Angular Roadmap

Last updated: 2022-04-01

Angular receives a large number of feature requests, both from inside Google and from the broader open-source community. At the same time, our list of projects contains plenty of maintenance tasks, code refactorings, and potential performance improvements. We bring together representatives from developer relations, product management, and engineering to prioritize this list. As new projects come into the queue, we regularly position them based on relative priority to other projects. As work gets done, projects move up in the queue.

The following projects are not associated with a particular Angular version. We'll release them on completion, and they will be part of a specific version based on our release schedule, following semantic versioning. For example, features are released in the next minor after they are complete, or the next major if they include breaking changes.

In progress

Rollout strict typings for `@angular/forms`

In Q4 2021 we designed a solution for introducing strict typings for forms and in Q1 2022 we concluded the corresponding request for comments. Currently, we're implementing a rollout strategy with an automated migration step that will enable the improvements for existing projects. We're first testing the solution with 2,500+ projects at Google to ensure a smooth migration path for the external community.

Implement APIs for optional NgModules

In the process of making Angular simpler, we're working on introducing APIs that allow developers to bootstrap applications, instantiate components, and use the router without NgModules. We'll share an RFC to start a design discussion with the community, followed by a series of pull request with the implementation. In the first part of this project we finalized the design of standalone components, directives, and pipes (RFC).

Improve image performance

The Aurora and the Angular teams have started a collaboration to design and implement a new directive that would help Angular users to leverage various image optimization techniques. We are in the process of initial research and design.

Investigate micro frontend architecture for scalable development processes

We conducted a series of 40 interviews to understand the requirements for micro-frontend architecture of the community. We'll follow up with a broader community survey and share analysis of the results publicly.

Investigate modern bundles

To improve development experience by speeding up build times research modern bundles. As part of the project experiment with esbuild and other open source solutions, compare them with the state of the art tooling in Angular CLI, and report the findings.

New CDK primitives

We're working on new CDK primitives to facilitate creating custom components based on the WAI-ARIA design patterns for Menu & Menubar, Listbox, Combobox, and Dialog.

Enhanced Angular Material components by integrating MDC Web

MDC Web is a library created by Google's Material Design team that provides reusable primitives for building Material Design components. The Angular team is incorporating these primitives into Angular Material. Using MDC Web aligns Angular Material more closely with the Material Design specification, expand accessibility, improve component quality, and improve our team's velocity.

Angular component accessibility

We're evaluating components in Angular Material against accessibility standards such as WCAG and working to fix any issues that arise from this process.

Documentation refactoring

Ensure all existing documentation fits into a consistent set of content types. Update excessive use of tutorial-style documentation into independent topics. We want to ensure the content outside the main tutorials is self-sufficient without being tightly coupled to a series of guides.

Future

Explore hydration and server-side rendering usability improvements

As part of this effort we'll explore the problem space of hydration with server-side rendering, different approaches, and opportunities for Angular. As outcome of this project we'll have validation of the effort as well as a plan for action.

Revamp performance dashboards to detect regressions

We have a set of benchmarks that we run against every code change to ensure Angular aligns with our performance standards. To ensure the framework’s runtime does not regress after a code change, we need to refine some of the existing infrastructure the dashboards step on.

Leverage full framework capabilities with Zone.js opt-out

We are going to design and implement a plan to make Zone.js optional from Angular applications. This way, we simplify the framework, improve debugging, and reduce application bundle size. Additionally, this lets us take advantage of built-in async/await syntax, which currently Zone.js does not support.

Improved build performance with ngc as a tsc plugin distribution

Distributing the Angular compiler as a plugin of the TypeScript compiler will substantially improve developers' build performance and reduce maintenance costs.

Support adding directives to host elements

A long-standing feature request is to add the ability to add directives to host elements. The feature lets developers augment their own components with additional behaviors without using inheritance. The project requires substantial effort in terms of the definition of APIs, semantics, and implementation.

Ergonomic component level code-splitting APIs

A common problem with web applications is their slow initial load time. A way to improve it is to apply more granular code-splitting on a component level. To encourage this practice, we’ll be working on more ergonomic code-splitting APIs.

Publish guides on advanced concepts

Develop and publish an in-depth guide on change detection. Develop content for performance profiling of Angular applications. Cover how change detection interacts with Zone.js and explain when it gets triggered, how to profile its duration, as well as common practices for performance optimization.

Ensure smooth adoption for future RxJS changes (version 8 and beyond)

We want to ensure Angular developers are taking advantage of the latest capabilities of RxJS and have a smooth transition to the next major releases of the framework. For this purpose, we will explore and document the scope of the changes in v7 and beyond RxJS, and plan an update strategy.

▼

Completed

Show all Hide all *expand_more*

Remove legacy View Engine

Completed Q1 2022

After the transition of all our internal tooling to Ivy is completed, we will remove the legacy View Engine for reduced Angular conceptual overhead, smaller package size, lower maintenance cost, and lower codebase complexity.

Simplified Angular mental model with optional NgModules

Completed Q1 2022

To simplify the Angular mental model and learning journey, we'll be working on making NgModules optional. This work lets developers develop standalone components and implement an alternative API for declaring the component's compilation scope. We kicked this project off with high-level design discussions that we captured in an RFC.

Design strict typing for `@angular/forms`

Completed Q1 2022

We will work on finding a way to implement stricter type checking for reactive forms with minimal backward incompatible implications. This way, we let developers catch more issues during development time, enable better text editor and IDE support, and improve the type checking for reactive forms.

Improve Angular DevTools' integration with framework

Completed Q1 2022

To improve the integration of Angular DevTools with the framework, we are working on moving the codebase to the angular/angular monorepository. This includes transitioning Angular DevTools to Bazel and integrating it into the existing processes and CI pipeline.

Launch advanced compiler diagnostics

Completed Q1 2022

Extend the diagnostics of the Angular compiler outside type checking. Introduce other correctness and conformance checks to further guarantee correctness and best practices.

Update our e2e testing strategy

Completed Q3 2021

To ensure we provide a future-proof e2e testing strategy, we want to evaluate the state of Protractor, community innovations, e2e best practices, and explore novel opportunities. As first steps of the effort, we shared an RFC and worked with partners to ensure smooth integration between the Angular CLI and state of the art tooling for e2e testing. As the next step, we need to finalize the recommendations and compile a list of resources for the transition.

Angular libraries use Ivy

Completed Q3 2021

Earlier in 2020, we shared an RFC for Ivy library distribution. After invaluable feedback from the community, we developed a design of the project. We are now investing in the development of Ivy library distribution, including an update of the library package format to use Ivy compilation, unblock the deprecation of the View Engine library format, and ngcc.

Improve test times and debugging with automatic test environment tear down

Completed Q3 2021

To improve test time and create better isolation across tests, we want to change `TestBed` to automatically clean up and tear down the test environment after each test run.

Deprecate and remove IE11 support

Completed Q3 2021

Internet Explorer 11 (IE11) has been preventing Angular from taking advantage of some of the modern features of the Web platform. As part of this project we are going to deprecate and remove IE11 support to open the path for modern features that evergreen browsers provide. We ran an RFC to collect feedback from the community and decide on next steps to move forward.

Leverage ES2017+ as the default output language

Completed Q3 2021

Supporting modern browsers lets us take advantage of the more compact, expressive, and performant new syntax of JavaScript. As part of this project we'll investigate what the blockers are to moving forward with this effort, and take the steps to enable it.

Accelerated debugging and performance profiling with Angular DevTools

Completed Q2 2021

We are working on development tooling for Angular that provides utilities for debugging and performance profiling. This project aims to help developers understand the component structure and the change detection in an Angular application.

Streamline releases with consolidated Angular versioning & branching

Completed Q2 2021

We want to consolidate release management tooling between Angular's multiple GitHub repositories (angular/angular, angular/angular-cli, and angular/components). This effort lets us reuse infrastructure, unify and simplify processes, and improve our release process's reliability.

Higher developer consistency with commit message standardization

Completed Q2 2021

We want to unify commit message requirements and conformance across Angular repositories (angular/angular, angular/components, angular/angular-cli) to bring consistency to our development process and reuse infrastructure tooling.

Transition the Angular language service to Ivy

Completed Q2 2021

The goal of this project is to improve the experience and remove legacy dependency by transitioning the language service to Ivy. Today the language service still uses the View Engine compiler and type checking, even for Ivy applications. We want to use the Ivy template parser and improved type checking for the Angular Language service to match application behavior. This migration is also a step towards unblocking the removal of View Engine, which will simplify Angular, reduce the npm package size, and improve the framework's maintainability.

Increased security with native Trusted Types in Angular

Completed Q2 2021

In collaboration with Google's security team, we're adding support for the new Trusted Types API. This web platform API helps developers build more secure web applications.

Optimized build speed and bundle sizes with Angular CLI webpack 5

Completed Q2 2021

As part of the v11 release, we introduced an opt-in preview of webpack 5 in the Angular CLI. To ensure stability, we'll continue iterating on the implementation to enable build speed and bundle size improvements.

Faster apps by inlining critical styles in Universal applications

Completed Q1 2021

Loading external stylesheets is a blocking operation, which means that the browser can't start rendering your application until it loads all the referenced CSS. Having render-blocking resources in the header of a page can significantly impact its load performance, for example, its first contentful paint. To make apps faster, we've been collaborating with the Google Chrome team on inlining critical CSS and loading the rest of the styles asynchronously.

Improve debugging with better Angular error messages

Completed Q1 2021

Error messages often bring limited actionable information to help developers resolve them. We've been working on making error messages more discoverable by adding associated codes, developing guides, and other materials to ensure a smoother debugging experience.

Improved developer onboarding with refreshed introductory documentation

Completed Q1 2021

We will redefine the user learning journeys and refresh the introductory documentation. We will clearly state the benefits of Angular, how to explore its capabilities and provide guidance so developers can become proficient with the framework in as little time as possible.

Expand component harnesses best practices

Completed Q1 2021

Angular CDK introduced the concept of component test harnesses to Angular in version 9. Test harnesses let component authors create supported APIs for testing component interactions. We're continuing to improve this harness infrastructure and clarifying the best practices around using harnesses. We're also working to drive more harness adoption inside of Google.

Author a guide for content projection

Completed Q2 2021

Content projection is a core Angular concept that does not have the presence it deserves in the documentation. As part of this project we want to identify the core use cases and concepts for content projection and document them.

Migrate to ESLint

Completed Q4 2020

With the deprecation of TSLint we will be moving to ESLint. As part of the process, we will work on ensuring backward compatibility with our current recommended TSLint configuration, implement a migration strategy for existing Angular applications and introduce new tooling to the Angular CLI toolchain.

Operation Bye Bye Backlog (also known as Operation Byelog)

Completed Q4 2020

We are actively investing up to 50% of our engineering capacity on triaging issues and PRs until we have a clear understanding of broader community needs. After that, we'll commit up to 20% of our engineering capacity to keep up with new submissions promptly.