**Angular 5 New Features\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

The main focus behind the changes in angular 5 is to make the very framework smaller, faster and easier for a developer to work on. Many of the changes in angular 5 has happend under the hood but, let me mention the important ones over here:

**Progressive Web Application Support**

As you know, Google has been working on PWA or Progressive Web Application for quite a long time. Basically, PWA is a web application that can be “installed” on your system. It works offline when you don’t have an internet connection, leveraging data cached during your last interactions with the app.

Angular has a package called @angular/service-worker, which was in a different repository until angular 5 where the angular team put it to the main repository. I believe the intent is to make it out of beta soon!

Now, in case you are not familiar with service workers, you can think of them as small proxies in your browser. Once activated, it lets you to cache static assets and therefore, you don’t need to fetch them on every reload thus, improving the performances. In fact, you can even go offline, and your app can still respond!

**Build Optimizer**

The build optimizer is a tool which is included in the ng CLI. It makes your bundles smaller using the semantic understanding of your Angular application.

Basically, it keeps those parts of your application which are needed and mark these as pure. This improves the tree shaking process that is provided by the existing tools that finally let you to remove the additional or extra part of the application from the build.

The second thing the build optimizer does is to remove Angular decorators from your application’s runtime code. Decorators are used by the compiler, and aren’t needed at runtime and therefore, should be removed. Finally, all of these things let you to decrease the size of your JavaScript bundles, therefore letting you to boot your application faster.

**TypeScript Update**

Now, let us talk about TypeScript. Angular 5 supports the new TypeScript version 2.4. Because of this now, you can take advantages of all the new TypeScript features in your angular code bas. The two main new Typescript features:

* **String-based Enums:**TypeScript 2.4 now allows enum members to contain string initializers. For example:

*enum Colors {*

*Red = "RED",*

*Green = "GREEN",*

*Blue = "BLUE",*

*}*

* **Weak-Type-Detection:**Any type that contains nothing but a set of all-optional properties is considered to be *weak*. Like in the example below, options is a weal type containing all optional properties:

*interface Options{*

*color?: string,*

*font?: string,*

*size?: string*

*}*

So, with typescript 2.4, you can’t assign an object to a weak type when there’s no overlap in the properties of the two. For example:

*function sendMessage(options: Options) {*

*// ... }*

*const opts = {*

*background: “#334433",*

*font-size: ”24px” }*

*sendMessage(opts); // error*

**Compiler Improvements**

Angular works in two ways when it comes to compiling your typescript or angular codes i.e. JiT and Aot. In Jit the templates are compiled in the browser at runtime whereas in case of aot the the templates are compiled at build time.

Now, if you think about the later i.e. ahead of time compilation has lots of advantage i.e. this let your browser to download the pre-compiled version of your application. Therefore, it can render the application immediately, without waiting for the compilation of the app to finish. But this compiler was a bit slow before Angular 5.0, and as a result, most of us were using the JiT mode in development and the AoT mode only for production.

The main reason for the slow compilation was that every template change triggered a full compilation of the application. Now, in angular 5, the compiler has been improved. Behind the scene, the angular compiler works as a TypeScript transform that allows you to hook into the standard typescript compilation pipeline. Because of this the very angular compiler is now able to compile only those parts that is necessary. So, you will find significant improvements in incremental compilation.

In fact, I would suggest you guys to check out this on your own using the flag –aot in your ng serve command.

**Preserve Whitespaces**

Earlier all the tabs, newlines, and spaces in the templates were recreated and then, included in the build by the compiler. But, things have changed w.r.t. angular 5. You can now choose whether to preserve white spaces or not. Again, this will help you to reduce the build size of your bundles. You have two ways to set the preserve white spaces options:

* **Using Component (This will work on a component level)**



* **Using tsconfig.app.json (This will be set for the entire application)**



In general, the component-level specifications override application-wide specifications. In future, the team might set the very preserve whitespace feature to false by default.

**Router**

Now, let us talk about router. Additional events are added to the router life cycle. These events are ActivationStart andActivationEnd or ChildActivationStart and ChildActivationEnd. The entire life cycle of routes now look as:

***GuardsCheckStart > ChildActivationStart > ActivationStart > GuardsCheckEnd > ResolveStart > ResolveEnd > ActivationEnd > ChildActivationEnd***

For example:

import { ActivatedRoute } from '@angular/router';

export class AboutComponent implements OnInit {

constructor(private route: ActivatedRoute) {

this.route.params.subscribe(res => console.log(res.id));

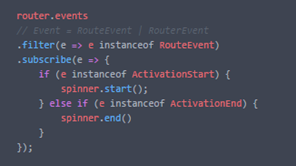
}

ngOnInit() {

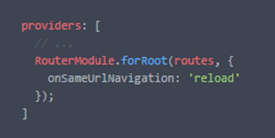
}

}

Additional events makes life of the developer easier by allowing them to track the router life cycle more efficiently and for example one can display a spinner while some children components are being loaded as shown in the code:



Also, it’s now possible to reload a page when the router receives a request to navigate to the same URL. Earlier, it was ignoring such a request, So, it was impossible to build a “refresh” button.



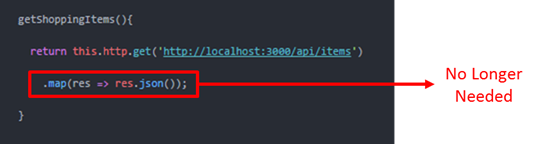
**HttpClient**

The old @angular/http module is now officially deprecated. In version 4.3, HttpClient was shipped in @angular/common which made it very easy to make web requests in Angular. So, given the positive response to the http module, they decided to go ahead with newer HttpClient module.

For using the new HttpClient, you’ll need to import HttpClientModule from @angular/common/http in each of your module.

https://qph.ec.quoracdn.net/main-qimg-ca15715cecace19525a6dbd9cc284b56

If you are replacing HttpModule with the newer HttpCLientmodule in your current codebase, you need to remove the map(res => res.json()) calls as JSON is an assumed as default and therefore, it does not require to be parsed explicitly.



**RxJS Update**

Angular 5 supports RxJS 5.5.2 or later. This has changed the way we used to import RxJS operators.



**New Pipes**

Angular team has introduced the new number, date, and currency pipes that increase standardization across browsers. Earlier, one needs to rely on the browser to provide number, date, and currency formatting using browser i18n APIs. So, inconsistencies between different browsers led to different bugs. All of this has been fixed in angular 5 with the new pipes which do not rely on i18n polyfills anymore.

If you don’t do anything when you upgrade, you’ll use the new pipes by default. In case, you want to use the previous implementation, you can do that as shown below:



**Forms Update**

You can now specify when validators should be executed in forms using option updateOn. This feature was there in angular js (1.x) but was not added into the newer angular until angular 5. The following update options can now be used in Angular 5 forms:

* *change*: change is the default mode. By using this update option the form / form control is updated after every single change.
* *blur*: the blur change mode is only updated the from values / validity status after a form control lost the focus.
* *submit*: updates are only done after form submit.

**Example - Template Driven Forms:**



**Other Changes:**

* Export Component and Directives with different Alias using exportAs



* Zones has been made Faster by Default
* Also, it is possible to bypass the Zones for Better Performance

https://qph.ec.quoracdn.net/main-qimg-961846ba6b1281beadd9beda25b4d85c