

JavaScript Assignment

Important

Please find the webpage for the assignment hosted at <https://bit.ly/js-vishnu>

Question 2 (Array Methods):

- Concat()

```
> var myGirls = ["Cecilie", "Lone"];
  var myBoys = ["Emil", "Tobias", "Linus"];
  var myChildren = myGirls.concat(myBoys);
< undefined
> myChildren
<=> (5) ["Cecilie", "Lone", "Emil", "Tobias", "Linus"]
> |
```

Figure 1 Screenshot showing concat()

- Every()

```
> var ages = [32, 33, 16, 40];
 
  function checkAdult(age) {
    return age >= 18;
  }
< undefined
> ages.every(checkAdult)
<=> false
>
```

Figure 2 Working of every()

- Filter()

```
> var ages = [32, 33, 16, 40];
  function checkAdult(age) {
    return age >= 18;
}
<- undefined
> ages.filter(checkAdult)
<- ▶ (3) [32, 33, 40]
```

Figure 3 Working on filter()

- forEach()

```
> var fruits = ["apple", "orange", "cherry"];
  fruits.forEach(myFunction);
  function myFunction(item, index) {
    console.log( index + ":" + item + "<br>");
  }
0:apple<br>                                         VM161:4
1:orange<br>                                         VM161:4
2:cherry<br>                                         VM161:4
```

Figure 4 Working of ForEach()

- indexOf()

```
> var str = "Hello world, welcome to the universe.";
  var n = str.indexOf("welcome");
  console.log(n)
13                                              VM245:3
```

Figure 5 Working of indexOf()

- `join()`

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  var energy = fruits.join();
  energy
< "Banana,Orange,Apple,Mango"
```

Figure 6 Working of `Join()`

- `lastIndexOf()`

```
> var str = "Hello planet earth, you are a great planet.";
  var n = str.lastIndexOf("planet");
  n
< 36
```

Figure 7 Working of `lastIndexOf()`

- `map()`

```
> var numbers = [4, 9, 16, 25];
  var x = numbers.map(Math.sqrt)
  x
< ▶ (4) [2, 3, 4, 5]
```

Figure 8 Working of `map()`

- pop()

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.pop();
<- "Mango"
```

Figure 9 Working of POP()

- Push()

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.push("Kiwi");
<- 5
> fruits
<- ▶ (5) ["Banana", "Orange", "Apple", "Mango", "Kiwi"]
> |
```

Figure 10 Working of Push()

- Reducer()

```
> const array1 = [1, 2, 3, 4];
  const reducer = (accumulator, currentValue) => accumulator + currentValue;
  console.log(array1.reduce(reducer));
10
```

Figure 11 Working of Reduce ()

- `reduceRight()`

```
> var numbers = [175, 50, 25];
  function myFunc(total, num) {
    return total - num;
  }

  numbers.reduceRight(myFunc);
< -200
```

Figure 12 Working of `reduceRight()`

- `Reverse()`

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.reverse();
< ▶ (4) ["Mango", "Apple", "Orange", "Banana"]
```

Figure 13 Working of `Reverse()`

- `Shift()`

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.shift();
< "Banana"
```

Figure 14 Working of `Shift()`

- Slice()

```
> var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
  var citrus = fruits.slice(1, 3);
< undefined
> citrus
<=> (2) ["Orange", "Lemon"]
```

Figure 15 Working of Slice()

- Some()

```
> var ages = [3, 10, 18, 20];

function checkAdult(age) {
  return age >= 18;
}

ages.some(checkAdult)
<=> true
```

Figure 16 Working of Some()

- Sort()

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
  fruits.sort();
<=> (4) ["Apple", "Banana", "Mango", "Orange"]
> |
```

Figure 17 Working of sort()

- Splice()

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
   fruits.splice(2, 1, "Lemon", "Kiwi");
<- ► ["Apple"]
```

Figure 18 Working of Splice()

- toString()

```
> number=17
   number.toString()
<- "17"
```

Figure 19 Working of toString()

- unshift()

```
> var fruits = ["Banana", "Orange", "Apple", "Mango"];
   fruits.unshift("Lemon", "Pineapple");
<- 6
```

Figure 20 Working of unshift()

Screenshots:

The screenshot shows a web-based application interface with three main sections:

- Question 3**:
 - String starts with lion
 - String ends with
 - String has abc (b can be any number of times >1)

Lion is roaring

Result: true
Location: 0
- Question 4**:
 - Sort in ascending order
 - Multiply each number by 10
 - Return number divisible by 3

1,2,3,6,9

Result: 3,6,9
- Question 4 Explanation**:
 - Difference between == and === with example.

Answer: The == operator is capable to change the values being checked by performing some type conversions which JS thinks as human error but in case of ===, no conversions are performed and it is considered as a strict equality.

Like, 20=="20" will return **True** as RHS will be treated as number.

Whereas, 20==="20" will return **False** as RHS will be treated **strictly** as string.

Figure 21 Screenshot showing outputs of question 2, 3 & 4