Javascript Assignment 10

1. Are Higher Order functions and Call back functions the same ? If not,

briefly explain about both functions.

Ans: Higher-order functions and callback functions are related concepts in programming, but they are not the same thing. Let's briefly explain both:

1. Higher-Order Functions:

* A higher-order function is a function that takes one or more functions as arguments and/or returns a function as its result.
* In other words, it operates on functions, treating them as first-class citizens.
* Higher-order functions are a fundamental concept in functional programming and allow you to abstract over actions, making your code more modular and reusable.
* Examples of higher-order functions in JavaScript include **map**, **filter**, and **reduce**, which take functions as arguments to transform or process data.

Callback Functions:

* A callback function is a function that is passed as an argument to another function and is intended to be executed after a specific task or event has occurred.
* Callback functions are often used in asynchronous operations, event handling, and to define behavior that should occur when an operation is complete.
* They provide a way to control the flow of the program by specifying what should happen next.
* Common use cases for callback functions include handling asynchronous operations using functions like **setTimeout**, **fetch**, or handling events in event-driven programming.

2. Is filter a Higher Order function in Javascript ? If yes, why ?

Ans: Yes, **filter** is a higher-order function in JavaScript.

A higher-order function is a function that takes one or more functions as arguments and/or returns a function as its result. In the case of **filter**, it takes a function (a callback function) as an argument.

The **filter** function is used to filter elements from an array based on a given condition specified by the callback function. It iterates over each element in the array and passes each element to the callback function. The callback function should return **true** to include the element in the filtered result or **false** to exclude it.

const numbers = [1, 2, 3, 4, 5, 6];

// Using filter as a higher-order function with a callback

const evenNumbers = numbers.filter(function (num) {

return num % 2 === 0;

});

console.log(evenNumbers); // Output: [2, 4, 6]

3. Give an example of a Higher Order function and a call back function

used in the same program.

Ans:

// Higher-order function: forEach

// Callback function: logItem

function forEach(arr, callback) {

for (let i = 0; i < arr.length; i++) {

callback(arr[i], i);

}

}

// Callback function

function logItem(item, index) {

console.log(`Item at index ${index}: ${item}`);

}

const fruits = ["apple", "banana", "cherry", "date"];

// Using forEach as a higher-order function with the logItem callback

forEach(fruits, logItem);

4. Carefully check the example below:

a) What will be the output of this program ?

b) Which function is a Higher Order function here ?

const names= ['John', 'Tina','Kale','Max']

function useFunction(arr,fn){

for(let i=0; i<arr.length; i++){

fn(arr[I]);

}

}

function argFn (name){

console.log("Hello " + name );

}

useFunction(names,argFn);

Ans:

a)

* 1. Hello John
  2. Hello Tina
  3. Hello Kale
  4. Hello Max
* b) **useFunction** is a higher-order function because it takes another function (**argFn**) as an argument.
* **argFn** is the callback function that is passed to **useFunction** and is called for each element in the **names** array.