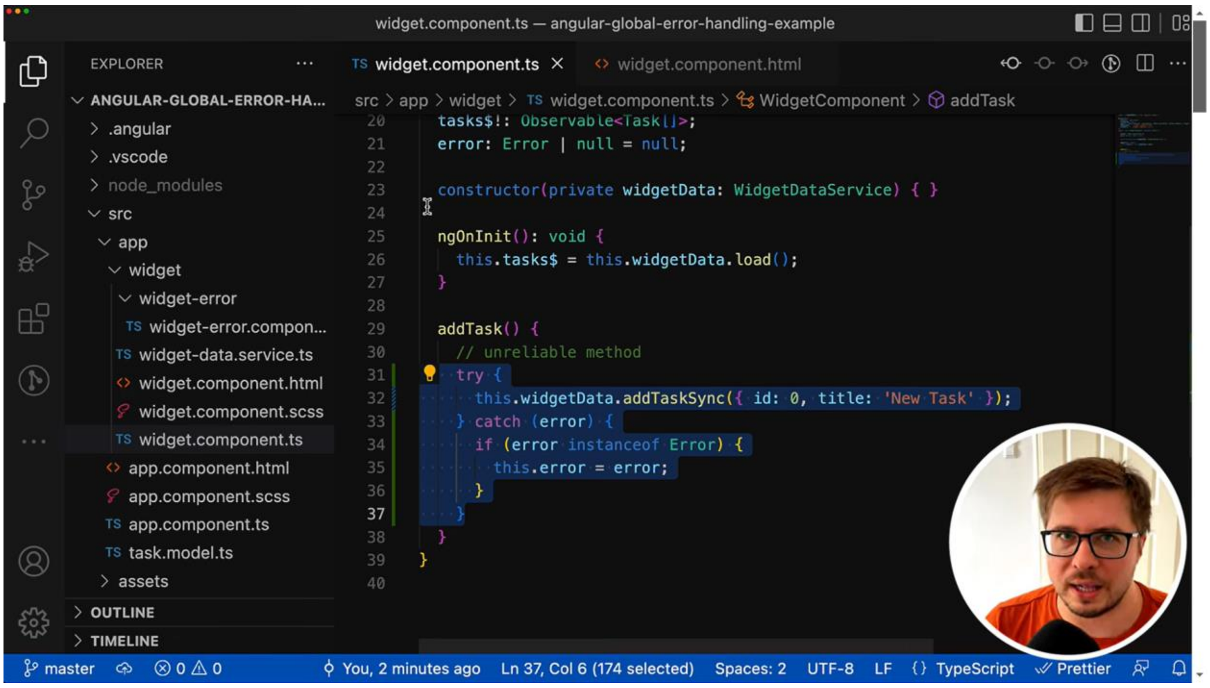
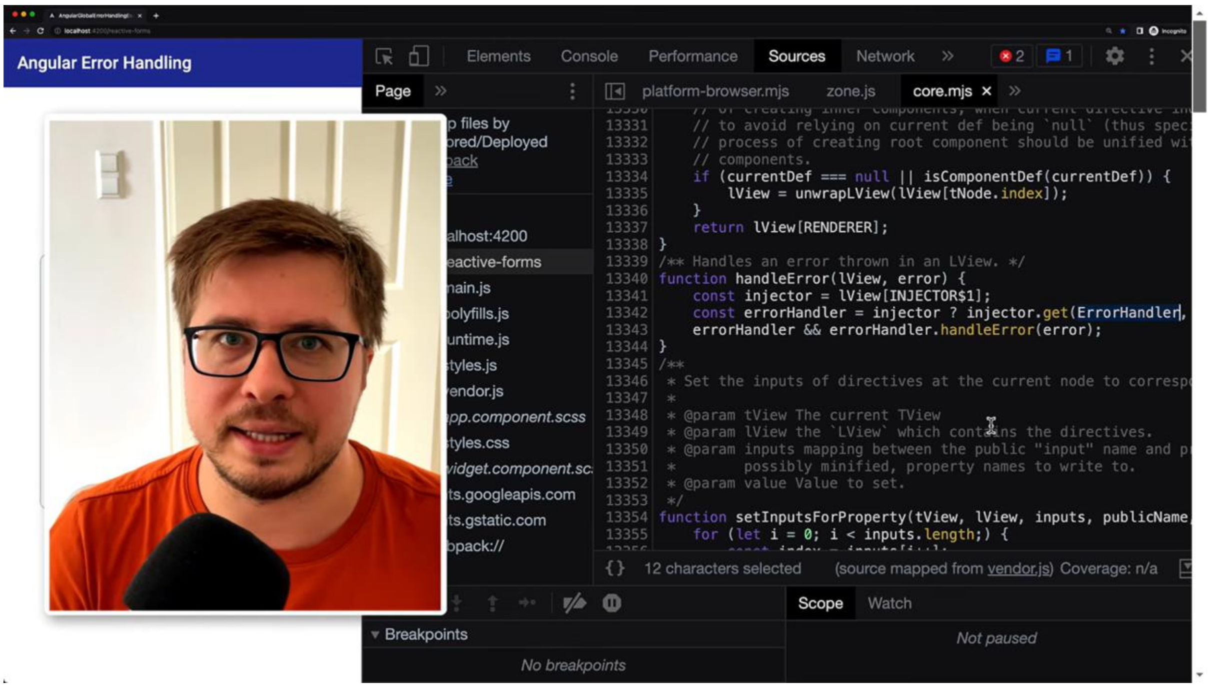
1. `try-catch` blocks



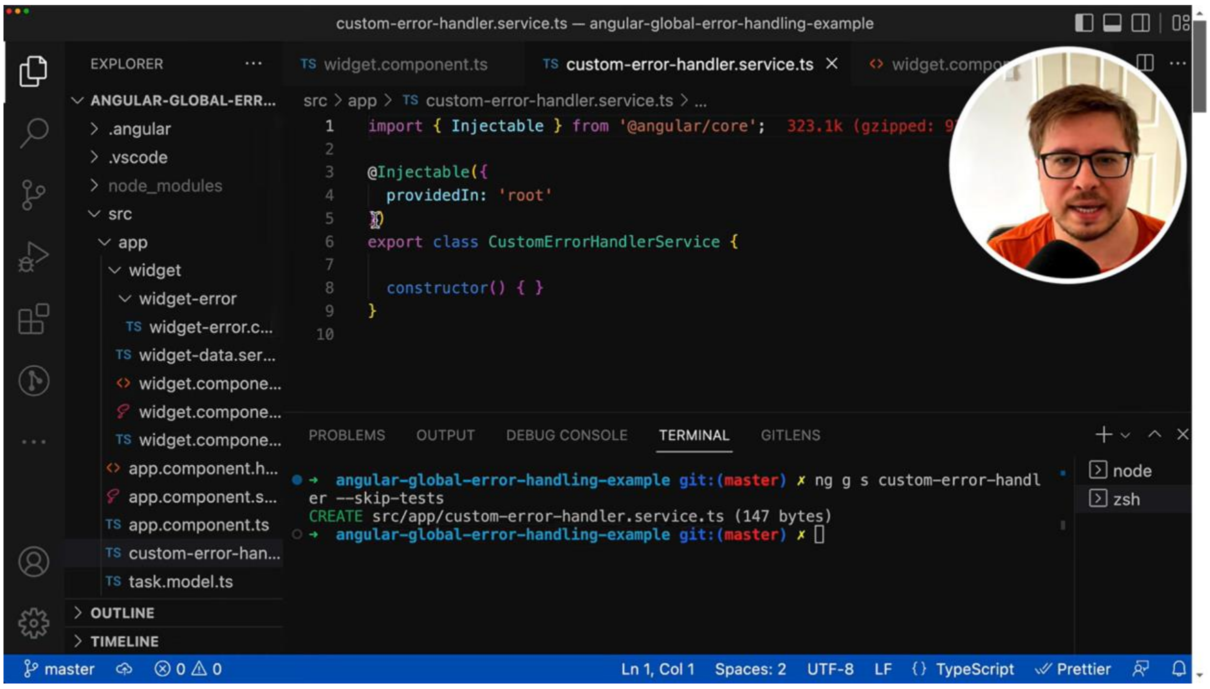
```typescript
  tasks$!: Observable<Task[]>;
  error: Error | null = null;

  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;
      }
    }
  }
}
```



```javascript
    // 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;) {
```

12 characters selected (source mapped from vendor.js) Coverage: n/a

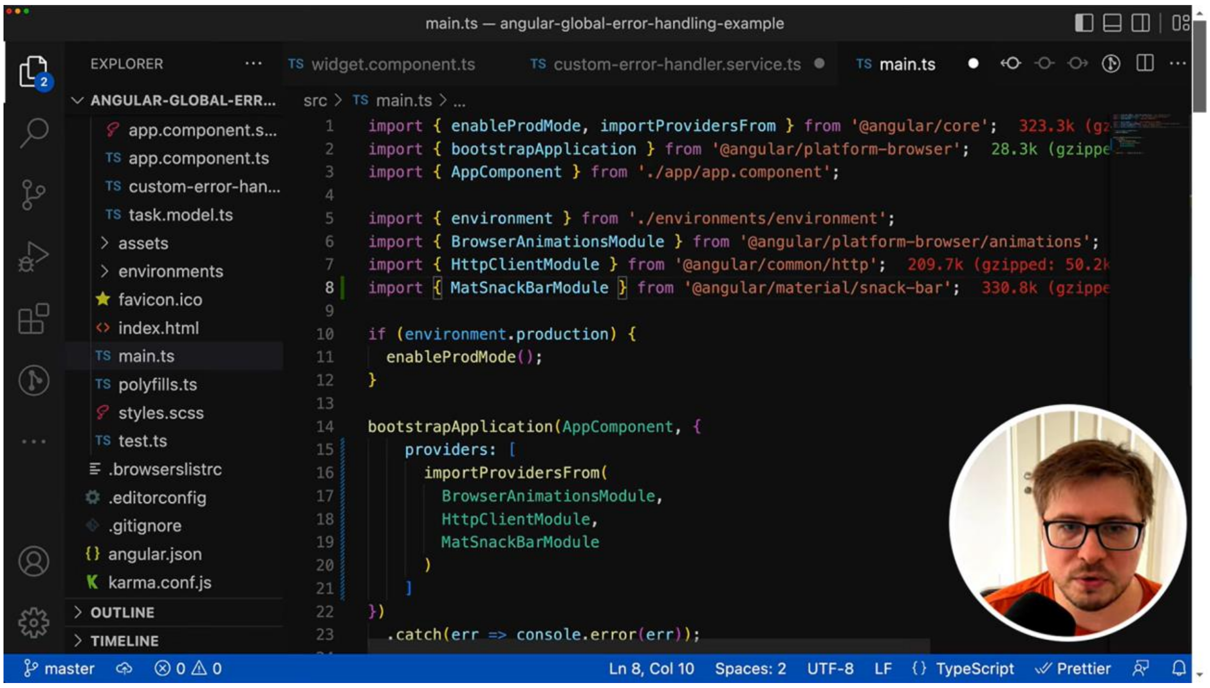
Created custom-error-handler.service



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

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

  constructor() { }
}
```

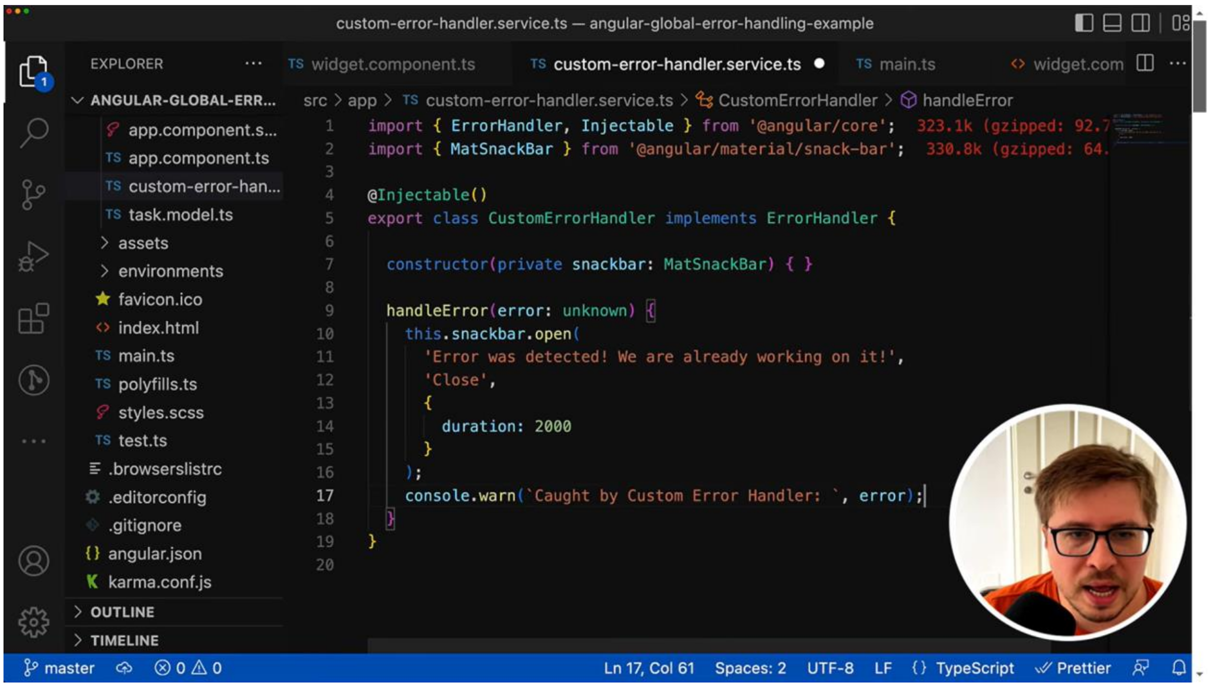
```
➜  angular-global-error-handling-example git:(master) ✗ ng g s custom-error-handl
er --skip-tests
CREATE src/app/custom-error-handler.service.ts (147 bytes)
➜  angular-global-error-handling-example git:(master) ✗
```

Import MatSnakbar



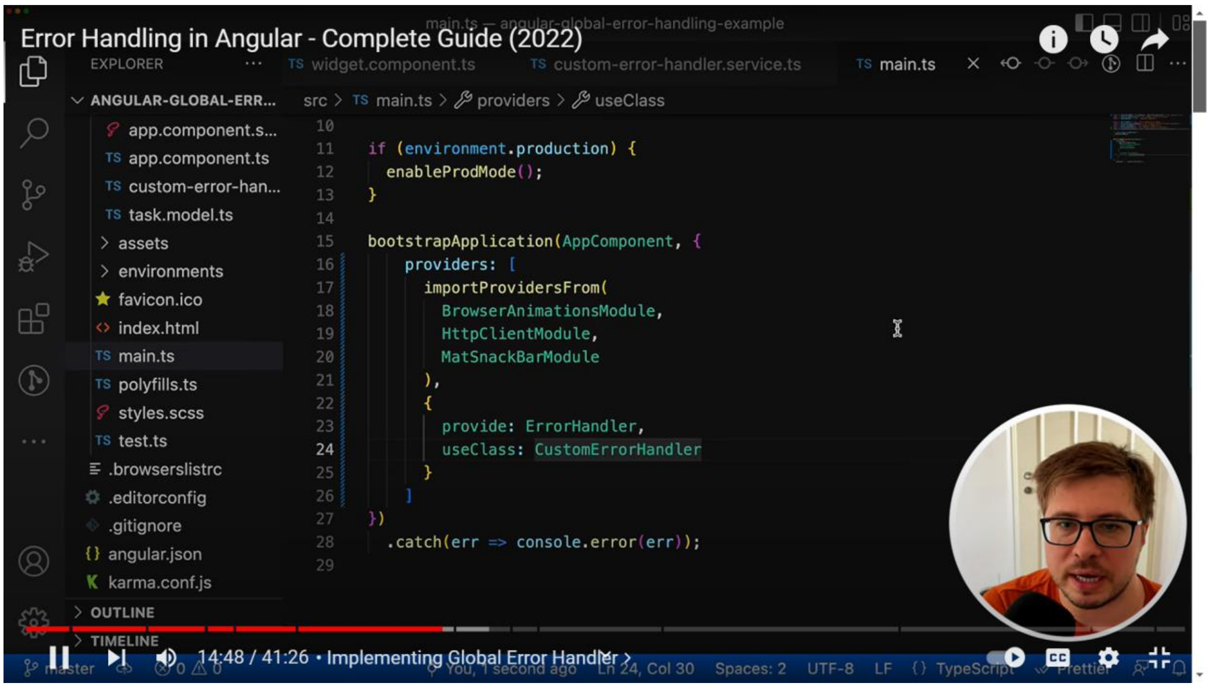
```
import { enableProdMode, importProvidersFrom } from '@angular/core';   323.3k (gz
import { bootstrapApplication } from '@angular/platform-browser';   28.3k (gzippe
import { AppComponent } from './app/app.component';

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



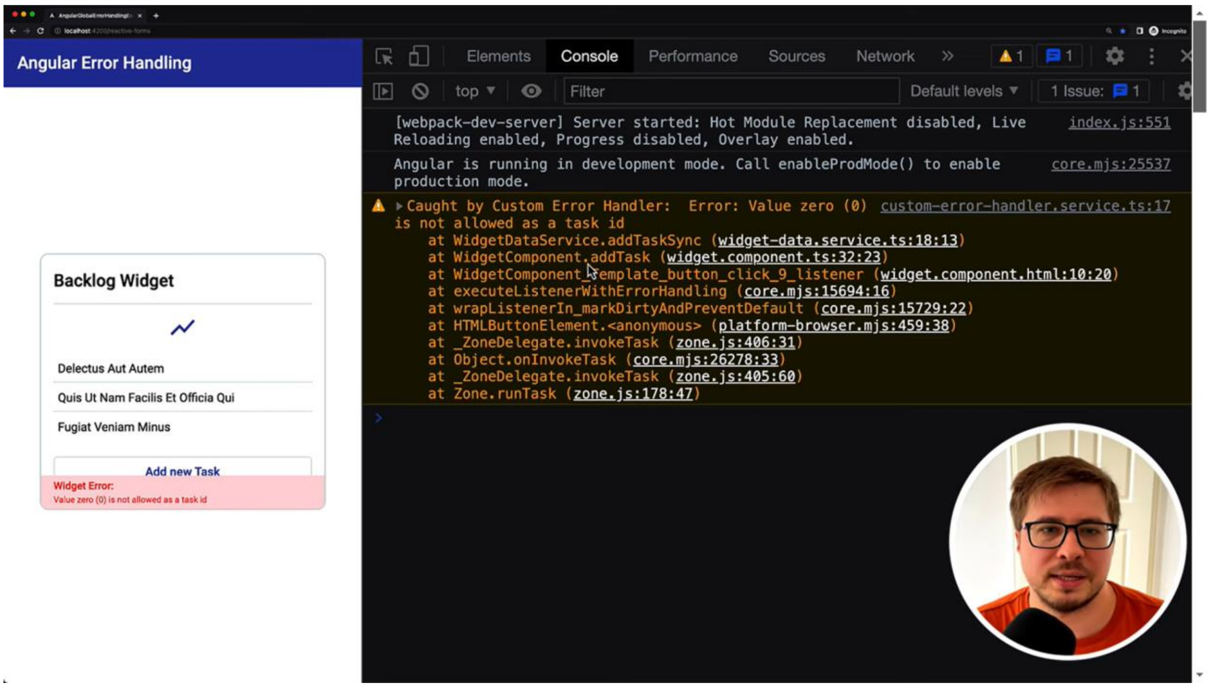
```typescript
import { ErrorHandler, Injectable } from '@angular/core';  323.1k (gzipped: 92.7
import { MatSnackBar } from '@angular/material/snack-bar';  330.8k (gzipped: 64.

@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



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

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

Throw error for capturing global based


Angular Error Handling

Backlog Widget

Delectus Aut Autem

Quis Ut Nam Facilis Et Officia Qui

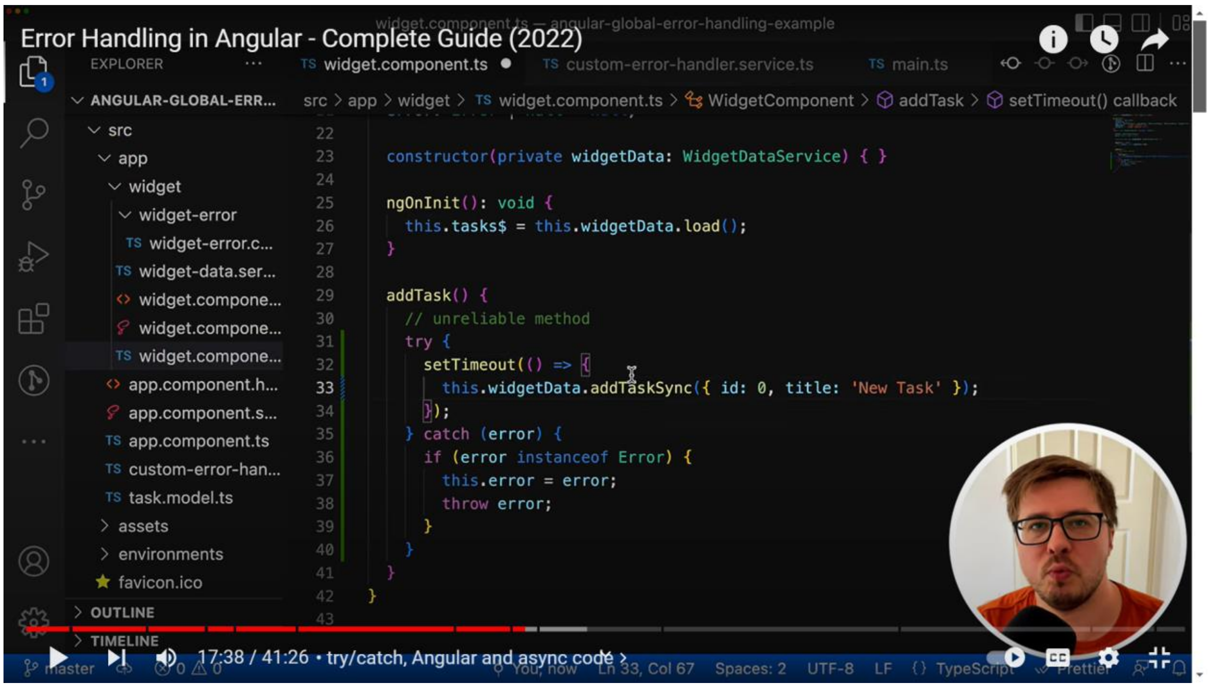
Fugiat Veniam Minus

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

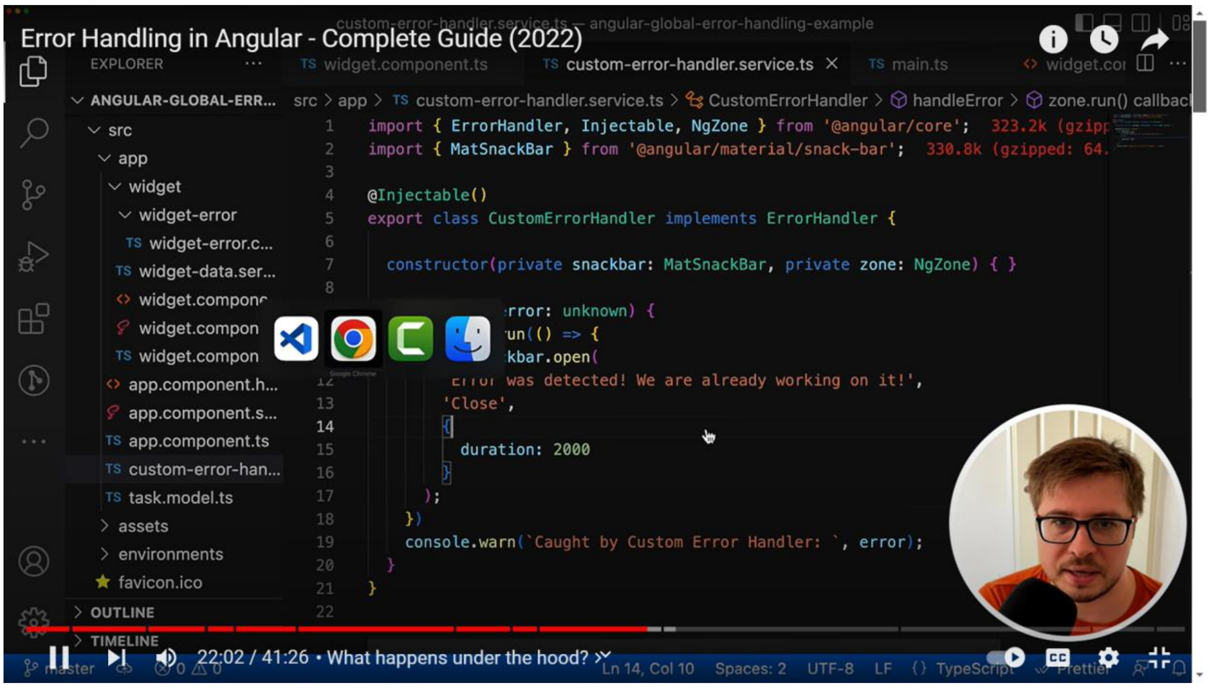

Error Handling in Angular - Complete Guide (2022)

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

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

```
  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 ›


Error Handling in Angular - Complete Guide (2022)
custom-error-handler.service.ts — angular-global-error-handling-example
EXPLORER
widget.component.ts
custom-error-handler.service.ts
main.ts
src > app > custom-error-handler.service.ts > CustomErrorHandler > handleError > zone.run() callbac
22:02 / 41:26 • What happens under the hood?
Ln 14, Col 10 Spaces: 2 UTF-8 LF {} TypeScript Prettier

```
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(
        Error 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
      }
```

Error Handling in Obserables

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