements of an array because it provides a more concise and readable syntax compared to traditional loops.

```
var arr2 =
[`React`,`Node`,`HTML`,`CSS`,`JavaScri
pt`];
for(var x of arr2) {
document.write(x, `<br>`);
 }
```

20) for….in(): The for...in loop is used to iterate over the enumerable properties of an object, including array indices. It is part of JavaScript's ES5 standard and is available in ES6 as well.

```
var obj3 = {
    "key1" : "Frontend",
    "key2" : "Backend",
    "key3" : "Database"
 };
 for(var x in obj3) {
 document.write(obj3[x]);
 }
//FrontendBackendDatabase
```

21) indexOf(): In ES6 (ECMAScript 2015), the indexOf() method is used to find the index of the first occurrence of a specified element in an array. If the element is found, indexOf() returns the index of that element. If the element is not found, it returns -1. It won't assign indexes to the repeated elements.

```
let arr8 = [10,20,30,10,20,30];
arr8.forEach((element,index)=>{
console.log(arr8.indexOf(element));
});
```

Q) Write a program that manipulates removing the duplicates from the array, using filter() and indexOf().

a) `let arr9 = [10,20,30,10,20,30];`

```
let new_arr3 =
arr9.filter((element,index)=>{
return arr9.indexOf(element) == index;
 });
console.log(new_arr3);  //[10, 20, 30]
```

22) fill(): In ES6 (ECMAScript 2015), the fill() method is used to modify all elements in an array with a static value, from a start index to an end index. It's a straightforward way to populate or replace array elements with a specified value.

```
let arr10 = [10,10,10,10,10];
console.log(
arr10.fill(20,1).fill(30,2,3).fill(40,
3,4).fill(50,4)
);          // [10, 20, 30, 40, 50]
```

23) length(): In JavaScript, the length property of an array is used to determine or set the number of elements in the array. It is a fundamental property that provides the size of the array.

```
let arr11 = [10,20,30,40,50];
console.log(arr11.length);   //5

let arr12 = [];
console.log(arr12.length);   //0

let arr13 = [];
arr13[100] = 1000;
console.log(arr13.length);  //101
```

24) delete(): In JavaScript, the delete operator is used to remove a property from an object, including properties of an array. Main drawback of delete() is… it deletes the element but memory filled with empty so index won't deleted.

```
let arr14 = [10,20,30,40,50];
delete arr14[0];
console.log(arr14.length);   //5
```

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

```
let arr15 = [20,30,40];
    arr15.push(50);
  console.log(arr15);
  //[20, 30, 40, 50]
```

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

```
let arr15 = [20,30,40,50];

    arr15.unshift(10);

  console.log(arr15);
//[10, 20, 30, 40, 50]
```

27) pop(): In JavaScript, the pop() method is used to remove the last element from an array. It modifies the original array and returns the removed element. This method is straightforward and commonly used for stack-like operations where you need to remove elements from the end of the array.

```
let arr15 = [10, 20, 30, 40, 50];
arr15.pop();
console.log(arr15);
//[10, 20, 30, 40]
```

28) shift(): In JavaScript, the shift() method is used to remove the first element from an array. It modifies the original array and returns the removed element. This method is useful when you need to remove elements from the beginning of the array, such as when implementing queue-like data structures.

```
let arr15 = [10,20,30,40];
arr15.shift();
console.log(arr15); //[20, 30, 40]
```

29) padStart(): In JavaScript, the padStart() method is actually a string method, not an array method. It is used to pad the beginning of a string with a specified character or string until it reaches a given length. This method is useful for formatting strings to a consistent length, such as when creating fixed-width columns or formatting numbers.

```
console.log(
"ExcelR".padStart(10,"*") //****ExcelR
);
```

30) padEnd(): In JavaScript, padEnd() is a string method, not an array method. It is used to pad the end of a string with a specified character or string until it reaches a given length. This method is useful when you need to format strings to a consistent length, especially for aligning text output.

```
console.log(
"ExcelR".padEnd(10,"*")
);
//ExcelR****
```

31) trimStart(): trimStart() method is used to remove whitespace characters from the beginning of a string. This method is particularly useful when you need to clean up user input or format strings by removing unnecessary leading spaces.

```
console.log(
" ExcelR ".length
);                  //8

console.log(
" ExcelR ".trimStart().length
);                 //7
```

32) trimEnd(): trimEnd() method is used to remove whitespace characters from the end of a string. This method is particularly useful for cleaning up text data or formatting strings by removing unnecessary trailing spaces.

```
console.log(
"ExcelR".trimEnd().length
);                  //7
```

33) trim(): It is used to remove whitespace characters from both the beginning and the end of a string. This method is particularly useful for cleaning up user input or formatting strings by removing unnecessary spaces at both ends.

```
console.log(
"ExcelR".trim().length
);          //6
```

34) sort(): In JavaScript, the sort() method is used to sort the elements of an array. The sort() method sorts the elements in place and returns the sorted array. By default, the sort() method sorts elements as strings in ascending order, which can lead to unexpected results when sorting numbers.

```
//ascending order
let arr16 = [10,20,30,40,50];
console.log(
arr16.sort((num1,num2)=>{
return num1-num2;//[10, 20, 30, 40, 50]
    })
);
//descending order
console.log(
arr16.sort((num1,num2)=>{
return num2-num1;//[50, 40, 30, 20, 10]
   })
);
```

35)reduceRight(): The reduceRight() method in JavaScript is used to apply a function to each element of an array (from right to left) to reduce the array to a single value. It works similarly to reduce(), but instead of processing the array from left to right, it processes the array from right to left.

```
let arr1 = ["ExcelR","to","Welcome"];
let res =
arr1.reduceRight((firstElement,nextEle
ment)=>{
return firstElement+nextElement;
});
console.log(res);     //WelcometoExcelR
```

36) lastIndexOf(): InJavaScript, the lastIndexOf() method is used to find the last occurrence of a specified element within an array. It searches the array from right to left (from the last element to the first) and returns the index of the last occurrence of the element. If the element is not found, it returns -1.

```
let arr2 = [10,20,30,40,50,10,20,10];
console.log(arr2.lastIndexOf(10));
  //7
```

37) copyWithin(): The copyWithin() method in JavaScript is used to copy a portion of an array to another location within the same array without modifying its length. It is a mutable method, meaning it changes the contents of the array in place and does not create a new array.

```
let arr3 =
[10,20,30,40,50,60,70,80,90,100];
arr3.copyWithin(5);
console.log(arr3);
//[10, 20, 30, 40, 50, 10, 20, 30, 40, 50]

let arr4 =
[10,20,30,40,50,60,70,80,90,100];
arr4.copyWithin(2,5);
console.log(arr4);          //[10, 20, 60,
70, 80, 90, 100, 80, 90, 100]

let arr5 =
[10,20,30,40,50,60,70,80,90,100];
arr5.copyWithin(2,5,7);
console.log(arr5);         //[10, 20, 60,
70, 50, 60, 70, 80, 90, 100]
```

38)repeat(): It is used to create a new string that contains a specified number of copies of the original string concatenated together.

```
let str = "ExcelR";
console.log(str.repeat(5));
//ExcelRExcelRExcelRExcelRExcelR
```

39)split(): The split() method is actually a string method, not an array method. It is used to split a string into an array of substrings based on a specified separator. The resulting array contains the substrings, and the original string remains unchanged.

```
let str1 = "Welcome to ExcelR";
 console.log( str1.split(" ")
);     //['Welcome', 'to', 'ExcelR']
```

40)substring(): The substring() method is actually a string method, not an array method, in JavaScript. It is used to extract a portion of a string and returns the extracted part as a new string, without modifying the original string. **It contains strating index and ending index.**

```
let str2 = "Welcome to ExcelR";
console.log( str2.substring(0,7)
);                            //Welcome
console.log( str2.substring(8,10)
);                            //to
console.log( str2.substring(11)
);                            //ExcelR
```

41)substr(): The substr() method is another string method in JavaScript, not an array method. It is used to extract a portion of a string, returning a specified number of characters from a starting index. **It contains starting index and no.of characters.**

```
let str2 = "Welcome to ExcelR";
console.log( str2.substr(0,7)
);                            //Welcome
console.log( str2.substr(8,2)
);                            //to
console.log( str2.substr(11)
);                          //ExcelR
```

42) replace(): The replace() method is a string method in JavaScript. It is used to replace a specified substring or pattern within a string with another substring.

```
const str = 'Hello, world!';

const result = str.replace('world', 'JavaScript');

console.log(result);

// Output: 'Hello, JavaScript!'
```