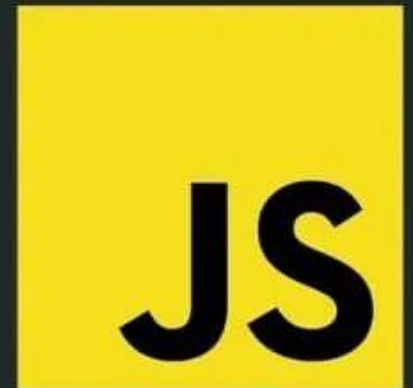


Array Methods...

- | | | |
|------------------|-------------------|---------------|
| 1. values() | 16. concat() | 31. isArray() |
| 2. length() | 17. some() | 32. filter() |
| 3. reverse() | 18. splice() | 33. keys() |
| 4. sort() | 19. flat() | 34. map() |
| 5. at() | 20. lastIndexOf() | |
| 6. fill() | 21. of() | |
| 7. from() | 22. every() | |
| 8. join() | 23. slice() | |
| 9. toString() | 24. flatMap() | |
| 10. pop() | 25. findIndex() | |
| 11. forEach() | 26. find() | |
| 12. shift() | 27. includes() | |
| 13. copyWithin() | 28. entries() | |
| 14. push() | 29. reduceRight() | |
| 15. unshift() | 30. reduce() | |



values(): This method returns an iterator that provides the values for each index in the array. It takes no arguments.



```
const arr = ['apple', 'banana', 'cherry']  
const iterator = arr.values();  
  
for (const value of iterator) {  
  console.log(value);  
} // Output: apple banana cherry
```

length(): This property returns the length of the array.



```
const arr = ['apple', 'banana', 'cherry']  
console.log(arr.length); // Output: 3
```



reverse(): This method reverses the order of the elements in the array.



```
const arr = ['apple', 'banana', 'cherry'];  
arr.reverse();  
console.log(arr); // Output: ['cherry', 'banana', 'apple']
```

sort(): This method sorts the elements of an array in place and returns the sorted array. It can take an optional compare function as an argument.



```
const arr = ['banana', 'apple', 'cherry'];  
arr.sort();  
console.log(arr); // Output: ['apple', 'banana', 'cherry']
```



from(): This method creates a new array from an array-like object or an iterable object. It can take up to two arguments: the object to convert to an array, and a mapping function to apply to each element of the new array.



```
const obj = { 0: 'apple', 1: 'banana', 2: 'cherry', length: 3 }  
const arr = Array.from(obj);  
console.log(arr); // Output: ['apple', 'banana', 'cherry']
```

join(): This method joins all the elements of an array into a string using a specified separator. It takes one optional argument: the separator to use.



```
const arr = ['apple', 'banana', 'cherry'];  
const str = arr.join(', ');  
console.log(str); // Output: 'apple, banana, cherry'
```



toString(): This method returns a string representing the array and its elements.



```
const arr = ['apple', 'banana', 'cherry'];  
const str = arr.toString();  
console.log(str); // Output: 'apple,banana,cherry'
```


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



```
const arr = ['apple', 'banana', 'cherry'];  
const last = arr.pop();  
console.log(last); // Output: 'cherry'  
console.log(arr); // Output: ['apple', 'banana']
```




forEach() method executes a provided function once for each array element. It doesn't return anything, it just executes the callback function on each element of the array.



```
let fruits = ['apple', 'banana', 'cherry']
fruits.forEach(function (item) {
  console.log(item);
}); // Output: apple, banana, cherry
```

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



```
let fruits = ['apple', 'banana', 'cherry'];
let shiftFruit = fruits.shift();
console.log(shiftFruit); // Output: 'apple'
console.log(fruits); // Output: ['banana', 'cherry']
```



copyWithin() method shallow copies part of an array to another location in the same array and returns the modified array without modifying its length. **Syntax** `.copyWithin(target, start, end)`



```
let numbers = [1, 2, 3, 4, 5];  
numbers.copyWithin(2, 0, 2);  
console.log(numbers); // Output: [1, 2, 1, 2, 5]
```

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



```
let fruits = ['apple', 'banana'];  
fruits.push('cherry', 'orange');  
console.log(fruits); // Output: ['apple', 'banana', 'cherry', 'orange']
```



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



```
let fruits = ['cherry', 'orange'];  
fruits.unshift('apple', 'banana');  
console.log(fruits); // Output: ['apple', 'banana', 'cherry', 'orange']
```

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



```
let fruits = ['apple', 'banana'];  
let moreFruits = ['cherry', 'orange'];  
let allFruits = fruits.concat(moreFruits);  
console.log(allFruits); // Output: ['apple', 'banana', 'cherry', 'orange']
```



splice() method changes the contents of an array by removing or replacing existing elements and/or adding new elements in place.

```
const fruits = ['apple', 'banana', 'cherry', 'orange'];  
//Syntax : arr.splice(start, deleteCount, item1, ..., itemN)  
fruits.splice(2, 1, 'mango', 'kiwi');  
console.log(fruits); // Output: [ 'apple', 'banana', 'mango', 'kiwi', 'orange'  
]
```

flat() This method creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

```
const numbers = [1, [2, [3]], 4];  
const flatNumbers = numbers.flat(Infinity);  
console.log(flatNumbers); // Output: [1, 2, 3, 4]
```



keys(): This method returns an array containing the keys of the given object.



```
const myObj = { a: 1, b: 2, c: 3 };  
const keysArray = Object.keys(myObj);  
console.log(keysArray); // Output: ["a", "b", "c"]
```

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



```
const numbers = [1, 2, 3, 4, 5];  
const squaredNumbers = numbers.map((number) => {  
  return number * number;  
});  
console.log(squaredNumbers); // Output: [1, 4, 9, 16, 25]
```

