**Arrays**

Examples: -

const names= ["Ajay", "Vijay"];

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

const arr= [1,2,3,4,5,”golu”];

1)Array() constructor **x**

let a = new Array(1,"fds",3,4)

Array() can be called with or without [new](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/new). Both create a new Array instance.

When only one parameter :

Array of that size is initialized

const c =new Array(3)

2) Array.from() **x**

Creates array from string or array like object

Can also take a function as parameter

console.log(Array.from("prateek"));

console.log(Array.from([1,2,3], x=>x\*2));

3) Array.isArray()**x**

console.log(Array.isArray([1, 3, 5]));

console.log(Array.isArray("[]"));""

console.log(Array.isArray([""]));

checks whether the value is array or not

returns true or false

4)Array.ofArray() **x**

Creates array from multiple no of arguments regardless of the datatype

console.log(Array.of("prateek", 2, "bar", true));

5) Symbol.species **x**

6) Array.at() ***done***

Takes index   
returns element at that index

const array1 = [5, 12, 8, 130, 44];

console.log(array1.at(3));

7) Array.concat ***done***

Merge two arrays

const array1 = [1,2,3,4,5];

const array2 = [534,6,34,314,634];

const array3 = array1.concat(array2);

console.log(array3);

8) copyWithin()

Copies element within the array

console.log([1, 2, 3, 4, 5].copyWithin(0, 3));

// [4, 5, 3, 4, 5]

console.log([1, 2, 3, 4, 5].copyWithin(0, 3, 4));

// [4, 2, 3, 4, 5]

Case 1 : copies elements from index 3 to end to index 0

Case 2 : copies element from index 3 only to index 0.

9) array.entries() **x**

Returns new iterator which contain key value pairs

const array1 = ["a", "b", "c"];

const iterator1 = array1.entries();

console.log(iterator1.next().value);

// [0, "a"]

console.log(iterator1.next().value);

//  [1, "b"]

console.log(iterator1.next().value);

//  [2, "c"]

10) .every() ***done***

Checks whether each element of an array passes a condition returns true false

const square = (x)=> {return x<20;}

const arr=[1,2,3,32]

const arr2=[1,2,3,4]

console.log(arr.every(square));

console.log(arr2.every(square));

11) .fill() ***done***

Changes value from the specified range

Case 1 : fills 0 in range 2-4

Case 2: fills 9 in range 1 to end

Case 3: fills 68 in entire array

const array1 = [1, 2, 3, 4];

console.log(array1.fill(0, 2, 4));

console.log(array1.fill(9, 1));

console.log(array1.fill(68));

output

[ 1, 2, 0, 0 ]

[ 1, 9, 9, 9 ]

[ 68, 68, 68, 68 ]

12) .filter() ***done***

Returns array elements that satisfies a specific condition

const square = (x)=> {return x<20;}

const arr=[1,2,3,32]

const arr2=[1,2,3,4]

console.log(arr.filter(square));

console.log(arr2.filter(square));

returns array with element which pass the condition

13) .find() ***done***

Takes one function as argument and returns the first element which satisfies that function’s condition.

const array1 = [5, 12, 8, 130, 44];

const a = array1.find((x) => x > 70);

console.log(a);

output = 130

14) .findIndex() ***done***

Takes one function as argument and returns the index of element which satisfies that function’s condition.

const array1 = [5, 12, 8, 130, 44];

const a = array1.findIndex((x) => x > 70);

console.log(a);

output = 3

14) .findLast()  ***done***

Takes one function as argument and returns the first element which satisfies that function’s condition from the last of array.

const array1 = [5, 12, 8, 130, 290 , 44];

const a = array1.findLast((x) => x > 70);

console.log(a);

// 290

14) .findLastIndex()  ***done***

Takes one function as argument and returns the index of first element which satisfies that function’s condition from the last of array.

const array1 = [5, 12, 8, 130, 290 , 44];

const a = array1.findLastIndex((x) => x > 70);

console.log(a);

// 4

15) flat ***done***

Flattens out array

const arr= [0, 1, [2, [3, [4, 5]]]];

console.log(arr.flat());

16) flatmap

First maps an array then flattens it if required .

const arr1 = [1, 2, 3, 4,5];

const result = arr1.flatMap((num) => [num, num\*10]);

console.log(result);

[

1, 10, 2, 20, 3, 30, 4, 40, 5, 50

]

basically jo function chal rha hai vo array return karega – mapping  
vo array flat hoga

17) forEach **x**

Used to iterate an array

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

arr.forEach((x)=>{

    console.log(x);

})

18) .includes() ***done***

Check whether an element is included in arr or not.

const array1 = [1, 2, 3];

console.log(array1.includes(2));

returns true

18) indexof()  ***done***

Returns index

const beasts = ["alu", "balu", "calu", "dalu", "malu"];

console.log(beasts.indexOf("malu"));

op= 4

19) .join() ***done***

Returns new string made by concatenating all elements

const beasts = ["alu", "balu", "calu", "dalu", "malu"];

console.log(beasts.join(""));

alubalucaludalumalu

20) .keys() **­x**

Returns iterator which contains the index of each element

const array1 = ["a", "b", "c"];

const iterator = array1.keys();

for (const key of iterator) {

  console.log(key);

}

21) lastIndexOf() ***done***

Returns last index of a value

const arr= [1,2,2,,2,2];

console.log(arr.lastIndexOf(2));

output - 5

22) map ***done***

Creates a new array which applies a function on an arr that was passed

const array1 = [1, 4, 9, 16];

const map1 = array1.map((x) => x \* 2);

console.log(map1)

23) pop  ***done***

Removes last element

const ar=[1,2,2,3,4];

console.log(ar.pop());

24) push ***done***

Pushes element to last

const ar=[1,2,2,3,4];

ar.push(332);

console.log(ar);

25) .reduce

Element keeps accumulating

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

const x=arr.reduce((accumulated, remaining)=>accumulated+remaining);

console.log(x);

op-15

accumulated – value jo update huyi ja rhi hai   
remaining – jiske through iterate kar rhe hai

26)reduceRight

Instead of accumulating from left to right

It does same in right to left

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

const x=arr.reduce((accumulated, remaining)=>accumulated+remaining);

console.log(x);

op-15

27)reverse ***done***

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

const x=arr.reverse();

console.log(x);

28) shift  ***done***

Similar to pop

Removes first element

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

const x=arr.shift();

console.log(arr);

29) slice ***done***

Returns arr element from a specified range

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

console.log(arr.slice(2,4));

[ 3, 4 ]

30 ) some ***done***

Checks whether any element satisfied a condition

Returns true or false

const arr = [1, 2, 3, 4];

console.log(arr.some((x) => x % 2 === 0));

true

31) sort ***done***

const arr= [9,6,4,3,2];

arr.sort();

console.log(arr);

32) splice ***done***

Can replace / add element at an index

Syntax

splice(start, deleteCount, item1)

starting index , elements to be deleted / can be zero , elements to be added

const arr= ["a", "b", "c", "d", "e", "f", "g", "h"];

arr.splice(1,3,"prateek", "gfd","gsfdgfds", "dsds");

console.log(arr);

33) tolocalestring

Converts date object to string

const d = new Date();

let text = d.toLocaleString();

console.log(text);

console.log(d);

34) to Reversed

Creates a new reverse arr

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

const x = arr.toReversed();

console.log(x);

35 ) toSorted

Same as toReverse

36) toSplice

Same

37) tostring

const array1 = [1, 322, "afsds", "1fsda"];

console.log(array1.toString());

arr to string

38) unshift  ***done***

Add element/elements to start of arr

const array1 = [1, 2, 3];

console.log(array1.unshift(4, 5));

console.log(array1);

5

[ 4, 5, 1, 2, 3 ]

39 ) .values ()

Similar to keys

Instead returns iterator containing values

const array1 = ["a", "b", "c"];

const iterator = array1.values();

for (const x of iterator) {

  console.log(x);

}

40) With() ***done***

Replaces an element at specific index

Returns new array

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

console.log(arr); // [1, 2, 3, 4, 5]

console.log(arr.with(0, 6)); // [6, 2, 3, 4, 5]

Array.at()