Practical – 9

AIM : **Discuss & demonstrate the following Higher order function in JS.**
forEach()
map(),find(),filter() etc…
forEach() vs map() Use
Cases of Reduce() call(),
apply(), bind()

Source Code:

HTML Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Higher Order Functions</title>
</head>
<body>
<h3>Higher order functions</h3>
<p>forEach()</p>
<p id="func1"></p>
<p>map()</p>
<p id="func2"></p>
<p>find()</p>
<p id="func3"></p>
<p>filter()</p>
<p id="func4"></p>
<p>reduce()</p>
<p id="func5"></p>
<p>call()</p>
<p id="func6"></p>
<p>apply()</p>
<p id="func7"></p>
<p>bind()</p>
<p id="func8"></p>
<script src="./open.js"></script>
```

```
</body>
</html>
```

JS Code:

```
// forEach() const
number=[1,2,3,4,5]; let
sum=0;
number.forEach(fun1);
function fun1(numb){
sum += numb;
}
document.getElementById("func1").innerHTML=sum;
// map() const num=[10,20,30,40]; const
newNum=num.map(fun2); function fun2(num){ return
num*10; }
document.getElementById("func2").innerHTML=newNum;
// find() const ages = [3, 10, 18, 20];
document.getElementById("func3").innerHTML = ages.find(checkAge);
function checkAge(age) { return age > 18;
}
// filter()
const age = [32, 33, 16, 40];
document.getElementById("func4").innerHTML = age.filter(checkAdult);
function checkAdult(age) { return age >= 18; }
// reduce() const numbers = [12.5,5.7,8.1,13.8];
document.getElementById("func5").innerHTML = numbers.reduce(getSum, 0);
function getSum(total, num) { return total + Math.round(num);
}
// call()
const person = { fullName: function() {
return this.firstName + " " + this.lastName;
}
}
const person1 = {
firstName:"abc",
lastName: "abc"
} const person2 =
{
```

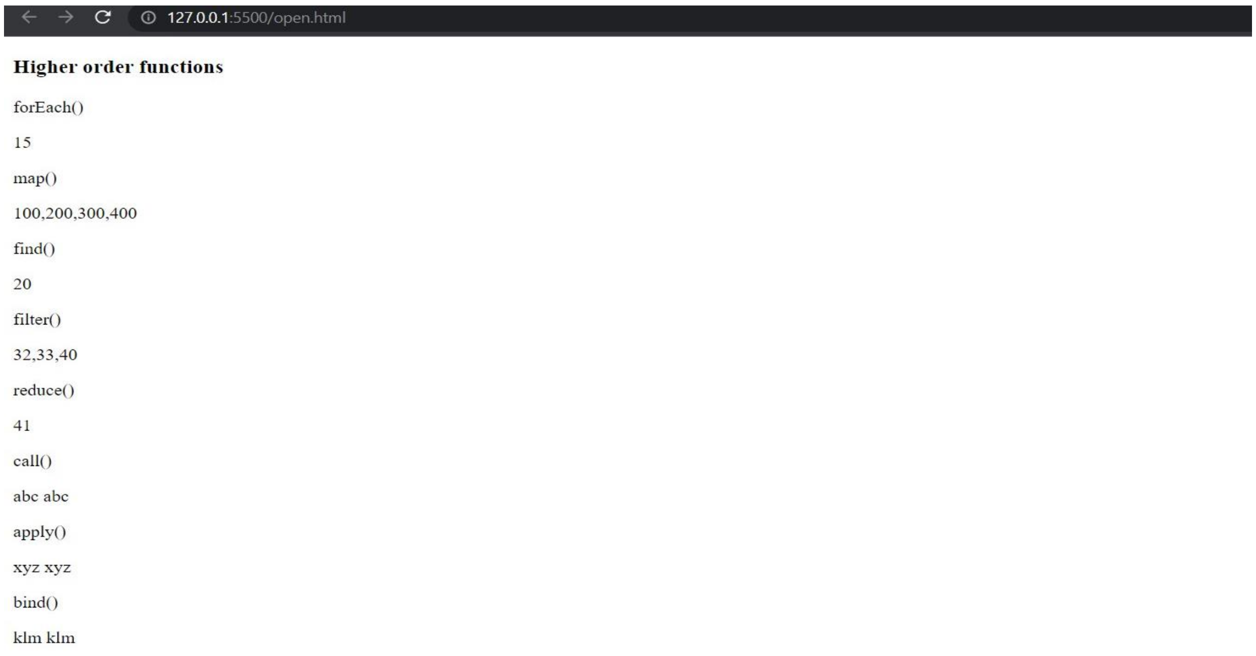
```
firstName:"mno",
lastName: "mno"
} document.getElementById("func6").innerHTML =
person.fullName.call(person1);
// apply()
const p = {
fullName: function() { return
this.firstName + " " + this.lastName;
}
}

const p1 = {
firstName: "xyz",
lastName: "xyz"
}

document.getElementById("func7").innerHTML=p.fullName.apply(p1);
// bind()
const p2 = {
firstName:"pqr", lastName: "pqr",
fullName: function () { return
this.firstName + " " + this.lastName;
}
}

const member = {
firstName:"klm",
lastName: "klm",
}

let fullName = person.fullName.bind(member);
document.getElementById("func8").innerHTML = fullName();
```
Output:



```
127.0.0.1:5500/open.html

Higher order functions

forEach()

15

map()

100,200,300,400

find()

20

filter()

32,33,40

reduce()

41

call()

abc abc

apply()

xyz xyz

bind()

klm klm
```

Conclusion:

A function which takes another function as an argument or returns a function is known as a higher order function.