Angular Roadmap

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, potential performance improvements, and so on. 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 will move up in the queue.

The projects below 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

Operation Bye Bye Backlog (aka Operation Byelog)

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.

Support TypeScript 4.0

We're working on adding support for TypeScript 4.0 ahead of its stable release. We always want Angular to stay up-to-date with the latest version of TypeScript so that developers get the best the language has to offer.

Update our e2e testing strategy

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.

Angular libraries use Ivy

We are investing in the design and development of Ivy library distribution plan, which will include an update of the library package format to use Ivy compilation, unblock the deprecation of the View Engine library format, and ngcc.

Evaluate future RxJS changes (v7 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 of RxJS and plan an update strategy.

Angular language service uses 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 will also be a step towards unblocking the removal of View Engine, which will simplify Angular, reduce the npm package size, and improve the framework's maintainability.

Expand component harnesses best practices

Angular CDK introduced the concept of component test harnesses to Angular in version 9. Test harnesses allow component authors to 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.

Support native Trusted Types in Angular

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

Integrate MDC Web into Angular Material

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 will align Angular Material more closely with the Material Design specification, expand accessibility, overall improve component quality, and improve our team's velocity.

Offer Google engineers better integration with Angular and Google's internal server stack

This is an internal project to add support for Angular front-ends to Google's internal integrated server stack.

Angular versioning & branching

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

Future

Refresh introductory documentation

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.

Strict typing for `@angular/forms`

We will work on implementing stricter type checking for reactive forms. This way, we will allow developers to catch more issues during development time, enable better text editor and IDE support, and improve the type checking for reactive forms.

webpack 5 in the Angular CLI

Webpack 5 brings a lot of build speed and bundle size improvements. To make them available for Angular developers, we will invest in migrating Angular CLI from using deprecated and removed webpack APIs.

Commit message standardization

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.

Optional Zone.js

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

Remove legacy View Engine

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

Angular DevTools

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

Optional NgModules

To simplify the Angular mental model and learning journey, we'll be working on making NgModules optional. This work will allow developers to develop standalone components and implement an alternative API for declaring the component's compilation scope.

Ergonomic component level code-splitting APIs

A common problem of 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.

Migration to ESLint

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.