WPT Assignment 4

Exercise 1:
Create a function processData that takes two parameters: a string and a callback function. Your task is to write a callback that converts the string to uppercase and then call it within processData.

Requirements:

- Define a function toUpperCase that will serve as a callback.
- Pass a string and toUpperCase to processData and log the output.

```
function processData(x, call){
    str = call(x);
    console.log(str);
}

processData("samruddhi", (x)=>{
    return x.toUpperCase();
});
```

Exercise 2:
Write a function forEachElement that accepts an array and a callback. This function should apply the callback to each element of the array.
Requirements:

- Pass an anonymous function as the callback that multiplies each element by 2 and logs the result with the index.

```
function operation(arr, call){
    console.log(call(arr));
}

let arr = [10,20,30,40,50];
```

```
operation(arr, ()=> {
    for(i=0; i<arr.length; i++){
        arr[i] = arr[i] + 1;
    }
    return arr;
});
```

Exercise 3:
Simulate a network request by creating a function fetchData that takes a URL and a
callback as parameters. Use setTimeout to simulate a delay and then call the callback with a string representing a response.
Requirements:
● After a delay, log the “response” to the console.

```
function fetchData(url, call){
    setTimeout(()=>{
        console.log("Data Received from "+url+"\n");
        if(call){
            call();
        }
    },2000);
}

console.log("Getting Data From url1...");
fetchData("url1",()=>{
    console.log("Getting Data From url2...");
    fetchData("url2", ()=>{
         console.log("Getting Data From url3...");
         fetchData("url3");
    })
});
```

Exercise 4:
Modify fetchData from Exercise 3 to include error handling.
Requirements:
● Call the callback with an error message if an error occurs; otherwise, pass the “response.”
● Handle the error gracefully by logging it if it occurs.

```
function fetchData(url, call){
    setTimeout(()=>{
        if(url.slice(0,3) == "url"){
            let res = "Data";
            console.log("Data Received from "+url+"\n");
            if(call){
                call(res);
            }
        }
        else{
            let err = "Error";
            console.log("Error: Cannot receive Data from "+ur
l+"\n");
            if(call){
                call(err);
            }
        }
    },2000);
}

console.log("Getting Data From url1...");
fetchData("url1", (x)=>{
    console.log("Getting Data From url2...");
    fetchData("arl2", (x)=>{
         console.log("Getting Data From url3...");
         fetchData("url3");
```

```
    })
});
```

Exercise 5:
Using fetchData from Exercise 4, create another function processData that simulates processing the fetched data. Chain these functions together using nested
callbacks.
Requirements:
● First, call fetchData. Once the response is received, pass it to processData.
● processData should modify the data and log the processed result.

```
function fetchData(url, call){
    setTimeout(()=>{
        if(url.slice(0,3) == "url"){
            let res = "Data";
            console.log("Data Received from "+url+"\n");
            if(call){
                call(res);
            }
        }
        else{
            let err = "Error";
            console.log("Error: Cannot receive Data from "+ur
l+"\n");
            if(call){
                call(err);
            }
        }
    },2000);
}

const processData = (Data)=> {
    if(Data!="Error")
```

```
        console.log("Processed "+Data+"\n");
    else
        console.log("Error Occured");
}

fetchData("url1",(x)=>{
    processData(x);
});

fetchData("arl1",(x)=>{
    processData(x);
});
```