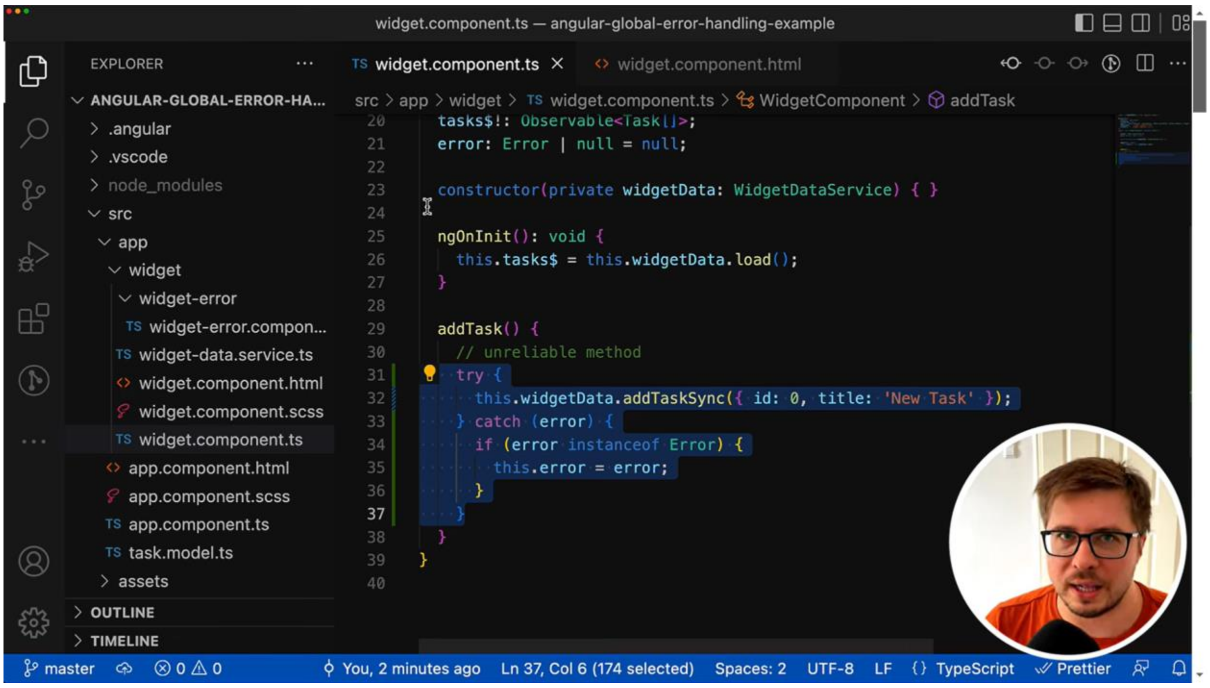
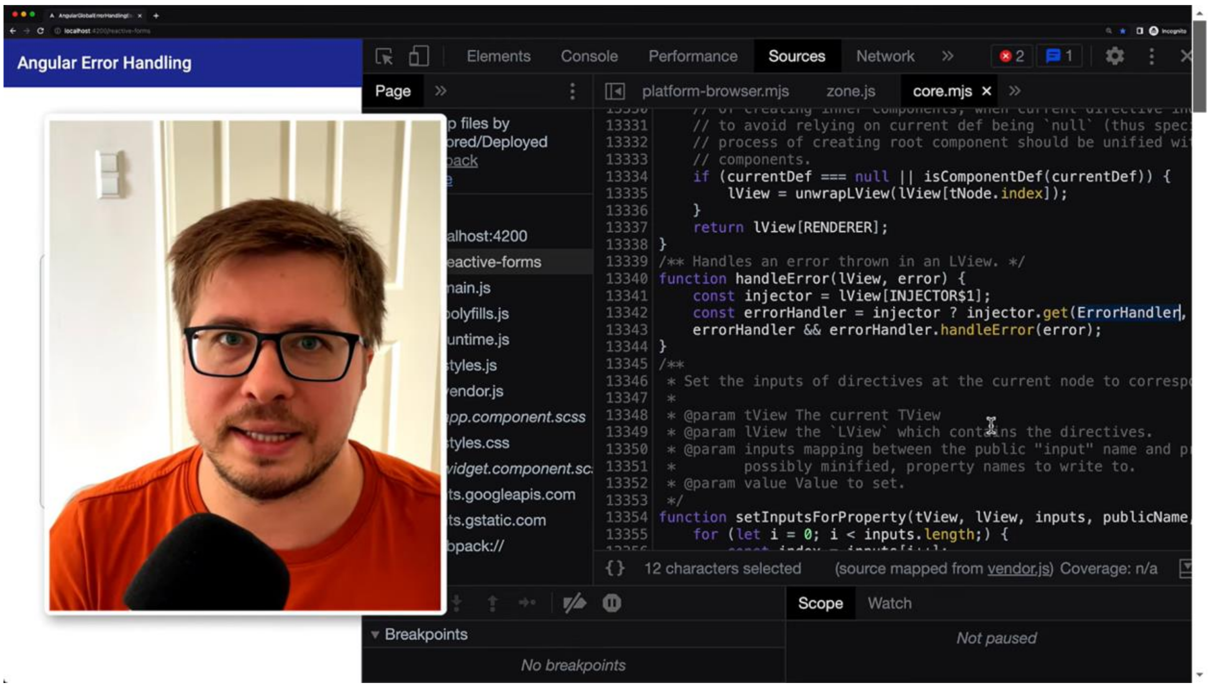
1. `try-catch` blocks

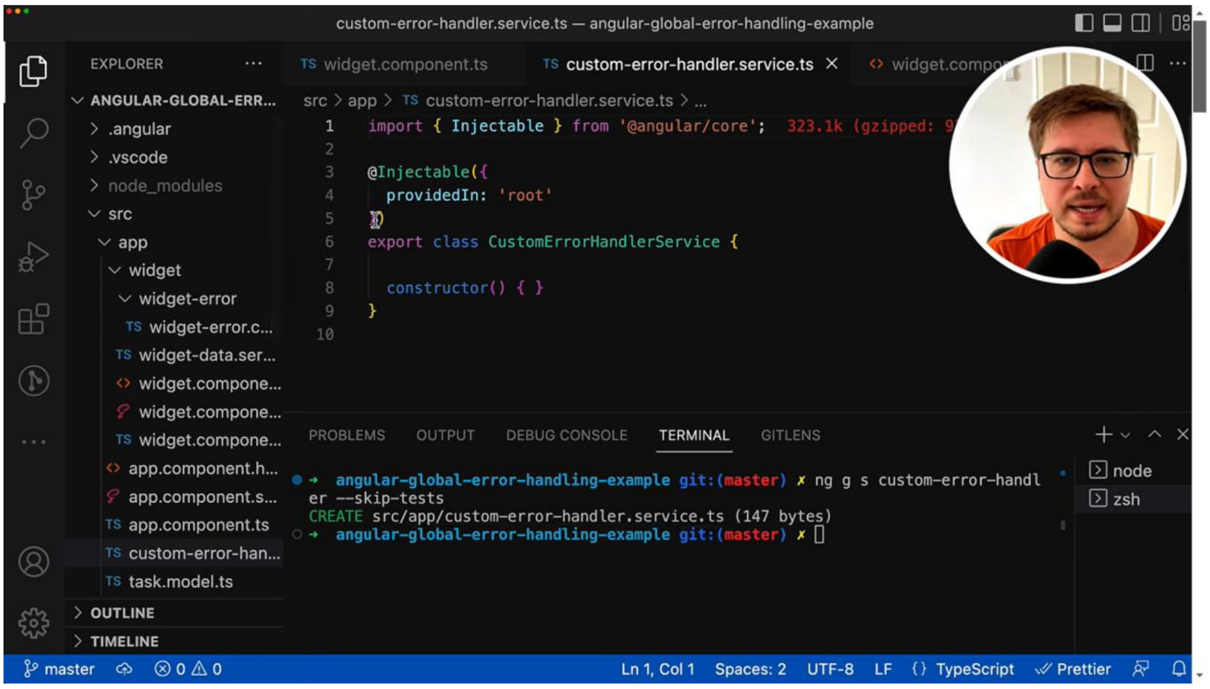


```
tasks$!: Observable<Task[]>;
error: Error | null = null;

constructor(private widgetData: WidgetDataService) { }

ngOnInit(): void {
  this.tasks$ = this.widgetData.load();
}

addTask() {
  // unreliable method
  try {
    this.widgetData.addTaskSync({ id: 0, title: 'New Task' });
  } catch (error) {
    if (error instanceof Error) {
      this.error = error;
    }
  }
}
}
```



```
    // to avoid relying on current def being `null` (thus spec
    // process of creating root component should be unified wi
    // components.
    if (currentDef === null || isComponentDef(currentDef)) {
        lView = unwrapLView(lView[tNode.index]);
    }
    return lView[RENDERER];
}
/** Handles an error thrown in an LView. */
function handleError(lView, error) {
    const injector = lView[INJECTOR$1];
    const errorHandler = injector ? injector.get(ErrorHandler,
    errorHandler && errorHandler.handleError(error);
}
/**
 * Set the inputs of directives at the current node to corresp
 *
 * @param tView The current TView
 * @param lView the `LView` which contains the directives.
 * @param inputs mapping between the public "input" name and p
 *         possibly minified, property names to write to.
 * @param value Value to set.
 */
function setInputsForProperty(tView, lView, inputs, publicName,
    for (let i = 0; i < inputs.length;) {
```

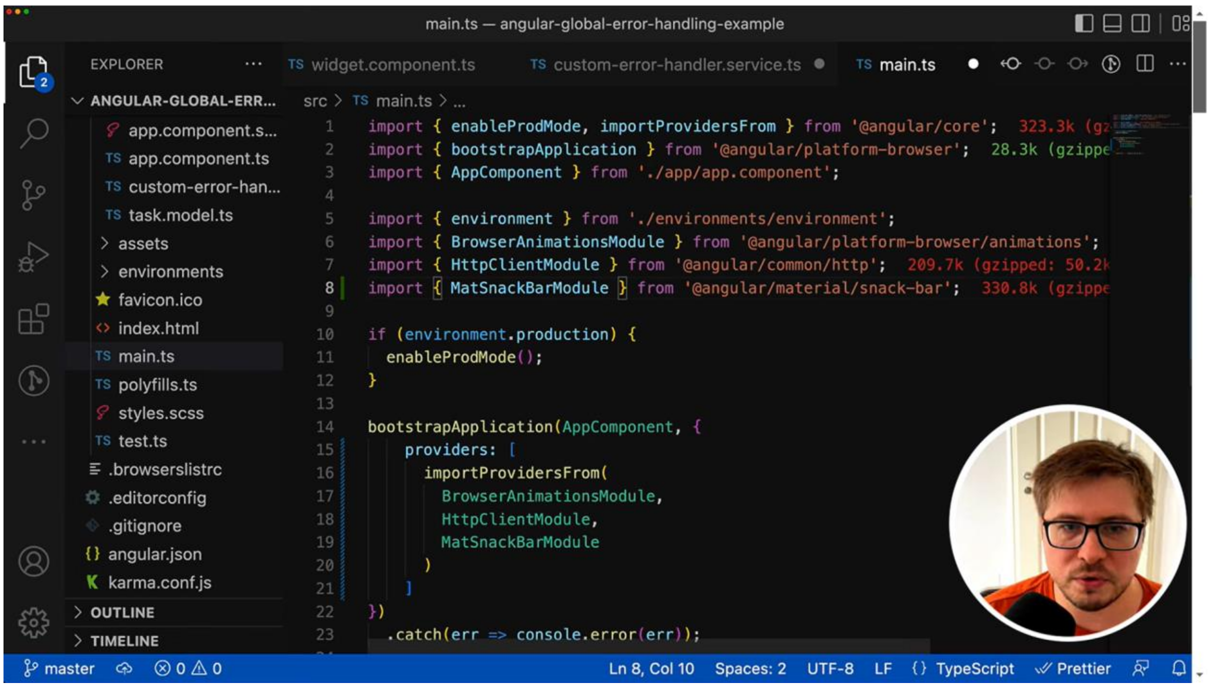
Created custom-error-handler.service



```
import { Injectable } from '@angular/core';  323.1k (gzipped: 9

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

  constructor() { }
}
```

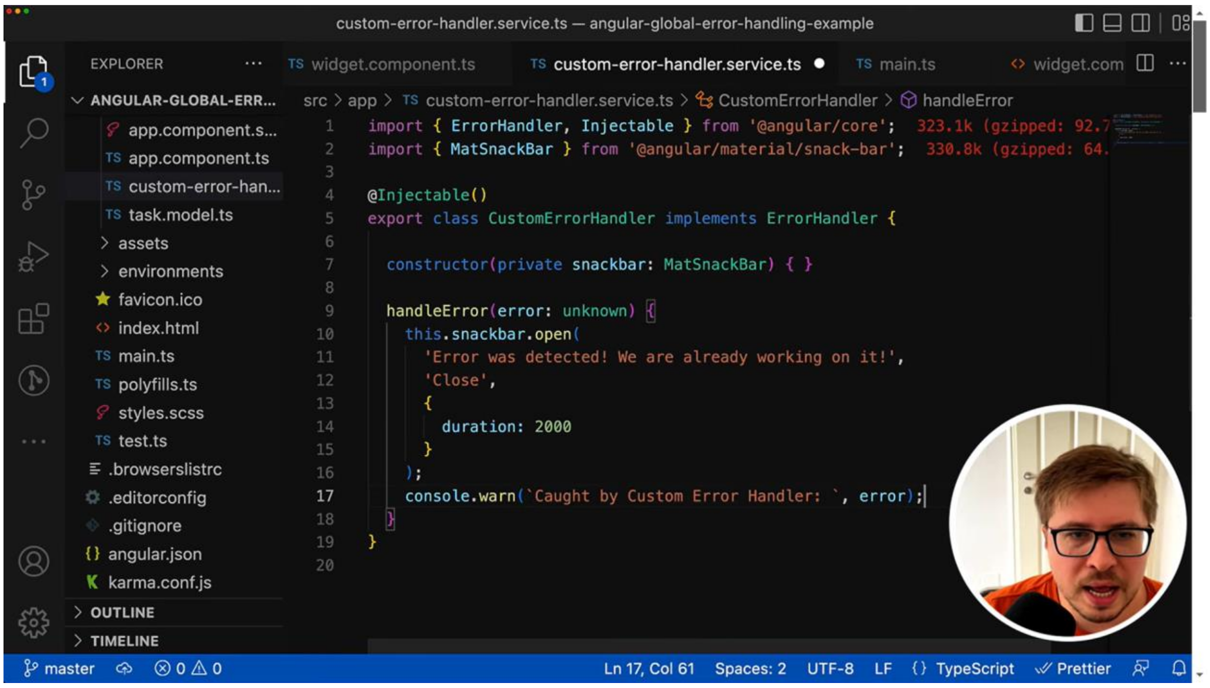
```
→ angular-global-error-handling-example git:(master) ✗ ng g s custom-error-handl
er --skip-tests
CREATE src/app/custom-error-handler.service.ts (147 bytes)
→ angular-global-error-handling-example git:(master) ✗
```

Import MatSnakbar



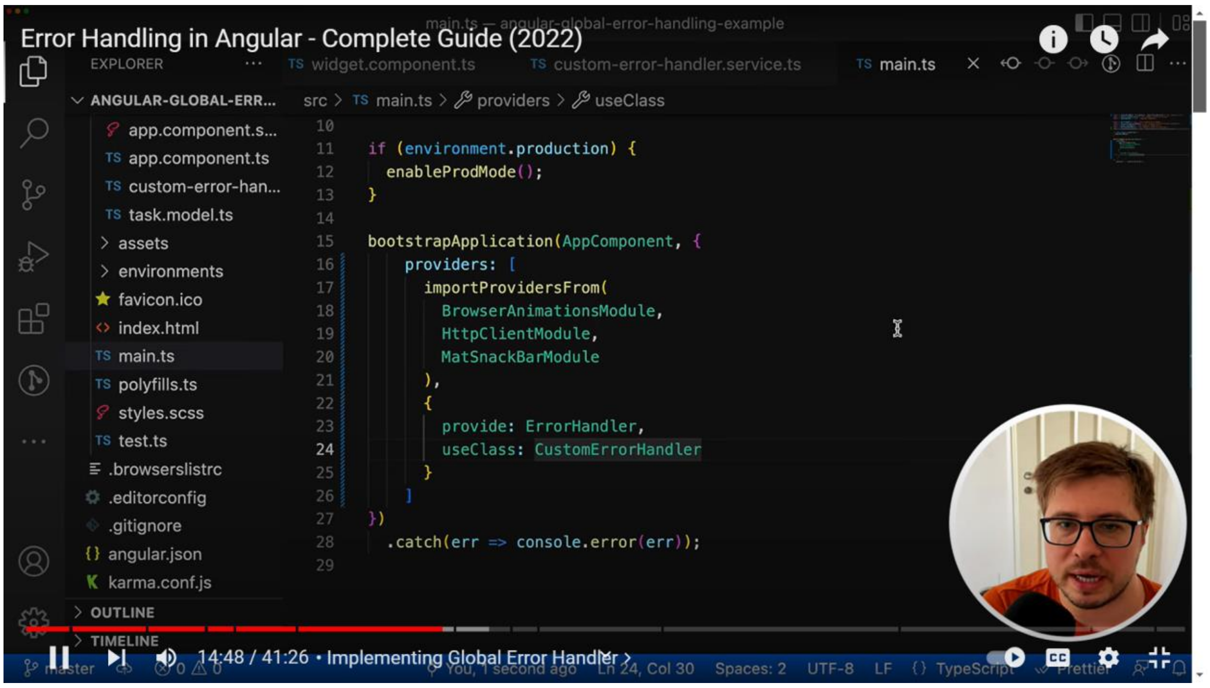
```
import { enableProdMode, importProvidersFrom } from '@angular/core';  323.3k (gz
import { bootstrapApplication } from '@angular/platform-browser';  28.3k (gzippe
import { AppComponent } from './app/app.component';

import { environment } from './environments/environment';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { HttpClientModule } from '@angular/common/http';  209.7k (gzipped: 50.2k
import { MatSnackBarModule } from '@angular/material/snack-bar';  330.8k (gzippe

if (environment.production) {
  enableProdMode();
}

bootstrapApplication(AppComponent, {
    providers: [
      importProvidersFrom(
        BrowserAnimationsModule,
        HttpClientModule,
        MatSnackBarModule
      )
    ]
})
  .catch(err => console.error(err));
```

Error handler method



```typescript
import { ErrorHandler, Injectable } from '@angular/core';
import { MatSnackBar } from '@angular/material/snack-bar';

@Injectable()
export class CustomErrorHandler implements ErrorHandler {

  constructor(private snackbar: MatSnackBar) { }

  handleError(error: unknown) {
    this.snackbar.open(
      'Error was detected! We are already working on it!',
      'Close',
      {
        duration: 2000
      }
    );
    console.warn(`Caught by Custom Error Handler: `, error);
  }
}
```

Inject this on main.ts or model for global level use

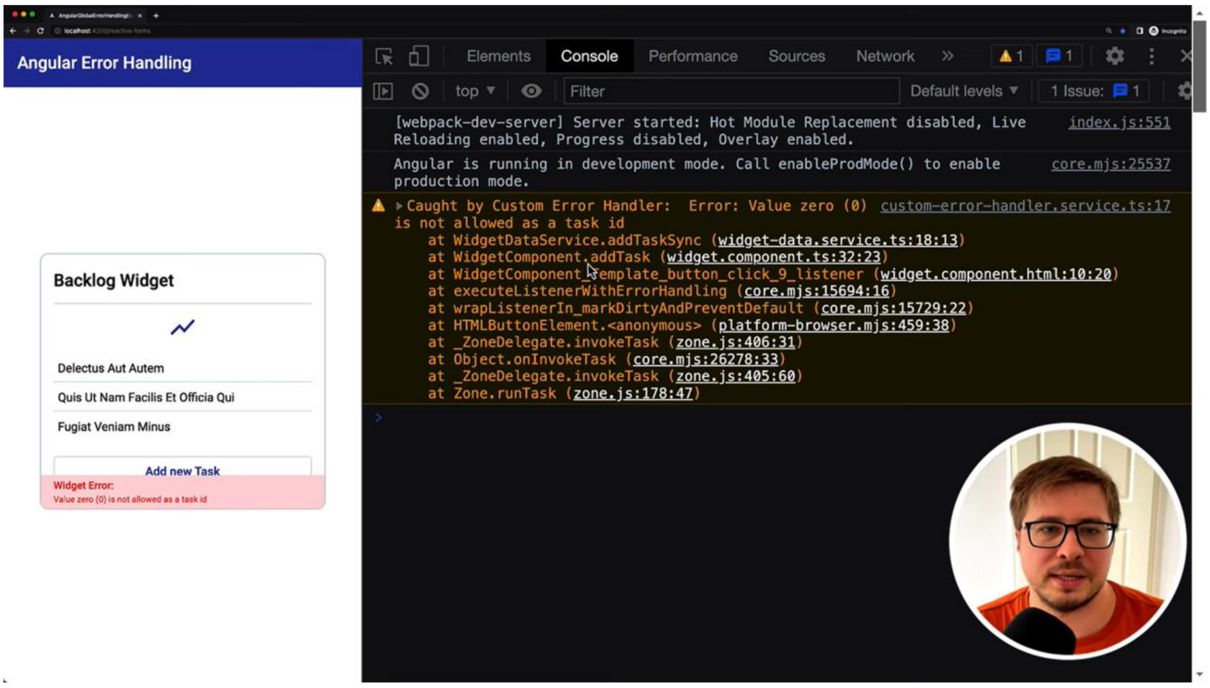


Error Handling in Angular - Complete Guide (2022)

```typescript
if (environment.production) {
  enableProdMode();
}

bootstrapApplication(AppComponent, {
  providers: [
    importProvidersFrom(
      BrowserAnimationsModule,
      HttpClientModule,
      MatSnackBarModule
    ),
    {
      provide: ErrorHandler,
      useClass: CustomErrorHandler
    }
  ]
})
  .catch(err => console.error(err));
```

14:48 / 41:26 • Implementing Global Error Handler

Throw error for capturing global based



Angular Error Handling

Backlog Widget

Delectus Aut Autem

Quis Ut Nam Facilis Et Officia Qui

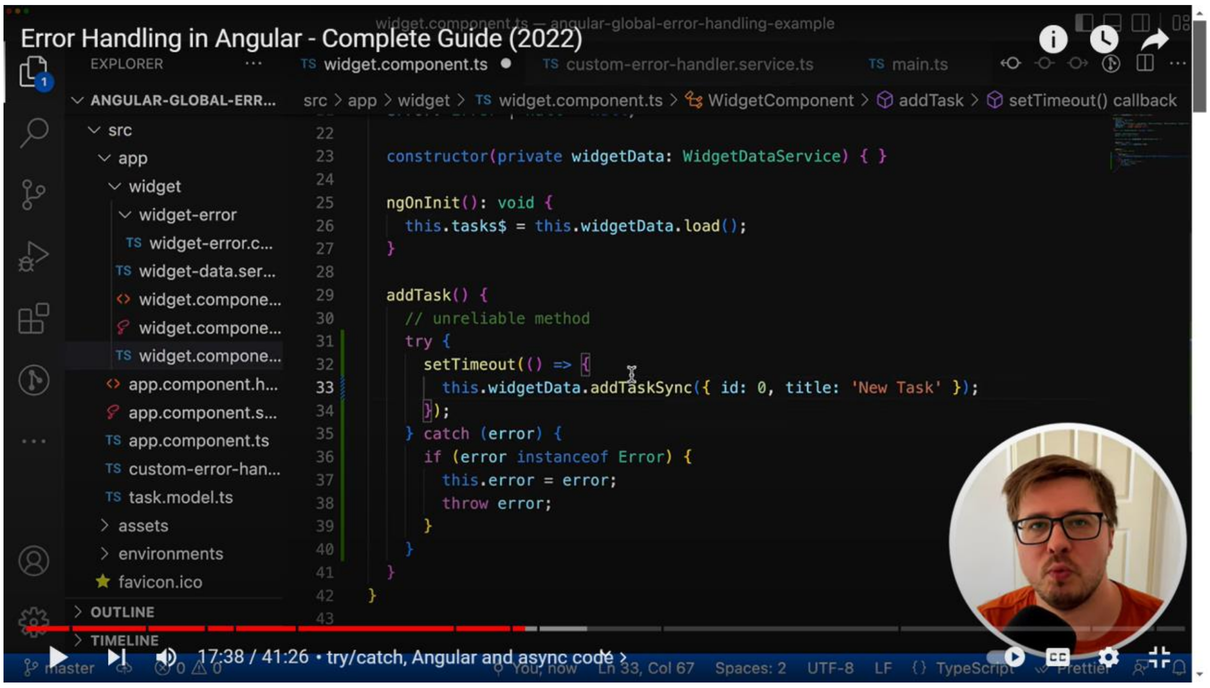
Fugiat Veniam Minus

Add new Task

Widget Error:
Value zero (0) is not allowed as a task id

```
[webpack-dev-server] Server started: Hot Module Replacement disabled, Live     index.js:551
Reloading enabled, Progress disabled, Overlay enabled.
Angular is running in development mode. Call enableProdMode() to enable     core.mjs:25537
production mode.
▸Caught by Custom Error Handler:  Error: Value zero (0) custom-error-handler.service.ts:17
is not allowed as a task id
    at WidgetDataService.addTaskSync (widget-data.service.ts:18:13)
    at WidgetComponent.addTask (widget.component.ts:32:23)
    at WidgetComponent_Template_button_click_9_listener (widget.component.html:10:20)
    at executeListenerWithErrorHandling (core.mjs:15694:16)
    at wrapListenerIn_markDirtyAndPreventDefault (core.mjs:15729:22)
    at HTMLButtonElement.<anonymous> (platform-browser.mjs:459:38)
    at _ZoneDelegate.invokeTask (zone.js:406:31)
    at Object.onInvokeTask (core.mjs:26278:33)
    at _ZoneDelegate.invokeTask (zone.js:405:60)
    at Zone.runTask (zone.js:178:47)
```

Try-catch does not handle this error but by global



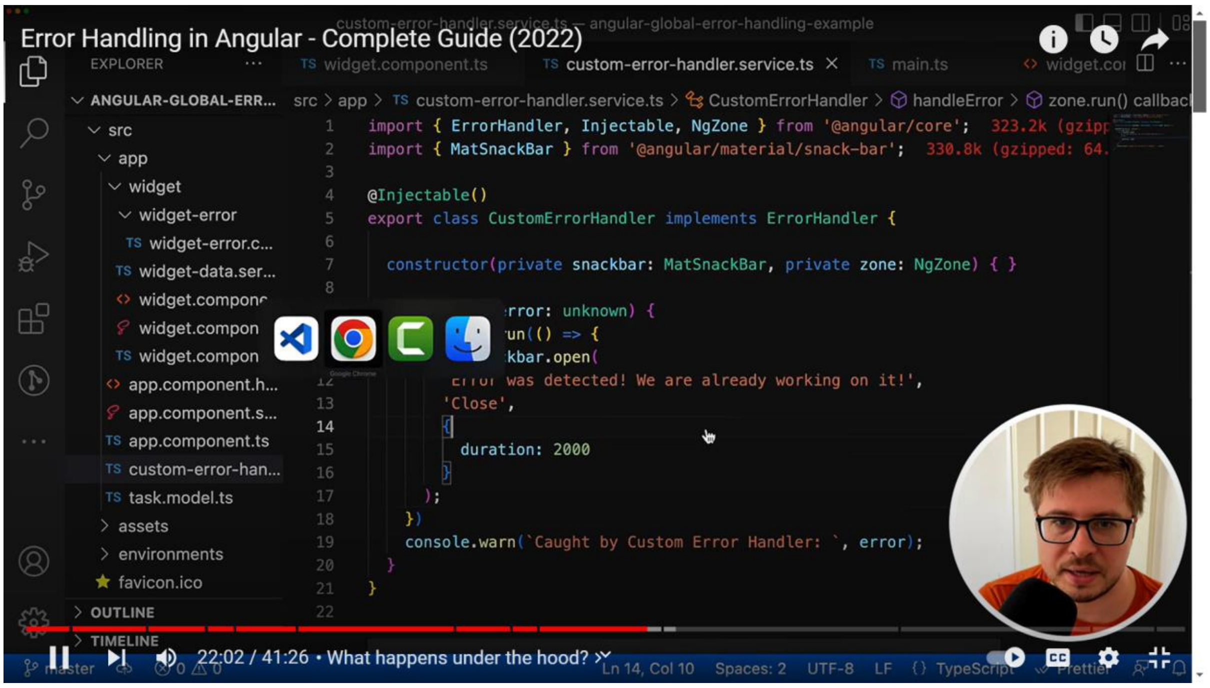
Error Handling in Angular - Complete Guide (2022)

widget.component.ts — angular-global-error-handling-example

src > app > widget > widget.component.ts > WidgetComponent > addTask > setTimeout() callback

```typescript
  constructor(private widgetData: WidgetDataService) { }

  ngOnInit(): void {
    this.tasks$ = this.widgetData.load();
  }

  addTask() {
    // unreliable method
    try {
      setTimeout(() => {
        this.widgetData.addTaskSync({ id: 0, title: 'New Task' });
      });
    } catch (error) {
      if (error instanceof Error) {
        this.error = error;
        throw error;
      }
    }
  }
}
```

17:38 / 41:26 • try/catch, Angular and async code ›



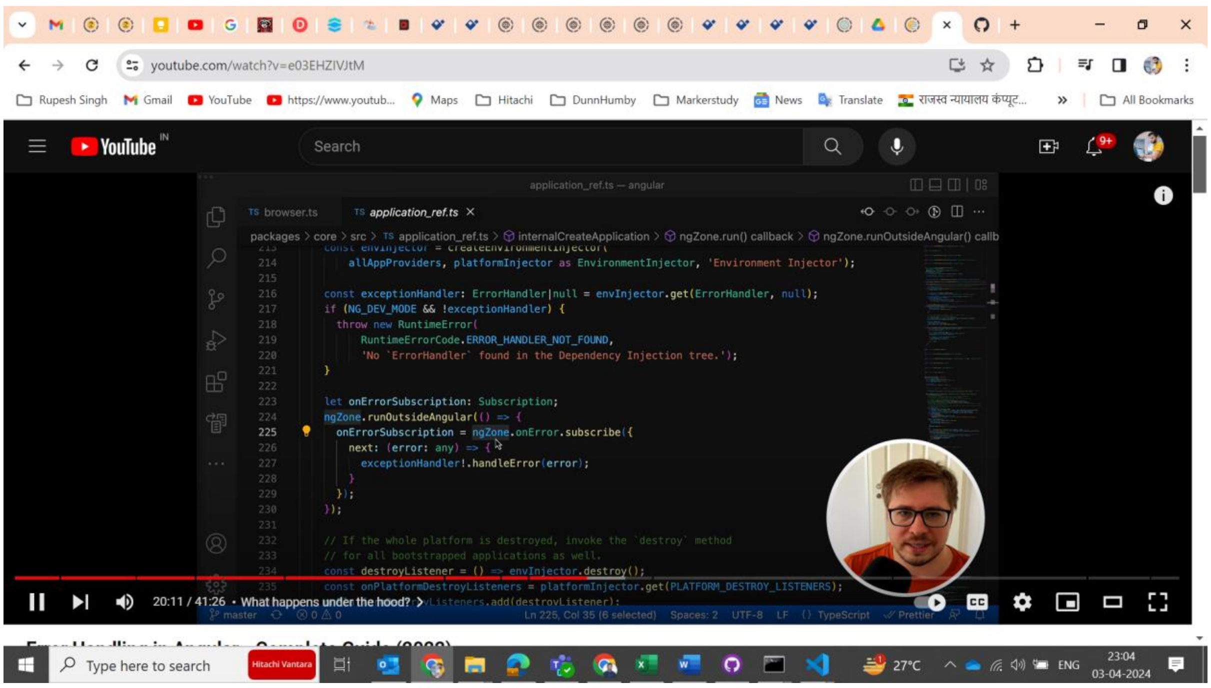
```
import { ErrorHandler, Injectable, NgZone } from '@angular/core';  323.2k (gzipp
import { MatSnackBar } from '@angular/material/snack-bar';  330.8k (gzipped: 64.

@Injectable()
export class CustomErrorHandler implements ErrorHandler {

  constructor(private snackbar: MatSnackBar, private zone: NgZone) { }

                  rror: unknown) {
                  un(() => {
                  kbar.open(
          [illegible] was detected! We are already working on it!',
          'Close',
          {
            duration: 2000
          }
        );
      })
      console.warn(`Caught by Custom Error Handler: `, error);
    }
}
```

```
 throw error // used for send global based error
```

Try-catch will not work with Async call

```
 try {
      // this is an example
      setTimeout(() => {
        this.widgetData.addTaskSync({ id: 0, title: 'New Task' });
      });

    }
    catch (error) {
      if (error instanceof Error) {
        this.error = error
        throw error // for global error
      }
```

But why because ngzone find bellow screen



```
const envInjector = createEnvironmentInjector(
    allAppProviders, platformInjector as EnvironmentInjector, 'Environment Injector');

const exceptionHandler: ErrorHandler|null = envInjector.get(ErrorHandler, null);
if (NG_DEV_MODE && !exceptionHandler) {
  throw new RuntimeError(
      RuntimeErrorCode.ERROR_HANDLER_NOT_FOUND,
      'No `ErrorHandler` found in the Dependency Injection tree.');
}

let onErrorSubscription: Subscription;
ngZone.runOutsideAngular(() => {
  onErrorSubscription = ngZone.onError.subscribe({
    next: (error: any) => {
      exceptionHandler!.handleError(error);
    }
  });
});

// If the whole platform is destroyed, invoke the `destroy` method
// for all bootstrapped applications as well.
const destroyListener = () => envInjector.destroy();
const onPlatformDestroyListeners = platformInjector.get(PLATFORM_DESTROY_LISTENERS);
```

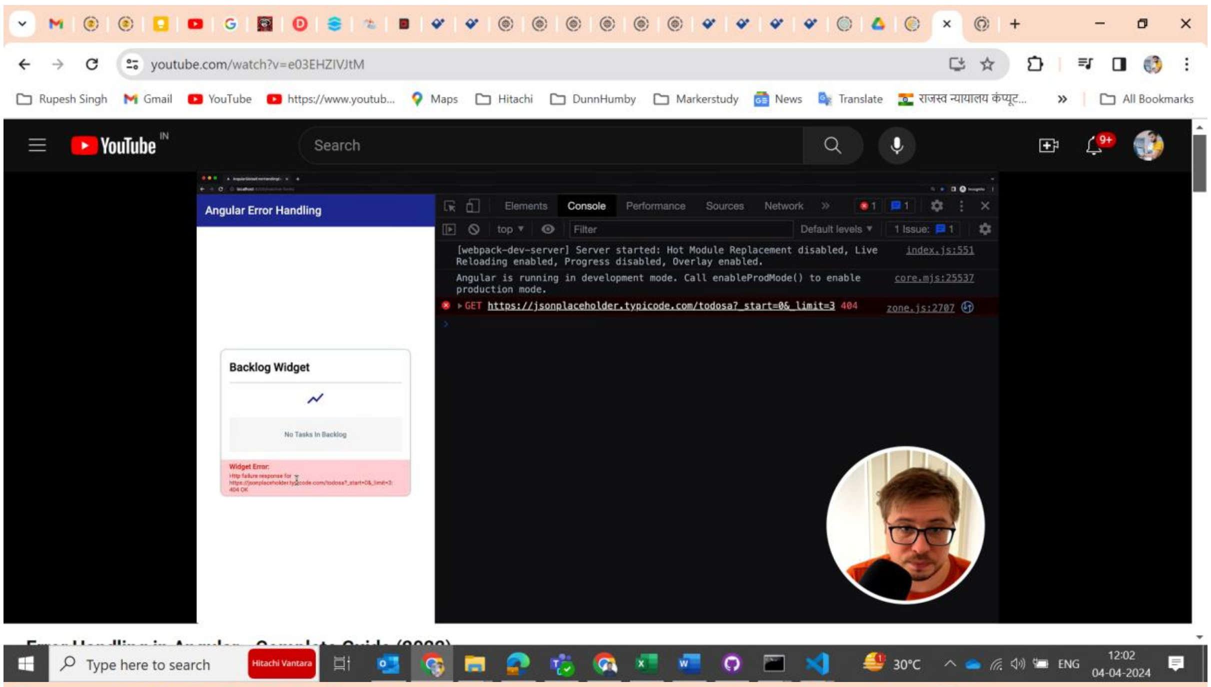
youtube.com/watch?v=e03EHZIVJtM
application_ref.ts — angular
browser.ts
application_ref.ts
packages > core > src > application_ref.ts > internalCreateApplication > ngZone.run() callback > ngZone.runOutsideAngular() callb
20:11 / 41:26 • What happens under the hood? >
master
Ln 225, Col 35 (6 selected) Spaces: 2 UTF-8 LF TypeScript Prettier
Type here to search
27°C ENG 23:04 03-04-2024

Pipe are used for error handling

```
 load() {
    return
this.http.get<Task[]>(`https://jsonplaceholder.typicode.comdfdsf/todos?_start=
0&_limit=3`).pipe(
      catchError(() => {
        console.info('Error handled by widget service...')
        return throwError(() => new Error('Could not load Data'));
      })
    )
  }
```

This code will show for both error and message

```
 ngOnInit(): void {
    this.tasks$ = this.widgetData.load().pipe(
      tap({
        error: (error: Error | null): any => this.error = error
      }),
      catchError(err => of([])),
    );
  }
```

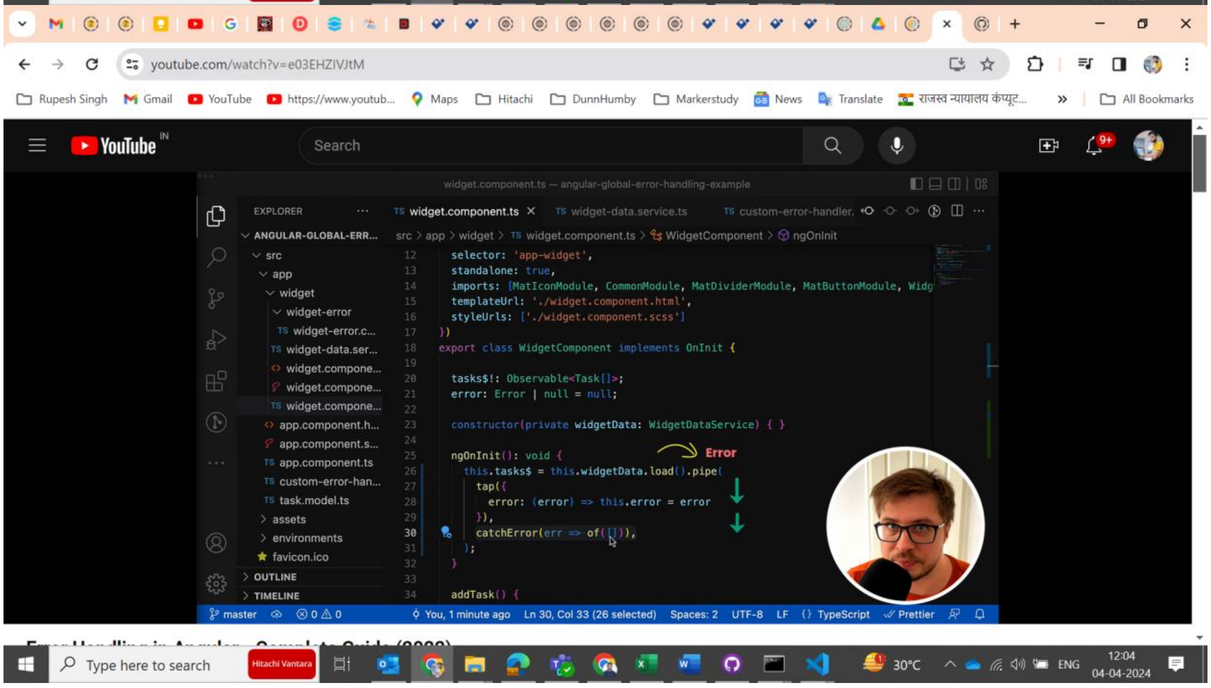


Backlog Widget

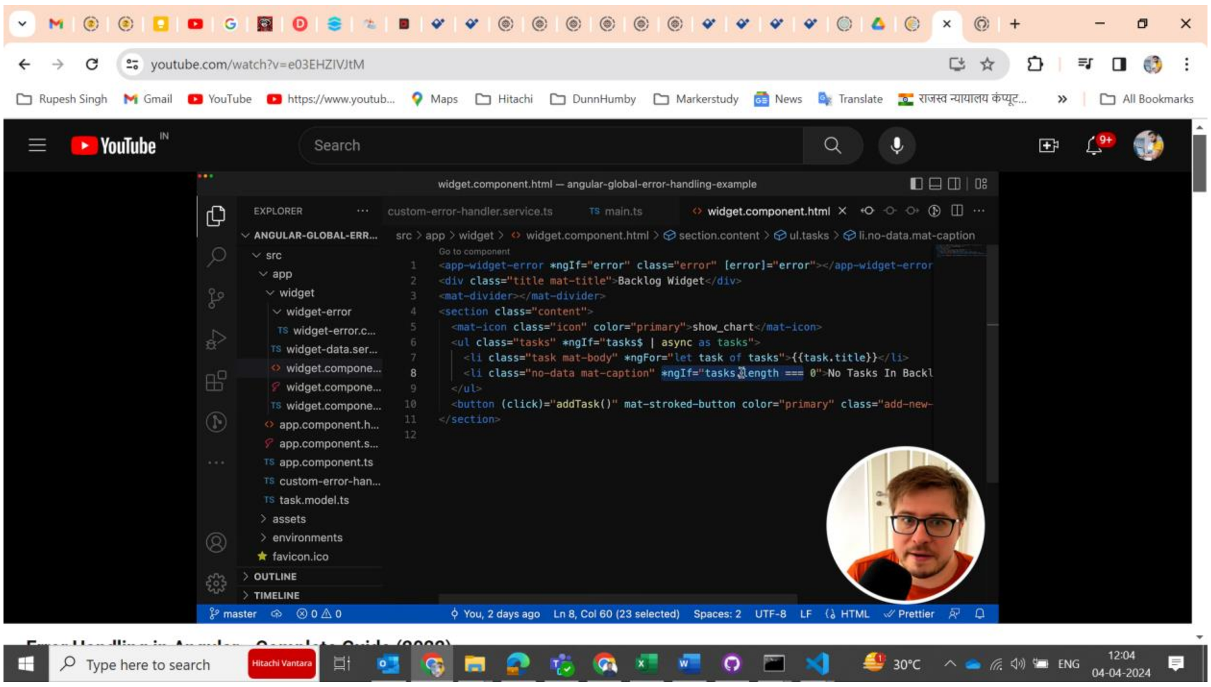
No Tasks In Backlog

Widget Error:
Http failure response for https://jsonplaceholder.typicode.com/todosa?_start=0&_limit=3: 404 OK

```
[webpack-dev-server] Server started: Hot Module Replacement disabled, Live      index.js:551
Reloading enabled, Progress disabled, Overlay enabled.
Angular is running in development mode. Call enableProdMode() to enable       core.mjs:25537
production mode.
▸GET https://jsonplaceholder.typicode.com/todosa?_start=0&_limit=3 404        zone.js:2707
```



```typescript
  selector: 'app-widget',
  standalone: true,
  imports: [MatIconModule, CommonModule, MatDividerModule, MatButtonModule, Widg
  templateUrl: './widget.component.html',
  styleUrls: ['./widget.component.scss']
})
export class WidgetComponent implements OnInit {

  tasks$!: Observable<Task[]>;
  error: Error | null = null;

  constructor(private widgetData: WidgetDataService) { }

  ngOnInit(): void {
    this.tasks$ = this.widgetData.load().pipe(
      tap({
        error: (error) => this.error = error
      }),
      catchError(err => of([])),
    );
  }

  addTask() {
```

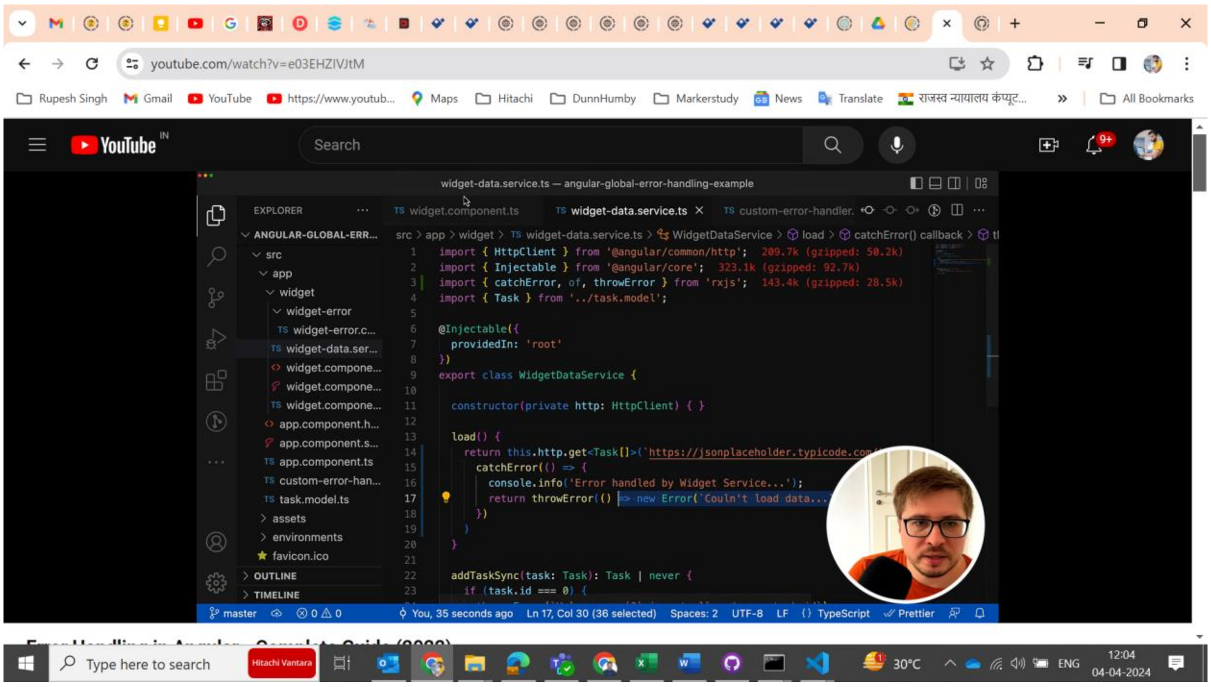


widget.component.html — angular-global-error-handling-example

src > app > widget > widget.component.html > section.content > ul.tasks > li.no-data.mat-caption

```html
<app-widget-error *ngIf="error" class="error" [error]="error"></app-widget-error
<div class="title mat-title">Backlog Widget</div>
<mat-divider></mat-divider>
<section class="content">
  <mat-icon class="icon" color="primary">show_chart</mat-icon>
  <ul class="tasks" *ngIf="tasks$ | async as tasks">
    <li class="task mat-body" *ngFor="let task of tasks">{{task.title}}</li>
    <li class="no-data mat-caption" *ngIf="tasks.length === 0">No Tasks In Backl
  </ul>
  <button (click)="addTask()" mat-stroked-button color="primary" class="add-new-
</section>
```

Downstream



widget-data.service.ts — angular-global-error-handling-example

src > app > widget > widget-data.service.ts > WidgetDataService > load > catchError() callback

```typescript
import { HttpClient } from '@angular/common/http';  209.7k (gzipped: 50.2k)
import { Injectable } from '@angular/core';  323.1k (gzipped: 92.7k)
import { catchError, of, throwError } from 'rxjs';  143.4k (gzipped: 28.5k)
import { Task } from '../task.model';

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

  constructor(private http: HttpClient) { }

  load() {
    return this.http.get<Task[]>(`https://jsonplaceholder.typicode.com
      catchError(() => {
        console.info('Error handled by Widget Service...');
        return throwError(() => new Error(`Couln't load data...
      })
    )
  }

  addTaskSync(task: Task): Task | never {
    if (task.id === 0) {
```

Global handling of HTTP errors

And retry http request

To create re

```
ng g interceptor global-http-error-handler --skip-tests
```

1. step

```
 intercept(request: HttpRequest<unknown>, next: HttpHandler):
Observable<HttpEvent<unknown>> {
    return next.handle(request).pipe(
      retry({
        count: 3,
        delay: (_, retryCont) => timer(retryCont * 1000), // 1sec ,2sec, 3sec
      }),
      catchError(err => {
        console.log('Error handled by HTTP interceptor...')
        return throwError(() => {
          console.log('Error rethrogh by Http Interceptor')
          return err
        })
      })
    );

  }
```

2. step

import this on main.ts

```
{provide:HTTP_INTERCEPTORS,useClass:GlobalHttpErrorHandlerInterceptor,multi:tr
ue}
```

3. component.ts

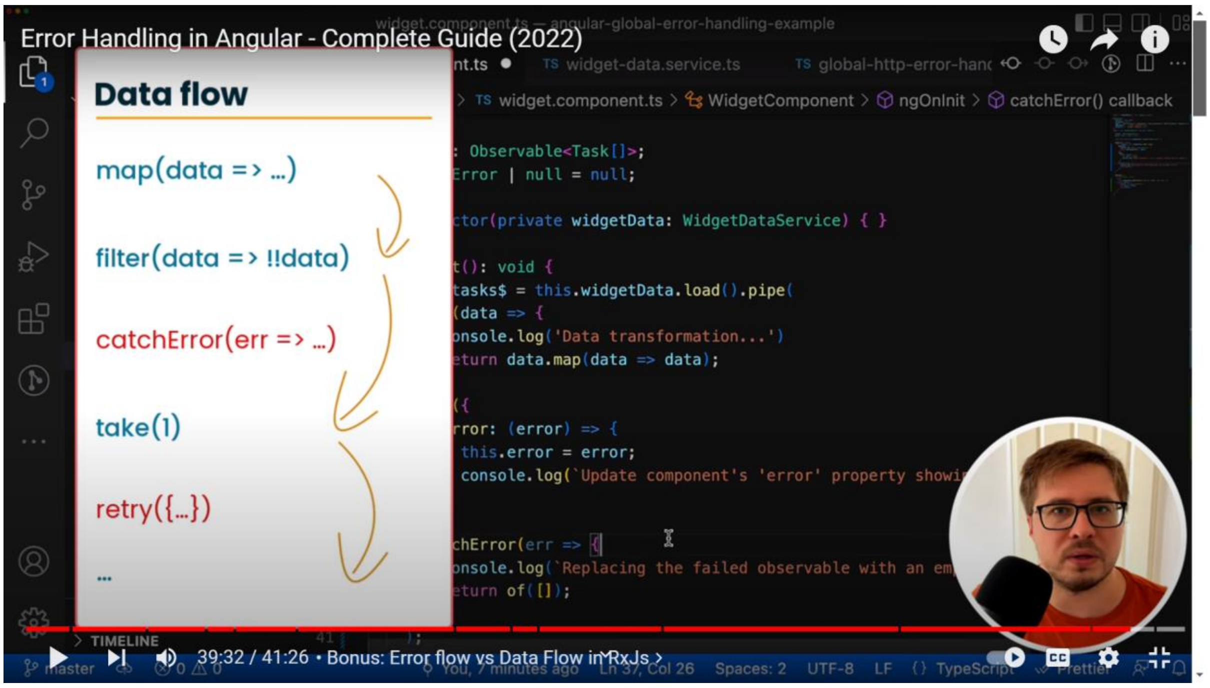
```
ngOnInit(): void {
    this.tasks$ = this.widgetData.load().pipe(
      tap({
        error: (error: Error | null): any => {
          this.error = error
          console.log('Update components error property showing ')
        }
      }),
      catchError(err => {
        console.log('replacing ther failed obeserables with an emoty array')
        return of([])
      }),
    );
  }
```

```
   4. Step   data.service.ts
```

```
 load() {
    return
this.http.get<Task[]>(`https://jsonplaceholder.typicode.com/todos?_start=0&_li
mit=3`).pipe(
      catchError(() => {
        console.info('Error handled by widget service...')
        return throwError(() => {
          console.log('Error rethrogh by widget service')
          return new Error('Could not load Data')
        });
      })
    )
  }
```

Map() operator are skiped

How data is flow if everything is fine



Error Handling in Angular - Complete Guide (2022)

Data flow

map(data => …)

filter(data => !!data)

catchError(err => …)

take(1)

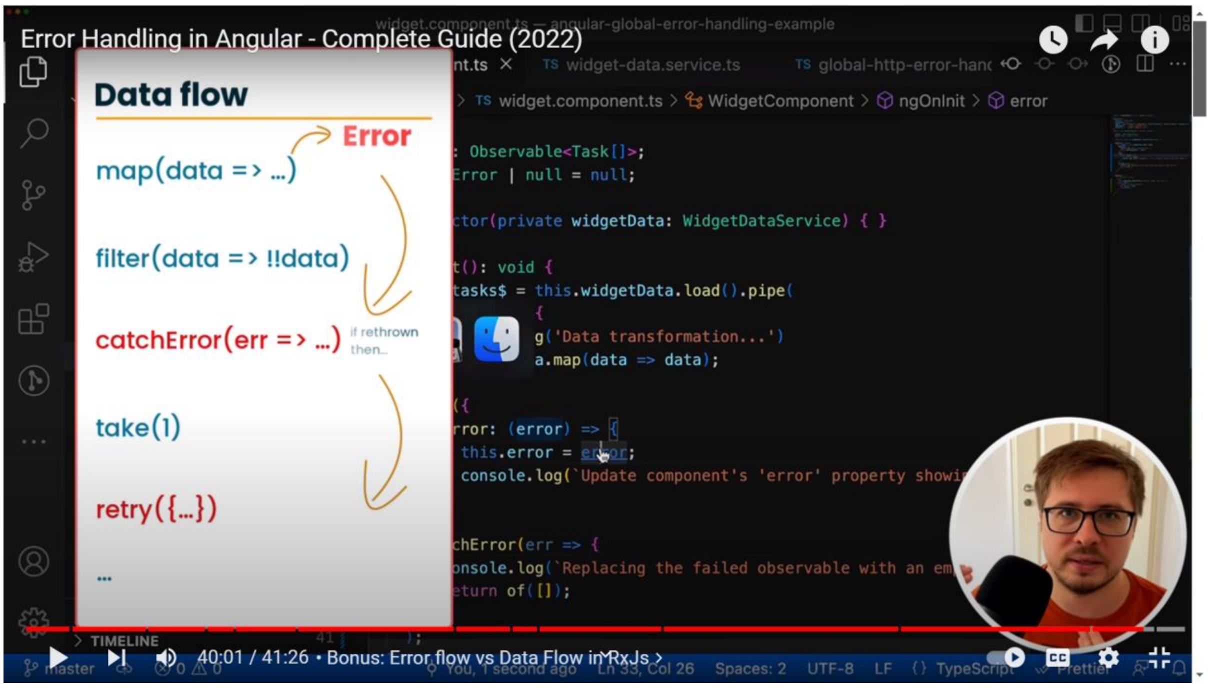
retry({…})

...

```
widget.component.ts > WidgetComponent > ngOnInit > catchError() callback

: Observable<Task[]>;
Error | null = null;

ctor(private widgetData: WidgetDataService) { }

t(): void {
tasks$ = this.widgetData.load().pipe(
(data => {
onsole.log('Data transformation...')
eturn data.map(data => data);

({
rror: (error) => {
 this.error = error;
 console.log(`Update component's 'error' property showi

chError(err => {
onsole.log(`Replacing the failed observable with an em
eturn of([]);
```

39:32 / 41:26 • Bonus: Error flow vs Data Flow in RxJs >

If error came how data flow



Error Handling in Angular - Complete Guide (2022)

Data flow

Error

map(data => …)

filter(data => !!data)

catchError(err => …) if rethrown then…

take(1)

retry({…})

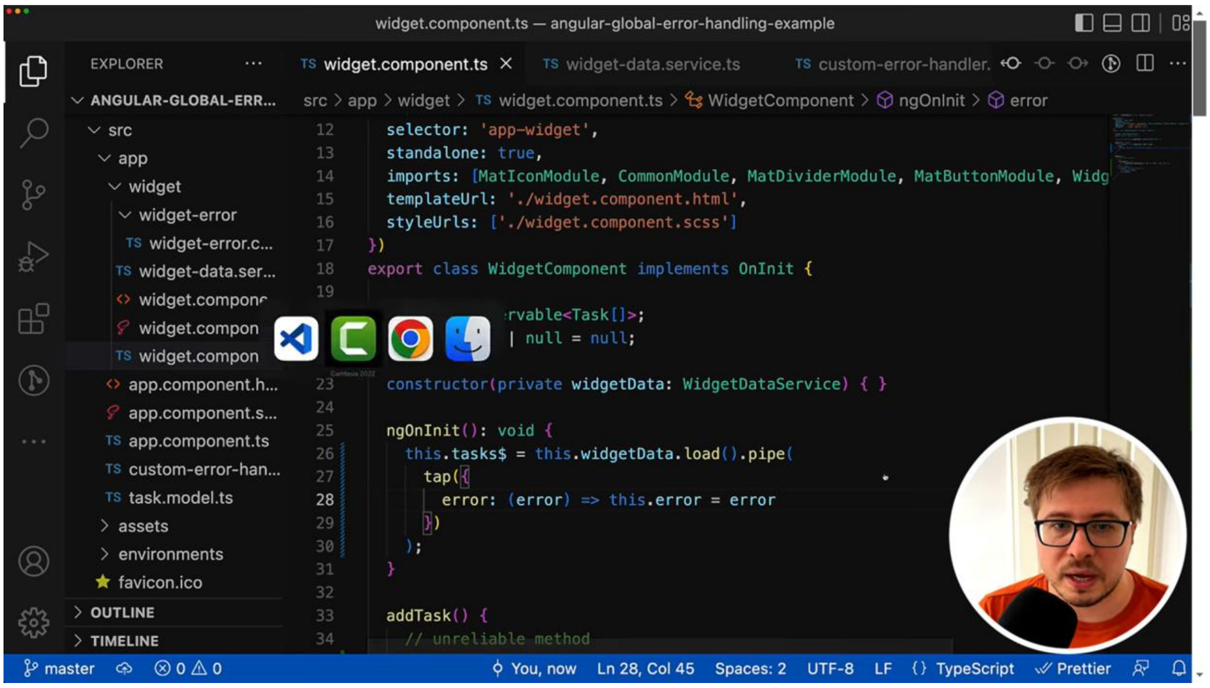
...

```
widget.component.ts > WidgetComponent > ngOnInit > error

: Observable<Task[]>;
Error | null = null;

ctor(private widgetData: WidgetDataService) { }

t(): void {
tasks$ = this.widgetData.load().pipe(
 {
 g('Data transformation...')
 a.map(data => data);

({
rror: (error) => {
 this.error = error;
 console.log(`Update component's 'error' property showi

chError(err => {
onsole.log(`Replacing the failed observable with an em
eturn of([]);
```

40:01 / 41:26 • Bonus: Error flow vs Data Flow in RxJs >

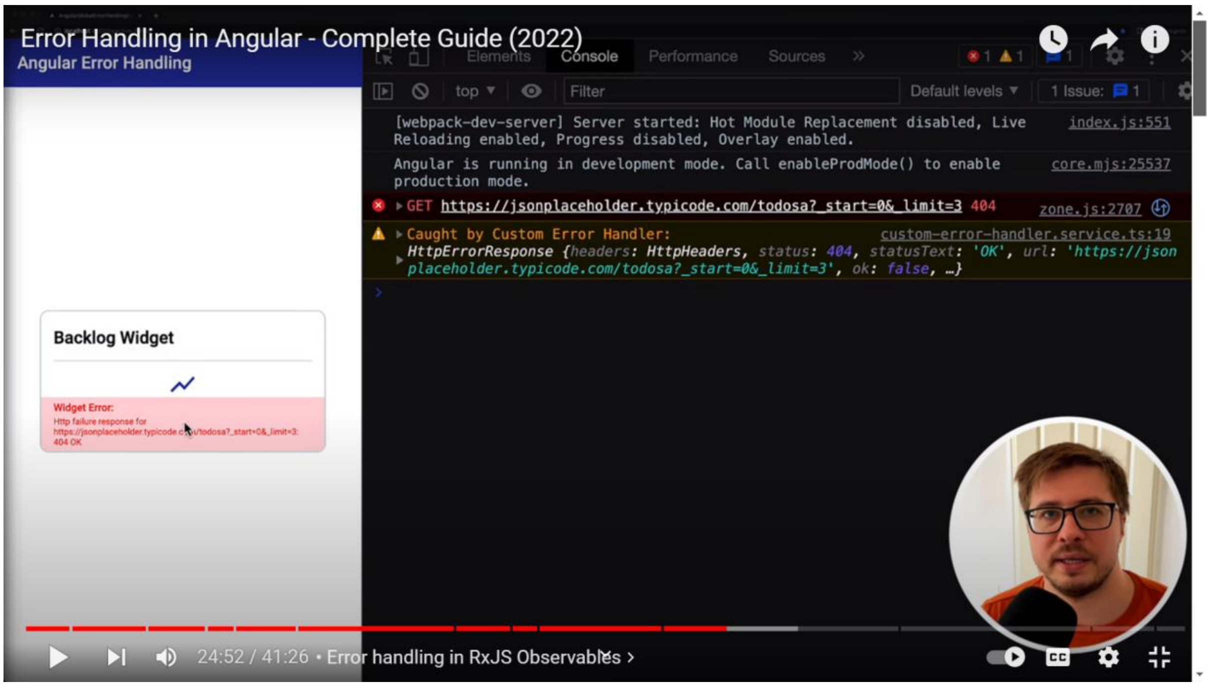
Map() operator are ignored

How pipe error are working let see



```
  selector: 'app-widget',
  standalone: true,
  imports: [MatIconModule, CommonModule, MatDividerModule, MatButtonModule, Widg
  templateUrl: './widget.component.html',
  styleUrls: ['./widget.component.scss']
})
export class WidgetComponent implements OnInit {

            rvable<Task[]>;
            | null = null;

  constructor(private widgetData: WidgetDataService) { }

  ngOnInit(): void {
    this.tasks$ = this.widgetData.load().pipe(
      tap({
        error: (error) => this.error = error
      })
    );
  }

  addTask() {
    // unreliable method
```

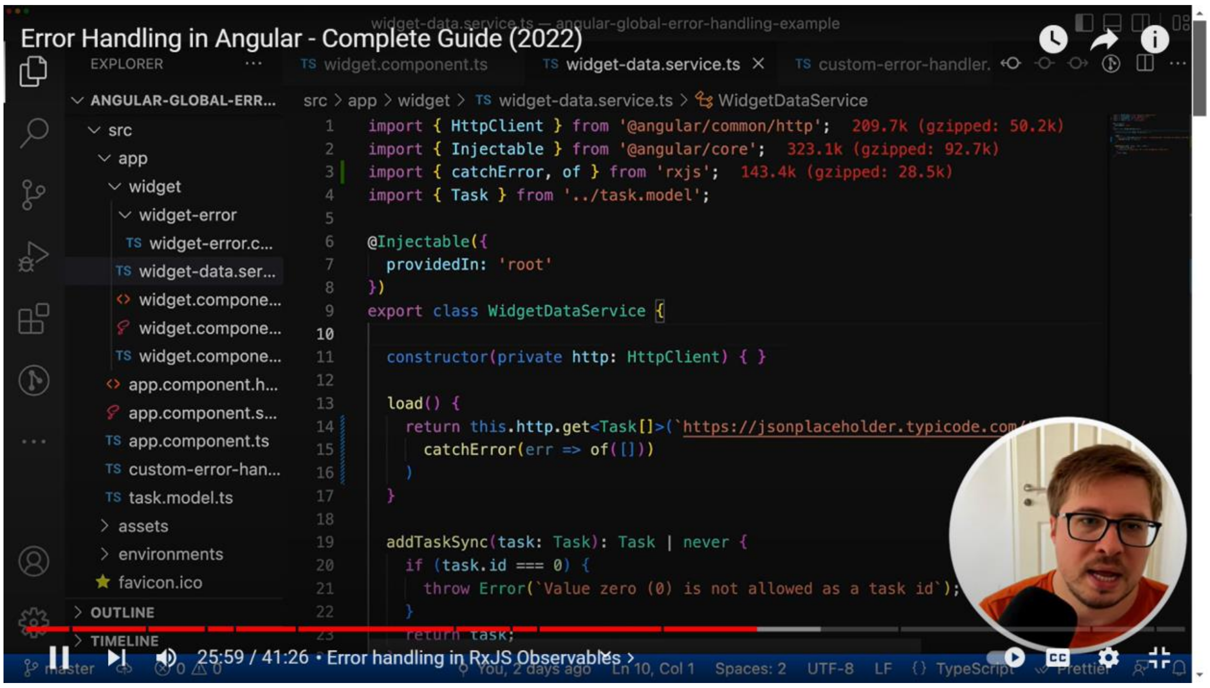
By using tap can trace error from service like this using async pipe


Error Handling in Angular - Complete Guide (2022)
Angular Error Handling
Backlog Widget
Widget Error:
Http failure response for https://jsonplaceholder.typicode.com/todosa?_start=0&_limit=3: 404 OK
24:52 / 41:26 • Error handling in RxJS Observables ›

```
[webpack-dev-server] Server started: Hot Module Replacement disabled, Live     index.js:551
Reloading enabled, Progress disabled, Overlay enabled.
Angular is running in development mode. Call enableProdMode() to enable      core.mjs:25537
production mode.
GET https://jsonplaceholder.typicode.com/todosa?_start=0&_limit=3 404         zone.js:2707
Caught by Custom Error Handler:                                custom-error-handler.service.ts:19
HttpErrorResponse {headers: HttpHeaders, status: 404, statusText: 'OK', url: 'https://json
placeholder.typicode.com/todosa?_start=0&_limit=3', ok: false, …}
```

Catch and replace strategy in error

From service


Error Handling in Angular - Complete Guide (2022)
widget-data.service.ts — angular-global-error-handling-example
widget.component.ts
widget-data.service.ts
custom-error-handler.
src > app > widget > widget-data.service.ts > WidgetDataService
25:59 / 41:26 • Error handling in RxJS Observables ›

```
import { HttpClient } from '@angular/common/http';  209.7k (gzipped: 50.2k)
import { Injectable } from '@angular/core';  323.1k (gzipped: 92.7k)
import { catchError, of } from 'rxjs';  143.4k (gzipped: 28.5k)
import { Task } from '../task.model';

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

  constructor(private http: HttpClient) { }

  load() {
    return this.http.get<Task[]>(`https://jsonplaceholder.typicode.com/
      catchError(err => of([]))
    )
  }

  addTaskSync(task: Task): Task | never {
    if (task.id === 0) {
      throw Error(`Value zero (0) is not allowed as a task id`);
    }
    return task;
```

`https://www.youtube.com/watch?v=e03EHZIVJtM`

Error Handling in Obserables

2. error handling middleware
3. global error handling,
4. error handling within components.