Stephen Fluin Follow
May 3 · 7 min read

The 6.0.0 release of Angular is here! This is a major release focused less on the underlying framework, and more on the toolchain and on making it easier to move quickly with Angular in the future.

a photo by Manu Murthy of the Angular Team

As a part of this release, we are synchronizing the major versions going forward for the framework packages (`@angular/core` , `@angular/common` , `@angular/compiler` , etc), the Angular CLI, and Angular Material + CDK. All are releasing as 6.0.0 today. We made this change to clarify cross compatibility. The minor and patch releases for these projects will be issued based on the project's needs.

See the full list of changes in our changelogs: framework, material+cdk, cli.

ng update

`ng update <package>` is a new CLI command that analyzes your `package.json` and uses its knowledge of Angular to recommend updates to your application. To see it in action, check out our update guide.

Not only will `ng update` help you adopt the right version of dependencies, and keep your dependencies in sync, but 3rd parties can provide update scripts using schematics. If one of your dependencies provides an `ng update` schematic, they can automatically update your code when they need to make breaking changes!

`ng update` will not replace your package manager, but uses npm or yarn under the hood to manage dependencies. In addition to updating dependencies and peer dependencies, `ng update` will apply needed transforms to your project.

For example, the command `ng update @angular/core` will update all of the Angular framework packages as well as RxJS and TypeScript, and will run any schematics available on these packages to keep you up to date. As part of this one command, we'll automatically install `rxjs-compat` into your application to make the adoption of RxJS v6 smoother.

We expect to see many more libraries and packages add `ng update` schematics over the coming months, and have already heard from enterprise component library teams that are planning to use `ng update` to push through important changes in an automated way to save their developers time.

Learn more about how the `ng update` command works. To get started creating your own `ng update` schematic, take a look at the entry in the `package.json` of rxjs and its associated `collection.json` .

ng add

Another new CLI command `ng add <package>` makes adding new capabilities to your project easy. `ng add` will use your package manager to download new dependencies and invoke an installation script (implemented as a schematic) which can update your project with configuration changes, add additional dependencies (e.g. polyfills), or scaffold package-specific initialization code.

Try out some of the following on your fresh `ng new` application:

- **`ng add @angular/pwa`** —Turn your application into a PWA by adding an app manifest and service worker
- **`ng add @ng-bootstrap/schematics`** —Add ng-bootstrap to your application
- **`ng add @angular/material`** —Install and setup Angular Material and theming and register new starter components into `ng generate`
- **`ng add @clr/angular`**—Install and setup Clarity from VMWare
- **`ng add @angular/elements`** —Add the needed `document-register-element.js` polyfill and dependencies for Angular Elements (see below)

Because `ng add` is built on top of schematics and the npm registry, our hope is that libraries and the community will help us build a rich ecosystem of `ng add` supporting packages.

Take a look at Angular Material's ng-add schematic for an example to help you get started building your own ng-add schematics.

Angular Elements

The first release of Angular Elements is focused on allowing you to bootstrap Angular components within an existing Angular application by registering them as Custom Elements. We use this extensively in angular.io as part of our content management system to allow dynamic bootstrapping of capabilities via embedded HTML. This replaces the need to manually bootstrap Angular components found in static html content.

Check out an example of registering a component as a custom element or learn more about Angular Elements.

One of our community members has also produced an Angular Elements Quick Start video that we highly recommend.

Angular Material + CDK Components

The biggest addition is the new tree component for displaying hierarchical data. Following patterns from the data-table component, the CDK houses the core tree directives, with Angular Material offering the same experience with Material Design styles on top. We recently gave a talk about the component, so check that out for more information (video, slides). These new tree components come in both styled (Material's `mat-tree`) and unstyled versions (CDK's `cdk-tree`).

Alongside the tree, we also have new badge and bottom-sheet components. Badges help display small bits of helpful information, such as unread item counts. Bottom-sheets are a special type of mobile-centric dialogs that come up from the bottom of the viewport, commonly used to present a list of options following an action.

The `@angular/cdk/overlay` package is one of the most powerful pieces of infrastructure in the CDK today. With the release of v6, this package now includes new positioning logic that helps make pop-ups that intelligently remain on-screen in all situations.

Angular Material Starter Components

ng generate for adding a dashboard to your project

Once you have run `ng add @angular/material` to add material to an existing application, you will also be able to generate 3 new starter components.

—

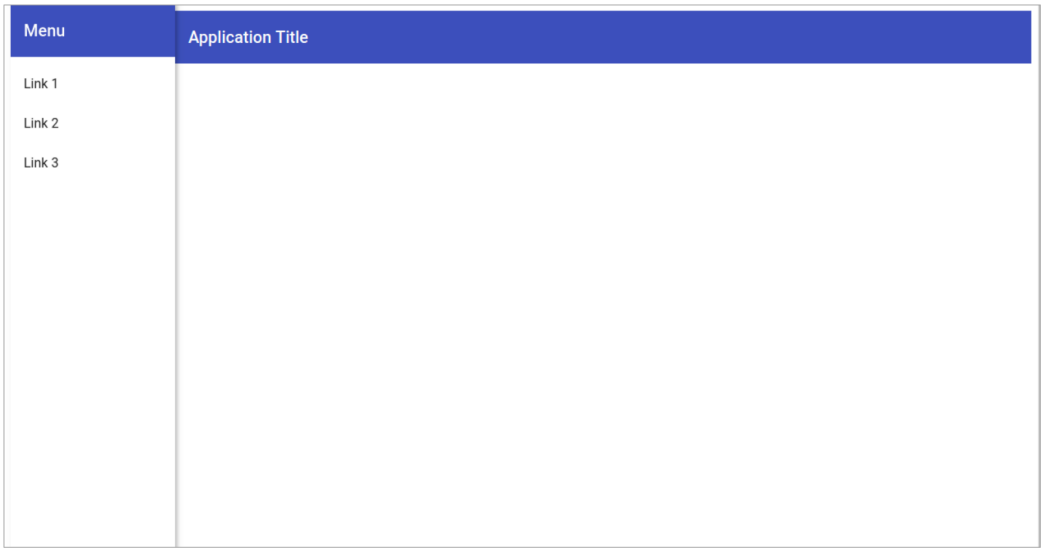
Material Sidenav

You can now generate a starter component including a toolbar with the app name and the side navigation. This component is responsive based on breakpoints.

Run:

```
ng generate @angular/material:material-nav --name=my-nav
```

This will create this starter component:



material-nav

—

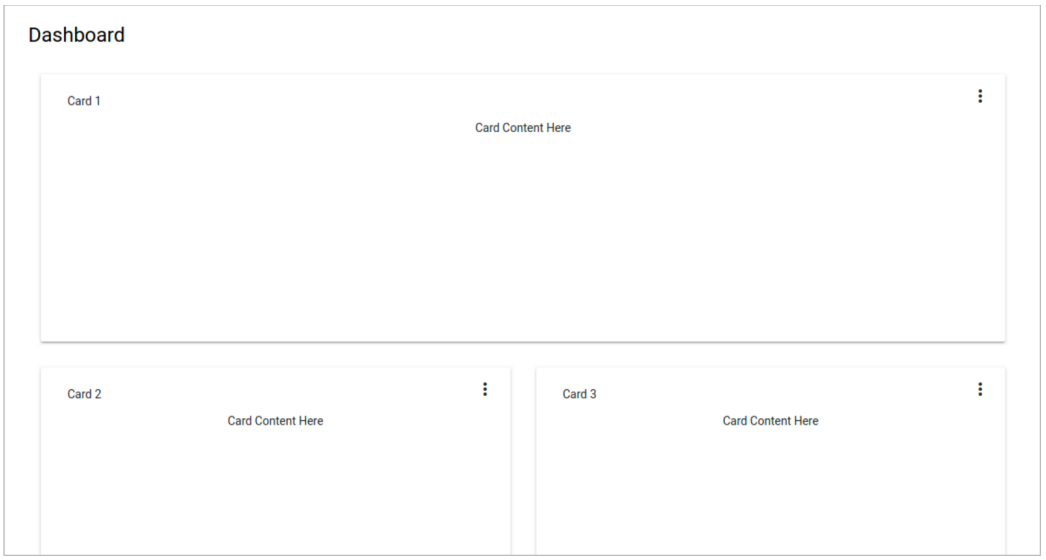
Material Dashboard

You can now generate a starter dashboard component containing a dynamic grid list of cards.

Run:

```
ng generate @angular/material:material-dashboard --name=my-dashboard
```

This will create this starter component:



material-dashboard

—

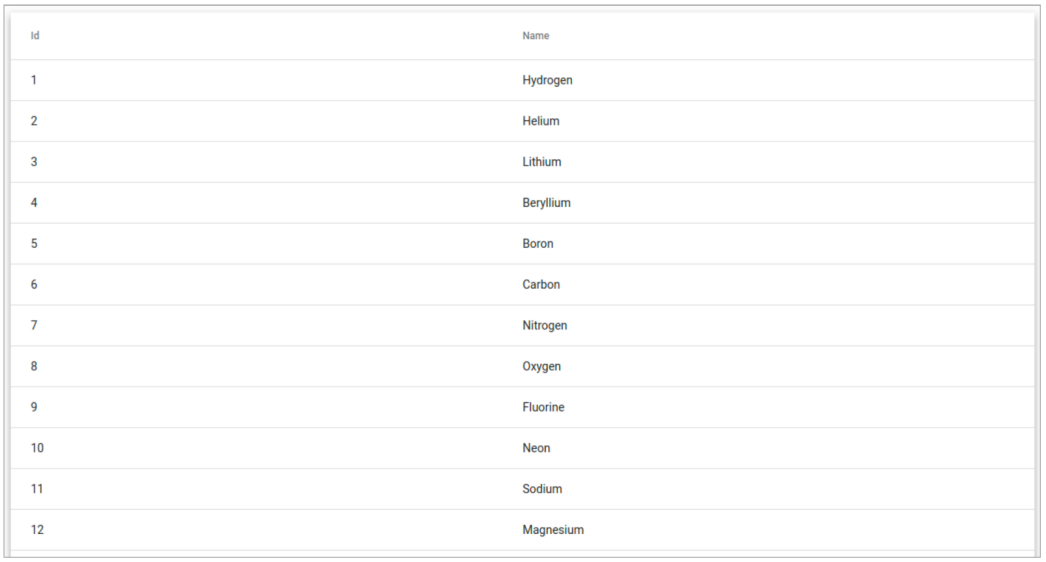
Material Data Table

You can generate a starter data table component that is pre-configured with a `datasource` for sorting and pagination.

Run:

```
ng generate @angular/material:material-table --name=my-table
```

This will create this starter component:



Id	Name
1	Hydrogen
2	Helium
3	Lithium
4	Beryllium
5	Boron
6	Carbon
7	Nitrogen
8	Oxygen
9	Fluorine
10	Neon
11	Sodium
12	Magnesium

material-table

—

Learn more about the available Angular Material Schematics.

CLI Workspaces

CLI v6 now has support for workspaces containing multiple projects, such as multiple applications or libraries. CLI projects will now use `angular.json` instead of `.angular-cli.json` for build and project configuration.

Each CLI workspace has projects, each project has targets, and each target can have configurations.

```
{
  "projects": {
    "my-project-name": {
      "projectType": "application",
      "architect": {
        "build": {
          "configurations": {
            "production": {},
            "demo": {},
            "staging": {},
          }
        },
        "serve": {},
        "extract-i18n": {}
```

Learn more about the new configuration file

Library Support

One of the most requested features for our CLI has been support for creating and building libraries, and we are proud to introduce:

```
ng generate library <name>
```

ng generate library within an existing project

This command will create a library project within your CLI workspace, and configure it for testing and for building.

Learn more about creating libraries with the Angular CLI

Tree Shakable Providers

To make your applications smaller, we've moved from modules referencing services to services referencing modules. This allows us to only bundle services into your code base in modules where they are injected.

Before

```
@NgModule({
  ...
  providers: [MyService]
})
```

```
import { Injectable } from '@angular/core';

@Injectable()
export class MyService {
  constructor() { }
```

After

No references are needed in our NgModule.

```
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root',
})
export class MyService {
```

Read more about Dependency Injection

Animations Performance Improvements

We've updated our implementation of Animations to no longer need the web animations polyfill. This means that you can remove this polyfill from your application and save approximately 47KB of bundle size, while increasing animations performance in Safari at the same time.

RxJS v6

Angular has been updated to use v6 of RxJS. RxJS is an independent project that released v6 several weeks ago. RxJS v6 brings with it several major changes, along with a backwards compatibility package `rxjs-compat` that will keep your applications working.

RxJS has been rearranged to make it more tree-shakable, ensuring that only the pieces of RxJS that you use are included in your production bundles.

If you use `ng update`, your application should keep working, but you can learn more about the 5.5 to 6.0 migration.

Long Term Support (LTS)

We are expanding our Long Term Support to all major releases.

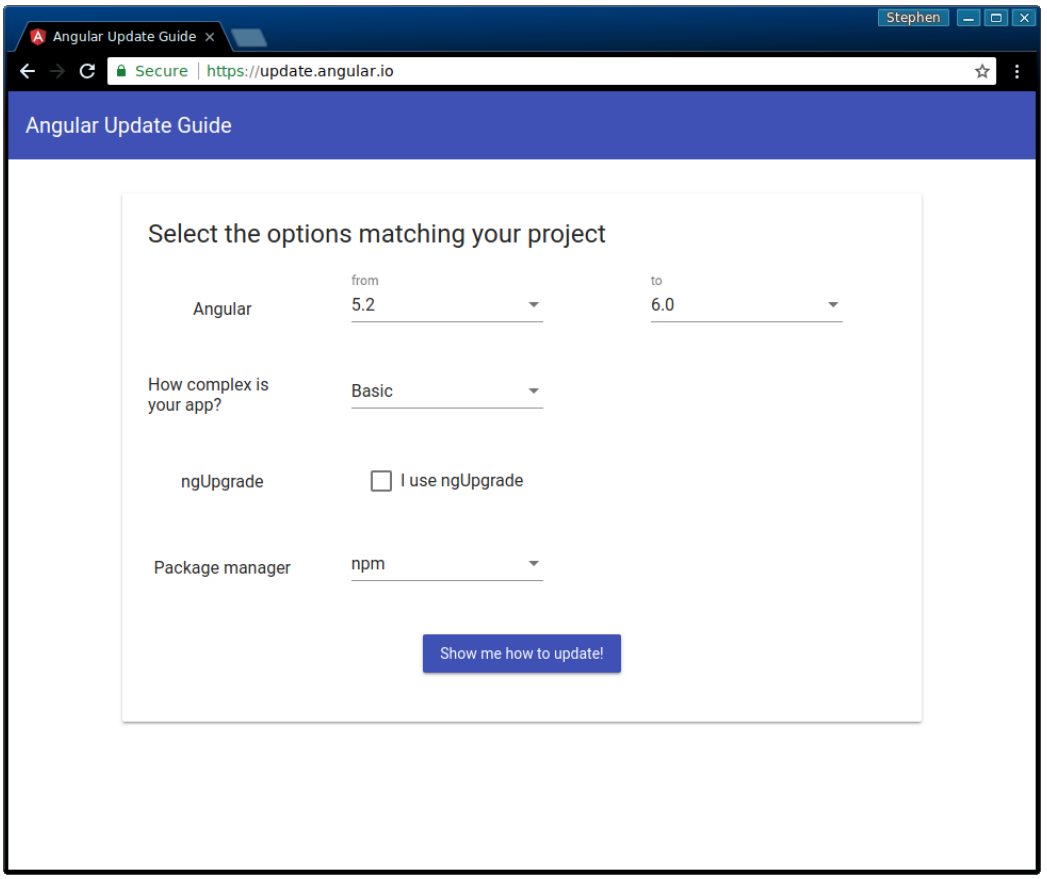
Previously we announced that only v4 and v6 would be LTS releases but in order to make updating from one major to the next easier, and give bigger projects more time to plan updates, we have decided to extend the long-term support to all major releases starting with v4.

Each major release will be supported for 18 months with around 6 months of active development followed by 12 months of critical bugfixes and security patches.

Learn more about how Angular versions and releases.

How to update to 6.0.0

Visit update.angular.io for information and guidance on updating your application.



https://update.angular.io

The update generally follows 3 steps, and will take advantage of the new `ng update` tool.

1. Update @angular/cli
2. Update your Angular framework packages
3. Update other dependencies

Making it easy for developers to stay up to date with the latest releases is extremely important to us, so let us know what you think about this release in the comments!

What about Ivy?

At ng-conf we mentioned a new initiative called Ivy—our next generation rendering pipeline. Ivy is currently under active development and is not part of the 6.0 release. We will announce an opt-in preview of Ivy as soon as it is ready in the coming months. Keep an eye on this blog for the latest information.