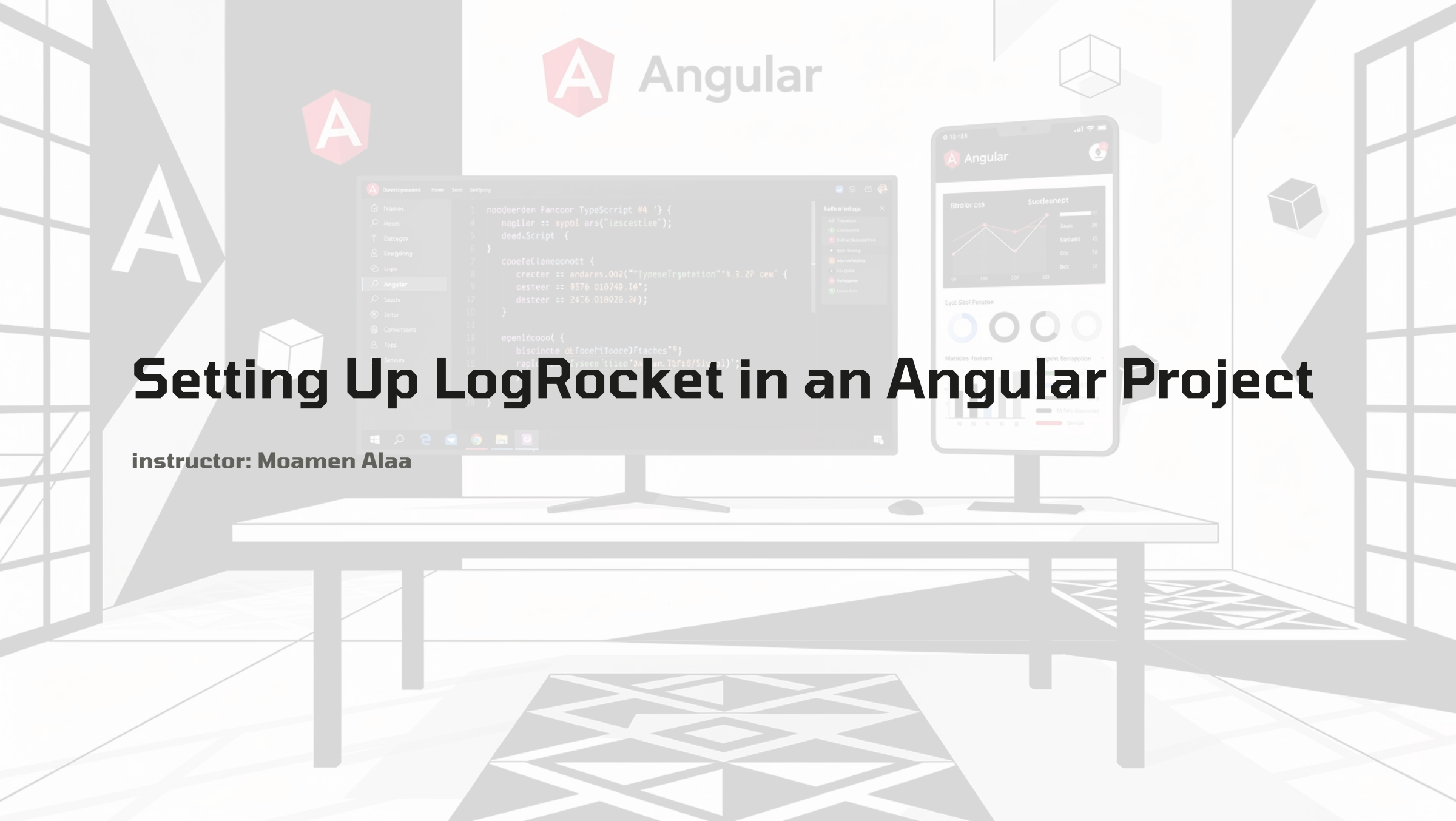


Setting Up LogRocket in an Angular Project

instructor: Moamen Alaa

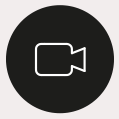
Why LogRocket is Essential for Modern Development

The Problem

Developers face a critical challenge: reproducing bugs reported by users. When a user encounters an error, traditional debugging methods fall short. You might have stack traces and error messages, but without context—what the user clicked, what state the application was in, which API calls failed—troubleshooting becomes guesswork. Users can't always articulate what went wrong, and "it doesn't work" provides little actionable information.

LogRocket solves this by recording everything users do in your application. Every click, navigation event, console log, network request, and Redux action is captured and can be replayed exactly as the user experienced it. This transforms debugging from an art into a science.

Key Benefits

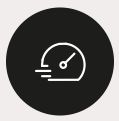


Session Replay

Watch user sessions like a movie to see exactly what happened

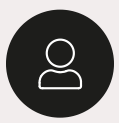
Error Tracking

Automatically capture JavaScript errors with full context and stack traces



Performance Monitoring

Track page load times and identify slow network requests



User Insights

Understand how users interact with your interface to improve UX

LogRocket integrates seamlessly with popular frameworks including Angular, React, and Vue, and connects with tools you already use like Slack, Jira, and GitHub. This means error notifications can flow directly into your team's workflow, enabling faster response times and more efficient debugging processes.

Installation and Configuration

01

Install the LogRocket Package

Use npm to add LogRocket to your Angular project dependencies

02

Initialize LogRocket

Configure LogRocket with your project credentials in the application bootstrap

03

Identify Users

Associate sessions with specific users for targeted analysis

04

Test and Verify

Run your application and confirm sessions appear in the dashboard

Step 1: Install LogRocket

Open your terminal in the root directory of your Angular project and run the following command to install LogRocket as a project dependency:

```
npm install logrocket --save
```

This command downloads the LogRocket package and adds it to your package.json file. The installation typically completes in under a minute, depending on your internet connection. Once installed, LogRocket's TypeScript definitions will be available, providing excellent IDE support and autocomplete functionality.

Step 2: Initialize LogRocket in Your Application

LogRocket must be initialized early in your application's lifecycle to capture all user interactions. The best place to do this is either in `main.ts` (before the application bootstraps) or in `app.component.ts` (in the constructor or ngOnInit). Here's how to set it up:

```
import LogRocket from 'logrocket';

LogRocket.init('your-username/your-project-id');
```

Replace `your-username/your-project-id` with the actual project ID from your LogRocket dashboard. You can find this ID in your project settings under the "Setup" section. The initialization should happen as early as possible—before any user interactions occur—to ensure complete session capture. For production applications, consider storing the project ID in your environment configuration files rather than hardcoding it.

Step 3: Identify Users (Recommended)

User identification is optional but highly recommended for production applications. By identifying users, you can search for specific sessions by name, email, or user ID, making it much easier to investigate issues reported by particular users. Implement user identification after authentication succeeds:

```
LogRocket.identify('THE_USER_ID_IN_YOUR_APP', {
  name: 'James Morrison',
  email: 'jamesmorrison@example.com',
  subscriptionType: 'pro'
});
```

The first parameter is a unique identifier for the user (typically their user ID from your database). The second parameter is an object containing additional user properties. You can include any custom properties relevant to your application, such as subscription tier, role, account type, or organization. These properties become searchable in the LogRocket dashboard, enabling you to filter sessions by user characteristics. Place this code in your authentication service or in a component that runs after successful login.

Running and Verifying Your Integration

Start Your Development Server

With LogRocket installed and configured, start your Angular development server using the Angular CLI:

```
ng serve
```

The application will compile and launch, typically on `http://localhost:4200`. Open your browser and navigate to this URL to begin interacting with your application.

Interact with Your Application

To generate a meaningful session for LogRocket to capture, interact with your application naturally. Click buttons, navigate between pages, fill out forms, trigger API calls, and perform actions that a typical user would. LogRocket captures everything: mouse movements, clicks, scrolls, keyboard input (sanitized for privacy), console logs, network requests and responses, DOM mutations, and Redux actions if applicable.

The more you interact, the richer your test session becomes. Try triggering different features and workflows. If your application has authentication, log in to test the user identification feature. Navigate through several pages to verify routing is captured. Submit forms to see how LogRocket handles user input. Interact for at least 2-3 minutes to create a substantial session that demonstrates LogRocket's capabilities.

Verify in the LogRocket Dashboard

After interacting with your application, open the LogRocket dashboard in a new browser tab. Navigate to the "Sessions" section where you should see your recent session appear. It may take 30-60 seconds for the session to process and appear in the dashboard. Click on the session to watch a replay of your interactions. You'll see a video-like playback showing exactly what you did, along with a timeline of events, network requests, console logs, and any errors that occurred.

1

LogRocket Package Installed

Verified in package.json dependencies

2

Initialization Code Added

LogRocket.init() called in main.ts or app.component.ts

3

User Identification Configured

LogRocket.identify() called after authentication (optional but recommended)

4

Sessions Appear in Dashboard

Interactions are recorded and playable in LogRocket

Troubleshooting and Best Practices

Common Issues and Solutions

Sessions Not Appearing

Verify your project ID is correct in the initialization code. Check the browser console for any LogRocket errors. Ensure you have internet connectivity, as sessions are uploaded to LogRocket's servers in real-time.

Initialization Timing

Make sure LogRocket.init() runs before any user interactions. If placed too late in the application lifecycle, early events won't be captured. Initialize in main.ts or in the root component's constructor.

User Identification Not Working

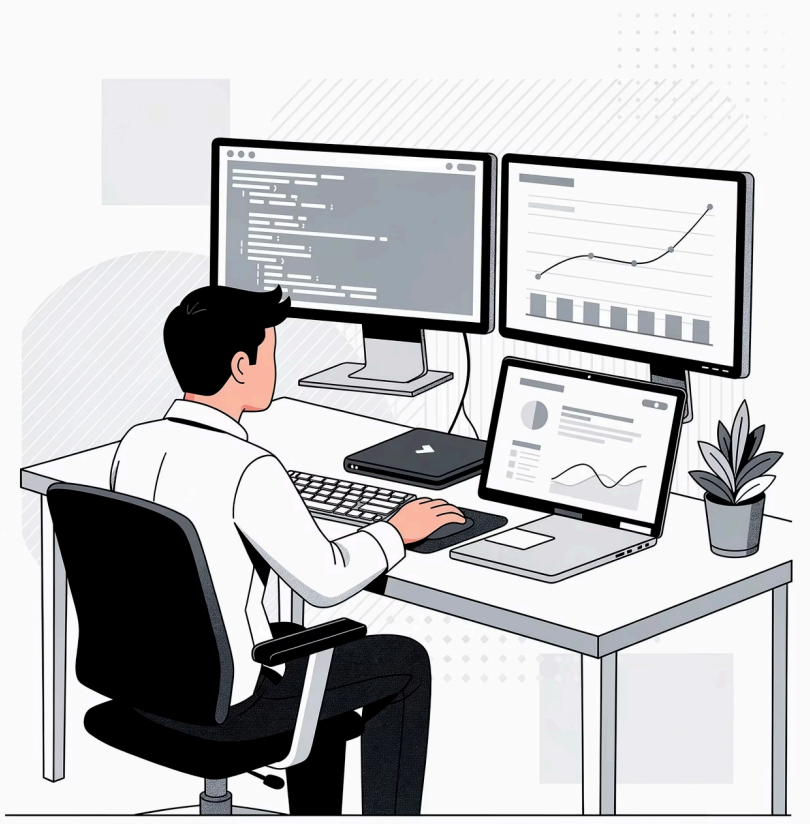
Ensure LogRocket.identify() is called after authentication completes. Check that you're passing valid user data. The user ID (first parameter) must be a string.

Performance Concerns

LogRocket is optimized for minimal performance impact, typically adding less than 100KB to your bundle and consuming minimal CPU. However, if you notice issues, verify you're on the latest version of the package.

Production Best Practices

- **Environment Configuration:** Store your LogRocket project ID in environment files (environment.ts and environment.prod.ts) rather than hardcoding it. This allows different IDs for development and production environments.
- **Privacy Considerations:** LogRocket automatically sanitizes sensitive input fields (passwords, credit cards). Use CSS classes to mark additional fields as sensitive if needed by adding the class "lr-block" or "lr-mask".
- **Error Integration:** Consider integrating LogRocket with your error handling service. You can attach LogRocket session URLs to error reports sent to services like Sentry or Rollbar for enhanced debugging context.
- **Sampling:** For high-traffic applications, implement session sampling to record only a percentage of sessions. This reduces costs while still providing valuable insights.
- **Console Sanitization:** Be mindful of what you log to the console in production, as LogRocket captures all console output. Avoid logging sensitive data like tokens or personal information.



Next Steps and Advanced Features

Now that you have LogRocket successfully integrated, explore advanced features to maximize its value. Set up custom events to track specific user actions important to your business. Configure alerts to notify your team when critical errors occur or when user behavior matches certain patterns. Integrate LogRocket with your existing tools—connect it to Slack for instant error notifications, link it to Jira to automatically create tickets with session replays, or integrate with GitHub to associate errors with specific commits. These integrations transform LogRocket from a debugging tool into a comprehensive user experience monitoring platform that helps your team build better software faster.