Task 1:

```
> var arr1 = [1,2,3,4];
< undefined
> var arr2 = [5,6,7,8];
< undefined

> var final_arr = arr1.concat(arr2);
< undefined
> final_arr
< (8) [1, 2, 3, 4, 5, 6, 7, 8]

> every_arr = final_arr.every(function (ar) {
    return ar > 0;
 });
< true

> filter_arr = final_arr.filter(function (ar) {
    return ar > 5;
 });
< (3) [6, 7, 8]

> foreach_arr = filter_arr.forEach(function (ar) {
    console.log(ar);
 });
< 6
   7
   8
   undefined

> indexofelem = filter_arr.indexOf(7);
< 1

> join_arr = filter_arr.join(6);
< "66768"

> var lio = final_arr.concat(filter_arr);
< undefined
> lio
< (11) [1, 2, 3, 4, 5, 6, 7, 8, 6, 7, 8]
> lastindexof_elem = lio.lastIndexOf(7);
< 9

> filter_arr;
< (3) [6, 7, 8]
> map_arr = filter_arr.map(ar => Math.floor(Math.PI * ar));
< (3) [18, 21, 25]
```

```
> pop_elem = lio.pop();
< 8
> lio;
< (10) [1, 2, 3, 4, 5, 6, 7, 8, 6, 7]

> final_arr;
< (8) [1, 2, 3, 4, 5, 6, 7, 8]
> push_arr = final_arr.push(9, 10);
< 10
> final_arr;
< (10) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

> reduce_elem = final_arr.reduce(function (ar1, ar2) {
    return ar1 + ar2;
 });
< 55

> reduce_elem = final_arr.reduceRight(function (ar1, ar2) {
    return ar1 - ar2;
 });
< -35

> reverse_arr = final_arr.reverse();
< (10) [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]

> shift_arr = reverse_arr.shift();
< 10
> reverse_arr;
< (9) [9, 8, 7, 6, 5, 4, 3, 2, 1]

> final_arr;
< (9) [9, 8, 7, 6, 5, 4, 3, 2, 1]
> slice_arr = final_arr.slice(2, 5);
< (3) [7, 6, 5]

> some_arr = final_arr.some(function (ar) {
    return ar < 3;
 });
< true

>>>>toSource is not working on chrome.

> sort_arr = final_arr.sort();
< (9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
> splice_arr = final_arr.splice(0, 4);
< (4) [1, 2, 3, 4]
> final_arr
< (5) [5, 6, 7, 8, 9]

> tostring_arr = final_arr.toString();
< "5,6,7,8,9"

> unshift_arr = final_arr.unshift(1,2,3,4);
< 9
> final_arr
< (9) [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Task 2:

```
> var add1 = (function ()
    {var counter = 0; console.log(counter); return function () {return counter +=  1;}})()
    add1();
    add1();
    add1();
0
< 3
```

In this question, the answer returned is 3 since we are calling the add1 function 3 times. For simplicity, we will call the outside function as funcA and the inside function as funcB(the function which is in the return statement of add1 function). Here, funcA is getting executed only the first time and then funcB is executed after that. The expression or code in the funcA is replaced by funcB. Now, the counter will be updated by +1. Next, we are calling add1 two more times. Now, funcA contains (counter += 1), so counter will be 3 after all the add1 functions are executed. Like you can see in the following example, it's 11(1+5+5) following the above concept.

Example -

```
> var counter = 0; counter += 1;
    var add1 = (function () {  console.log(counter); return function () {return counter += 5;}})()
    add1();
    add1();
1
< 11
```

Task 3:

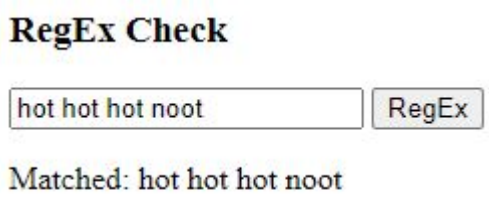
Difference between '\n' and '\r' -

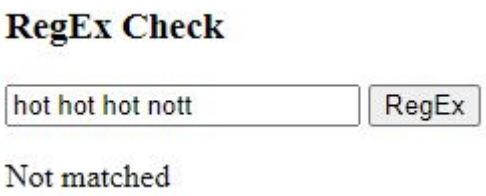
New line can be used as a beginning of a new sentence in a different line or as a beginning of a paragraph.

Whereas, carriage return was used to denote the continuation of a previous line/previous sentence.

Task 4:

Check Regex - (Filename - RegEx.html)





Array Methods - (Filename - ArrayMethods.html, myFunction.js)
Input - Use comma separated numbers