JS Functions

1. concat()

```
> var arr = [1, 2, 3, 4, 5];
  console.log(arr.concat([6, 7, 8, 9, 10]));
```

```
▶ (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

2. every()

```
> var arr = [1,2,3,4,5];
  console.log(arr.every((element) => element < 10));
```

```
true
```

3. filter()

```
> var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
  console.log(arr.filter((element) => element % 3 == 0));
```

```
▶ (3) [3, 6, 9]
```

4. forEach()

```
var arr = [11, 22, 33, 44, 55];
arr.forEach((item,index) => console.log(item+index));
```

```
11
23
35
47
59
```

5. indexOf()

```
> var arr = ['a', 'b', 'c', 'd', 'e'];
  console.log(arr.indexOf('c'));
```

```
2
```

6. join()

```
var arr = ["Omkar", "Darshan", "Rohan"];
console.log(arr.join("\n"));
```

```
Omkar
Darshan
Rohan
```

7. lastIndexOf()

```
> var arr = [1, 2, 3, 2, 3, 2, 3, 5, 3, 6, 7];
  console.log(arr.lastIndexOf(3));
```

```
8
```

8. map()

```
var arr = [1 ,2 ,3 ,4, 5];
console.log(arr.map(item => item += 1));
```

```
▶ (5) [2, 3, 4, 5, 6]
```

9. pop()

```
var arr = [5, 4, 3, 2, 1];
console.log(arr.pop());
console.log(arr.pop());
console.log(arr.pop());
console.log(arr.pop());
console.log(arr.pop());
```

```
1
2
3
4
5
```

10. push()

```
var arr=[1, 2, 3, 4, 5];
console.log(arr.push(6));
console.log(arr.push(7));
console.log(arr.push(8));
console.log(arr.push(8));
console.log(arr);
```

```
6
7
8
9
▶ (9) [1, 2, 3, 4, 5, 6, 7, 8, 8]
```

11. reduce()

```
var arr = [10, 15, 20, 25, 30];
var sum = arr.reduce((sum, curr) => sum += curr);
console.log(sum);
```

```
100
```

12. reduceRight()

```
var arr = ['00', '11', '22', '33', '44', '55'].reduceRight((str, curr) => str+curr);
console.log(arr);
```

```
554433221100
```

13. reverse()

```
var arr = [10, 15, 20, 25, 30];
console.log(arr.reverse());
```

```
▶ (5) [30, 25, 20, 15, 10]
```

14. shift()

```
var arr = [10, 15, 20, 25, 30];
console.log(arr.shift());
console.log(arr);
console.log(arr.shift());
console.log(arr);
console.log(arr.shift());
console.log(arr);
```

```
10
▶ (4) [15, 20, 25, 30]
15
▶ (3) [20, 25, 30]
20
▶ (2) [25, 30]
```

15. slice()

```
var arr = [10, 20, 30, 40, 50];
console.log(arr.slice(3, 5));
```

```
▶ (2) [40, 50]
```

16. some()

```
var arr = [11, 21, 35, 41, 54];
console.log(arr.some(item => item % 11 == 0));
```

```
true
undefined
```

```
var arr = [11, 21, 35, 41, 54];
console.log(arr.some(item => item % 12 == 0));
```

```
false
```

17. toSource()

```
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
person.toSource();
console.log(person.toSource());
```

```
▶ Uncaught TypeError: person.toSource is not a function
    at <anonymous>:2:8
```

18. sort()

```
var arr = [311, 121, 235, 541, 454];
console.log(arr.sort((a,b) => a - b));
```

```
▶ (5) [121, 235, 311, 454, 541]
```

```
undefined
```

```
var arr = ["Omkar", "Darshan", "Rohan"];
console.log(arr.sort());
```

```
▶ (3) ["Darshan", "Omkar", "Rohan"]
```

19. splice()

```
var arr = [11, 22, 33, 44, 55];
arr.splice(2, 0, 66, 77 ,88);
console.log(arr);
```

```
▶ (8) [11, 22, 66, 77, 88, 33, 44, 55]
```

```
undefined
```

```
var arr = [11, 22, 33, 44, 55];
arr.splice(2, 2, 66, 77 ,88);
console.log(arr);
```

```
▶ (6) [11, 22, 66, 77, 88, 55]
```

20. toString()

```
var arr = [11, 22, 33, 44, 55];
var str= arr.toString();
console.log(str);
```

```
11,22,33,44,55
```

21. unshift()

```
var arr = [11, 22, 33, 44, 55];
arr.unshift(0);
console.log(arr);
```

```
▶ (6) [0, 11, 22, 33, 44, 55]
```

Difference between \r & \n:

Ascii Code of \n is 10 & that of \r is 13

Unix	Windows	Mac
\n is end-of-line \r means nothing special	Old Windows \r\n was end of line	Old Mac \r end of line New Mac \n end of line

In new systems \r and \n act similarly, however \n is the way to go in the modern context of writing to a text file.

\r is good to use when writing to a character terminal (console window) and want the next line you write to overwrite the last one that was just wrote.

For electro-mechanical teletype-like terminals:
- \r commands the carriage to go back leftwards until it hits the leftmost stop (a very slow operation)
- \n commands the roller to roll up one line (a much faster operation)
- Thus \r before \n, so that the roller can move while the carriage is still going leftwards!-)