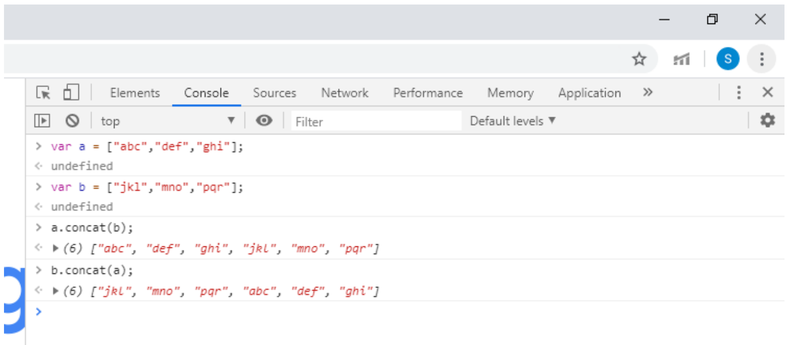
Assignment Q1 and Q2 by Sheenam Yadav .

a) concat()

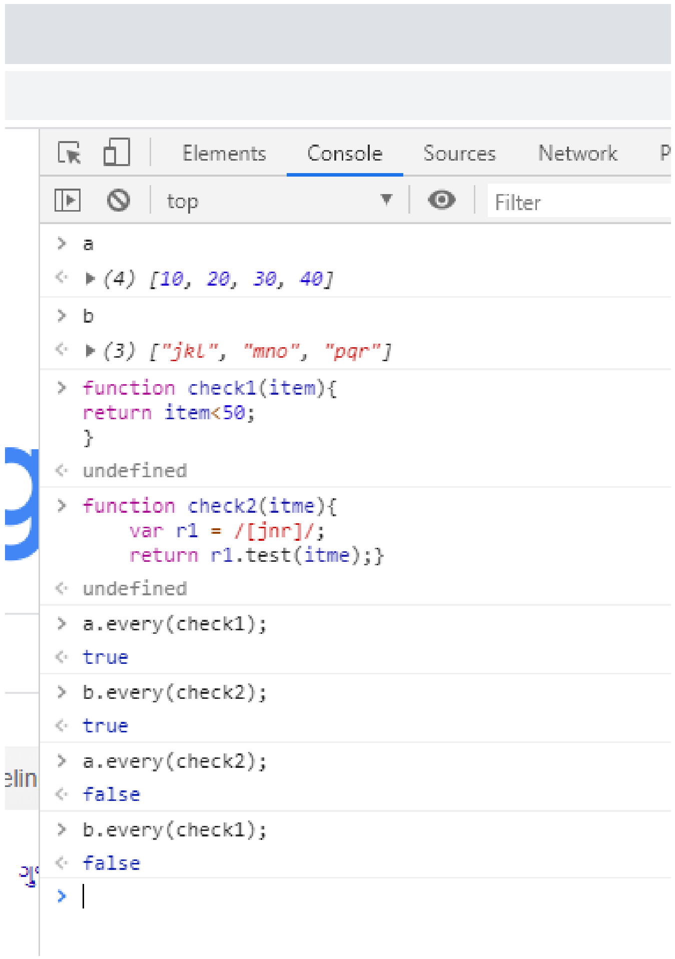
```
var a = ["abc","def","ghi"];
undefined
var b = ["jkl","mno","pqr"];
undefined
a.concat(b);
 ["abc", "def", "ghi", "jkl", "mno", "pqr"]
b.concat(a);
 ["jkl", "mno", "pqr", "abc", "def", "ghi"]
```



```
> var a = ["abc","def","ghi"];
< undefined
> var b = ["jkl","mno","pqr"];
< undefined
> a.concat(b);
< ▶(6) ["abc", "def", "ghi", "jkl", "mno", "pqr"]
> b.concat(a);
< ▶(6) ["jkl", "mno", "pqr", "abc", "def", "ghi"]
>
```

b) every()

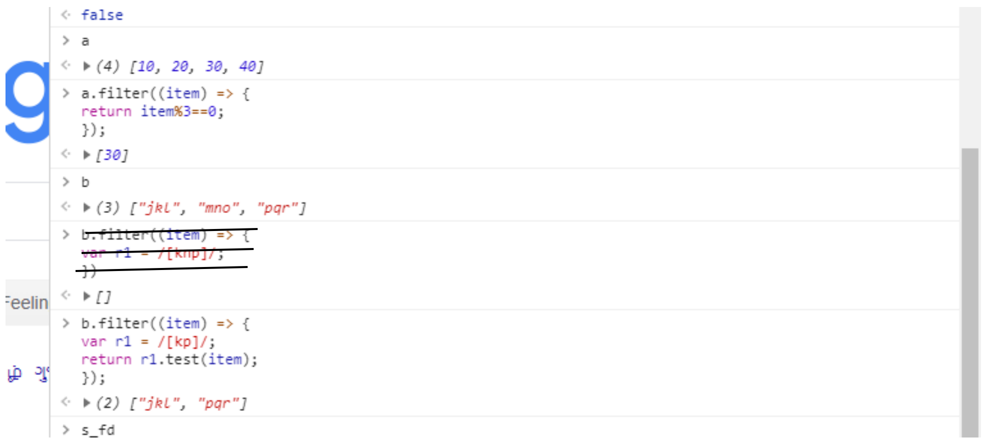
```
a
[10, 20, 30, 40]
b
 ["jkl", "mno", "pqr"]
function check1(item){
return item<50;
}
undefined
function check2(itme){
    var r1 = /[jnr]/;
    return r1.test(itme);}
undefined
a.every(check1);
true
b.every(check2);
true
a.every(check2);
false
b.every(check1);
false
```



```
> a
< ▶(4) [10, 20, 30, 40]
> b
< ▶(3) ["jkl", "mno", "pqr"]
> function check1(item){
  return item<50;
  }
< undefined
> function check2(itme){
      var r1 = /[jnr]/;
      return r1.test(itme);}
< undefined
> a.every(check1);
< true
> b.every(check2);
< true
> a.every(check2);
< false
> b.every(check1);
< false
> |
```

c) filter()

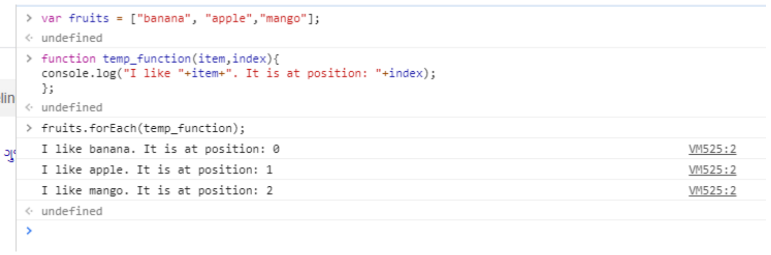
```
a
[10, 20, 30, 40]
a.filter((item) => {
return item%3==0;
});
[30]
b
 ["jkl", "mno", "pqr"]
b.filter((item) => {
var r1 = /[kp]/;
return r1.test(item);
});
["jkl", "pqr"]
```



```
< false
> a
< ▶(4) [10, 20, 30, 40]
> a.filter((item) => {
  return item%3==0;
  });
< ▶[30]
> b
< ▶(3) ["jkl", "mno", "pqr"]
> ~~b.filter((item) => {~~
  ~~var r1 = /[knp]/;~~
  ~~})~~
< ▶[]
> b.filter((item) => {
  var r1 = /[kp]/;
  return r1.test(item);
  });
< ▶(2) ["jkl", "pqr"]
> s_fd
```

d) forEach()

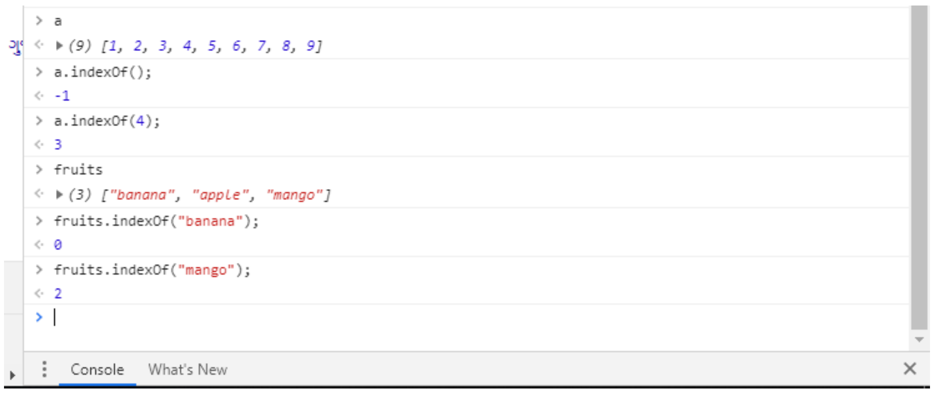
```
var fruits = ["banana", "apple","mango"];
function temp_function(item,index){
console.log("I like "+item+". It is at position
};
fruits.forEach(temp_function);
 I like banana. It is at position: 0
 I like apple. It is at position: 1
 I like mango. It is at position: 2
```



```
> var fruits = ["banana", "apple","mango"];
< undefined
> function temp_function(item,index){
  console.log("I like "+item+". It is at position: "+index);
  };
< undefined
> fruits.forEach(temp_function);
  I like banana. It is at position: 0                                  VM525:2
  I like apple. It is at position: 1                                   VM525:2
  I like mango. It is at position: 2                                   VM525:2
< undefined
>
```

e) indexOf()

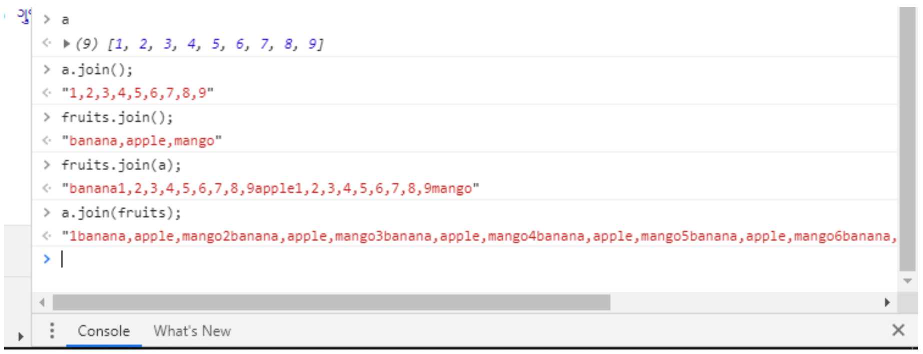
```
a
 [1, 2, 3, 4, 5, 6, 7, 8, 9]
a.indexOf();
-1
a.indexOf(4);
3
fruits
["banana", "apple", "mango"]
fruits.indexOf("banana");
0
fruits.indexOf("mango");
2
```



```
> a
< ▸(9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
> a.indexOf();
< -1
> a.indexOf(4);
< 3
> fruits
< ▸(3) ["banana", "apple", "mango"]
> fruits.indexOf("banana");
< 0
> fruits.indexOf("mango");
< 2
> |
```

f) join()

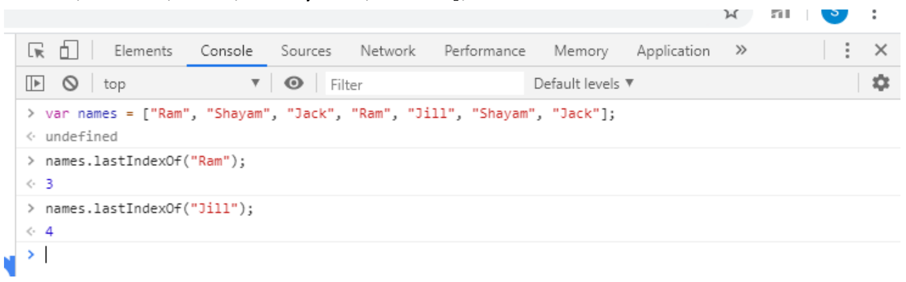
```
a
[1, 2, 3, 4, 5, 6, 7, 8, 9]
a.join();
"1,2,3,4,5,6,7,8,9"
fruits.join();
"banana,apple,mango"
fruits.join(a);
"banana1,2,3,4,5,6,7,8,9apple1,2,3,4,5,6,7,8,9mango"
a.join(fruits);
"1banana,apple,mango2banana,apple,mango3banana,apple,mango4banana,apple,mango5bana
na,apple,mango6banana,apple,mango7banana,apple,mango8banana,apple,mango9"
```



```
> a
< ▸(9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
> a.join();
< "1,2,3,4,5,6,7,8,9"
> fruits.join();
< "banana,apple,mango"
> fruits.join(a);
< "banana1,2,3,4,5,6,7,8,9apple1,2,3,4,5,6,7,8,9mango"
> a.join(fruits);
< "1banana,apple,mango2banana,apple,mango3banana,apple,mango4banana,apple,mango5banana,apple,mango6banana,
> |
```

g) lastIndexOf() *//starts searching for a given element from the end of the array.*

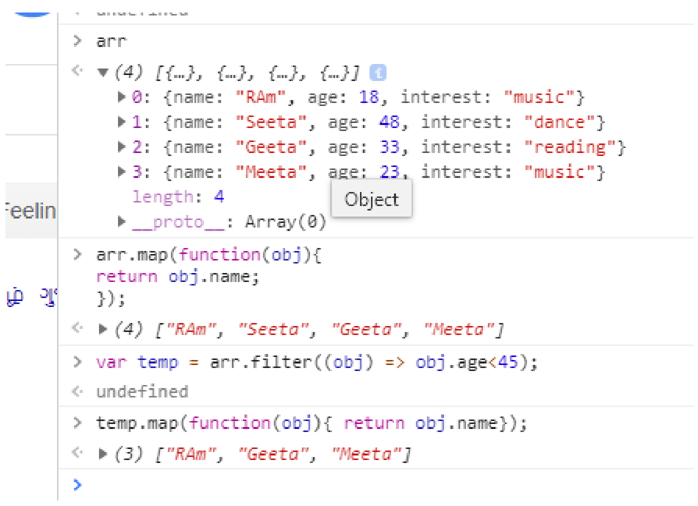
```
var names = ["Ram", "Shayam", "Jack", "Ram", "Jill", "Shayam", "Jack"];
undefined
names.lastIndexOf("Ram");
3
names.lastIndexOf("Jill");
4
```



```
> var names = ["Ram", "Shayam", "Jack", "Ram", "Jill", "Shayam", "Jack"];
< undefined
> names.lastIndexOf("Ram");
< 3
> names.lastIndexOf("Jill");
< 4
> |
```

h) map()

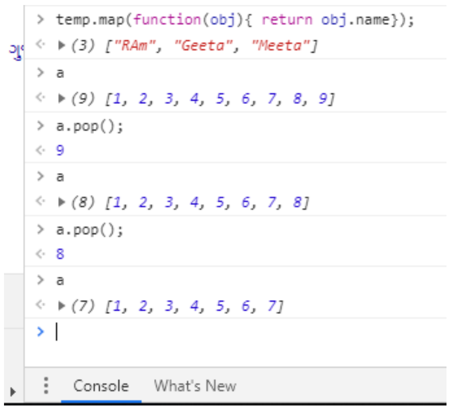
```
arr
0: {name: "RAm", age: 18, interest: "music"}
1: {name: "Seeta", age: 48, interest: "dance"}
2: {name: "Geeta", age: 33, interest: "reading"}
3: {name: "Meeta", age: 23, interest: "music"}
length: 4
__proto__: Array(0)
arr.map(function(obj){
return obj.name;
});
(4) ["RAm", "Seeta", "Geeta", "Meeta"]
var temp = arr.filter((obj) => obj.age<45);
undefined
temp.map(function(obj){ return obj.name});
(3) ["RAm", "Geeta", "Meeta"]
```



```
> arr
< ▼(4) [{…}, {…}, {…}, {…}]
    ▶0: {name: "RAm", age: 18, interest: "music"}
    ▶1: {name: "Seeta", age: 48, interest: "dance"}
    ▶2: {name: "Geeta", age: 33, interest: "reading"}
    ▶3: {name: "Meeta", age: 23, interest: "music"}
     length: 4
    ▶__proto__: Array(0)
> arr.map(function(obj){
  return obj.name;
  });
< ▶(4) ["RAm", "Seeta", "Geeta", "Meeta"]
> var temp = arr.filter((obj) => obj.age<45);
< undefined
> temp.map(function(obj){ return obj.name});
< ▶(3) ["RAm", "Geeta", "Meeta"]
>
```

i) pop()

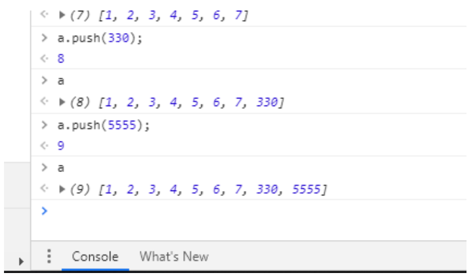
```
a
(9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
a.pop();
9
a
(8) [1, 2, 3, 4, 5, 6, 7, 8]
a.pop();
8
a
(7) [1, 2, 3, 4, 5, 6, 7]
```



```
> temp.map(function(obj){ return obj.name});
< ▶(3) ["RAm", "Geeta", "Meeta"]
> a
< ▶(9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
> a.pop();
< 9
> a
< ▶(8) [1, 2, 3, 4, 5, 6, 7, 8]
> a.pop();
< 8
> a
< ▶(7) [1, 2, 3, 4, 5, 6, 7]
> |
```

j) push()

```
a.push(330);
8
a
(8) [1, 2, 3, 4, 5, 6, 7, 330]
a.push(5555);
9
a
(9) [1, 2, 3, 4, 5, 6, 7, 330, 5555]
```

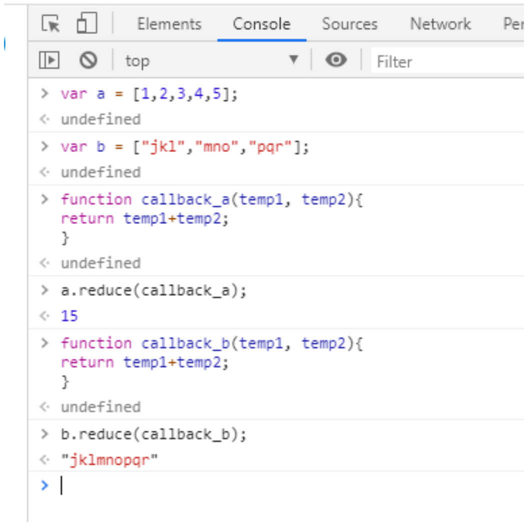


```
< ▶(7) [1, 2, 3, 4, 5, 6, 7]
> a.push(330);
< 8
> a
< ▶(8) [1, 2, 3, 4, 5, 6, 7, 330]
> a.push(5555);
< 9
> a
< ▶(9) [1, 2, 3, 4, 5, 6, 7, 330, 5555]
>
```

k) reduce()

In call back function temp1 (accumulator) is the value obtaind by applying the operation specified in callback function and temp2 is the currentvalue…it one by one assumes all the values of given array.

```
var a = [1,2,3,4,5];
var b = ["jkl","mno","pqr"];
function callback_a(temp1, temp2){
return temp1+temp2;
}
a.reduce(callback_a);
15
function callback_b(temp1, temp2){
return temp1+temp2;
}
b.reduce(callback_b);
"jklmnopqr"
```

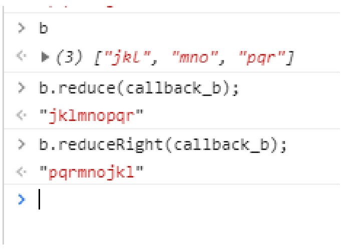


```
> var a = [1,2,3,4,5];
< undefined
> var b = ["jkl","mno","pqr"];
< undefined
> function callback_a(temp1, temp2){
  return temp1+temp2;
  }
< undefined
> a.reduce(callback_a);
< 15
> function callback_b(temp1, temp2){
  return temp1+temp2;
  }
< undefined
> b.reduce(callback_b);
< "jklmnopqr"
>
```

l) reduceRight()

it is same as right()…only difference is that it starts accumulating from right side.

```
b
 ["jkl", "mno", "pqr"]
b.reduce(callback_b);
"jklmnopqr"
b.reduceRight(callback_b);
"pqrmnojkl"
```

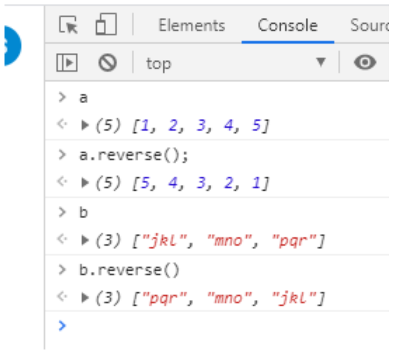


```
> b
< ▸(3) ["jkl", "mno", "pqr"]
> b.reduce(callback_b);
< "jklmnopqr"
> b.reduceRight(callback_b);
< "pqrmnojkl"
>
```

m) reverse()

inplace reverse.

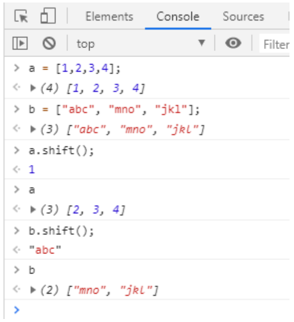
```
a
(5) [1, 2, 3, 4, 5]
a.reverse();
(5) [5, 4, 3, 2, 1]
b
(3) ["jkl", "mno", "pqr"]
b.reverse()
(3) ["pqr", "mno", "jkl"]
```



```
> a
< ▸(5) [1, 2, 3, 4, 5]
> a.reverse();
< ▸(5) [5, 4, 3, 2, 1]
> b
< ▸(3) ["jkl", "mno", "pqr"]
> b.reverse()
< ▸(3) ["pqr", "mno", "jkl"]
>
```

n) shift() *remves first element from left and return it.*

```
a = [1,2,3,4];
(4) [1, 2, 3, 4]
b = ["abc", "mno", "jkl"];
(3) ["abc", "mno", "jkl"]
a.shift();
1
a
(3) [2, 3, 4]
b.shift();
"abc"
b
(2) ["mno", "jkl"]
```



```
> a = [1,2,3,4];
< ▸(4) [1, 2, 3, 4]
> b = ["abc", "mno", "jkl"];
< ▸(3) ["abc", "mno", "jkl"]
> a.shift();
< 1
> a
< ▸(3) [2, 3, 4]
> b.shift();
< "abc"
> b
< ▸(2) ["mno", "jkl"]
>
```

o) slice()

shallow copy of sliced array from 'strt' to 'end-1'

var names = ["Ram", "Shayam", "Jack", "Ram", "Jill", "Shayam", "Jack"];

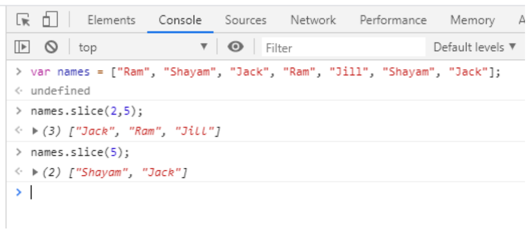
undefined

names.slice(2,5);

(3) ["Jack", "Ram", "Jill"]

names.slice(5);

(2) ["Shayam", "Jack"]



```
> var names = ["Ram", "Shayam", "Jack", "Ram", "Jill", "Shayam", "Jack"];
<· undefined
> names.slice(2,5);
<· ▶(3) ["Jack", "Ram", "Jill"]
> names.slice(5);
<· ▶(2) ["Shayam", "Jack"]
> |
```

p) some()

checks specified condition for all elemts of array and if any on element satisfies the condition it wil return true.

var names = ["Ram", "Shayam", "Jack", "Ram", "Jill", "Shayam", "Jack"];

names.some((item) => {

return item=="Jack"

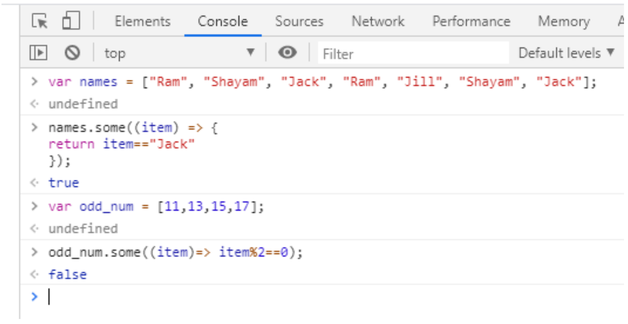
});

true

var odd_num = [11,13,15,17];

odd_num.some((item)=> item%2==0);

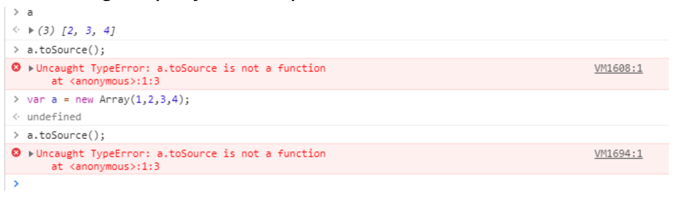
false



```
> var names = ["Ram", "Shayam", "Jack", "Ram", "Jill", "Shayam", "Jack"];
<· undefined
> names.some((item) => {
  return item=="Jack"
  });
<· true
> var odd_num = [11,13,15,17];
<· undefined
> odd_num.some((item)=> item%2==0);
<· false
> |
```

q) toSource()

it is a non-standard function, returns originally defined array.



```
> a
<· ▶(3) [2, 3, 4]
> a.toSource();
⊗ ▶Uncaught TypeError: a.toSource is not a function                VM1608:1
      at <anonymous>:1:3
> var a = new Array(1,2,3,4);
<· undefined
> a.toSource();
⊗ ▶Uncaught TypeError: a.toSource is not a function                VM1694:1
      at <anonymous>:1:3
>
```

Not working in the browser.

r) sort()

for sorting array elements. Returns sorted array.

names.sort();

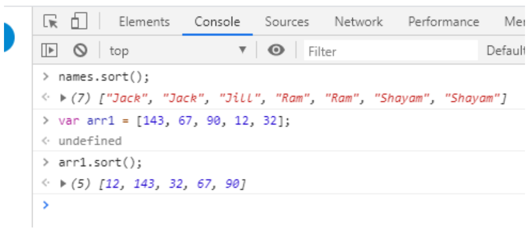
(7) ["Jack", "Jack", "Jill", "Ram", "Ram", "Shayam", "Shayam"]

var arr1 = [143, 67, 90, 12, 32];

undefined

arr1.sort();

(5) [12, 143, 32, 67, 90]

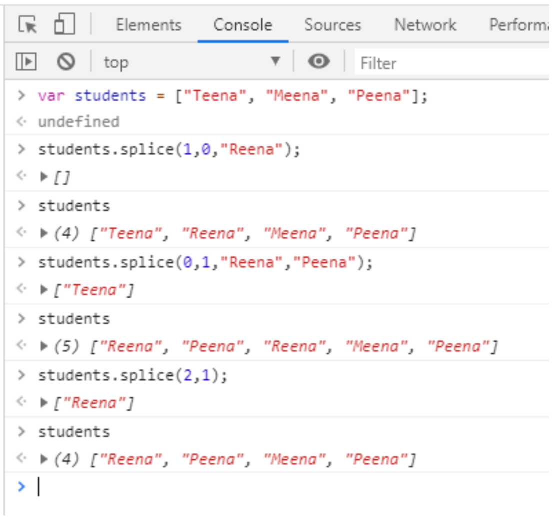


```
> names.sort();
<· ▶(7) ["Jack", "Jack", "Jill", "Ram", "Ram", "Shayam", "Shayam"]
> var arr1 = [143, 67, 90, 12, 32];
<· undefined
> arr1.sort();
<· ▶(5) [12, 143, 32, 67, 90]
>
```

s) splice()

a.splice(no_of_elements_to_insert (index_of_the_elements_to_be_deleted), no_of_elemets_to_be_deleted , list_of_elements_to_insert); //if list_of_elements_to_insert is not given it will only remove and not add anything.

```
var students = ["Teena", "Meena", "Peena"];
undefined
students.splice(1,0,"Reena");
[]
students
(4) ["Teena", "Reena", "Meena", "Peena"]
students.splice(0,1,"Reena","Peena");
["Teena"]
students
(5) ["Reena", "Peena", "Reena", "Meena", "Peena"]
students.splice(2,1);
["Reena"]
students
(4) ["Reena", "Peena", "Meena", "Peena"]
```

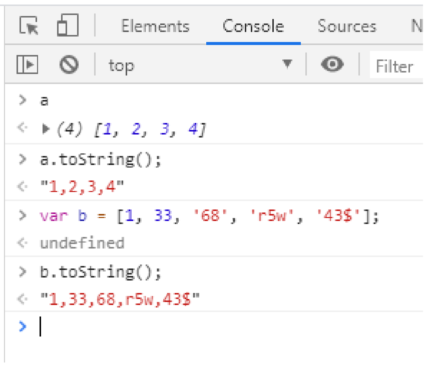


```
> var students = ["Teena", "Meena", "Peena"];
<· undefined
> students.splice(1,0,"Reena");
<· ▶ []
> students
<· ▶ (4) ["Teena", "Reena", "Meena", "Peena"]
> students.splice(0,1,"Reena","Peena");
<· ▶ ["Teena"]
> students
<· ▶ (5) ["Reena", "Peena", "Reena", "Meena", "Peena"]
> students.splice(2,1);
<· ▶ ["Reena"]
> students
<· ▶ (4) ["Reena", "Peena", "Meena", "Peena"]
> |
```

t) toString()

returns a string of the elements of specified array.

```
a
(4) [1, 2, 3, 4]
a.toString();
"1,2,3,4"
var b = [1, 33, '68', 'r5w', '43$'];
undefined
b.toString();
"1,33,68,r5w,43$"
```

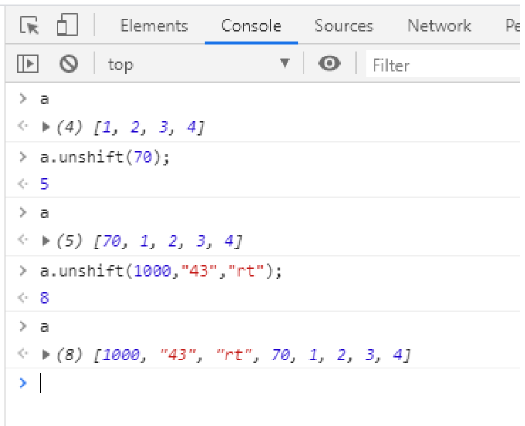


```
> a
<· ▶ (4) [1, 2, 3, 4]
> a.toString();
<· "1,2,3,4"
> var b = [1, 33, '68', 'r5w', '43$'];
<· undefined
> b.toString();
<· "1,33,68,r5w,43$"
> |
```

u) unshift()

used to add elements to the beginning of the array. It returns the size of array afeter inserting elements.

```
a
(4) [1, 2, 3, 4]
a.unshift(70);
5
a
(5) [70, 1, 2, 3, 4]
a.unshift(1000,"43","rt");
8
a
(8) [1000, "43", "rt", 70, 1, 2, 3, 4]
```



```
> a
<· ▶ (4) [1, 2, 3, 4]
> a.unshift(70);
<· 5
> a
<· ▶ (5) [70, 1, 2, 3, 4]
> a.unshift(1000,"43","rt");
<· 8
> a
<· ▶ (8) [1000, "43", "rt", 70, 1, 2, 3, 4]
> |
```

Q2. What is the difference between '\n' new line(line feed) and '\r' carriage return.

Answer-

\n has ASCII code 10 and \r has ASCII code 13.

Special ASCII characters were used to tell the printers what to do, '\r' was used to move to the left side of the paper, whereas '\n' is used to move to next line.

Most of the operating systems use '\n' or '\n\r' or '\r\n'.