

ANGULAR HEALTH CHECK

Sample: Angular RealWorld app

ang2-conduit · Angular 12.2.17



Codebase

[github.com/gothinkster/angular-
realworld-example-app](https://github.com/gothinkster/angular-realworld-example-app) @
57eeb6a

Date

2026-10-23

Prepared by

Stefan Kuburovic,
stefankuburovic.github.io

SUMMARY

Angular 12.2.17, 10 majors behind 22. Of 11 graded areas, 6 are poor, 1 fair and 4 good, for an overall score of 53 out of 100. Reaching Angular 22 takes 10 `ng update` steps, with 7 pieces of work along the way, listed below. The most urgent: unit tests run on Karma, which is deprecated; no unit tests found; end-to-end tests use Protractor, which is end of life.

Score 53/100 average of the graded areas	Angular 12.2.17 latest is 22	Upgrade steps 10 one major at a time	Known vulnerabilities 23 in direct dependencies, from osv.dev
Next: Angular 13 0 removed imports the next major no longer exports	To act on 16 7 high priority	Code 1 589 lines in 71 files	

WHAT TO ACT ON

Priority	Finding	Area	Places
High	Angular 12 is 10 majors behind 22 and out of support	Angular version & tooling	-
High	Unit tests run on Karma, which is deprecated	Testing	-
High	No unit tests found	Testing	-
High	End-to-end tests use Protractor, which is end of life	Testing	-
High	<code>strictNullChecks</code> is off	Type safety	-
High	<code>strictTemplates</code> is off	Type safety	-
High	23 known vulnerabilities in 4 dependencies (15 high or critical)	Security	-
Medium	1 project still built with the webpack builder	Angular version & tooling	-

Priority	Finding	Area	Places
Medium	<code>protractor</code> is end of life	<u>Angular version & tooling</u>	-
Medium	<code>tslint</code> is end of life	<u>Angular version & tooling</u>	-
Medium	<code>codelyzer</code> is end of life	<u>Angular version & tooling</u>	-
Medium	18 components use eager change detection (100%)	<u>Change detection & zoneless readiness</u>	18
Medium	18 components still declared in NgModules (16 NgModules in total)	<u>Components & templates</u>	18
Medium	9 <code>@Input()</code> decorators and 3 <code>@Output()</code> decorators left	<u>Components & templates</u>	12
Medium	<code>noImplicitAny</code> is off	<u>Type safety</u>	-
Medium	13 explicit <code>any</code> in application code (8.2 per 1 000 lines)	<u>Type safety</u>	7

ROADMAP: ANGULAR 12 → 22

Step	What it involves	Toolchain
First	Move from TSLint to ESLint	
Angular 13	The update itself	TypeScript >=4.4.2 <4.7
Angular 14	The update itself	no change
Angular 15	The update itself	no change
Angular 16	Convert NgModule components to standalone (18)	zone.js ~0.13.0
Angular 17	Move templates to built-in control flow (13)	no change
Angular 18	Switch to the esbuild application builder (1)	no change
Angular 19	Convert @Input() and @Output() to signal inputs and outputs (12)	no change

Step	What it involves	Toolchain
Angular 20	The update itself	no change
Angular 21	Move unit tests from Karma to Vitest Prepare for zoneless: move components to OnPush, then drop zone.js (18)	no change
Angular 22	The update itself	no change

Counts are places in the code each change touches. Commands and details per step are in the appendix.

Summary

The codebase is small and clean: no import cycles, no third-party package tied to Angular, most routes already lazy. What holds it back is age. Angular 12 has been out of support since 2022, and 23 published advisories affect its direct dependencies. None of that needs a rewrite. The upgrade is ten `ng update` steps of routine work, and the modern patterns (standalone components, signals, built-in control flow, zoneless) can land along the way.

Recommended order

1. **Before the first step:** switch on `strictNullChecks` folder by folder and add specs for the services. Today there are none, and every later step is safer with them.
2. **Angular 13 to 15:** the plain upgrade, in three reviewed merge requests. Replace Karma with Jest or Vitest when the toolchain allows it.
3. **Angular 16 and 17:** standalone components and built-in control flow, both with Angular's own migrations.
4. **Angular 18 to 22:** the esbuild builder, signal inputs and outputs, then OnPush and zoneless change detection.

Each step ships on its own, behind CI, while the team keeps building features.

Next steps

This is a sample report on a public open-source app, so it carries no estimate. For your codebase, the Health Check is free: the report, a manual review of what a tool cannot see, and this plan with an effort estimate per step, walked through with your team.

Appendix: area by area

Poor

Angular version & tooling

Angular 12.2.17, 10 majors behind 22.

Good

Deprecated APIs

Nothing deprecated as of Angular 12 is in use; APIs deprecated in later versions surface at each upgrade step.

Good

Third-party compatibility

No third-party package depends on Angular.

Poor

Change detection & zoneless readiness

Runs on zone.js; 0% of 18 components use OnPush.

Poor

Components & templates

18 components: 0% standalone, 0% signal inputs, 0% built-in control flow.

Fair

State management

No classic NgRx, 0 signal()/computed() calls, 0 signal-store files.

Good

Architecture

79% of 14 routes lazy, 0 files over 500 lines, 28 classes with constructor injection.

Poor

Testing

Karma, 0 spec files, 0% of components and services with a spec.

Poor

Type safety

strictNullChecks off, strictTemplates off, 13 explicit any.

Poor

Security

No sanitizer bypasses, direct DOM HTML writes or eval in 69 files; 1 sanitized [innerHTML] binding; 23 known vulnerabilities in dependencies.

Good

Circular dependencies

No import cycles between application files.

Info

Bundle size

No build stats found, so the bundle size is not part of this report.

UPGRADE PLAN: ANGULAR 12 → 22

10 steps, one major at a time, as `ng update` requires. Requirements come from each version's published package metadata.

First Before you start

- **Move from TSLint to ESLint**

```
ng add @angular-eslint/schematics
```

TSLint is deprecated and later versions of the Angular CLI no longer support it.

→ 13 Angular 13.4.0

```
ng update @angular/core@13 @angular/cli@13
```

Node.js ^12.20.0 || ^14.15.0 || >=16.10.0

TypeScript >=4.4.2 <4.7 (now 4.3.5)

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.11.4

→ 14 Angular 14.3.0

```
ng update @angular/core@14 @angular/cli@14
```

Node.js ^14.15.0 || >=16.10.0

TypeScript >=4.6.2 <4.9

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.11.4 || ~0.12.0

→ 15 Angular 15.2.10

```
ng update @angular/core@15 @angular/cli@15
```

Node.js ^14.20.0 || ^16.13.0 || >=18.10.0

TypeScript >=4.8.2 <5.0

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.11.4 || ~0.12.0 || ~0.13.0

→ 16 Angular 16.2.12

```
ng update @angular/core@16 @angular/cli@16
```

Node.js ^16.14.0 || >=18.10.0

TypeScript >=4.9.3 <5.2

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.13.0 (now 0.11.8)

- **Convert NgModule components to standalone** 18

```
ng generate @angular/core:standalone
```

Three passes: convert declarations, remove unneeded NgModules, switch to standalone bootstrap.

→ 17 **Angular 17.3.12**

```
ng update @angular/core@17 @angular/cli@17
```

Node.js ^18.13.0 || >=20.9.0

TypeScript >=5.2 <5.5

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.14.0

- **Move templates to built-in control flow** 13

```
ng generate @angular/core:control-flow
```

→ 18 **Angular 18.2.14**

```
ng update @angular/core@18 @angular/cli@18
```

Node.js ^18.19.1 || ^20.11.1 || >=22.0.0

TypeScript >=5.4 <5.6

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.14.10

- **Switch to the esbuild application builder** 1

```
ng update @angular/cli --name use-application-builder
```

→ 19 **Angular 19.2.25**

```
ng update @angular/core@19 @angular/cli@19
```

Node.js ^18.19.1 || ^20.11.1 || >=22.0.0

TypeScript >=5.5 <5.9

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.15.0

- **Convert @Input() and @Output() to signal inputs and outputs** 12

```
ng generate @angular/core:signal-input-migration && ng generate @angular/core:output-migration
```

→ 20 **Angular 20.3.34**

```
ng update @angular/core@20 @angular/cli@20
```

Node.js ^20.19.0 || ^22.12.0 || >=24.0.0

TypeScript >=5.8 <6.0

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.15.0

→ 21 Angular 21.2.25

```
ng update @angular/core@21 @angular/cli@21
```

Node.js ^20.19.0 || ^22.12.0 || >=24.0.0

TypeScript >=5.9 <6.1

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.15.0 || ~0.16.0

- **Move unit tests from Karma to Vitest**

Vitest is the Angular CLI default from 21 (@angular/build:unit-test). Jasmine-specific calls (spyOn, jasmine.*) need rewriting.

- **Prepare for zoneless: move components to OnPush, then drop zone.js 18**

Upgraded apps keep zone.js through provideZoneChangeDetection() until it is removed.

→ 22 Angular 22.2.3

```
ng update @angular/core@22 @angular/cli@22
```

Node.js ^22.22.3 || ^24.15.0 || >=26.0.0

TypeScript >=6.0 <6.1

RxJS ^6.5.3 || ^7.4.0

zone.js ~0.15.0 || ~0.16.0

Angular version & tooling

Poor

Angular 12.2.17, 10 majors behind 22.

Installed

12.2.17

package-lock.json

Latest

22

npm registry

Majors behind

10

Build system

webpack

High Angular 12 is 10 majors behind 22 and out of support

Angular supports each major for 18 months, so only the newest three get fixes; 12 receives no security patches. `ng update` moves one major at a time, which makes this 10 sequential upgrade steps, each with its own migrations to review.

Medium 1 project still built with the webpack builder

The esbuild-based `application` builder has been the default since Angular 17 and builds several times faster. `ng update @angular/cli --name use-application-builder` migrates the configuration. Projects: `ang2-conduit`.

Medium **protractor is end of life**

End of life since 2023. Replaced by Playwright or Cypress.

Medium **tslint is end of life**

Deprecated since 2019. Replaced by ESLint with `angular-eslint`.

Medium **codeLyzer is end of life**

A set of TSLint rules, so it died with TSLint. Replaced by `angular-eslint`.

Deprecated APIs Good

Nothing deprecated as of Angular 12 is in use; APIs deprecated in later versions surface at each upgrade step.

Deprecated symbols

0

Files affected

0

Legacy template directives

13

not deprecated in this version

Angular 13

0 removed, 0 deprecated

imports, compared with 13.4.0

Third-party compatibility Good

No third-party package depends on Angular.

Depend on Angular

0

third-party packages

Block the upgrade

0

to Angular 22

Peer mismatches

0

with Angular 12

Deprecated on npm

0

Change detection & zoneless readiness Poor

Runs on zone.js; 0% of 18 components use OnPush.

Mode zone.js	OnPush 0% 0 of 18 components	Files injecting NgZone 0	Zone test helpers 0
------------------------	---	------------------------------------	-------------------------------

Note **Zoneless change detection needs Angular 20.2 or newer**

Upgrade first; OnPush coverage is the preparation that pays off now.

Medium **18 components use eager change detection (100%)**

Without `changeDetection: ChangeDetectionStrategy.OnPush` a component is checked on every change detection cycle and usually relies on zone.js to notice mutations. These are the components to review before going zoneless.

```
src/app/app.component.ts:5
src/app/article/article-comment.component.ts:6
src/app/article/article.component.ts:14
src/app/auth/auth.component.ts:7
src/app/editor/editor.component.ts:7
and 13 more
```

Components & templates **Poor**

18 components: 0% standalone, 0% signal inputs, 0% built-in control flow.

Components 18	Standalone 0% 16 NgModules	Signal inputs 0% 0 of 9	Signal outputs 0% 0 of 3
Built-in control flow 0% 0 of 13 blocks	@defer blocks 0		

Medium 18 components still declared in NgModules (16 NgModules in total)

Standalone components are the default since Angular 19 and what every new API assumes. `ng generate @angular/core:standalone` converts declarations, removes the emptied NgModules and switches bootstrapping, in three passes that can be reviewed one at a time.

```
src/app/app.component.ts:5
src/app/article/article-comment.component.ts:6
src/app/article/article.component.ts:14
src/app/auth/auth.component.ts:7
src/app/editor/editor.component.ts:7
and 13 more
```

Medium 9 `@Input()` decorators and 3 `@Output()` decorators left

Signal inputs (`input()`) make inputs readable in `computed()` and effects without `ngOnChanges` , and are what zoneless change detection tracks. `ng generate @angular/core:signal-input-migration` and `ng generate @angular/core:output-migration` convert most of them; the rest are usually inputs written to from inside the component.

Low 13 uses of `*ngIf` , `*ngFor` and `[ngSwitch]`

Built-in control flow (`@if` , `@for` , `@switch`) is available from Angular 17. `ng generate @angular/core:control-flow` migrates most templates automatically.

State management Fair

No classic NgRx, 0 `signal()/computed()` calls, 0 signal-store files.

Classic NgRx files

0

Signal store files

0

`signal()` / `computed()`

0

0 to `Signal()`

BehaviorSubject services

1

Low 1 service holds state in a **BehaviorSubject**

Each is a candidate for a `signal()`-based service: templates read it without the async pipe, `computed()` replaces `combineLatest` chains, and zoneless change detection tracks it automatically. `toObservable()` keeps existing RxJS consumers working during the move.

```
src/app/core/services/user.service.ts:12
```

Architecture **Good**

79% of 14 routes lazy, 0 files over 500 lines, 28 classes with constructor injection.

Lazy routes

79%

11 of 14

Large files

0

over 500 lines

Constructor injection

28

classes; 0 inject() calls

Class guards

7

and interceptors

Hotspots

0

from 0 commits in the
last year

Low 7 class-based guards and interceptors

Resolve, CanActivate, HttpInterceptor. Functional guards (`CanActivateFn` , `ResolveFn`) and interceptors (`HttpInterceptorFn` with `withInterceptors`) are the current style: a function with `inject()` inside, no class, no provider. Class interceptors also need `withInterceptorsFromDi()` to keep working with `provideHttpClient` .

```
src/app/article/article-resolver.service.ts:8
src/app/auth/no-auth-guard.service.ts:8
src/app/core/interceptors/http.token.interceptor.ts:7
src/app/core/services/auth-guard.service.ts:8
src/app/editor/editable-article-resolver.service.ts:8
and 2 more
```

Low 28 classes inject through the constructor

`inject()` is the current default (0 calls already in the code). It works in field initialisers and functions, and it is what signal-based code expects. `ng generate @angular/core:inject` migrates constructors automatically.

```
src/app/app.component.ts:10
src/app/article/article-comment.component.ts:11
src/app/article/article-resolver.service.ts:10
src/app/article/article.component.ts:28
src/app/auth/auth.component.ts:18
and 23 more
```

Testing

Poor

Karma, 0 spec files, 0% of components and services with a spec.

Unit-test runner

Karma

Spec files

0

With a spec

0%

0 of 32
component/service files

End-to-end

Protractor

High Unit tests run on Karma, which is deprecated

Karma was deprecated in 2023 and receives no further development. Vitest is the Angular CLI's default runner since v21 (`@angular/build:unit-test`). With no specs written yet, moving costs only the configuration, and the first specs can be written for Vitest directly.

High No unit tests found

32 component and service files without a single spec. Every refactoring and upgrade step is verified by hand.

High End-to-end tests use Protractor, which is end of life

Protractor reached end of life in 2023 and gets no fixes, including for new browser versions. Playwright and Cypress are the usual replacements.

Type safety Poor

strictNullChecks off, strictTemplates off, 13 explicit any.

strictNullChecks

off

default in TypeScript 4

strictTemplates

off

default in Angular 12

noImplicitAny

off

default in TypeScript 4

Explicit any

13

8.2 per 1 000 lines

Non-null assertions

0

High **strictNullChecks is off**

Every value that may be `null` or `undefined` is typed as if it never is, so the most common runtime error in an Angular app is invisible to the compiler. In a large codebase switching it on is staged: a separate tsconfig that checks one folder at a time, with a count that may only go down.

High **strictTemplates is off**

Template bindings are only loosely type-checked: a wrong input type, a misspelled output payload or a possibly-undefined value passed to a component compiles and fails at runtime. Turning it on typically surfaces a batch of real bugs alongside the type noise.

Medium **noImplicitAny is off**

An unannotated parameter silently becomes `any`, and no lint rule that counts the word `any` can see it.

Medium **13 explicit any in application code (8.2 per 1 000 lines)**

Files with the most are listed first. Each `any` switches type checking off for everything that flows through it.

```
src/app/core/services/api.service.ts:14
src/app/core/interceptors/http.token.interceptor.ts:11
src/test.ts:16
src/app/article/article-resolver.service.ts:18
src/app/editor/editable-article-resolver.service.ts:19
and 2 more
```

Low 1 @ts-ignore / @ts-expect-error comment

Each switches off checking for the next line, including errors nobody has seen yet.

Security Poor

No sanitizer bypasses, direct DOM HTML writes or eval in 69 files; 1 sanitized [innerHTML] binding; 23 known vulnerabilities in dependencies.

Direct DOM HTML writes

0

innerHTML =,
insertAdjacentHTML

Sanitizer bypasses

0

bypassSecurityTrust*

eval / new Function

0

[innerHTML] bindings

1

sanitized by Angular

Known vulnerabilities

23

in 35 direct
dependencies checked

High 23 known vulnerabilities in 4 dependencies (15 high or critical)

@angular/common 12.2.17: 8 advisories (CVE-2026-54266, CVE-2026-54268, CVE-2025-66035, ...), all fixed by 20.3.34; @angular/compiler 12.2.17: 6 advisories (CVE-2026-54265, CVE-2026-50557, CVE-2026-88057, ...), all fixed by 20.3.28; @angular/core 12.2.17: 7 advisories (CVE-2026-52725, CVE-2026-50557, CVE-2026-88057, ...), all fixed by 20.3.28; @angular/router 12.2.17: 2 advisories (CVE-2026-111752, CVE-2026-101896), all fixed by 20.3.33. Direct dependencies at their exact versions, from the advisories in osv.dev; the full dependency tree is npm audit 's job.

Low 1 [innerHTML] binding in 1 template

Angular sanitizes these, so they are safe on their own: scripts and event handlers are stripped. They become a hole only when the bound value went through bypassSecurityTrustHtml, which is worth checking for each one.

```
src/app/article/article.component.html
```

Circular dependencies

Good

No import cycles between application files.

Tangles

0

groups of files in a loop

Files involved

0

Largest tangle

0

files

Bundle size

Info

No build stats found, so the bundle size is not part of this report.

Note

Run `ng build --stats-json` (or `nx build <app> --stats-json`) and re-run to include the bundle size

Score: Average of the graded areas: good counts 100, fair 60, poor 20. Areas marked info are not counted. Measured by static analysis of the source, without building or running it and without the code leaving the machine. Deprecations come from the Angular packages' own `@deprecated` tags for the versions in use.

Generated with ng-checkup 1.1.0 on 2026-10-23.