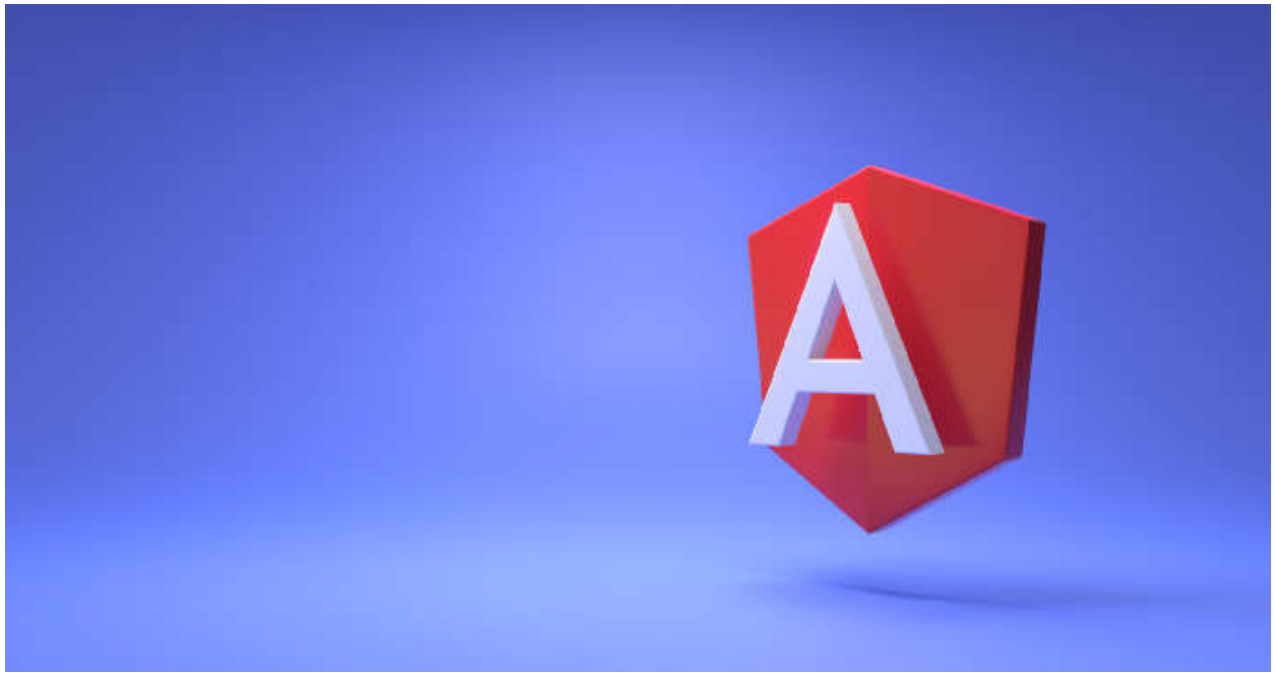
Difference Among Angular 8, 7, 6, 5, 4, 3, 2 — Breakdown, New Features, and Changes

Life N Shades Follow
Feb 10 · 7 min read



Difference Among Angular 8, 7, 6, 5, 4, 3, 2 — Breakdown, New Features, and Changes

Hey! I'm Sunny and I work at Cognizant. Follow me on LinkedIn and let me know what you're working on!

The first version of Angular was released in the year of 2010. Some people call this as AngularJS and some people call as Angular 1. But it is officially named as AngularJS.

Released in 2016

- Complete rewrite of Angular 1
- Written entirely in typescript
- Component-based instead of Controller
- ES6 and typescript supported
- More testable as component-based
- Support for Mobile/Low-end devices
- Up to typescript 1.8 is supported

68

Angular 3:

- Why we don't have Angular 3? — Angular is being developed in a MonoRepo it means a single repo for everything. @angular/core, @angular/compiler, @angular/router etc are in the same repo and may have their own versions. — The angular router was already in v3 and releasing angular 3 with router 4 will create confusion — To avoid this confusion they decided to skip the version 3 and release with version 4.0.0 so that every major dependency in the MonoRepo are on the right track.

Angular 4

- Released in 2017
- Changes in core library
- Angular 4 is simply the next version of angular 2, the underlying concept is the same & is an inheritance from Angular 2
- Lot of performance improvement is made to reduce size of AOT compiler generated code
- Typescript 2.1 & 2.2 compatible — all feature of ts 2.1 & 2.2 are supported in Angular 4 application
- Animation features are separated from @angular/core to @angular/animation — don't import @animation packages into the application to reduce bundle size and it gives the performance improvement.
- Else block in *ngIf introduced: — Instead of writing 2 ngIf for else , simply add below code in component template:

```
*ngIf="yourCondition; else myFalsyTemplate"
"<ng-template #myFalsyTemplate>Else Html</ng-template>"
```

Angular 5

- Released 1st November 2017

- Compiler Improvements: This is one of the very nice features of Angular 5, which improved the support of incremental compilation of an application.
- Preserve White space: To remove unnecessary new lines, tabs and white spaces we can add below code(decrease bundle size)

```
// in component decorator you can now add:
"preserveWhitespaces: false"
// or in tsconfig.json:
"angularCompilerOptions": { "preserveWhitespaces": false}`
```

68

- Increased the standardization across all browsers: For internationalization we were depending on `i18n` , but in ng 5 provides a new date, number, and currency pipes which increases the internationalization across all the browsers and eliminates the need of i18n polyfills.
- exportAs: In Angular 5, multiple names support for both directives and components
- HttpClient: until Angualar 4.3 @angular/HTTP was been used which is now depreciated and in Angular 5 a new module called HttpClientModule is introduced which comes under @angular/common/http package.
- Few new Router Life-cycle Events being added in Angular 5: In Angular 5 few new life cycle events being added to the router and those are:

```
ActivationStart, ActivationEnd, ChildActivationStart,
ChildActivationEnd, GuardsCheckStart, GuardsCheckEnd, ResolveStart
and ResolveEnd.
```

- Angular 5 supports TypeScript 2.3 version.
- Improved in faster Compiler support: A huge improvement made in an Angular compiler to make the development build faster. We can now take advantage of by running the below command in our development terminal window to make the build faster. ng serve/s — aot

Angular 6

- Released on April 2018
- This release is focused less on the underlying framework, and more on tool-chain and on making it easier to move quickly with angular in the future
- No major breaking changes

Material + CDK *All of the above are now version 6.0.0, minor and patch releases though are completely independent and can be changed based on a specific project.*

- Remove support for <*template*> tag and "<*ng-template*>" should be used.
- Registering provider: To register new service/provider, we import Service into module and then inject in provider array. e.g:

68

```
// app.module.ts
import {MyService} from './my-service';
...
providers: [...MyService]
...
```

But after this upgrade you will be able to add providedIn property in injectable decorator. e.g:

```
// MyService.ts
@Injectable({ providedIn: 'root'})
export class MyService{}
```

- The way ngModelChange event works: Let's understand this with output produced by older and this version:

```
// Angular 5:
<input [(ngModel)]='name' (ngModelChange)='onChange($event)' />
 onChange(value){ console.log(value); } // Would log updated value

<input #modelDir='ngModel' [(ngModel)]='name'
(ngModelChange)='onChange(modelDir)' />
 onChange(ngModel: NgModel){ console.log(ngModel.value); } // Would
log old value, not updated
// Angular 6:
 onChange(ngModel: NgModel){ console.log(ngModel.value); } // Would
log updated value
```

- CLI Changes: Two new commands have been introduced — ng update <package> * Analyse package.json and recommend updates to your application * 3rd parties can provide update

@angular/material. Behind the scene it add bit of necessary code and changes project where needed to add it the thing we just told it to add. * Now adding things like angular material, progressive web app, service workers & angular elements to your existing ng application will be easy.

- CLI + Material starter templates: Let angular create code snippet for your basic components. e.g: — Material Sidenav * ng generate @angular/material:material-nav — name=my-nav *Generate a starter template including a toolbar with app name and then the side navigation & it's also responsive* — Dashboard * ng generate @angular/material:material-dashboard — name=my-dashboard *Generates Dynamic list of cards* — Datatable * ng generate @angular/material:material-table — name=my-table *Generates Data Table with sorting, filtering & pagination*
- It uses angular.json instead of .angular-cli.json
- Support for multiple projects: Now in angular.json we can add multiple projects
- initial release of **Angular Elements** which gives us ability to use our angular components in other environments like a Vue.js application. Its potential is truly amazing but unfortunately this release only works for angular application, we need to wait for next release to wrap out angular component into custom element and use it with framework like **Vue.js**

Angular 7:

- Released on October 2018
- This is a major release and expanding to the entire platform including- — Core framework, — Angular Material, — CLI
- **CLI Prompts**: The CLI will now prompt users as when running common commands likeng new or ng add @angular/material with the intend of getting aid for building a new project using SCSS.
- Added a new interface — UrlSegment[] to CanLoad interface
- Added a new interface — DoBootstrap interface
- Angular 7 added a new compiler — Compatibility Compiler (ngcc)
- Introduce a new Pipe called — KeyValuePipe
- Angular 7 now supporting to TypeScript 2.9.
- Added a new elements features — enable Shadow DOM v1 and slots
- Added a new router features — warn if navigation triggered outside Angular zone

- Added a new ability to recover from malformed URLs
- Added a new compiler support dot (.) in import statements and also avoid a crash in ngc-wrapped
- Update compiler to flatten nested template fns

Angular 8:

- Releasing March/April 2019
- Being smaller, faster and easier to use and it will be making Angular developers life easier.
- Added Support for TypeScript 3.2
- Added a Navigation Type Available during Navigation in the Router
- Added pathParamsOrQueryParamsChange mode for runGuardsAndResolvers in the Router
- Allow passing state to routerLink Directives in the Router
- Allow passing state to NavigationExtras in the Router
- Restore the whole object when navigating back to a page managed by Angular Router
- Added support for SASS
- Resolve generated Sass/Less files to .css inputs
- Added Predicate function mode for runGuardsAndResolvers:- This option means guards and resolvers will ignore changes when a provided predicate function returns `false`. This supports use cases where an application needs to ignore some param updates but not others. For example, changing a sort param in the URL might need to be ignored, whereas changing the `project` param might require a re-run of guards and resolvers.
- Added functionality to mark a control and its descendant controls as touched: — add markAllAsTouched () to AbstractControl
- Added an ng-new command that builds the project with Bazel
- Use image based cache for windows BuildKite
- Export NumberValueAccessor & RangeValueAccessor directives
- Use shared DomElementSchemaRegistry instance for improve performance of platform-server(@angular/platform-server):- Right now the *ServerRendererFactory2* creates a new instance of the *DomElementSchemaRegistry* for each and every request, which is quite costly (for the Tour of Heroes SSR example this takes around **15%** of the overall execution time)
- Now the Performance Improvements on the core, more consistent about “typeof checks”: - When testing whether `value` is an object, use the ideal sequence of strictly not equal to `null` followed by `typeof value === ‘object’` consistently. Specifically, there’s no point in using double equal with `null` since `undefined` is ruled out by the `typeof` check. Also avoid the

68

hook)

- In the Compiler-CLI, expose ngtsc as a TscPlugin
- Restore whole object when navigating back to a page managed by Angular Router:- This feature adds a few capabilities. First, when a `popstate` event fires the value of `history.state` will be read and passed into `NavigationStart`. In the past, only the `navigationId` would be passed here. Additionally, `NavigationExtras` has a new public API called `state` which is any object that will be stored as a value in `history.state` on navigation. For example, the object `{foo: 'bar'}` will be written to `history.state` here: -`router.navigateByUrl('/simple', {state: {foo: 'bar'}});`

68

JavaScript Angular Angular2 Angular 4 Angular Cli

WRITTEN BY

Life N Shades

Travel | Code | Eat | youtube.com/lifenshades | instagram.com/lifenshades

Follow

More From Medium

Related reads

68

Event Emitters in Angular

Netanel Basal in...
Mar 7, 2017 · 4 mi... 2.5K

Related reads

RxJS: How to Observe an Object

Nicholas Jamieson...
Jun 14, 2018 · 2 mi... 1.7K

Related reads

Understanding provider scope in Angular

J Stepanyan in...
May 22, 2018 · 7 m... 1.4K