Functional Programing

1) We returned one function from another.
2) We can pass a function to another function as a parameter.

```
var a = function(){
}
a();

function printName(cb, cb2) {
	console.log(“name”);
	cb();
	cb2();
}

function printAge() {
	console.log(24);
}

function printLastName() {
	console.log(“Taneja”);
}

printName(printAge, printLastName);
```

Callbacks: when you pass a function as a parameter to another function this becomes your callback function.

Pure functions and Impure functions

Pure function: Returns the same value always for the same parameters.

```
function sum(a, b){
  return a + b;
}
```

Impure function: Would not return the same value with the same parameters. Because of an external factor changing the value on every return.

```
var c = 0;
function sum(a, b){
  return a + b + c++;
}
sum(2,3);
```

```
let myRadiusArray = [2, 3, 4, 5, 8];
```

function when it returns it should return the area of a circle in an array.

```
function calculateArea(myRadiusArray) {
      let result = [];
   for(var i = 0; i< myRadiusArray.length; i++) {
      result.push(3.14* myRadiusArray[i] * myRadiusArray[i]);
   }
   return result;
}


function calculateCircumference(myRadiusArray) {
      let result = [];
   for(var i = 0; i< myRadiusArray.length; i++) {
      result.push(3.14 * myRadiusArray[i] * 2);
   }
   return result;
}


function calculateDiameter(myRadiusArray) {
      let result = [];
   for(var i = 0; i< myRadiusArray.length; i++) {
      result.push(myRadiusArray[i] * 2);
   }
```

```
    return result;
}
```

Not a **DRY** code.

DRY: Do not repeat yourself

KISS: keep it simple silly

Higher Order function

```
function circleArea(radius){
    return Math.PI * radius * radius;
}
function circleCircumference(radius){
    return 2 * Math.PI * radius;
}
function circleDiameter(radius){
    return 2 * radius;
}

function calculate(myRadiusArray, logic) {
        let result = [];
    for(var i = 0; i< myRadiusArray.length; i++) {
        result.push(logic(myRadiusArray[i]));
    }
    return result;
}

calculate(myRadiusArray, circleArea);
calculate(myRadiusArray, circleCircumference);
calculate(myRadiusArray, circleDiameter);
```

Map

```
var a = [1,2,3,4];
var b = [];
for(var i = 0; i< a.length; i++) {
	b.push(a[i] *2);
}
console.log(b);
```

Map
a.map(callback(item, index))

```
a.map(function(item, index) {
	console.log(item)
	console.log(index)
})
```

1. Map take an callback as a parameter
2. Callback has item and index as parameters
3. And it returns an array

```
var a = [1,2,3,4];
var x = a.map(function(item, index) {
    return item*2;
})
```

Using map improve this function

```
function calculateArea(myRadiusArray) {
        let result = [];
    for(var i = 0; i< myRadiusArray.length; i++) {
        result.push(3.14* myRadiusArray[i] * myRadiusArray[i]);
    }
    return result;
}
```

```
function calculateArea(myRadiusArray) {
        let result;
        result = myRadiusArray.map(function(r) {
                return 3.14* r* r
        })
        return result;
```

}

Homework:
what if i want to iterate from the last value to first value? like for (i=n-1 -> i=0)

Question

You are given a transaction array treat the transaction amount in rupees, and convert those amounts into dollars and conversion rate is also provided to us.

```
const transactions = [1000, 3000, 4000, 2000, - 898, 3800, -
4500];
const inrtToUsd = 80;


let conversionToDollars = transactions.map(function(amount){
    return amount / inrtToUsd;
})
console.log(conversionToDollars)
```

Filter

filter(cb(item))

I can filter based on a condition

```
let myArr = [1, 2, 5, 7, 8, 2, 6, 9, 13, 17]

return even numbers;

let evenArray = myArr.filter(function(num) {
	return num % 2 === 0;
})

console.log(evenArray)
```

I have list of transactions

```
const transactions = [1000, 3000, 4000, 2000, - 898, 3800, -
4500];


let positiveValue = transactions.filter(function(amount){
   return amount > 0;
})
console.log(positiveValue)
```

Question

Sum of the array

```
let arr = [1, 2, 3, 4, 5]
let sum = 0;
for(let  i = 0 ; i < arr.length ; i ++ ){
   sum = sum + arr[i]
}
console.log(sum)
```

Reduce: Reduce an array into one value using this method;

```
reduce(cb(acc, num), 0)
```

```
let arr = [1, 2, 3, 4, 5]
arr.reduce(function(acc, num) {
        acc = acc + num
        return acc;
}, 0)
```