**Angular 11**

**Angular 11** was released on **Nov 11, 2020**.

**Angular v11** is now available! Let’s see the main features:

1. **Automatic Inlining of Fonts**

During compile time Angular CLI will download and inline fonts that are being used and linked in the application. We enable this by default in apps built with version 11. All you need to do to take advantage of this optimization is update your app!

**Component Test Harnesses**

In Angular v9 we introduced Component Test Harnesses. They provide a robust and legible API surface to help with testing Angular Material components. It gives developers a way to interact with Angular Material components using the supported API during testing.

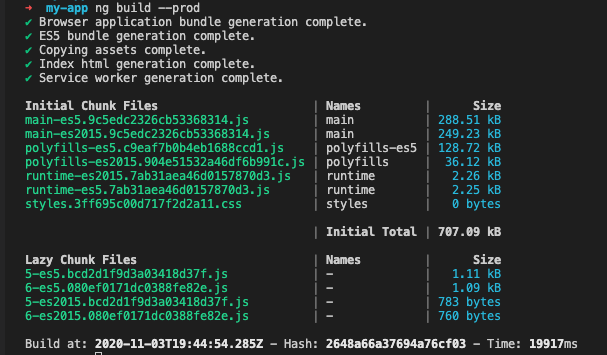
Releasing with version 11, we have harnesses for all of the components! Now developers can create more robust test suites.

They have also included performance improvements and new APIs. The parallel function makes working with asynchronous actions in your tests easier by allowing developers to run multiple asynchronous interactions with components in parallel. The manualChangeDetection function gives developers access to finer grained control of change detection by disabling automatic change detection in unit tests.

For more details and examples of these APIs and other new features, be sure to check out the [documentation for Angular Material](http://material.angular.io/cdk/test-harnesses/overview) Test Harnesses!

1. **Improved Reporting and Logging**

They have made changes to the builder phase reporting to make it even more helpful during development. We are bringing in new CLI output updates to make logs and reports easier to read.

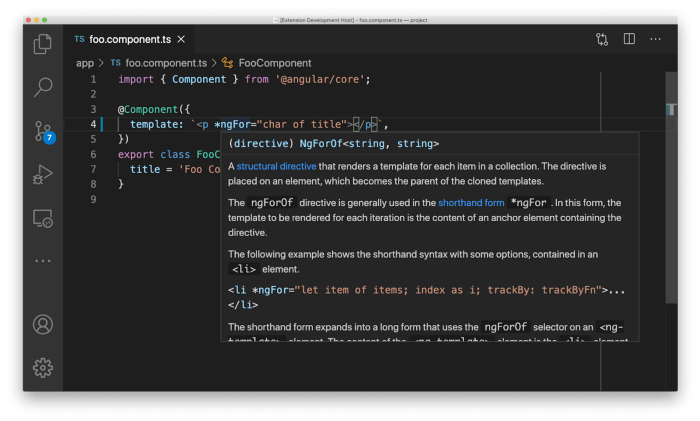
****

**Improved CLI output formatting**

1. **Updated Language Service Preview**

The Angular Language Service provides helpful tools to make development with Angular productive and fun. The current version of the language service is based on View Engine and today we’re giving a sneak peek of the Ivy-based language service. The updated language service provides a more powerful and accurate experience for developers.

Now, the language service will be able to correctly infer generic types in templates the same way the TypeScript compiler does. For example, in the screenshot below we’re able to infer that the iterable is of type string.

**Angular Language Service inferring iterable types in templates**

This powerful new update is still in development but we wanted to share an update as we keep preparing it for a full release in an upcoming version.

1. **Updated Hot Module Replacement (HMR) Support**

Angular has offered support for HMR but enabling it required configuration and code changes making it less than ideal to quickly include in Angular projects. In version 11 we’ve updated the CLI to allow enabling HMR when starting an application with ng serve. To get started, run the following command:

ng serve --hmr

After the local server starts the console will display a message confirming that HMR is active:

**NOTICE:** Hot Module Replacement (HMR) is enabled for the dev server.

See <https://webpack.js.org/guides/hot-module-replacement> for information on working with HMR for webpack.

Now during development the latest changes to components, templates and styles will be instantly updated into the running application. All without requiring a full page refresh. Data typed into forms are preserved as well as scroll position providing a boost to developer productivity.

## Faster Builds

Angular team bringing a faster development and build cycle by making updates to some key areas.

* When installing dependencies, the ngcc update process is now 2–4x faster.
* Faster compilation with TypeScript v4.0.

# **Experimental webpack 5 Support**

Now, teams can opt-in to webpack v5. Currently, you could experiment with [module federation](https://webpack.js.org/concepts/module-federation/). In the future, webpack v5 will clear the path for:

* Faster builds with persistent disk caching
* Smaller bundles thanks to [cjs tree-shaking](https://webpack.js.org/guides/tree-shaking/)

As you know, webpack is used to compile a large number of files into a single bundle or single file. Webpack 5 is the latest version and was released in the last month. It is still not fully stable.

However, Angular 11 provides experimental support for webpack 5 and you can use it with Angular 11 to try out new things. According to the release notes, the Angular team believe that they can extend this experimental support to achieve faster builds and small bundles once things get stable.

If you are up to try webpack 5, you can start by adding the following lines to your package.json file:

“resolutions”: {  
 “webpack”: “5.4.0”  
}

Currently, you’ll need to use **yarn** to test this as npm does not yet support the resolutions property.

## Linting

In previous versions of Angular, we’ve shipped a default implementation for linting (TSLint). Now, TSLint is deprecated by the project creators who recommend migration to ESLint. [James Henry](https://twitter.com/mrjameshenry) together with other folks from the open-source community developed a third-party solution and migration path via [typescript-eslint](https://github.com/typescript-eslint/typescript-eslint), [angular-eslint](https://github.com/angular-eslint/angular-eslint) and [tslint-to-eslint-config](https://github.com/typescript-eslint/tslint-to-eslint-config)! We’ve been collaborating closely to ensure a smooth transition of Angular developers to the supported linting stack.

We’re deprecating the use of TSLint and Codelyzer in version 11. This means that in future versions the default implementation for linting Angular projects will not be available. The ng lint command will function similarly to ng deploy, suggesting recommended implementations developers could add to their projects.

Head over to the [official project page](https://github.com/angular-eslint/angular-eslint#migrating-from-codelyzer-and-tslint) for a guide to incorporate angular-eslint in a project and migrate from TSLint.

## Housekeeping

In this update we’re removing support for IE9/IE10 and IE mobile. IE11 is the only version of IE [still supported](https://angular.io/guide/browser-support) by Angular. We’ve also [removed deprecated APIs](https://angular.io/guide/deprecations) and added a few to the deprecation list. Be sure to check this out to make sure you are using the latest APIs and following our recommended best practices.

# **IE9/IE10/IE Mobile are out**

Angular 11 has finally removed support for IE9/IE10 and IE Mobile. For obscure corporate reasons, support for IE 11 is still supported, but please don’t count on that staying true for long.

Now that we have Edge chromium, there’s no reason to let IE linger any longer in corporate environments.

# **TypeScript 4.0 Support**

With this latest update, the Angular team has dropped support for TypeScript 3.9. Now Angular 11 only supports TypeScript 4.0.

One of the main reasons behind this upgrade is to speed up the builds. Angular 11 ensures much faster builds than previous versions.

# **Other Changes:**

Apart from that, Angular 11 brings a large number of small changes like:

* New automated migrations and schematics.
* Service worker improvements.
* Lazy loading support for named outlets.
* Angular CLI can now generate resolve guards.
* Stricter types built in pipes.
* The **formatDate** function now supports ISO 8601 week-numbering year format.

## ****Some Breaking Changes:****

* Angular 11 removes IE 9, 10, and IE mobile support completely.
* TypeScript 3.9 is no longer supported.
* preserveQueryParams has been removed in router and queryParamsHandling=”preserve” can be used instead.
* Expressions within ICUs are now type-checked again.
* The async pipe no longer claims to return undefined for an input that was typed as undefined.
* platform-server: If you use useAbsoluteUrl to setup platform-server, you now need to also specify baseUrl. We are intentionally making this a breaking change in a minor release, because if useAbsoluteUrl is set to true then the behavior of the application could be unpredictable, resulting in issues that are hard to discover but could be affecting production environments.
* compiler: TypeScript 3.9 is no longer supported, please upgrade to TypeScript 4.0.
* router: \* The initialNavigation property for the options in RouterModule.forRoot no longer supports legacy\_disabled, legacy\_enabled, true, or false as valid values. legacy\_enabled (the old default) is instead enabledNonBlocking
* enabled is deprecated as a valid value for the RouterModule.forRoot initialNavigation option. enabledBlocking has been introduced to replace it
* router: preserveQueryParams has been removed, use queryParamsHandling=”preserve” instead
* router: If you were accessing the RouterLink values of queryParams, fragment or queryParamsHandling you might need to relax the typing to also accept undefined and null. ([#39151](https://github.com/angular/angular/issues/39151))
* core: \* ViewEncapsulation.Native has been removed. Use ViewEncapsulation.ShadowDom instead. Existing usages will be updated automatically by ng update.
* compiler-cli: Expressions within ICUs are now type-checked again, fixing a regression in Ivy. This may cause compilation failures if errors are found in expressions that appear within an ICU. Please correct these expressions to resolve the type-check errors.
* forms: Directives in the @angular/forms package used to have any[] as a type of validators and asyncValidators arguments in constructors. Now these arguments are properly typed, so if your code relies on directive constructor types it may require some updates to improve type safety.
* forms: Type of AbstractFormControl.parent now includes null. null is now included in the types of .parent. If you don't already have a check for this case, the TypeScript compiler might complain. A v11 migration exists which adds the non-null assertion operator where necessary. In an unlikely case your code was testing the parent against undefined with strict equality, you'll need to change this to === null instead, since the parent is now explicitly initialized with null instead of being left undefined.
* packaging: In v10, IE 9, 10, and IE mobile support was deprecated. In v11, Angular framework removes IE 9, 10, and IE mobile support completely. Supporting outdated browsers like these increases bundle size, code complexity, and test load, and also requires time and effort that could be spent on improvements to the framework. For example, fixing issues can be more difficult, as a straightforward fix for modern browsers could break old ones that have quirks due to not receiving updates from vendors.
* platform-webworker: @angular/platform-webworker and @angular/platform-webworker-dynamic have been removed as they were deprecated in v8
* common: The slice pipe now returns null for the undefined input value, which is consistent with the behavior of most pipes. If you rely on undefined being the result in that case, you now need to check for it explicitly.
* common: The typing of the keyvalue pipe has been fixed to report that for input objects that have number keys, the result will contain the string representation of the keys. This was already the case and the code has simply been updated to reflect this. Please update the consumers of the pipe output if they were relying on the incorrect types. Note that this does not affect use cases where the input values are Maps, so if you need to preserve numbers, this is an effective way.
* common: The signatures of the number pipes now explicitly state which types are accepted. This should only cause issues in corner cases, as any other values would result in runtime exceptions.
* common: The signature of the date pipe now explicitly states which types are accepted. This should only cause issues in corner cases, as any other values would result in runtime exceptions.
* common: The async pipe no longer claims to return undefined for an input that was typed as undefined. Note that the code actually returned null on undefined inputs. In the unlikely case you were relying on this, please fix the typing of the consumers of the pipe output.
* common: The case conversion pipes no longer let falsy values through. They now map both null and undefined to null and raise an exception on invalid input (0, false, NaN) just like most "common pipes". If your code required falsy values to pass through, you need to handle them explicitly.
* router: While the new parameter types allow a variable of type NavigationExtras to be passed in, they will not allow object literals, as they may only specify known properties. They will also not accept types that do not have properties in common with the ones in the Pick. To fix this error, only specify properties from the NavigationExtras which are actually used in the respective function calls or use a type assertion on the object or variable: as NavigationExtras.
* router: This commit changes the default value of relativeLinkResolution from 'legacy' to 'default'. If your application previously used the default by not specifying a value in the ExtraOptions and uses relative links when navigating from children of empty path routes, you will need to update your RouterModule to specifically specify 'legacy' for relativeLinkResolution. See <https://angular.io/api/router/ExtraOptions#relativeLinkResolution> for more details.
* core: If you call TestBed.overrideProvider after TestBed initialization, provider overrides are not applied. This behavior is consistent with other override methods (such as TestBed.overrideDirective, etc) but they throw an error to indicate that, when the check was missing in the TestBed.overrideProvider function. Now calling TestBed.overrideProvider after TestBed initialization also triggers an error, thus there is a chance that some tests.

(where TestBed.overrideProvider is called after TestBed initialization) will start to fail and require updates to move TestBed.overrideProvider calls before TestBed initialization is completed.

* router: This change corrects the argument order when calling RouteReuseStrategy#shouldReuseRoute. Previously, when evaluating child routes, they would be called with the future and current arguments would be swapped. If your RouteReuseStrategy relies specifically on only the future or current snapshot state, you may need to update the shouldReuseRoute implementation’s use of “future” and “current” ActivateRouteSnapshots.
* common: The locale data API has been marked as returning readonly arrays, rather than mutable arrays, since these arrays are shared across calls to the API. If you were mutating them (e.g. calling sort(), push(), splice(), etc) then your code will not longer compile. If you need to mutate the array, you should now take a copy (e.g. by calling slice()) and mutate the copy.
* common: When passing a date-time formatted string to the DatePipe in a format that contains fractions of a millisecond, the milliseconds will now always be rounded down rather than to the nearest millisecond. Most applications will not be affected by this change. If this is not the desired behaviour then consider pre-processing the string to round the millisecond part before passing it to the DatePipe.
* core: CollectionChangeRecord has been removed, use IterableChangeRecord instead
* forms: Previously if FormControl, FormGroup and FormArray class instances had async validators defined at initialization time, the status change event was not emitted once async validator completed. After this change the status event is emitted into the statusChanges observable.

(If your code relies on the old behaviour, you can filter/ignore this additional status change event)

## How to update to get version 11?

Run this command to update Angular and CLI:

**ng update @angular/cli @angular/core**

**THANK YOU**

[**https://github.com/PranamBhat**](https://github.com/PranamBhat)

[**https://www.linkedin.com/in/pranam-bhat-11670689/**](https://www.linkedin.com/in/pranam-bhat-11670689/)