Assignment 4

Exercise 1:

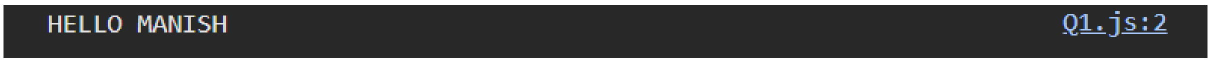
Create a function processData that takes two parameters: a string and a callback function. Your task is to write a callback that converts the string to uppercase and then call it within processData.

Requirements:

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

Ans:

```
function processData(str,Upper){
    console.log(Upper(str));
}
function toUpperCase(str){
    return str.toUpperCase();
}
processData("hello Manish",toUpperCase);
```



```
HELLO MANISH                                                    Q1.js:2
```

Exercise 2:

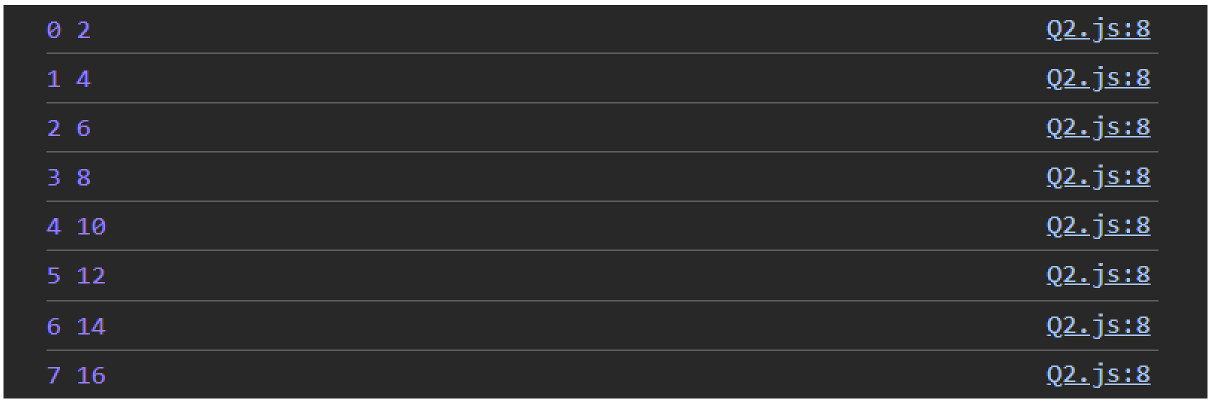
Write a function forEachElement that accepts an array and a callback. This function should apply the callback to each element of the array.

Requirements:

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

Ans:

```
function forEachElement(arr,call){
    for(let i=0;i<arr.length;i++){
        call(arr[i],i)
    }
}


forEachElement([1,2,3,4,5,6,7,8],(value,index)=>{
    console.log(index,value*2);
});
```



```
0 2                                                    Q2.js:8
1 4                                                    Q2.js:8
2 6                                                    Q2.js:8
3 8                                                    Q2.js:8
4 10                                                   Q2.js:8
5 12                                                   Q2.js:8
6 14                                                   Q2.js:8
7 16                                                   Q2.js:8
```

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.

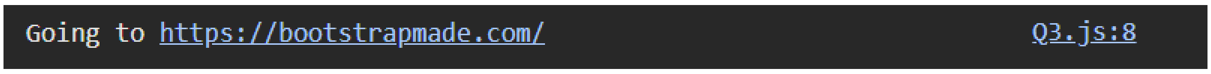
Requirements:

- **After a delay, log the “response” to the console.**

CDAC Mumbai

Ans:

```
function fetchData(url,call){
    setTimeout(()=>{
        let a= "Going to "+url;
        call(a);
    },10000);
}
fetchData("https://bootstrapmade.com/",(Response)=>{
   console.log(Response);
});
```



```
Going to https://bootstrapmade.com/                    Q3.js:8
```

Exercise 4:

Modify fetchData from Exercise 3 to include error handling.

Requirements:

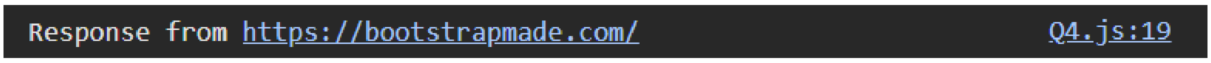
● **Call the callback with an error message if an error occurs; otherwise, pass the "response."**

● **Handle the error gracefully by logging it if it occurs.**

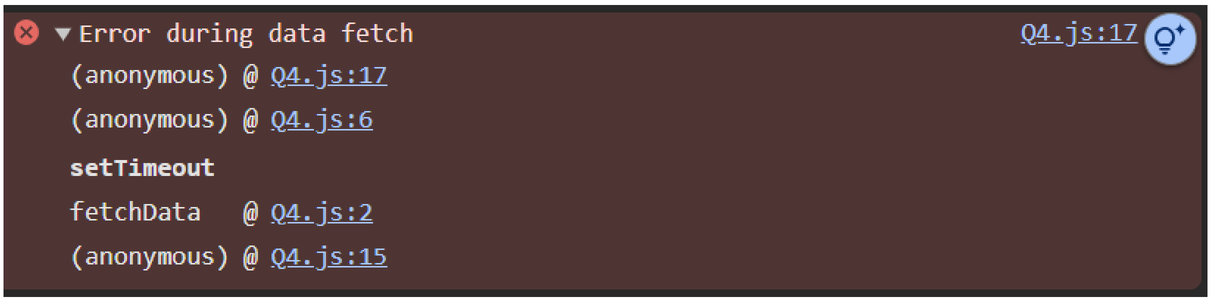
Ans:

```
function fetchData(url, callback) {
   setTimeout(() => {
      const error = Math.random() > 0.6;

      if (error) {
         callback('Error during data fetch', null);
      } else {
         const data = `Response from ${url}`;
         callback(null, data);
      }
   }, 5000);
}

fetchData("https://bootstrapmade.com/", (err, response) => {
   if (err) {
      console.error(err);
```

```
    } else {
        console.log(response);
    }
});
```



```
Response from https://bootstrapmade.com/                     Q4.js:19
```



```
⊗ ▼Error during data fetch                                    Q4.js:17
    (anonymous) @ Q4.js:17
    (anonymous) @ Q4.js:6
    setTimeout
    fetchData   @ Q4.js:2
    (anonymous) @ Q4.js:15
```

Exercise 5:

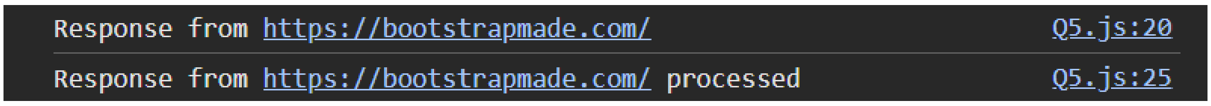
Using fetchData from Exercise 4, create another function processData that simulates processing the fetched data. Chain these functions together using nested callbacks.

Requirements:

- **First, call fetchData. Once the response is received, pass it to processData.**
- **processData should modify the data and log the processed result.**

Ans:

```
function fetchData(url, callback) {
    setTimeout(() => {
        const data = `Response from ${url}`;
        callback(null, data);
    }, 1000);
}

function processData(data, callback) {
    setTimeout(() => {
        const processedData = `${data} processed`;
        callback(null, processedData);
    }, 1000);
}

// Nested use of functions
fetchData("https://bootstrapmade.com/", (err, data) => {
    if (err) {
        console.error(err);
    } else {
        console.log(data);
        processData(data, (err, processedData) => {
            if (err) {
                console.error(err);
            } else {
                console.log(processedData);
            }
        });
    }
});
```



```
Response from https://bootstrapmade.com/                    Q5.js:20
Response from https://bootstrapmade.com/ processed          Q5.js:25
```