Here's a **concise guide** to understanding and debugging Angular applications, covering error messages, dev tools, and advanced debugging techniques:

**🔍 Understanding Angular Error Messages**

Angular error messages are often detailed but can be confusing. Here's how to read them:

**✅ Example Error:**

ERROR Error: NG0100: ExpressionChangedAfterItHasBeenCheckedError: Expression has changed after it was checked.

**🔑 Breakdown:**

* NG0100 → Angular-specific error code (look it up on Angular Error Guide)
* Description → Often means something changed *after* Angular’s change detection.

**🛠 Common Error Types:**

| **Code** | **Meaning** |
| --- | --- |
| NG0100 | Expression changed after check |
| NG0303 | Can't bind to property (possibly typo or null) |
| NG0900 | Injector error (missing provider) |
| NG2003 | Unexpected null/undefined |
| NG1001 | Template parse error |

**🧠 Using VS Code to Debug Angular Apps**

**✅ Tips for Setup:**

1. Install **Debugger for Chrome** extension.
2. Add a .vscode/launch.json:

{

"version": "0.2.0",

"configurations": [{

"name": "Launch Chrome",

"type": "chrome",

"request": "launch",

"url": "http://localhost:4200",

"webRoot": "${workspaceFolder}"

}]

}

**✅ Use:**

* Set breakpoints in .ts files.
* Use debugger; statements in code.
* Watch variables, inspect call stack.

**🔧 Using Angular DevTools**

**Angular DevTools** is a Chrome/Edge extension for runtime inspection.

**🔥 Features:**

* Inspect component tree
* View @Input/@Output values
* Detect change detection cycles
* Profile performance

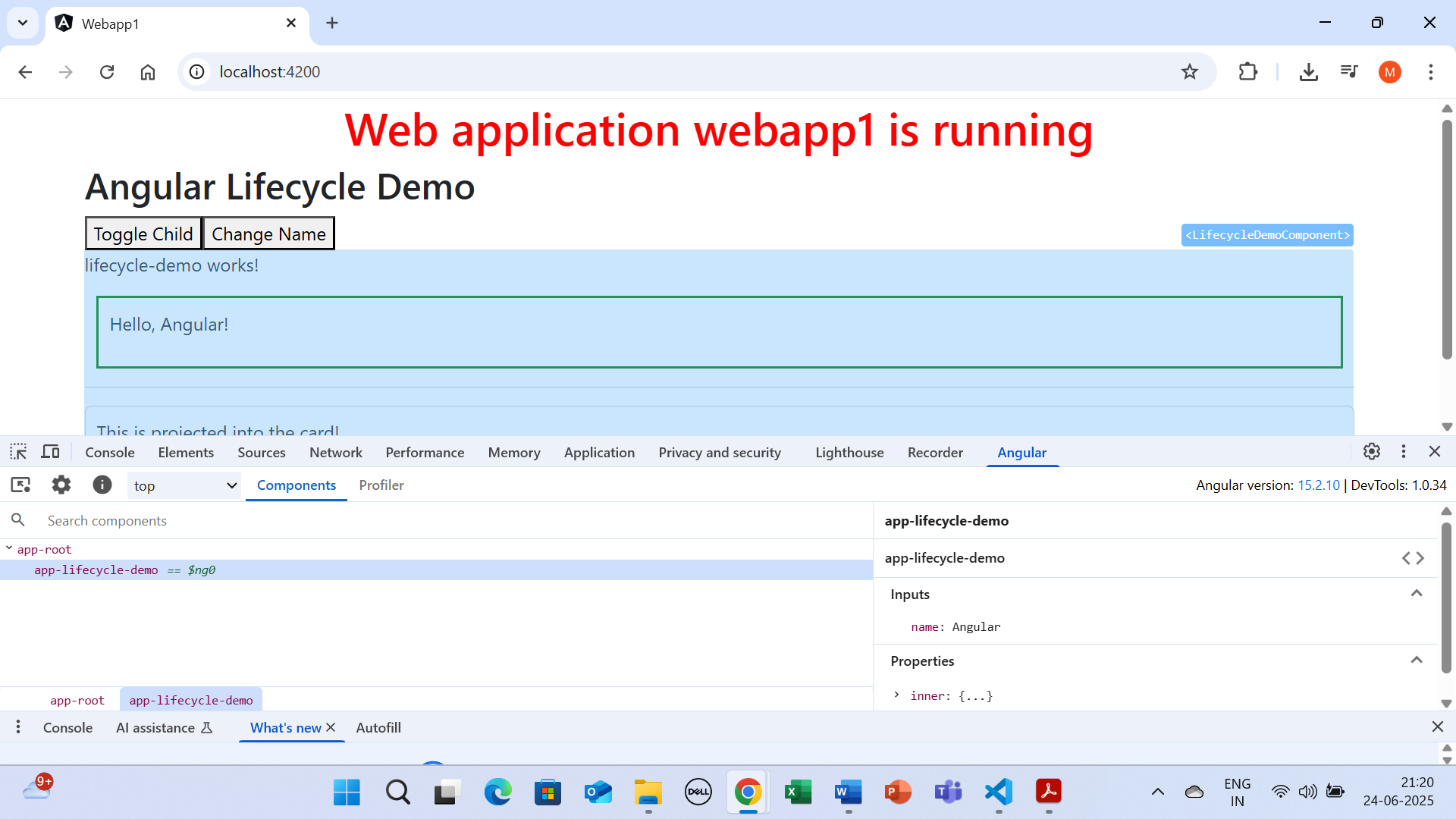
**🧪 Install:**

* From Chrome Web Store

**🚀 How to Use:**

1. Open app in Chrome.
2. Open DevTools → “Angular” tab.
3. Explore component hierarchy and state.

Sample screen



**🧱 Creating a Global Custom ErrorHandler**

Use this to **centralize logging, notifications, and fallback UI**.

**✅ Step 1: Create a Class**

import { ErrorHandler, Injectable } from '@angular/core';

@Injectable()

export class GlobalErrorHandler implements ErrorHandler {

handleError(error: any): void {

console.error('Custom Global Error:', error);

// Optionally log to server, show toast, etc.

}

}

**✅ Step 2: Provide it in AppModule**

import { NgModule, ErrorHandler } from '@angular/core';

import { GlobalErrorHandler } from './global-error-handler';

@NgModule({

providers: [

{ provide: ErrorHandler, useClass: GlobalErrorHandler }

]

})

export class AppModule { }

**🧬 Using the ng Object in Console**

Angular exposes a powerful ng object in DevTools console (in development mode only).

**🧪 Examples:**

ng.getComponent($0) // Get the component instance of selected DOM

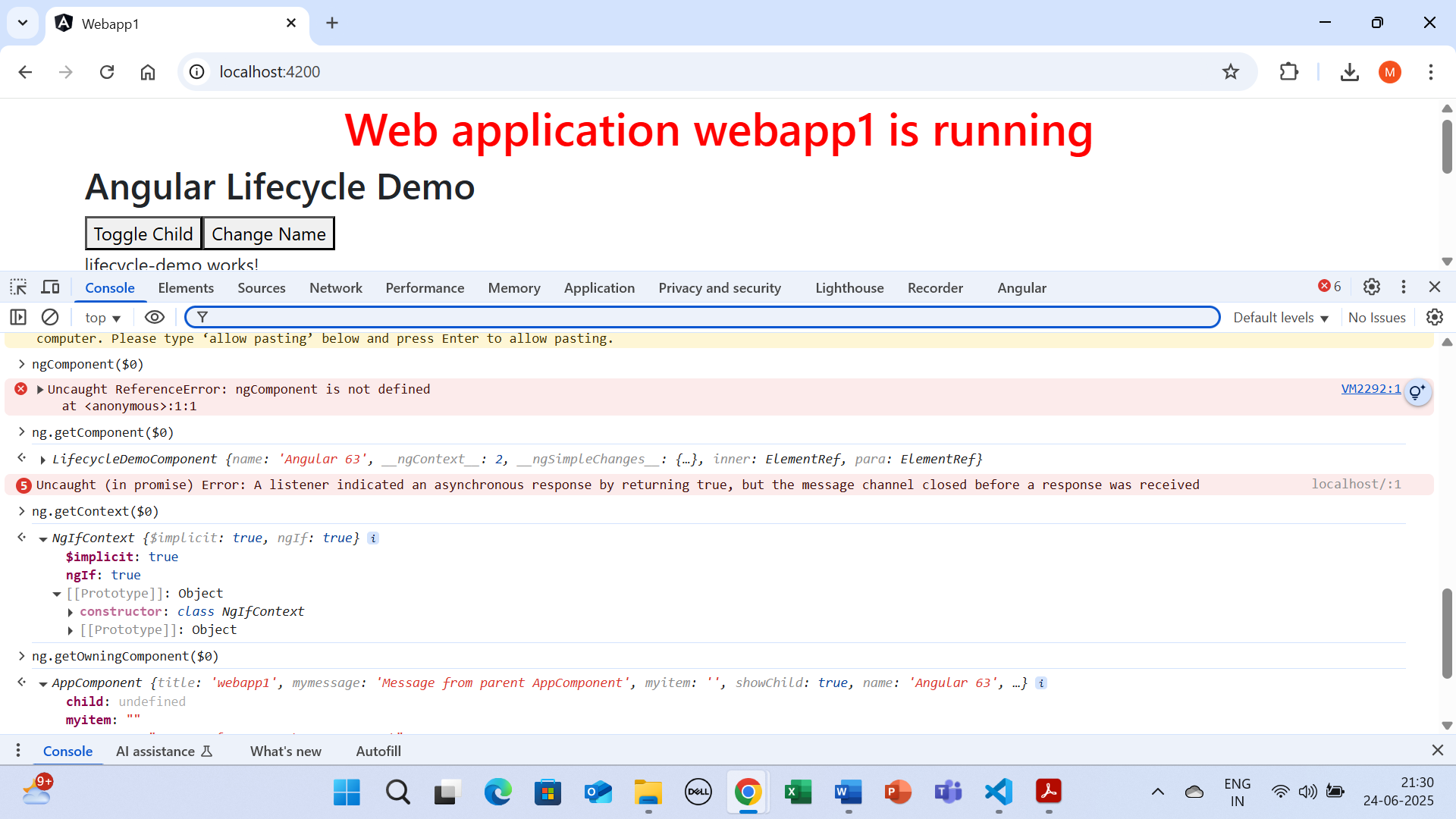
ng.getContext($0) // Get template context

ng.getOwningComponent($0) // Traverse hierarchy

ng.getInjector($0) // Access DI for selected element

$0 refers to the element selected in Elements tab.

Sample screen of use of ng object at console



**🧠 Summary Table**

| **Tool/Concept** | **Use Case** |
| --- | --- |
| **Angular Error Codes** | Diagnose runtime/template issues |
| **VS Code Debugger** | Breakpoints, step-through, variable watching |
| **Angular DevTools** | Runtime state and structure inspection |
| **Global ErrorHandler** | Centralized error logging/reporting |
| **ng Object** | Live inspection of component/DOM in browser |

Let's now simulate **realistic Angular errors** that are more like what you’d encounter in a real app. Here are **3 examples**, and we’ll build one into the demo:

**✅ Realistic Angular Error Example: undefined Property Access**

**💥 Scenario: Accessing a property of undefined (common null pointer)**

causeError() {

const user: any = null;

console.log(user.name); // ❌ Cannot read properties of null

}

**Expected Console Output:**

🔥 Global Error Handler: TypeError: Cannot read properties of null (reading 'name')

**🧪 Updated AppComponent for Realistic Error**

**✅ app.component.ts**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html'

})

export class AppComponent {

title = 'Realistic Error Demo';

causeUndefinedPropertyError() {

const user: any = null;

// ❌ Simulate error: Cannot read property 'name' of null

console.log('User Name:', user.name);

}

causeHttpError() {

// Simulate service call failure (without HttpClient for simplicity)

Promise.reject(new Error('500 Internal Server Error'))

.catch(err => {

// Let Angular catch it as an uncaught Promise rejection

throw err;

});

}

}

**✅ app.component.html**

<h2>Angular Global Error Handler Demo</h2>

<button (click)="causeUndefinedPropertyError()">Trigger Null Property Error</button>

<button (click)="causeHttpError()">Trigger Simulated HTTP Error</button>

**✅ Console Output**

When you click:

* **"Trigger Null Property Error"**:

🔥 Global Error Handler: TypeError: Cannot read properties of null (reading 'name')

* **"Trigger Simulated HTTP Error"**:

🔥 Global Error Handler: Error: 500 Internal Server Error

**✅ Optional: Customize the Handler Further**

Update global-error-handler.ts for smart logging:

handleError(error: any): void {

console.error('🔥 Global Error Handler:', error.message || error);

if (error instanceof TypeError) {

console.warn('⚠️ A TypeError occurred:', error.message);

} else if (error.message?.includes('500')) {

console.warn('🚨 Backend Error:', error.message);

}

// Optional: navigate to fallback UI, report to server, etc.

}

**🧱 Summary**

| **Error Type** | **Simulated By** | **Common In** |
| --- | --- | --- |
| TypeError | Accessing .name on null | Services, Inputs, ViewChild |
| Uncaught Promise Error | Promise.reject(...) without catch | HTTP calls, async tasks |
| Routing/DI Errors | Wrong path/module, missing service | Lazy-loaded apps |