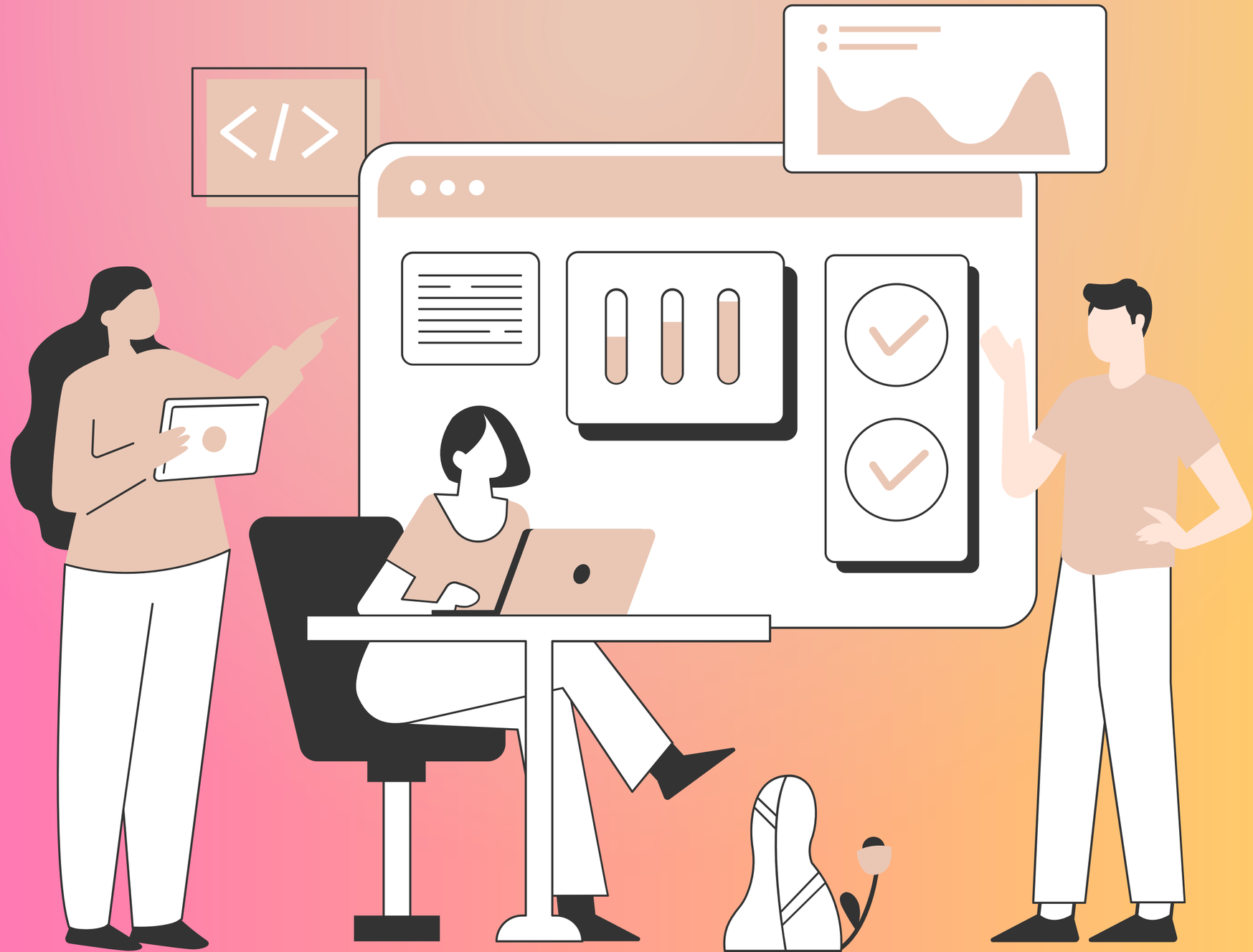
MONK-CODES

WEB-DEVELOPER

JavaScript Interview Preparation in 10 Days

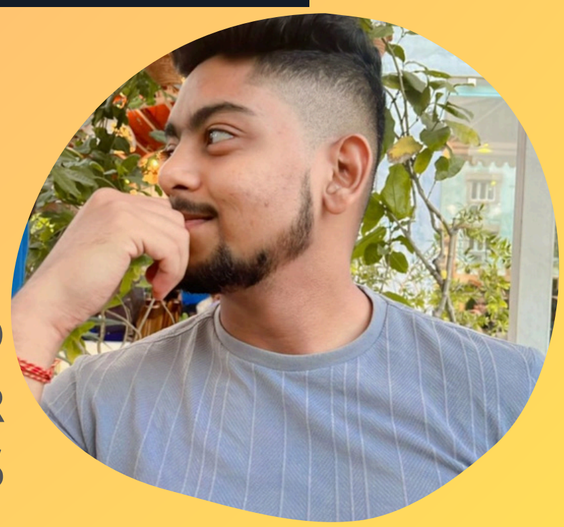


WHAT IS ASYNCHRONOUS PROGRAMMING IN JAVASCRIPT?

Asynchronous programming allows tasks to run in the background while other tasks continue to run, without blocking the main thread.

index.js

```
console.log("Start");

setTimeout(() => {
   console.log("This runs after 2 seconds");
}, 2000);

console.log("End");
```



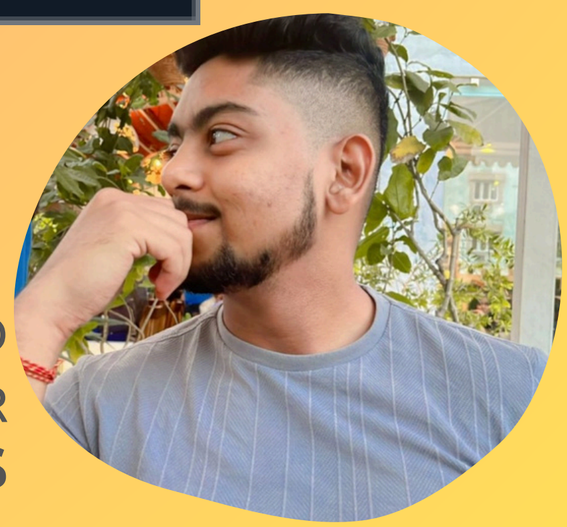
REACT | NEXT JS | FRONTEND DEVELOPER
MONK CODES

WHAT ARE CALLBACKS IN JAVASCRIPT?

A callback is a function passed into another function as an argument and executed after the completion of that function

index.js

```
function fetchData(callback) {
  setTimeout(() => {
    callback("Data fetched");
  }, 1000);
}

fetchData((message) => {
  console.log(message); // "Data fetched"
});
```



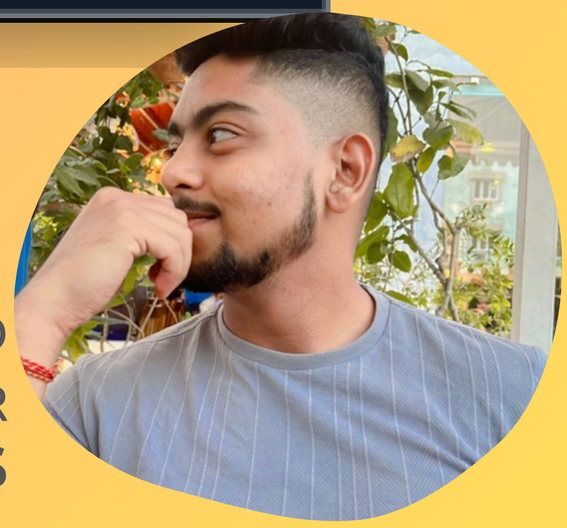
REACT | NEXT JS | FRONTEND DEVELOPER
MONK CODES

WHAT ARE PROMISES IN JAVASCRIPT?

A Promise is an object representing the eventual completion or failure of an asynchronous operation

index.js

```
let promise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("Promise resolved");
  }, 1000);
});

promise.then((message) => {
  console.log(message); // "Promise resolved"
});
```



REACT | NEXT JS | FRONTEND DEVELOPER

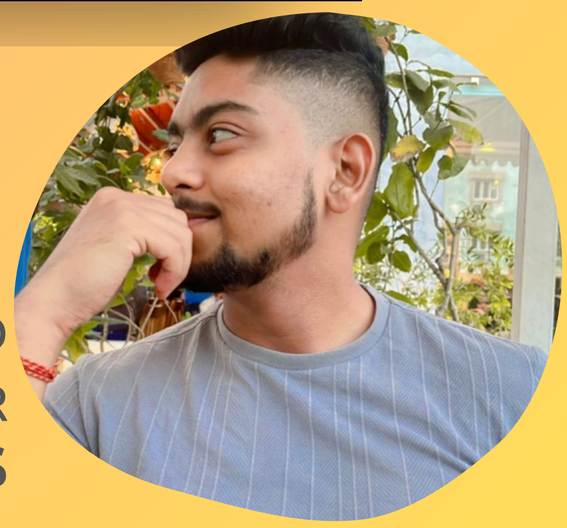
MONK CODES

WHAT DOES THE THEN() METHOD DO?

The then() method is used to specify what to do when a Promise is resolved

index.js

```
let promise = new Promise((resolve, reject) => {
  setTimeout(() => {
    resolve("Data loaded");
  }, 1000);
});

promise.then((data) => {
  console.log(data); // "Data loaded"
});
```



REACT | NEXT JS | FRONTEND DEVELOPER

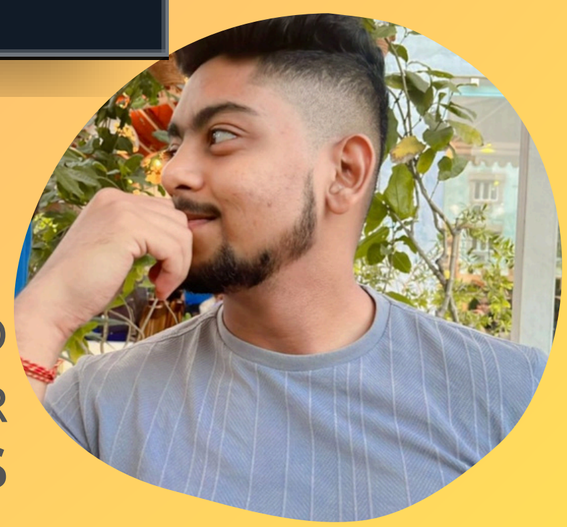
MONK CODES

WHAT DOES THE CATCH() METHOD DO?

The catch() method is used to specify what to do when a Promise is rejected

index.js

```
let promise = new Promise((resolve, reject) => {
  setTimeout(() => {
    reject("Error occurred");
  }, 1000);
});

promise
  .then((data) => {
    console.log(data);
  })
  .catch((error) => {
    console.log(error); // "Error occurred"
  });
```



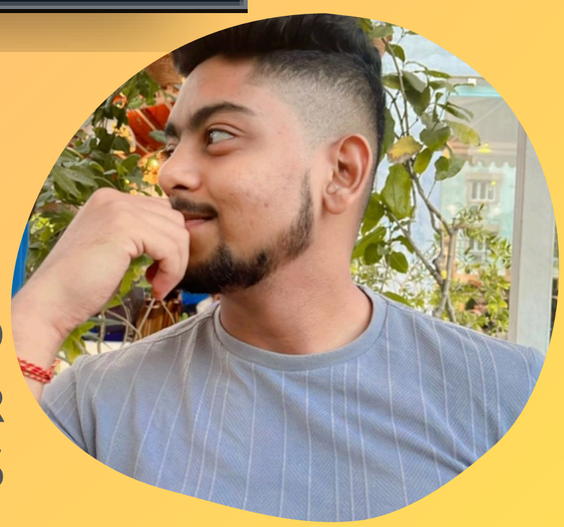
REACT | NEXT JS | FRONTEND DEVELOPER
MONK CODES

WHAT IS ASYNC/AWAIT IN JAVASCRIPT?

async/await is syntactic sugar for working with Promises, making asynchronous code look and behave more like synchronous code

index.js

```
async function fetchData() {
  let promise = new Promise((resolve) => {
    setTimeout(() => resolve("Data fetched"), 1000);
  });

  let result = await promise;
  console.log(result); // "Data fetched"
}

fetchData();
```



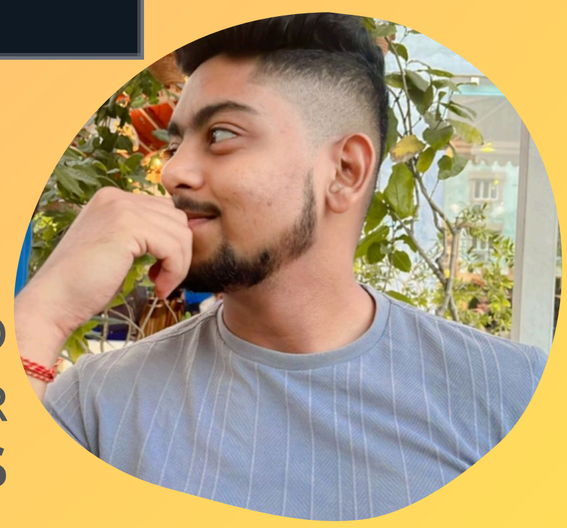
REACT | NEXT JS | FRONTEND DEVELOPER
MONK CODES

HOW DO YOU HANDLE ERRORS IN ASYNC/AWAIT?

Errors in async/await can be handled using try/catch blocks

index.js

```
async function fetchData() {
  try {
    let promise = new Promise((resolve, reject) =>
      {
      setTimeout(() => reject("Error occurred")
                 ,1000);
    });

    let result = await promise;
    console.log(result);
  } catch (error) {
    console.log(error); // "Error occurred"
  }
}
fetchData();
```



REACT | NEXT JS | FRONTEND DEVELOPER

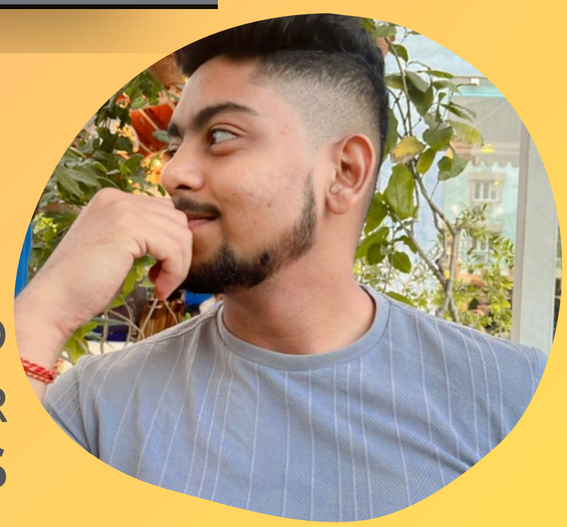
MONK CODES

WHAT IS THE EVENT LOOP IN JAVASCRIPT?

The Event Loop is a mechanism that handles the execution of multiple chunks of code, enabling non-blocking asynchronous operations

index.js

```
console.log("Start");

setTimeout(() => {
  console.log("This runs last");
}, 0);

console.log("End");
// Output: "Start", "End", "This runs last"
```



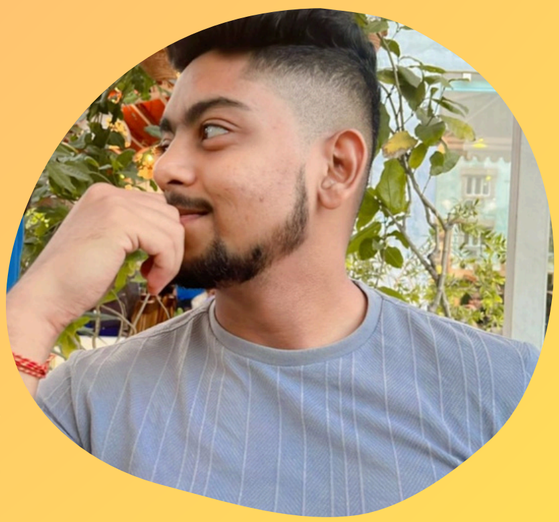
REACT | NEXT JS | FRONTEND DEVELOPER
MONK CODES

WHAT IS THE FETCH API?

The fetch API provides a way to make network requests and handle responses in a more modern and flexible way compared to XMLHttpRequest

index.js

```
fetch("https://api.example.com/data")
  .then((response) => response.json())
  .then((data) => console.log(data))
  .catch((error) => console.error("Error:", error));
```



REACT | NEXT JS | FRONTEND DEVELOPER
MONK CODES

HOW DO YOU USE PROMISE.ALL IN JAVASCRIPT?

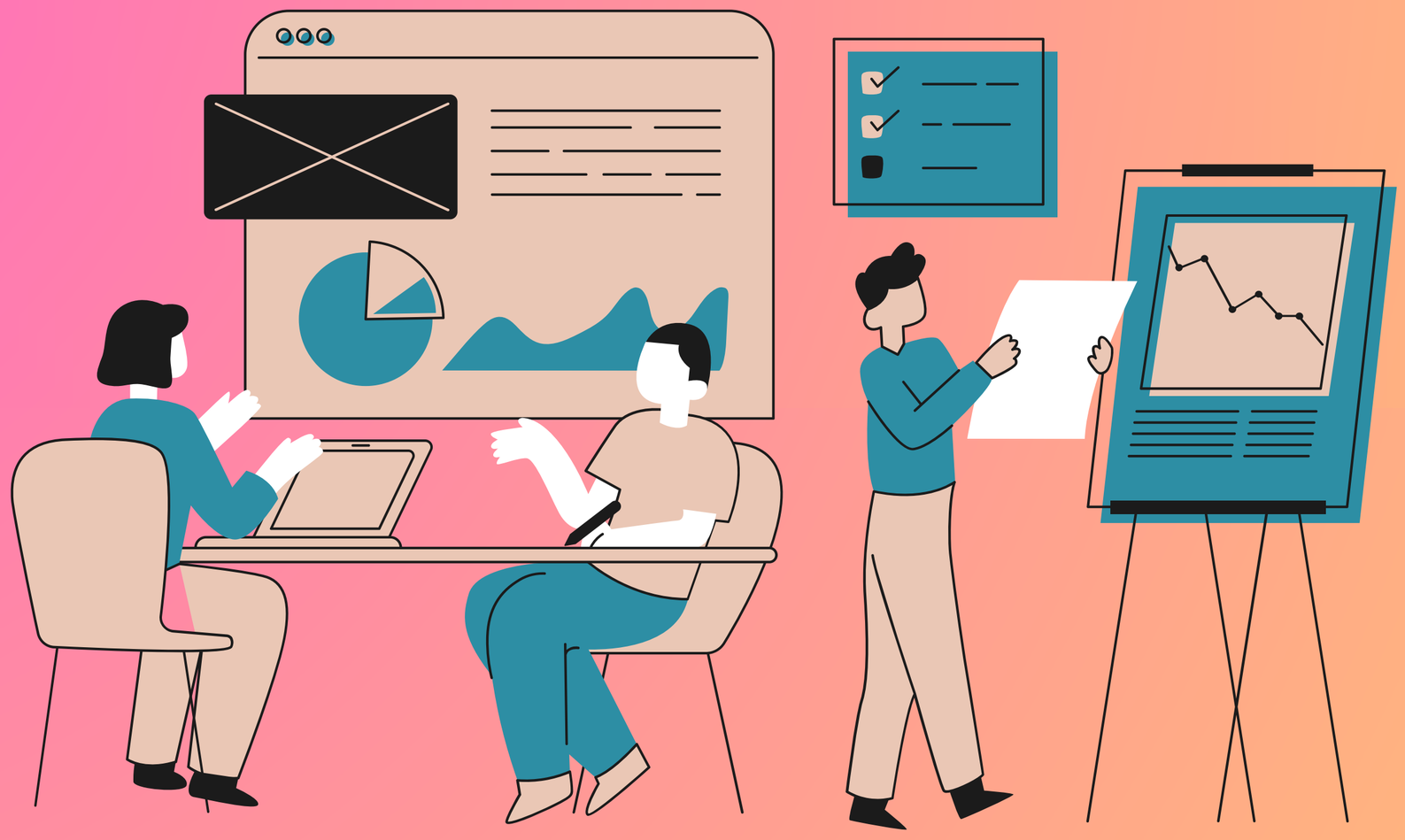
Promise.all takes an array of Promises and returns a single Promise that resolves when all of the input Promises have resolved

index.js

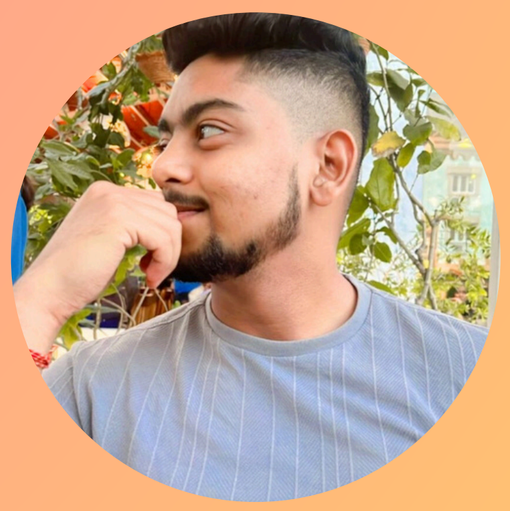
```
let promise1 = Promise.resolve(3);
let promise2 = 42;
let promise3 = new Promise((resolve) => {
  setTimeout(resolve, 100, "foo");
});

Promise.all([promise1, promise2, promise3])
  .then((values) => {
  console.log(values); // [3, 42, "foo"]
});
```

REACT | NEXT JS | FRONTEND DEVELOPER

MONK CODES



Contact Us



Phone Number

+91-8961894629

Email

monkcodesdev@gmail.com

Website

linkedin.com/in/dxpratikk