****

**Angular 6**

**Submitted By:**

**Shaikh Sameer**

Angular 6 was released on May 4th, 2018. That is 6 months after its predecessor's (Angular 5) release. The highlights of Angular 6 include the **Angular Command Line Interface** (CLI), **The Component Development KIT**(CDK) and the **Angular Material package**update. The cherry on top, all three are shipped as part of Angular 6, not separate updates.

Let's get to the gritty details by first knowing that Angular 6 uses the RXJS library, so hurray for reactive programming for web!

Moving on, this version release is more focused on the tooling and support rather than the whole framework.

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.

As a part of this release, the major versions going forward for the framework packages are synchronized (@angular/core, @angular/common, @angular/compiler, etc), the Angular CLI, and Angular Material + CDK. The changes are made to clarify cross compatibility. The minor and patch releases for these projects will be issued based on the project’s needs.

**Major Changes**

1. **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.

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 updateschematics 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.

**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. 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 @angular/elements** — Add the needed document-register-element.js polyfill and dependencies for Angular Elements

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.

### 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. These new tree components come in both styled (Material’s mat-tree) and unstyled versions (CDK’s cdk-tree).

Alongside the tree, it also has 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.

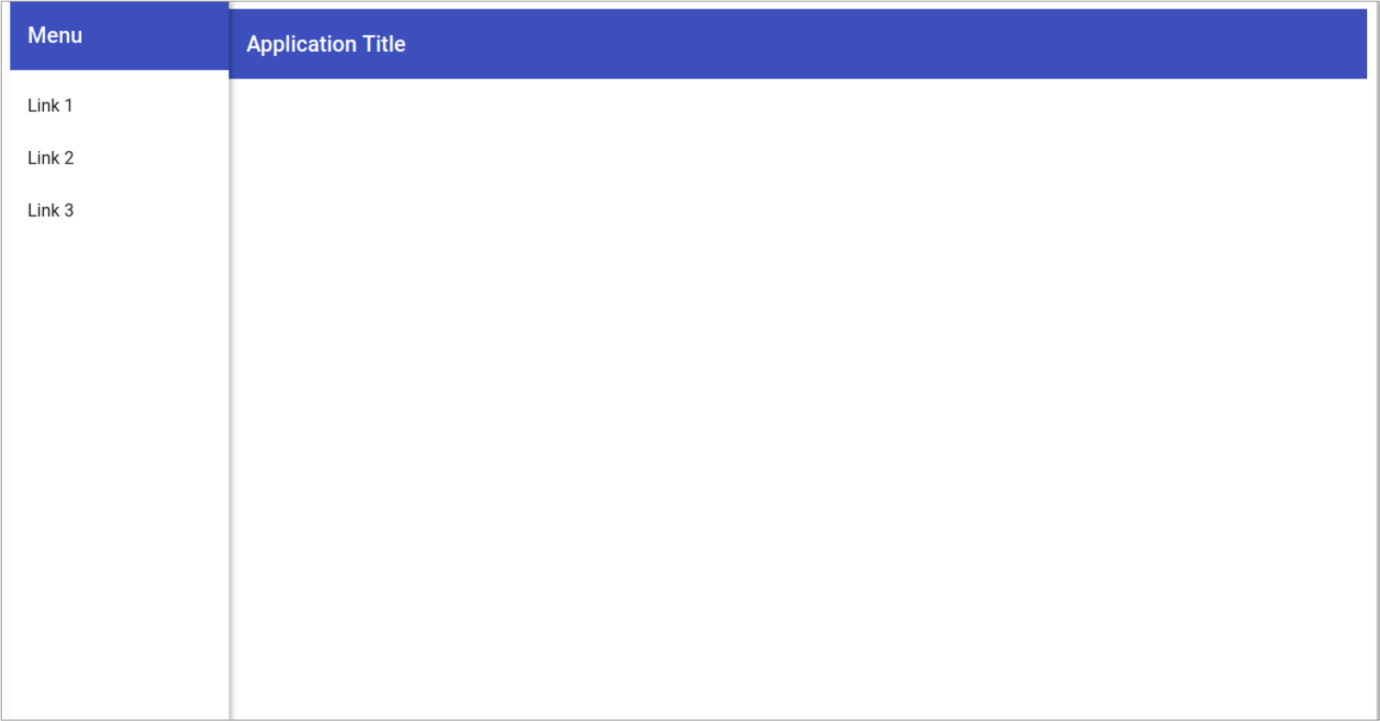
**Material Sidenav**

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

It will render the following component



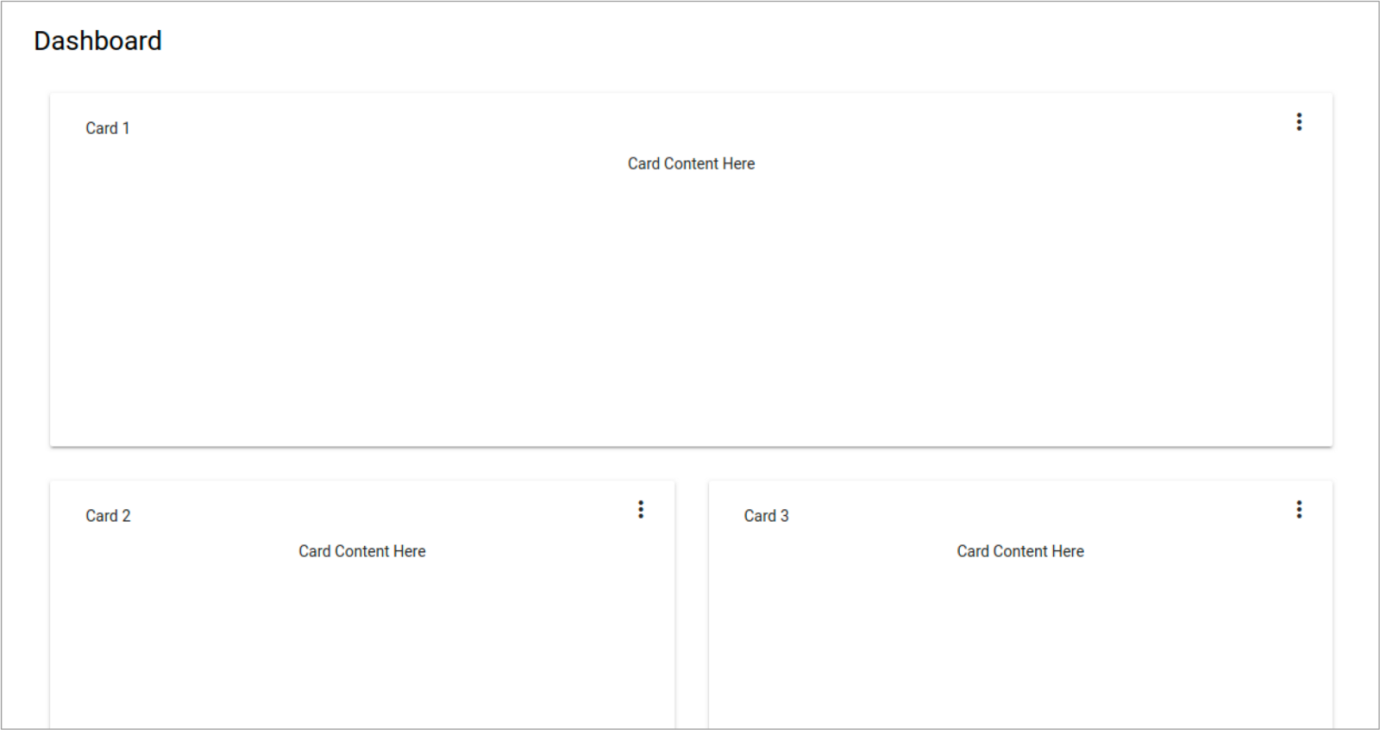
**Material Dashboard**

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

It will render the below Dashboard Component



### CLI Workspaces

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

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



**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:

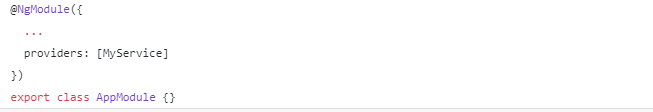
ng generate library <name>

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

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.

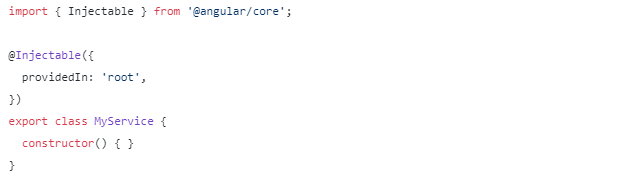
**Before**



### C:\DeepakJagtap\Sameer\Winflex\Angular 6\Capture2.PNG

**After**

No references are needed in our NgModule.

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.

How to update to 6.0.0

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

**Angular service worker introduction**

Service workers augment the traditional web deployment model and empower applications to deliver a user experience with the reliability and performance on par with natively-installed code. Adding a service worker to an Angular application is one of the steps for turning an application into a Progressive Web App (also known as a PWA).

## Adding a service worker to your project

To set up the Angular service worker in your project, use the CLI command ng add @angular/pwa. It takes care of configuring your app to use service workers by adding the service-worker package along with setting up the necessary support files.

add @angular/pwa --project \*project-name\*

The above command completes the following actions:

1. Adds the @angular/service-worker package to your project.
2. Enables service worker build support in the CLI.
3. Imports and registers the service worker in the app module.
4. Updates the index.html file:
   * Includes a link to add the manifest.json file.
   * Adds meta tags for theme-color.
5. Installs icon files to support the installed Progressive Web App (PWA).
6. Creates the service worker configuration file called [ngsw-config.json](https://v6.angular.io/guide/service-worker-config), which specifies the caching behaviors and other settings.

Now, build the project:

build --prod

The CLI project is now set up to use the Angular service worker.

### Simulating a network issue

To simulate a network issue, disable network interaction for your application. In Chrome:

1. Select **Tools** > **Developer Tools** (from the Chrome menu located at the top right corner).
2. Go to the **Network tab**.
3. Check the **Offline box**.

The offline checkbox in the Network tab is checked

If you look at the Network tab, you can verify that the service worker is active.



Notice that under the "Size" column, the requests state is (from ServiceWorker). This means that the resources are not being loaded from the network. Instead, they are being loaded from the service worker's cache.

# Service worker communication

Importing ServiceWorkerModule into your AppModule doesn't just register the service worker, it also provides a few services you can use to interact with the service worker and control the caching of your app.

## SwUpdate service

The SwUpdate service gives you access to events that indicate when the service worker has discovered an available update for your app or when it has activated such an update—meaning it is now serving content from that update to your app.

The SwUpdate service supports four separate operations:

* Getting notified of available updates. These are new versions of the app to be loaded if the page is refreshed.
* Getting notified of update activation. This is when the service worker starts serving a new version of the app immediately.
* Asking the service worker to check the server for new updates.
* Asking the service worker to activate the latest version of the app for the current tab.

### Available and activated updates

The two update events, available and activated, are Observable properties of SwUpdate:

@Injectable()

export class LogUpdateService {

constructor(updates: SwUpdate) {

updates.available.subscribe(event => {

console.log('current version is', event.current);

console.log('available version is', event.available);

});

updates.activated.subscribe(event => {

console.log('old version was', event.previous);

console.log('new version is', event.current);

});

}

}

### Checking for updates

It's possible to ask the service worker to check if any updates have been deployed to the server. You might choose to do this if you have a site that changes frequently or want updates to happen on a schedule.

Do this with the checkForUpdate() method:

import { interval } from 'rxjs';

@Injectable()

export class CheckForUpdateService {

constructor(updates: SwUpdate) {

interval(6 \* 60 \* 60).subscribe(() => updates.checkForUpdate());

}

}

### Forcing update activation

If the current tab needs to be updated to the latest app version immediately, it can ask to do so with the activateUpdate() method:

@Injectable()

export class PromptUpdateService {

constructor(updates: SwUpdate) {

updates.available.subscribe(event => {

if (promptUser(event)) {

updates.activateUpdate().then(() => document.location.reload());

} });}}

The src/ngsw-config.json configuration file specifies which files and data URLs the Angular service worker should cache and how it should update the cached files and data. The CLI processes the configuration file during ng build --prod. Manually, you can process it with the ngsw-config tool:

ngsw-config dist src/ngsw-config.json /base/[href](https://v6.angular.io/api/router/RouterLinkWithHref" \l "href)

The configuration file uses the JSON format. All file paths must begin with /, which is the deployment directory—usually dist in CLI projects.

### ****Multiple Validators For Your Forms****

Those of you who had to fuss about passing more than one validator in your Formbuilders, Angular 6 now allows you to pass multiple validators to the formBuilder.

**Angular 6 Changes :**

1) Typescript 2.7+ supports

2) Added Angular Material and CDK Stable

3) Component Dev Kit (CDK) - CDK allows you to build your own library of UI components using Angular Material.

4) Improved decorator error messages

5) Fix platform-detection example for Universal

6) Ivy Renderer - It is a new backward compatible and main focused area - speed improvements, size reduction, and increased flexibility.

7) Add afterContentInit and afterContentChecked to render

8) Added to supports of nativeElement

9) Added Optional generic type for ElementRef The Example looks like - @ViewChild('your-element') yourElement:ElementRef;

10) Bazel Compiler - Bazel only rebuilds what is necessary.

11) Added Test Comment

12) Add missing lifecycle tests for projected components

13) Closure Compiler - Closure Compiler consistently generates smaller bundles.

14) Rename QueryPredicate to LQuery and LQuery to LQueries

15) Service Worker - Service worker is a script that runs in the web browser. It also manages caching for an application.

16) Added multiple validators for array method of FormBuilder

17) Handle string with and without line boundary - Now Handle string with and without line boundary (^ & $) on pattern validators. Previously, it works with string not boundaries.

18) AbstractControl statusChanges - Previous version, not emits an event when you called “markAsPending” but now emits an event of "PENDING" when we call AbstractControl markAsPending.

19) Updates on NgModelChange - Now emitted after value and validity is updated on its control. Previously, it was emitted before updated.

20) Allow HttpInterceptors to inject HttpClient – Previously, an interceptor attempting to inject HttpClient directly would receive a circular dependency error, as HttpClient was constructed via a factory which injected the interceptor instances. Users want to inject HttpClient into interceptors to make supporting.