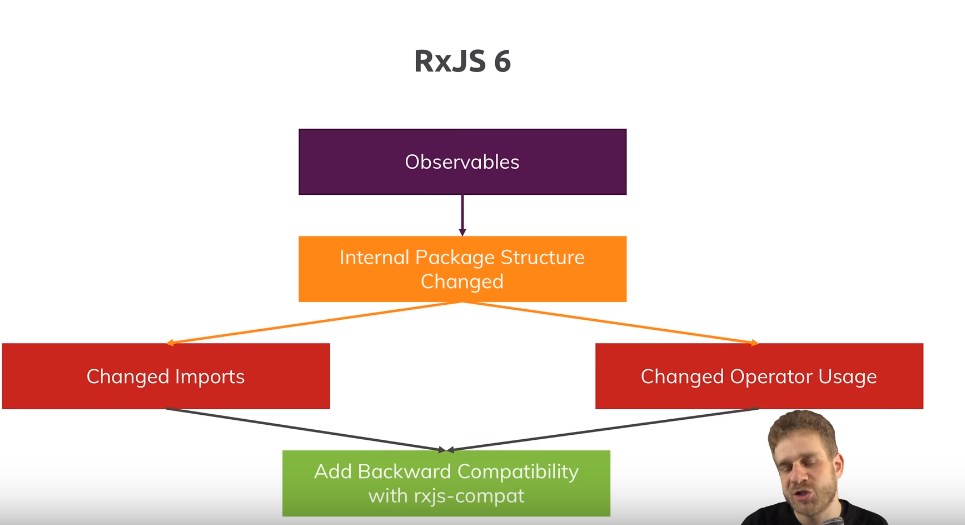
Angular 6 does not include any breaking changes, RXJS 6 which is core dependency of angular does have breaking changes. But its easy to fix.

Lets strat with breaking change. Angular 6 uses RXJs version 6, RXJS is library which is used for observables and observable operators. Angular uses that library a lot behind the scenes. You typically also use that library in angular applications for ex while using HttpClient, then you probably laos use some rxjs operators like map. With rxjs 6, internal package structute of rxjs6 is changed to ensure that future bundles are smaller and import statements are easier.

So you have to adjust your import statements and ay you use operators also changed. How ever there is a package tat you can install to still have your old code work with new rxjs version, i.e to get that backward compatability. That packge is called **rxjs-compat.**  So lets see how we can use this package and how we can update our code without using package.



For this we have project. This project is updte to angular 6 version. We are also using rxjsversion 6 and that is version that breaks your code. Now in app.component,w e are sending a http request, then we are using some operators before subscribing to observable. This code used to work before but now with angular 6 , it fails. Code-

ngOnInit() {

this.http

.get<{ id: number; userId: number; title: string; body: string }[]>(

"https://jsonplaceholder.typicode.com/posts"

)

.map(data => {

return data.map(el => ({ title: el.title, body: el.body }));

})

// .mergeMap(transformedData => transformedData)

.catch(error => {

return Observable.throw('Something went wrong!');

})

.subscribe((transformedData: {title: string, body: string}[]) => {

this.blogPosts = transformedData;

});

}

We get this error-

**ERROR in node\_modules/rxjs/Observable.d.ts(1,15): error TS2307: Cannot find module 'rxjs-compat/Observable'.**

**node\_modules/rxjs/Rx.d.ts(1,15): error TS2307: Cannot find module 'rxjs-compat'.**

**src/app/app.component.ts(4,10): error TS2305: Module '"D:/Angular 6 code/ng6-update-example-project-starting-code-ng6/node\_modules/rxjs/Observable"' has no exported member 'Observable'.**

**src/app/app.component.ts(38,8): error TS2339: Property 'map' does not exist on type 'Observable<{ id: number; userId: number; title: string; body: string; }[]>'.**

So we get the error that it cannot find that rxjs- compat module. So it is already referring to this compatibility module, that we talked about. Now lets install that compatibility packge-

**npm install --save rxjs-compat**

now if you run **ng serve**, your project will run.

Now lets say we want to upgrade without using that package, althrough that package is very light. First uninstall that package. Run-

**npm uninstall –-save rxjs-compat**

lets see how we need to adjust our code to work again. We have to adjust some imports.

We have this import-

import { Observable } from "rxjs/Observable";

now anything that was exported from rxjs/something (like Observable, Subject,Subscrption)will be exported from rxjs. Now all these operators are also imported differently –

import 'rxjs/Rx'; // This or ...

// all the below imports

import "rxjs/add/observable/of";

import "rxjs/add/observable/throw";

import "rxjs/add/operator/map";

import "rxjs/add/operator/mergeMap";

first import was importing everything so we can get rid of that. We import operators like this-

import { map, catchError } from 'rxjs/operators';

so all rxjs operaors are imported like this. Catch is renamed to catchError, this is because of conflict with js reserved keyword.so we operators like functions.

Import of Observable creation methods is also changed. Right now they are imported like this-

import "rxjs/add/observable/of";

import "rxjs/add/observable/throw";

they are also imported from rxjs. Like this-

import { Observable, of, throwError } from 'rxjs';

throw is also renamed to throwError. Now with all these import updates, question is, how do we change our code(where we use them). Right now our code is-

ngOnInit() {

this.http

.get<{ id: number; userId: number; title: string; body: string }[]>(

"https://jsonplaceholder.typicode.com/posts"

)

.map(data => {

return data.map(el => ({ title: el.title, body: el.body }));

})

// .mergeMap(transformedData => transformedData)

.catch(error => {

return Observable.throw('Something went wrong!');

})

.subscribe((transformedData: {title: string, body: string}[]) => {

this.blogPosts = transformedData;

});

}

Instead of chaining operators to observable like this, now we have new method provded by rxjs i.e pipe.pipe takes infineite amount of arguments, each argument is operator that you want to apply to your observable. We dnt use Observable.throw, we use throwError. Now we use them like this-

ngOnInit() {

this.http

.get<{ id: number; userId: number; title: string; body: string }[]>("https://jsonplaceholder.typicode.com/posts")

.pipe(map(data => {

return data.map(el => ({ title: el.title, body: el.body }));

}), catchError (error => {

return throwError("Something went wrong!");

})) // .mergeMap(transformedData => transformedData)

.subscribe((transformedData: { title: string; body: string }[]) => {

this.blogPosts = transformedData;

});

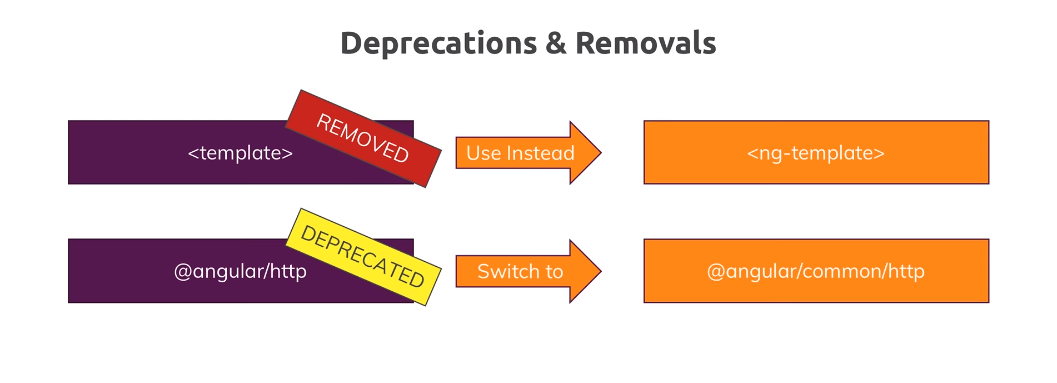
}

With this change our code compiles without any error. Note now are not using any compatibility package.

This is not all that angular 6 brings, but it is ony breaking change. Now in angular 6, we also got deprecations and removals.

First of call template is removed. This does not mean template property of your component decorator, it means if in html you are using <template>, you need to use <ng-template>.

Regarding http client, old one is still fine to use, sill safe nothing wrong with it. But is deprecated. Actually it was already deprecated with angular 5. Now you should slowly switch to new http clinet which is imported from @angular/common/http. With angular 7, old one will be removed too. In code we are already using new http client.



So this was second change deprecations and removals.

There is also a change regarding services. In code we have alert service, if we want to use same instance of that service throughout entire app, we will provide that service in provider array of app.module. but in angular 6 we can do it on alternate way.

import { Injectable } from "@angular/core";

@Injectable({providedIn: 'root'})

export class AlertService {

showAlert() {

alert('Hello!');

}

}

Angular elemnts is addition shipping with angular. Angular elements allow us to transform our angular components to native web components. Native web components are not conncted with angular, they use vanilla js and they allow us to create custom html elements. theoratically we can use these lement in any web application we create , even if its is not using angular, though with time being with angular 6, anguar elements allow us to create custom elemnts that we can use only inside angular app. Question is, why woud we use this feature then, we already got angular components that we can use inside angular app, why would we create html elemnts, that work there too? Reason is dynamic content. If you got some content management system from which you are fetching content that you wanna render, lets say you are fetching prepared code because you got some editors on backend that are writing html code and you want to give these people a chance of using your angular component in html code they are preparing. For details see video by max.

Lecture skipped, here max is talking bout angular elments.

Angular 6 also include a new renderer ivy. What this has to do with you? First fo all , in the end it is a renderer that has a target of drasitically reducing bundle sizes. Hello word bi=udle should be down to 3kb. It is still in early pahse, so it is not default in angular application and its very unstable. So play around with it but don’t use. It will come out in angular7.

In this part if video max showed, how wecan enable setting to use this rendered. But if we compile our code , it will fail. This is because it is early release now.

Now  **ng update** will update you angur project and its dependencies to angular 6. So yor update straetegy should be this-

