JavaScript Assignment

Q1)Give examples for each function:

1)concat()

```
> var ar1=[1,2,3]
  var ar2=[4,5,6]
  var ar3 = ar1.concat(ar2)
  console.log(ar3)
  ▶ (6) [1, 2, 3, 4, 5, 6]
```

2) every()

```
> var ar1=[0,2,4]
  var res = ar1.every((val)=>val%2==0)
  console.log(res)
  true
```

3) filter()

```
> var ar1=[-3,-2,-1,0,1,2,3]
  var ar2=[4,5,6]
  var res = ar1.filter((val)=>val<0)
  console.log(res)
  ▶ (3) [-3, -2, -1]
```

4) forEach()

```
var ar1=[0,2,4]
ar1.forEach((val)=>console.log(val*2))
0                                    VM764:2
4                                    VM764:2
8                                    VM764:2
undefined
```

5) indexOf()

```
> var ar1=[0,2,4,2]
  ar1.indexOf(2)
< 1
```

6) join()

```
> var ar1=[2,4,2]
  console.log(ar1.join())
  console.log(ar1.join(""))
  2,4,2
  242
```

7) lastIndexOf()

```
> var ar1=[0,2,4,2]
  ar1.lastIndexOf(2)
< 3
```

8) map()

```
> var ar1=[1,2,3]
  var ar2 = ar1.map((val)=>val*3)
  console.log(ar2)
  ▶ (3) [3, 6, 9]
```

9) pop()

```
> var ar1=[1,2,3]
  ar1.pop()

< 3
> console.log(ar1)
  ▶ (2) [1, 2]
```

10) push()

```
> var ar1=[1,2,3]
  ar1.push(4)
  console.log(ar1)
  ▶ (4) [1, 2, 3, 4]
```

11) reduce()

```
> const sum2 = (accumulator,val) =>{
  return accumulator+val;
  }
  ar1=[1,2,3,4]
  ar1.reduce(sum2,0)
< 10
```

12) reduceRight()

```
> var revConcate5 = (accumulator,val) =>{
  return accumulator + val.toString();
  }
  ar1=[1,2,3,4]
  ar1.reduceRight(revConcate5,'')
< "4321"
```

13) reverse()

```
> var ar1=[1,2,3]
  var rev_arr = ar1.reverse()
  console.log(rev_arr)
  ▶ (3) [3, 2, 1]
```

14) shift()

```
> var ar1=[1,2,3]
  console.log(ar1.shift())
  1
  undefined
> console.log(ar1)
```

15) slice()

```
> var ar1=[1,2,3,5,6,7]
  var ar2 = ar1.slice(2,4)
  console.log(ar2)

  ▶ (2) [3, 5]
```

16) some()

```
> var ar1=[1,3,5]
  ar1.some((val)=>val%7==0)

  false
> var ar1=[1,3,5]
  ar1.some((val)=>val%3==0)

  true
```

17)sort

```
> var ar1=[8,3,15,2]
  ar1.sort((a,b)=>a-b)
  console.log(ar1)

  ▶ (4) [2, 3, 8, 15]
```

18) splice

```
> ar1=[1,2,3,4,5]
  ar1.splice(1,0,"hehe","haha")
  ▶ []
> ar1
  ▶ (7) [1, "hehe", "haha", 2, 3, 4, 5]
```

--

```
> ar1=[1,2,3,4,5]
  ar1.splice(1,2)
  ▶ (2) [2, 3]
> ar1
  ▶ (3) [1, 4, 5]
>
```

19) toString()

```
> var ar1=['b',3,'a',2]
  ar1.toString()

  "b,3,a,2"
>
```

20) unshift()

```
> var ar1=[8,3,15,2]
  ar1.unshift(0,1)

<- 6
> ar1
<- ▶ (6) [0, 1, 8, 3, 15, 2]
```

Q2) What is the difference between \n and \r?

Previously(in the era of typewriters), \n was meant to bring the carriage down to the next line while \r was meant to bring the carriage to the left most position. So \r\n was used together after the end of line.

In current context \r is kind of obsolete:

- Unix and new macs use \n for End of Line
- Older macs use \r
- Windows use \r\n for End of Line

Solution 3) question3.html(contains inline js)