1. **Concat():**

   The concat() method is used to join two or more strings or arrays.

```
> var s1="Akshat"
  var s2="jain"
  var res=s1.concat(s2);
  res;
"Akshatjain"
> var s1=["a","k","s","h","a","t"]
  var s2=["j","a","i","n"]
  s1.concat(s2);
▶ (10) ["a", "k", "s", "h", "a", "t", "j", "a", "i", "n"]
>
```

2. **Indexof():**

   The indexOf() method searches the array for the specified item, and returns its position

```
> var s1 = "Hello, wassup ? is everything alright";
  s1.indexOf("is");
16
> var a1=[10,15,20,25,30,35,40,45,50,55,60];
  a1.indexOf(45);
7
```

3. **Every():**

   The every() method checks if all elements in an array pass a test.

```
> var a1=[10,15,20,25,30,35,40,45,50,55,60];
  function eval(item){
      return item <=30;}
  a1.every(eval);
false
>
```

4. **Join():**

The join() method returns the array as a string.

```
> var s1 = ["Hello", "wassup","is","everything","alright","?"];
  s1.join();
"Hello,wassup,is,everything,alright,?"
> var s1 = ["Hello", "wassup","is","everything","alright","?"];
  s1.join(" ");
"Hello wassup is everything alright ?"
> var s1 = ["Hello", "wassup","is","everything","alright","?"];
  s1.join(":");
"Hello:wassup:is:everything:alright:?"
>
```

5. **Lastindexof():**

The lastIndexOf() method searches the array for the specified item, and returns its position

6. **Pop()**

The pop() method removes the last element of an array, and returns that element.

```
> a1;
◂ ▶(12) [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 45]
> a1.pop();
◂ 45
> a1;
◂ ▶(11) [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
> s1;
◂ ▶(7) ["Hello", "wassup", "is", "everything", "alright", "?", "i"]
> s1.pop();
  s1;
◂ ▶(6) ["Hello", "wassup", "is", "everything", "alright", "?"]
>
```

7.Reduce()

The reduce() method reduces the array to a single value.

```
> a1;
◂ ▶(11) [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
> var res=0;
  res=a1.reduce(function(res,item){
  return res+item;});
  res;
◂ 385
```

8.reduceRight()

The reduceRight() method reduces the array to a single value

```
> var res=0;
  res=a1.reduceRight(function(res,item){
  return res+item;});
  res;
◂ 385
>
```

9.reverse():

The reverse() method reverses the order of the elements in an array.

```
> a1;
  a1.reverse();
  a1;
◂ ▸(11) [60, 55, 50, 45, 40, 35, 30, 25, 20, 15, 10]
> a1;
  a1.reverse();
◂ ▸(11) [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
> a1;

◂ ▸(11) [10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
> a1.reverse();
◂ ▸(11) [60, 55, 50, 45, 40, 35, 30, 25, 20, 15, 10]
```

10.Shift():

The shift() method removes the first item of an array.

```
a1;
▸(11) [60, 55, 50, 45, 40, 35, 30, 25, 20, 15, 10]
a1.shift();
60
a1;
▸(10) [55, 50, 45, 40, 35, 30, 25, 20, 15, 10]
```

11.Slice()

The slice() method returns the selected elements in an array, as a new array object

```
> a1;
◂ ▸(10) [55, 50, 45, 40, 35, 30, 25, 20, 15, 10]
> a1.slice(3,8);
◂ ▸(5) [40, 35, 30, 25, 20]
>
```

12.some()

The some() method checks if any of the elements in an array pass a test (provided as a function).

```
> a1;
▶ (10) [55, 50, 45, 40, 35, 30, 25, 20, 15, 10]
> function myfunc(item){
      return item >=30;}
  a1.some(myfunc);
true
> function myfunc(item){
      return item >=80;}
  a1.some(myfunc);
false
>
```

13.Sort()

The sort() method sorts the items of an array.The sort order can be either alphabetic or numeric, and either ascending (up) or descending (down).

```
> s1;
▶ (6) ["Hello", "wassup", "is", "everything", "alright", "?"]
> s1.sort();
▶ (6) ["?", "Hello", "alright", "everything", "is", "wassup"]
>
```

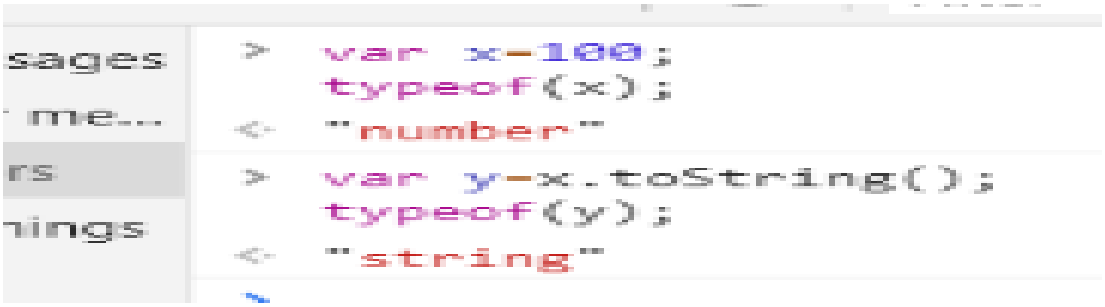
14.Splice()

The splice() method adds/removes items to/from an array, and returns the removed item(s).

```
> s1;
  s1.splice(2,0,"NO","Ofcourse");
  s1.splice(2,2,"seriously","yes");
▶ (2) ["NO", "Ofcourse"]
> var s1 = ["Hello", "wassup","is","everything","alright","?"];
  s1.splice(2,0,"NO","Ofcourse");
  s1;
  s1.splice(2,2,"seriously","yes");
  s1;
▶ (8) ["Hello", "wassup", "seriously", "yes", "is", "everything", "alright", "?"]
>
```

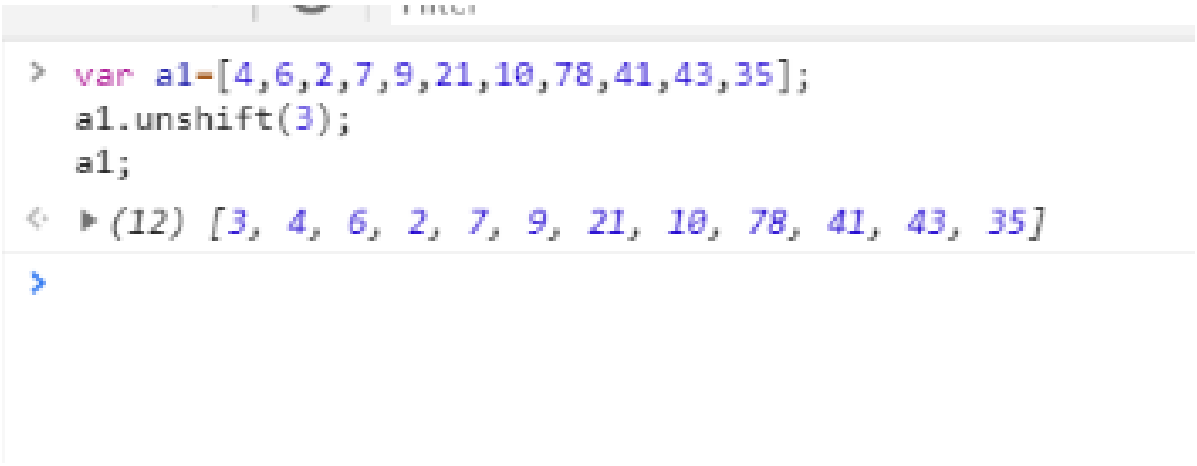
15.tostring()

The tostring() method returns a string with all the array values, separated by commas.



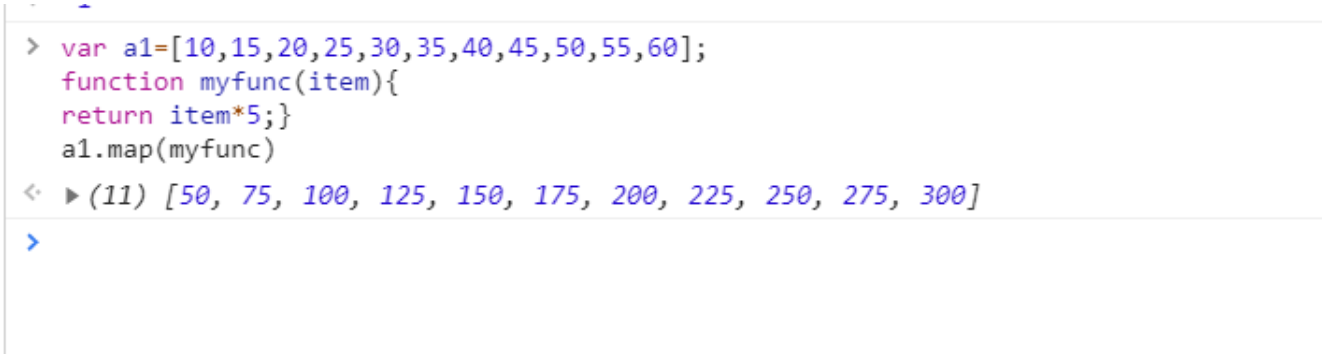
16. Unshift()

The unshift() method adds new items to the beginning of an array, and returns the new length



17.map()

The map() method creates a new array with the results of calling a function for every array element.



18.push()

The push() method adds new items to the end of an array, and returns the new length.

```
> var a1=[4,6,2,7,9,21,10,78,41,43,35];
  a1.push(100);
  a1;
◂ ▶ (12) [4, 6, 2, 7, 9, 21, 10, 78, 41, 43, 35, 100]
>
```

19.filter()

The filter() method creates an array filled with all array elements that pass a test (provided as a function)

```
> var a1=[4,6,2,7,9,21,10,78,41,43,35];
  a1.push(100);
  a1;
◂ ▶ (12) [4, 6, 2, 7, 9, 21, 10, 78, 41, 43, 35, 100]
> var a1=[10,15,20,25,30,35,40,45,50,55,60];
  function eval(item){
      return item%2==0;}
  a1.filter(eval);
◂ ▶ (6) [10, 20, 30, 40, 50, 60]
>
```

20. Difference between \r and \n ?

The \n moves to the beginning of the next line. The **n** stands for "new line",

The \r moves to the beginning of the current line. The **r** stands for "return" or "carriage return"