callback function in typescript example

In TypeScript, callback functions are commonly used to handle asynchronous operations, such as fetching data from a server, reading files, or executing tasks that take some time to complete. Here's an example of how to define and use a callback function in TypeScript:

data.service.ts:

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class DataService {

  private apiUrl = 'https://jsonplaceholder.typicode.com/posts'; // Example API URL, replace with your API endpoint

  constructor(private http: HttpClient) { }

  // Fetch product item data from a fake API asynchronously
  fetchDataFromAPI(): Observable<ProductItem[]> {
    return this.http.get<ProductItem[]>(this.apiUrl);
  }
}

export interface ProductItem {
  userId: number;
  id: number;
  title: string;
```

```
  body: string;
}
```

app.component.ts:

```
import { Component, OnInit } from '@angular/core';
import { DataService, ProductItem } from './data.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent implements OnInit {

  fetchedData: ProductItem[];

  constructor(private dataService: DataService) {}

  ngOnInit() {
    // Call the fetchDataFromAPI function from the DataService and subscribe to the Observable
    this.dataService.fetchDataFromAPI().subscribe(
      (data: ProductItem[]) => {
        // Handle the fetched data
        this.fetchedData = data;
      },
      (error) => {
        console.error('Error fetching data:', error);
      }
    );
  }
```

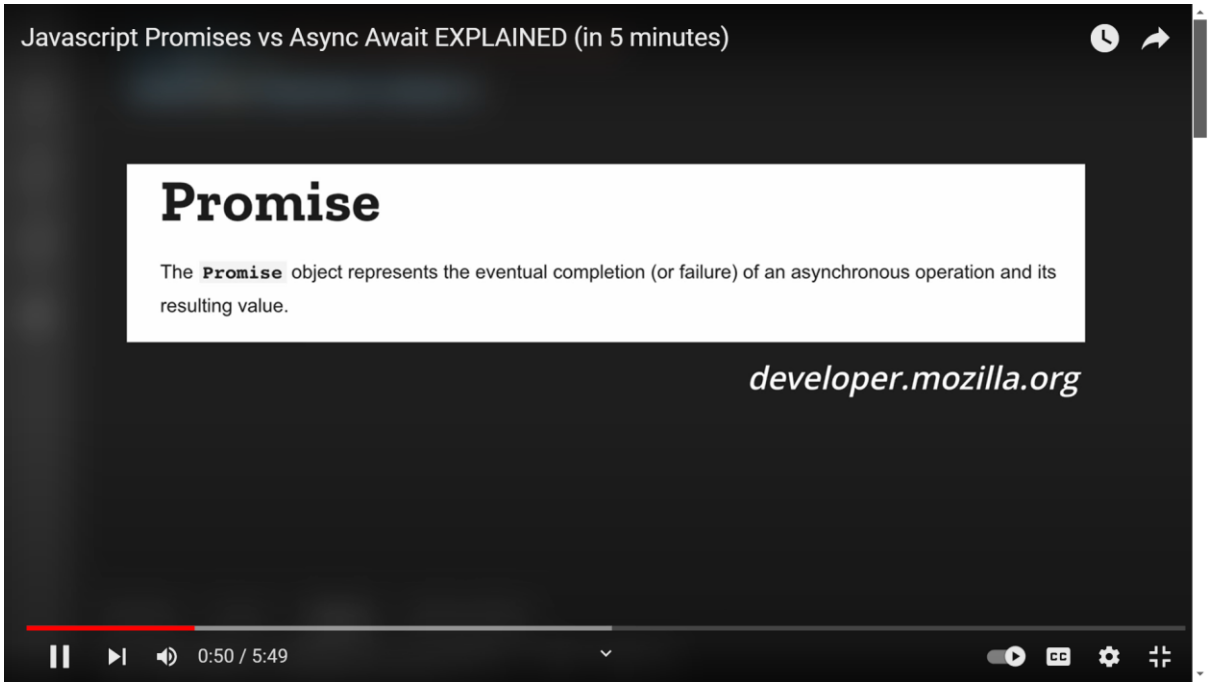
```
}
```

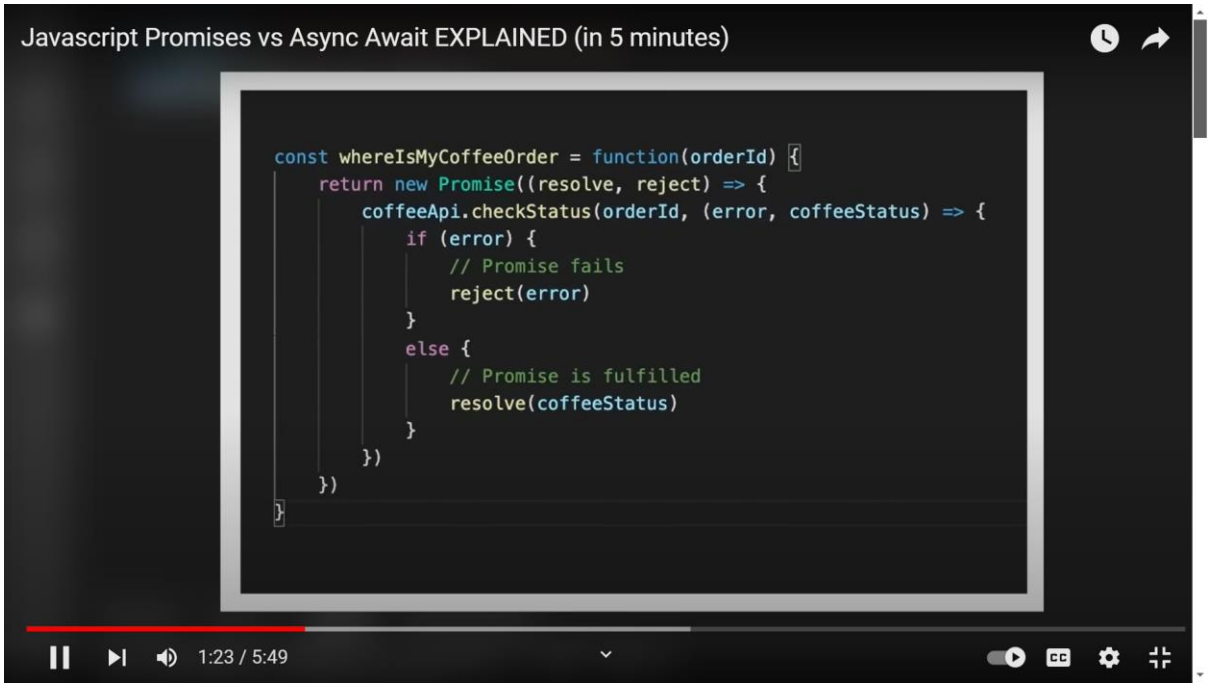
app.component.html:

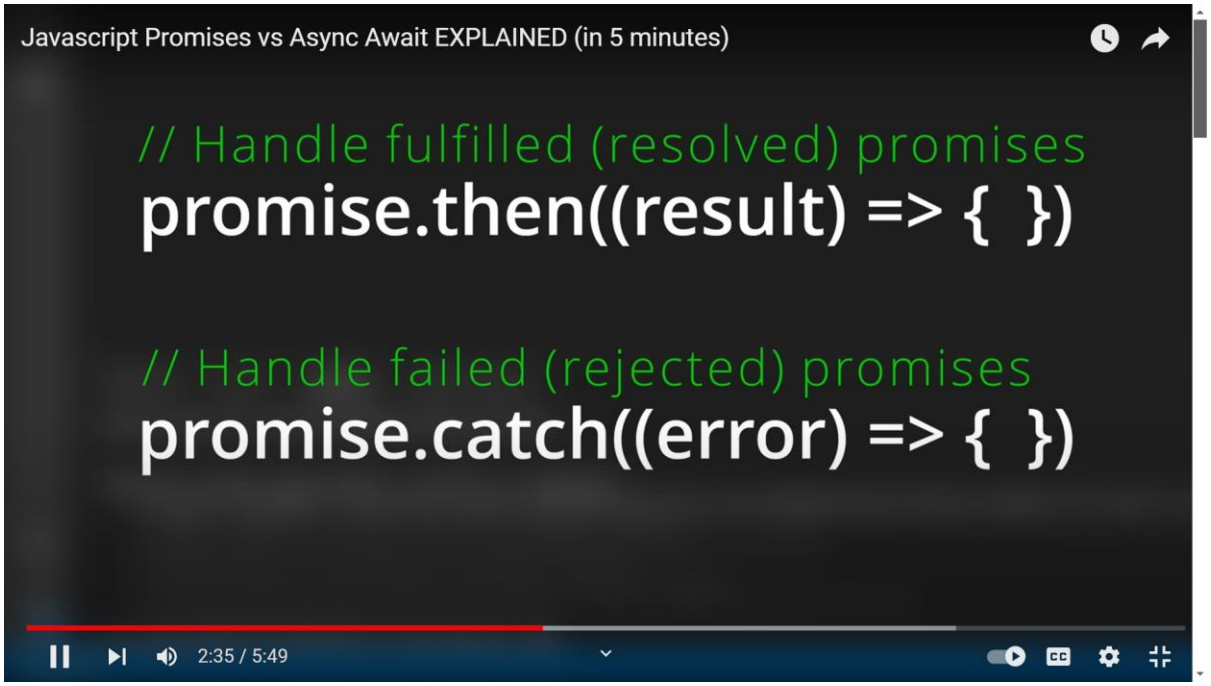
```
<div>
  <h1>Fetched Products:</h1>
  <ul>
    <li *ngFor="let item of fetchedData">
      <strong>Title:</strong> {{ item.title }}<br>
      <strong>Body:</strong> {{ item.body }}
    </li>
  </ul>
</div>
```

what is promises in Typescript with code examples Angular?

In Angular applications, promises are often used to handle asynchronous operations such as making HTTP requests to fetch data from a server. Angular's HttpClient module returns Observables by default for handling HTTP requests, but you can convert these Observables into promises using the `.toPromise()` method if needed. Here's an example of using promises in Angular:


Javascript Promises vs Async Await EXPLAINED (in 5 minutes)
Promise
The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.
developer.mozilla.org
0:50 / 5:49


Javascript Promises vs Async Await EXPLAINED (in 5 minutes)
const whereIsMyCoffeeOrder = function(orderId) {
return new Promise((resolve, reject) => {
coffeeApi.checkStatus(orderId, (error, coffeeStatus) => {
if (error) {
// Promise fails
reject(error)
}
else {
// Promise is fulfilled
resolve(coffeeStatus)
}
})
})
}
1:23 / 5:49


Javascript Promises vs Async Await EXPLAINED (in 5 minutes)
// Handle fulfilled (resolved) promises
promise.then((result) => { })
// Handle failed (rejected) promises
promise.catch((error) => { })
2:35 / 5:49

```
const axiosRequest = require('axios')

axiosRequest
    .get('https://www.boredapi.com/api/activity')
    .then(response => {
        console.log(`You could ${response.data.activity}`)
    })
    .catch(error => {
        console.error(`ERROR! ${error}`)
    })
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples %
```

Javascript Promises vs Async Await EXPLAINED (in 5 minutes)

```
axiosRequest
    .get('https://httpstat.us/404')
    .then(response => {
        console.log(`You could ${response.data.activity}`)
    })
    .catch(error => {
        console.error(`ERROR! ${error}`)
    })
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

3:15 / 5:49

```
chrisroberts@Chriss-MacBook-Air api-examples %
```

```
})
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples % node 3-promise-then.js
ERROR! Error: Request failed with status code 404
chrisroberts@Chriss-MacBook-Air api-examples %
```

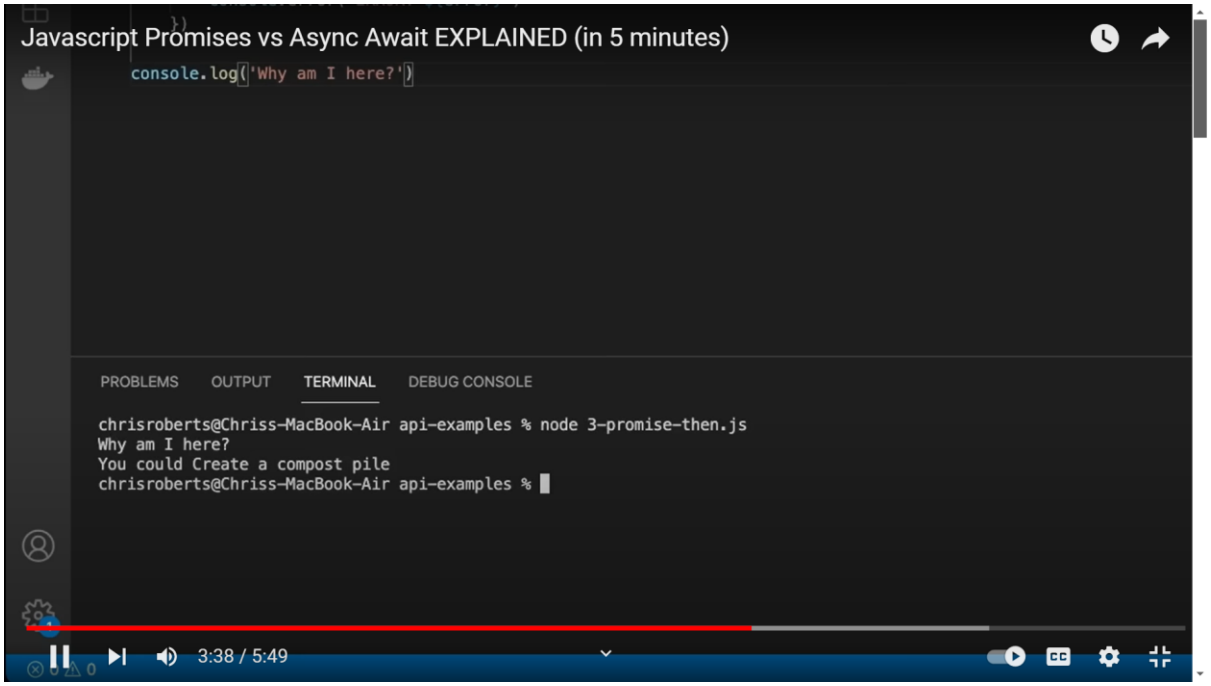
⊗ 0 ⚠ 0

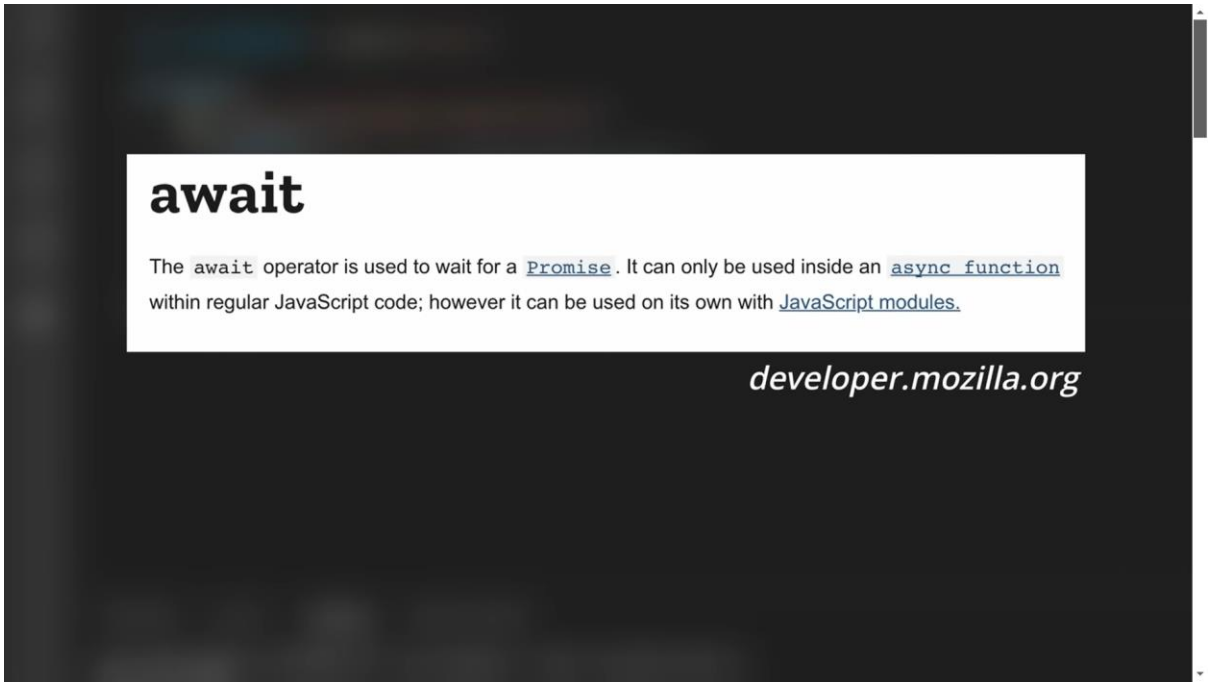
```
})

console.log('Why am I here?')
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples % nod
```

⊗ 0 ⚠ 0


Javascript Promises vs Async Await EXPLAINED (in 5 minutes)
console.log('Why am I here?')
PROBLEMS
OUTPUT
TERMINAL
DEBUG CONSOLE
chrisroberts@Chriss-MacBook-Air api-examples % node 3-promise-then.js
Why am I here?
You could Create a compost pile
chrisroberts@Chriss-MacBook-Air api-examples %
3:38 / 5:49


await
The `await` operator is used to wait for a `Promise`. It can only be used inside an `async function` within regular JavaScript code; however it can be used on its own with JavaScript modules.
developer.mozilla.org

```
const axiosRequest = require('axios')

async function getActivity() {
    let response = await axiosRequest.get('https://www.boredapi.com/api/activity')
    console.log(`You could ${response.data.activity}`)
}

getActivity()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples %
```

Javascript Promises vs Async Await EXPLAINED (in 5 minutes)

```
const axiosRequest = require('axios')
async function getActivity() {
    let response = await axiosRequest.get('https://www.boredapi.com/api/activity')
    console.log(`You could ${response.data.activity}`)
}

getActivity()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples % node 5-async-resolve.js
You could Have a jam session with your friends
chrisroberts@Chriss-MacBook-Air api-examples %
```

4:48 / 5:49

Javascript Promises vs Async Await EXPLAINED (in 5 minutes)

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples % node 5-async-resolve.js
You could Have a jam session with your friends
chrisroberts@Chriss-MacBook-Air api-examples %
```

4:48 / 5:49

```
const axiosRequest = require('axios')

async function getActivity() {
    try {
        let response = await axiosRequest.get('https://httpstat.us/500')
        console.log(`You could ${response.data.activity}`)
    } catch (error) {
        console.error(`ERROR: ${error}`)
    }
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples %
```

```
}

getActivity()
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
chrisroberts@Chriss-MacBook-Air api-examples % node 6-async-reject.js
ERROR: Error: Request failed with status code 500
chrisroberts@Chriss-MacBook-Air api-examples %
```

⊗ 0 ⚠ 0