1. **Double using callback.**
Write a function that takes in an array of integers and a callback function, and returns a new array where each element is doubled using the callback.

```
function doubleArray(arr, callback) {
  const doubledArr = arr.map((num) => {
    return callback(num);
  });
  return doubledArr;
}

const originalArray = [1, 2, 3, 4];

function callback(num) {
  return num * 2;
}

const doubledArray = doubleArray(originalArray, callback);

console.log(doubledArray); // Output: [2, 4, 6, 8]
```

2. String Manipulation.
Write a function “manipulateString” that takes in a string and converts the characters to uppercase letters. The Write a function "manipulateString" that takes in a string and converts the characters to uppercase letters. The function should return a callback function "logString" that logs the sentence "The manipulated string is: " along with the manipulated string or the new string to the console.

```
function manipulateString(inputString, callback) {
  const manipulatedString = inputString.toUpperCase();

  callback(manipulatedString);
}

function logString(manipulatedString) {
  console.log(`The manipulated string is: ${manipulatedString}`);
}

// Expected Output:

manipulateString("hello, world!", logString); // The manipulated string is:
HELLO, WORLD!
```

3. Age in Days.
Write a function that takes in an array of integers and a callback function, and returns a new array where each Write a JavaScript function called ageInDays that accepts an object containing a person's first name, last name, and age in years as input. The function should concatenate the first and last name into a single string and store it in a variable called fullName. It should then calculate the person's age in days and store it in a variable called ageInDays.

The ageInDays function should then return a callback function that logs a message to the console. The message should include the person's full name and age in days, and should be in the format: "The person's full name is [full name] and their age in days is [age in days]."

Note that the ageInDays function should not log the message to the console directly, but should instead return a callback function that can be used to log the message at a later time.

```
const person = {
  firstName: "Mithun",
  lastName: "S",
  age: 20,
};

function ageInDays(personObject, callback) {
  const fullName = `${personObject.firstName} ${personObject.lastName}`;
  const ageInDays = personObject.age * 365;
  callback(fullName, ageInDays);
}

function logResult(fullName, ageInDays) {
  console.log(
    `The person's full name is ${fullName} and their age in days is
${ageInDays}.`
  );
}

// Expected Output.

ageInDays(person, logResult); // The person's full name is Mithun S and
their age in days is 7300.
```

4. Arrange in alphabetical order.

Write a program that accepts a list of objects representing books [title, author, and year] and a callback function. The program should use the map function to create a new list containing only the titles of the books, and then pass this new list to the callback function. The callback function should then log the titles to the console in alphabetical order.

```
const books = [
  {
    title: "The Great Gatsby",
    author: "F. Scott Fitzgerald",
    year: 1925,
  },
  {
    title: "To Kill a Mockingbird",
    author: "Harper Lee",
    year: 1960,
  },
  {
    title: "Who are You?",
    author: "George Orwell",
    year: 1949,
  },
  {
    title: "Pride and Prejudice",
    author: "Jane Austen",
    year: 1813,
  },
];

function logTitles(titles) {
  titles.sort();
  console.log(titles.join(", "));
}

function extractTitles(books, callback) {
  const titles = books.map((book) => book.title);
  callback(titles);
}

extractTitles(books, logTitles);
```

5. Greeting Promise.
You need to write a function that takes a name as input and returns a promise that resolves with a greeting message. The function should greet the person using their name, with a message in the format "Hello, {name}!".

For example, if the input to the function is "Mithun", the promise should resolve with the string "Hello, Mithun!".

```
function greet(name) {
  return new Promise((resolve) => {
    const greeting = `Hello, ${name}!`;
    resolve(greeting);
  });
}

greet("Mithun").then((message) => console.log(message)); // "Hello,
Mithun!"
```

6. Fetch results asynchronously.
Write a function that asynchronously fetches data from an API
[https://jsonplaceholder.typicode.com/todos/1]and logs the result to the console.

```
async function fetchData() {
  const response = await
fetch("https://jsonplaceholder.typicode.com/todos/1");
  const data = await response.json();
  console.log(data);
}

fetchData();
```

7. Multiple requests.
Create an asynchronous function that retrieves data from two different API endpoints: "https://jsonplaceholder.typicode.com/todos/1" and "https://jsonplaceholder.typicode.com/posts/1". The first API returns a to-do task, while the second API provides post details. The function should combine the results from both APIs and log them as an object, where the keys are "todo" and "post", and the corresponding values are the responses from the respective APIs.

```
async function getCombinedData() {
  const [data1, data2] = await Promise.all([
    fetch("https://jsonplaceholder.typicode.com/todos/1").then((response)
=>
      response.json()
    ),
    fetch("https://jsonplaceholder.typicode.com/posts/1").then((response)
=>
      response.json()
    ),
  ]);
  const combinedData = {
    todo: data1,
    post: data2,
  };
  return combinedData;
}

getCombinedData().then((data) => console.log(data));
```

8. Get Data from API and Display it on the browser console.
Write a JavaScript program that uses the Fetch method to retrieve data from an API, and then logs the data to the console. For example, you could use the API at https://jsonplaceholder.typicode.com/posts to retrieve a list of posts, and then display them to the browser console.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Get Data from API and Display it on the browser console.</title>
  </head>
  <body>
    <!-- JS Starts -->
    <script>
      fetch("https://jsonplaceholder.typicode.com/posts")
        .then((response) => response.json())
        .then((data) => console.log(data))
        .catch((error) => console.error(error));
    </script>
    <!-- JS Ends -->
  </body>
</html>
```

9. Error Handling

Write a JavaScript program that uses the Fetch method to retrieve data from an API, and then handles errors that may occur. For example, you could use the API at https://jsonplaceholder.typicode.com/posts/123456789 to simulate an error, and then display an error message on the webpage.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
/>
    <title>Error Handling</title>
  </head>
  <body>
    <!-- JS Starts -->
    <script>
      fetch("https://jsonplaceholder.typicode.com/posts/123456789").then(
        (response) => {
          if (!response.ok) {
            throw new Error("Network response was not ok");
          }
          return response.json();
        }
      );
    </script>
    <!-- JS Ends -->
  </body>
</html>
```