

Assignment 04

Module: WPT Topic: Lab Assignment Based on Callback Function

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`.

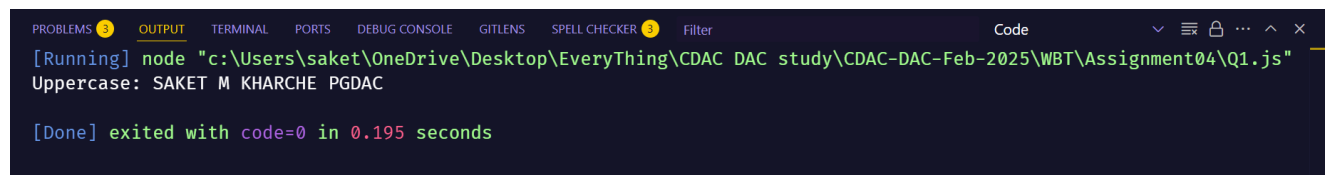
Requirements:

- Define a function `toUpperCase` that will serve as a callback.

- Pass a string and `toUpperCase` to `processData` and log the output.

Ans:-

```
// Callback function
function toUpperCase(str) {
    return str.toUpperCase();
}
function processData(str, callback) {
    const result = callback(str);
    console.log("Uppercase:", result);
}
processData("sakat m kharche pgdac", toUpperCase);
```



The screenshot shows a code editor with a dark theme. The top bar includes tabs for PROBLEMS, OUTPUT, TERMINAL, PORTS, DEBUG CONSOLE, GITLENS, SPELL CHECKER, and Filter. The main area shows the code being executed. The terminal output at the bottom reads: [Running] node "c:\Users\saket\OneDrive\Desktop\Everything\CDAC DAC study\CDAC-DAC-Feb-2025\WBT\Assignment04\Q1.js" Uppercase: SAKET M KHARCHE PGDAC [Done] exited with code=0 in 0.195 seconds.

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, callback) {
  for (let i = 0; i < arr.length; i++) {
    callback(arr[i], i);
  }
}
forEachElement([1, 2, 3, 4], function (element, index) {
  console.log(`Index ${index}:`, element * 2);
});
```

```
[Running] node "c:\Users\saket\OneDrive\Desktop\Everything\CDAC DAC study\CDAC-DAC-Feb-2025\WBT\Assignment04\Q2.js"
Index 0: 2
Index 1: 4
Index 2: 6
Index 3: 8
[Done] exited with code=0 in 0.109 seconds
```

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.

Ans:-

```
function fetchData(url, callback) {
  console.log("Fetching data from", url);
  setTimeout(() => {
    const response = `Response from ${url}`;
    callback(response);
  }, 2000);
}
fetchData("https://www.youtube.com", function (response) {
  console.log("Received:", response);
});
```

```
[Running] node "c:\Users\saket\OneDrive\Desktop\Everything\CDAC DAC study\CDAC-DAC-Feb-2025\WBT\Assignment04\tempCodeRunnerFile.js"
Fetching data from https://www.youtube.com
Received: Response from https://www.youtube.com
[Done] exited with code=0 in 2.128 seconds
```

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) {
  console.log("Fetching data from", url);
  setTimeout(() => {
    const success = Math.random() > 0.3;
    if (success) {
      const response = `Response from ${url}`;
      callback(null, response);
    } else {
      callback("Network error occurred!", null);
    }
  }, 2000);
}

// Using fetchData with a callback to handle response or error
fetchData("https://http.cat/404", function (error, response) {
  if (error) {
    console.error("Error:", error);
  } else {
    console.log("Success:", response);
  }
});
```

```
[Running] node "c:\Users\saket\OneDrive\Desktop\Everything\CDAC DAC study\CDAC-DAC-Feb-2025\WBT\Assignment04\Q4.js"
Fetching data from https://http.cat/404
Success: Response from https://http.cat/404

[Done] exited with code=0 in 2.1 seconds

[Running] node "c:\Users\saket\OneDrive\Desktop\Everything\CDAC DAC study\CDAC-DAC-Feb-2025\WBT\Assignment04\Q4.js"
Fetching data from https://http.cat/404
Error: Network error occurred!
```

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) {
  console.log("Fetching data from", url);
  setTimeout(() => {
    const success = Math.random() > 0.3;
    if (success) {
      const response = `Response from ${url}`;
      callback(null, response);
    } else {
      callback("Network error occurred!", null);
    }
  }, 2000);
}

// processData
function processData(data) {
  const processed = `Processed Data: ${data.toUpperCase()}`;
  console.log(processed);
}

// nested callbacks
fetchData("https://www.google.com", function (error, response) {
  if (error) {
    console.error("Error:", error);
  } else {
    processData(response);
  }
});
```

```
[Running] node "c:\Users\saket\OneDrive\Desktop\Everything\CDAC DAC study\CDAC-DAC-Feb-2025\WBT\Assignment04\Q5.js"
Fetching data from https://www.google.com
Processed Data: RESPONSE FROM HTTPS://WWW.GOOGLE.COM

[Done] exited with code=0 in 2.111 seconds
```