Browser support

Angular supports most recent browsers. This includes the following specific versions:

Browser

Supported versions

Chrome

latest

Firefox

latest and extended support release (ESR)

Edge

2 most recent major versions

Safari

2 most recent major versions

iOS

2 most recent major versions

Android

2 most recent major versions

Angular's continuous integration process runs unit tests of the framework on all of these browsers for every pull request, using Sauce Labs.

Polyfills

Angular is built on the latest standards of the web platform. Targeting such a wide range of browsers is challenging because they do not support all features of modern browsers. You compensate by loading polyfill scripts ("polyfills") for the browsers that you must support. See instructions on how to include polyfills into your project below.

The suggested polyfills are the ones that run full Angular applications. You might need additional polyfills to support features not covered by this list. Note that polyfills cannot magically transform an old, slow browser into a modern, fast one.

Enabling polyfills with CLI projects

The Angular CLI provides support for polyfills. If you are not using the CLI to create your projects, see Polyfill instructions for non-CLI users.

When you create a project with the `ng new` command, a `src/polyfills.ts` configuration file is created as part of your project folder. This file incorporates the mandatory and many of the optional polyfills as JavaScript `import` statements.

- The npm packages for the mandatory polyfills (such as `zone.js`) are installed automatically for you when you create your project with `ng new`, and their corresponding `import` statements are already enabled in the `src/polyfills.ts` configuration file.
- If you need an *optional* polyfill, you must install its npm package, then uncomment or create the corresponding import statement in the `src/polyfills.ts` configuration file.

{@a non-cli}

Polyfills for non-CLI users

If you are not using the CLI, add your polyfill scripts directly to the host web page (`index.html`).

For example:

<!– pre-zone polyfills –> <script src="node_modules/core-js/client/shim.min.js"></script> <script> /** * you can configure some zone flags which can disable zone interception for some * asynchronous activities to improve startup performance - use these options only * if you know what you are doing as it could result in hard to trace down bugs.. */ // ___Zone_disable_requestAnimationFrame = true; // disable patch requestAnimationFrame // ___Zone_disable_on_property = true; // disable patch onProperty such as onclick // ___zone_symbol___UNPATCHED_EVENTS = ['scroll', 'mousemove']; // disable patch specified eventNames / in Edge developer tools, the addEventListener will also be wrapped by zone.js * with the following flag, it will bypass `zone.js` patch for Edge */ // ___Zone_enable_cross_context_check = true; </script> <!– zone.js required by Angular –> <script src="node_modules/zone.js/bundles/zone.umd.js"></script> <!– application polyfills –>