Javascript,Typescript and angular

Table of Contents

[JavaScript 3](#_Toc72039931)

[JavaScript 3](#_Toc72039932)

[JSON 3](#_Toc72039933)

[Closure : 5](#_Toc72039934)

[ECMA 6/ES 6 : Javascript version 6](#_Toc72039935)

[let and constant 7](#_Toc72039936)

[Destructring 8](#_Toc72039937)

[spread operator for cloning 9](#_Toc72039938)

[Template Literals 9](#_Toc72039939)

[\*\*\*Promise : Promise API ==> Asynchronous code execution 9](#_Toc72039940)

[Export and Import : 10](#_Toc72039941)

[Transpilers/Compilers 11](#_Toc72039942)

[Node.js 12](#_Toc72039943)

[webpack 13](#_Toc72039944)

[TypeScript datatypes: 14](#_Toc72039945)

[Metadata 15](#_Toc72039946)

[Angular SetUp 17](#_Toc72039947)

[SPA frameworks and angular 19](#_Toc72039948)

[Angular building blocks 20](#_Toc72039949)

[Template : 22](#_Toc72039950)

[Component 22](#_Toc72039951)

[Services 26](#_Toc72039952)

[Module 26](#_Toc72039953)

[Angular Flow 28](#_Toc72039954)

[Data Binding 31](#_Toc72039955)

[@Input() ==> Property from parent to child, @Output() ==> EventEmitter 34](#_Toc72039956)

[@output 35](#_Toc72039957)

[@ViewChild 37](#_Toc72039958)

[Angular Pipes 39](#_Toc72039959)

[Angular Routing 41](#_Toc72039960)

[Router navigation 45](#_Toc72039961)

[Angular Services 45](#_Toc72039962)

[UNIT and e2e(EndToEnd) TESTING 56](#_Toc72039963)

## JavaScript

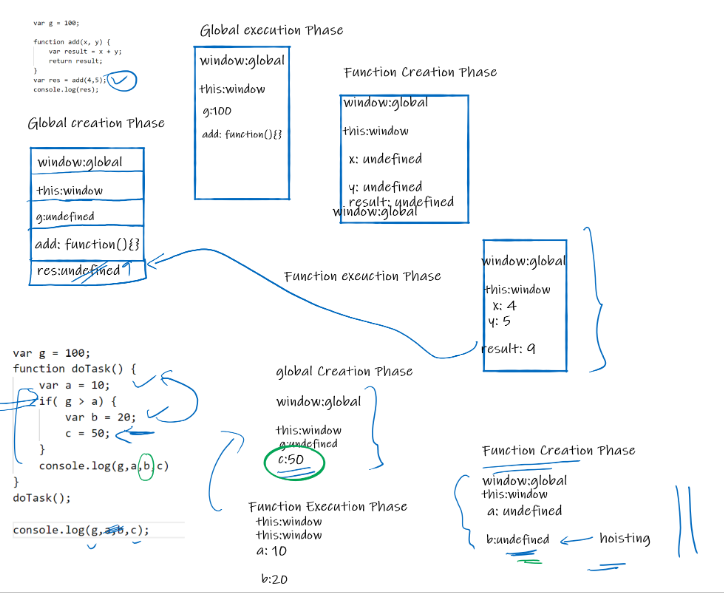
-------------------------------------------------------

JavaScript: Scripting language, loosely typed

Runs on JS engine [developed using C++] ==> V8 [google], SpiderMonkey, Chakra, NashHorn, ...

========================

JSON ==> JavaScript Object Notation {} ==> usually used to represent state ==> carriers of data

  
HOF :Higher Order Function

***functional style of programming where func accepts a function or returns a function***

-=> functions which can be used on any type of data [ objects, prmitives,..]

Example: filter, iterate, map,...

Functional style of Programming uses High Order Functions:

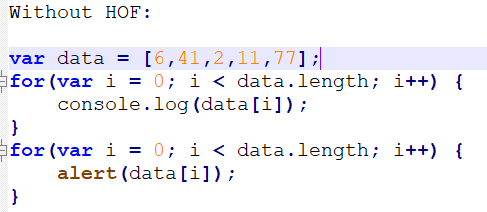
1) functions which accept function as argument

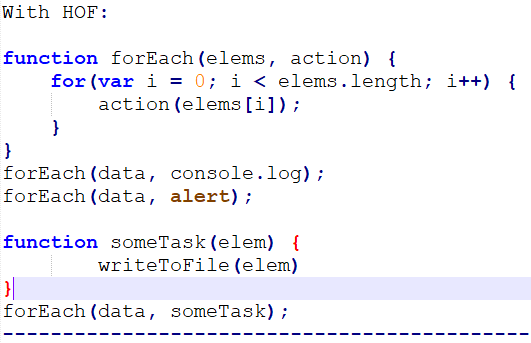
2) function which returns a function

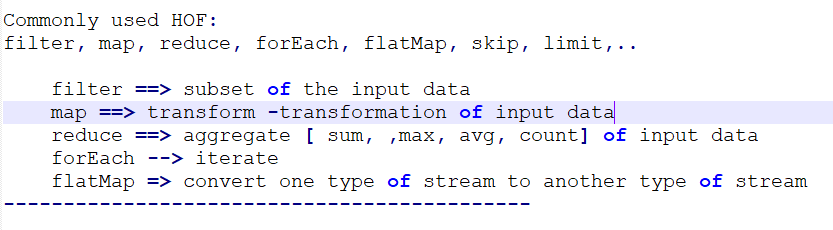
==> treat function as first-class members [ primitive , object]

***Extentensively used in typescript,reactjs and other libarariries***

Example :





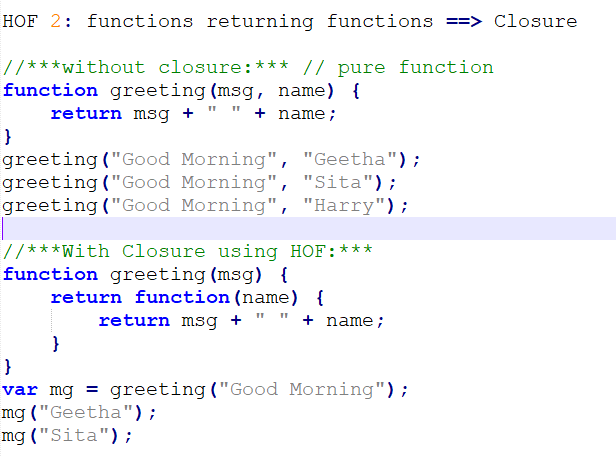




HOF : <https://rxmarbles.com/>

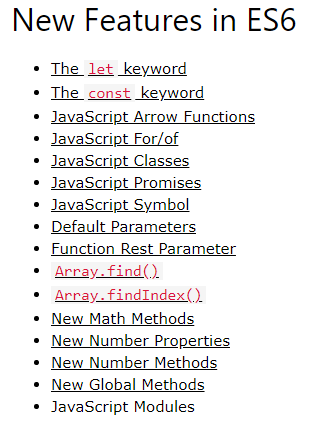
### Closure :

***Closure is a concept wherein inner function has an access to members of outer functions***



### ECMA 6/ES 6 : Javascript version

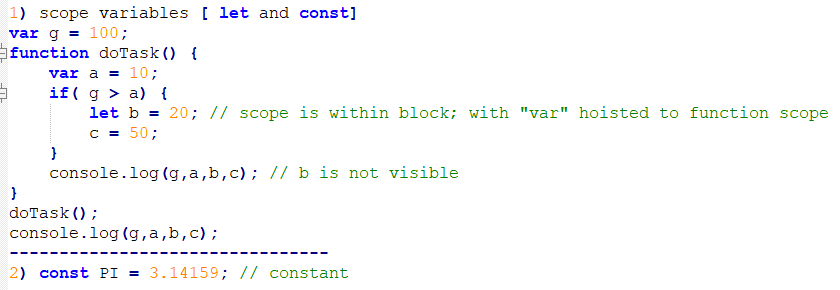
**ES6** stands for ECMAScript 6. ECMAScript was created to standardize JavaScript, and **ES6** is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015

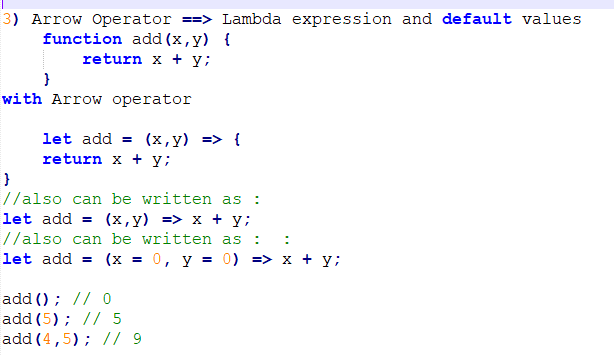


<https://www.w3schools.com/js/js_es6.asp#mark_let>

### let and constant

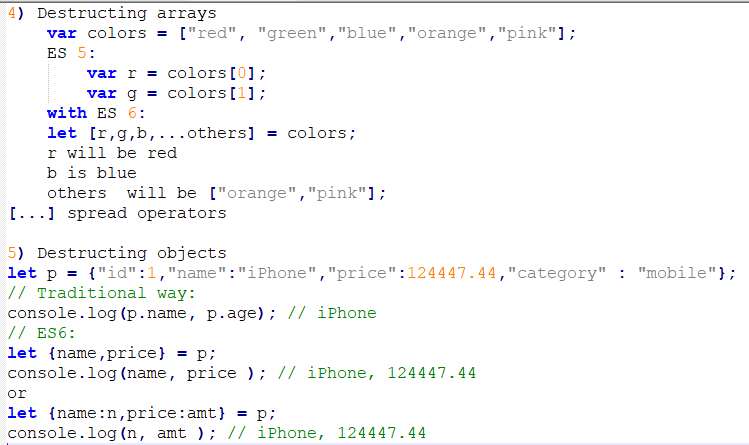
**let** allows you to declare variables that are limited to the scope of a block statement, or expression on which it is used, unlike the var keyword, which declares a variable globally, or locally to an entire function regardless of block scope. **Constants** are similar to let variables, except that the value cannot be changed.



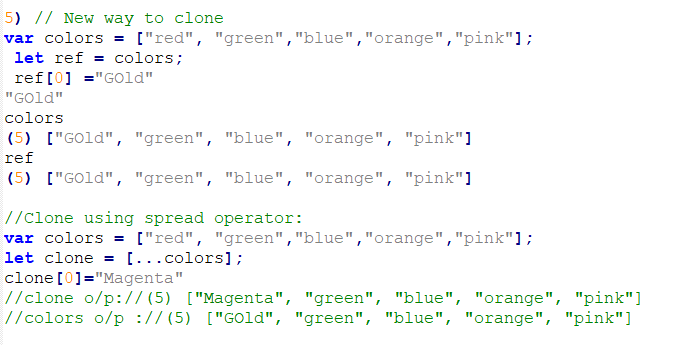


Check features and browser support : <https://caniuse.com/>

### Destructring



### spread operator for cloning

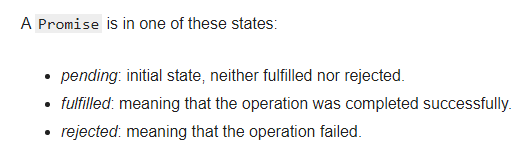


### Template Literals

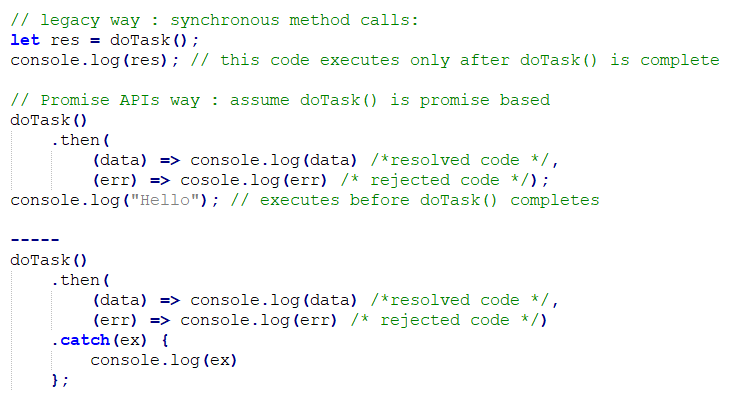


### \*\*\*Promise : Promise API ==> Asynchronous code execution

Promise: The **Promise** object represents the eventual completion (or failure) of an asynchronous operation and its resulting value

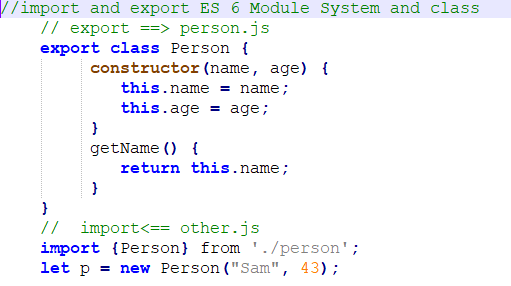


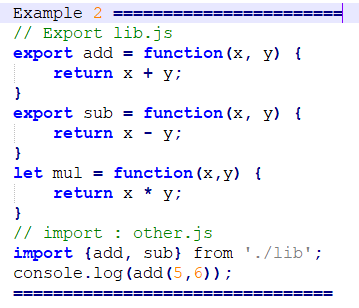




### Export and Import :

*The****export****statement is used when creating JavaScript modules to export live bindings to functions, objects, or primitive values from the module so they can be used by other programs with the*[***import***](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import)*statement. Bindings that are exported can still be modified locally; when imported, although they can only be read by the importing module the value updates whenever it is updated by the exporting module.*

**

**

## [Transpilers](https://scotch.io/tutorials/javascript-transpilers-what-they-are-why-we-need-them#toc-in-defense-of-transpilers)/Compilers

[Transpilers](https://scotch.io/tutorials/javascript-transpilers-what-they-are-why-we-need-them#toc-in-defense-of-transpilers) read CoffeeScript, TypeScript, and ES2015, and converts to JavaScript guaranteed to work anywhere.

Transcompiler , Transpiler ==> Babel, Traucer  
***ES 5 is supported by almost all current browsers***  
Development can happen in ES6, we use Transcompiler to convert to lower version ==> ES5

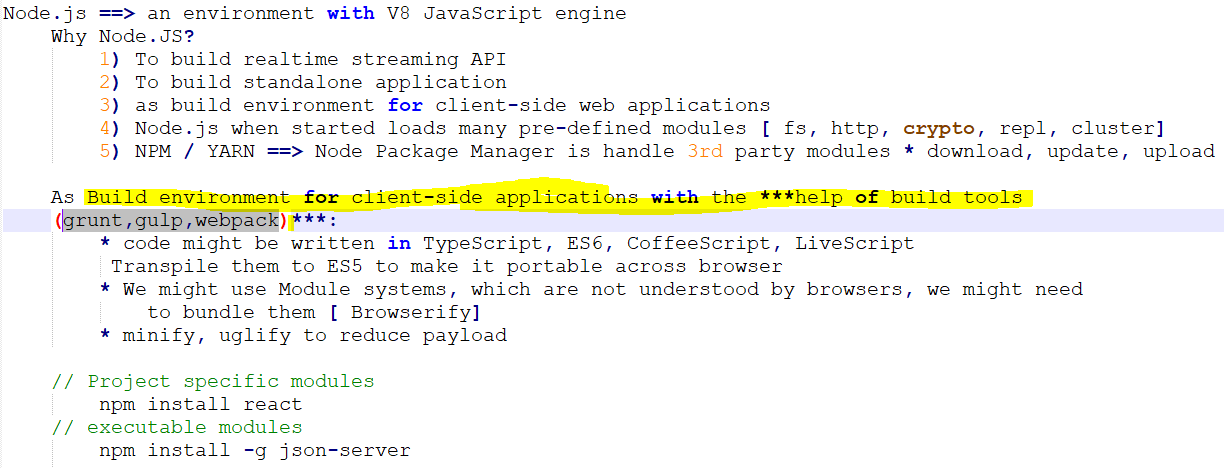
ESNext, TypeScript, CoffeeScript, DART, LiveScript are used only in development stage

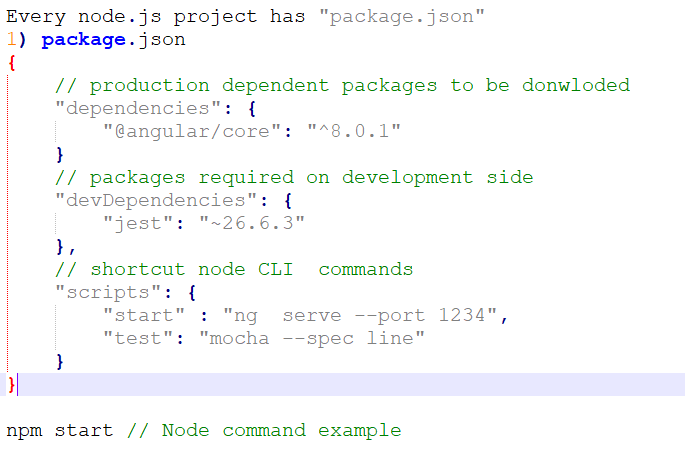
these are transcompiled into JS for production.

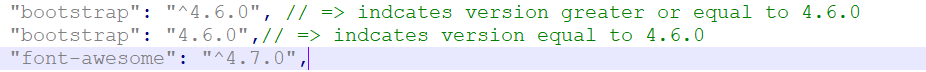
*Babel is a free and open-source JavaScript transcompiler that is mainly used to convert ECMAScript 2015+ code into a backwards compatible version of JavaScript however webpack is widely used*

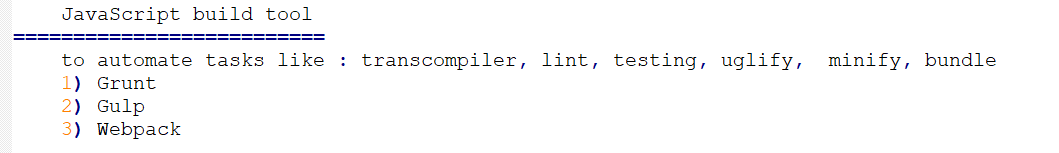
### 

### Node.js



**

**

**

***Grunt***is a JavaScript task runner, a tool used to automatically perform frequent tasks such as minification, compilation, unit testing, and linting.

webpack *:*

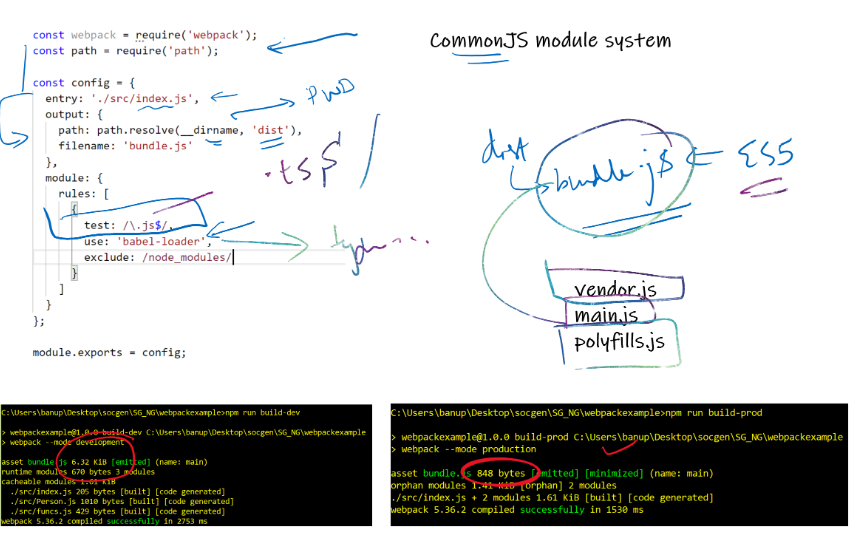
\*\*\*Webpack is a popular module bundler, a tool for bundling application source code in convenient chunks and for loading that code from a server into a browser

\*\*\*Webpack behind the scenes to transpile TypeScript to JavaScript,transform Sass files to CSS,

and many other tasks

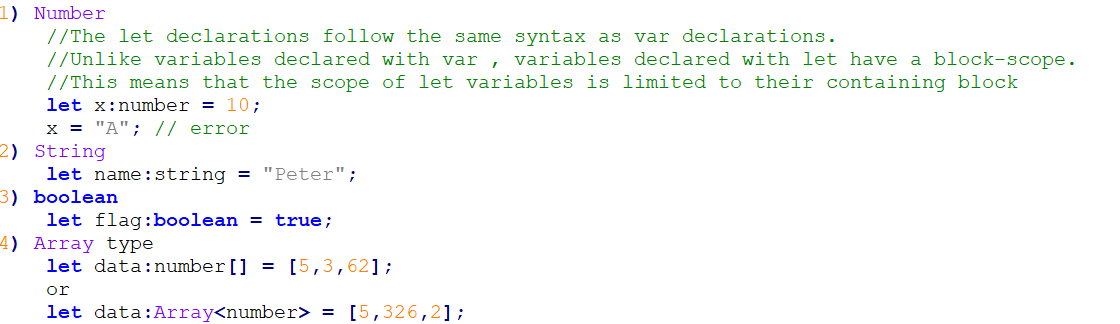
You determine what Webpack does and how it does it with a JavaScript configuration file, webpack.config.js

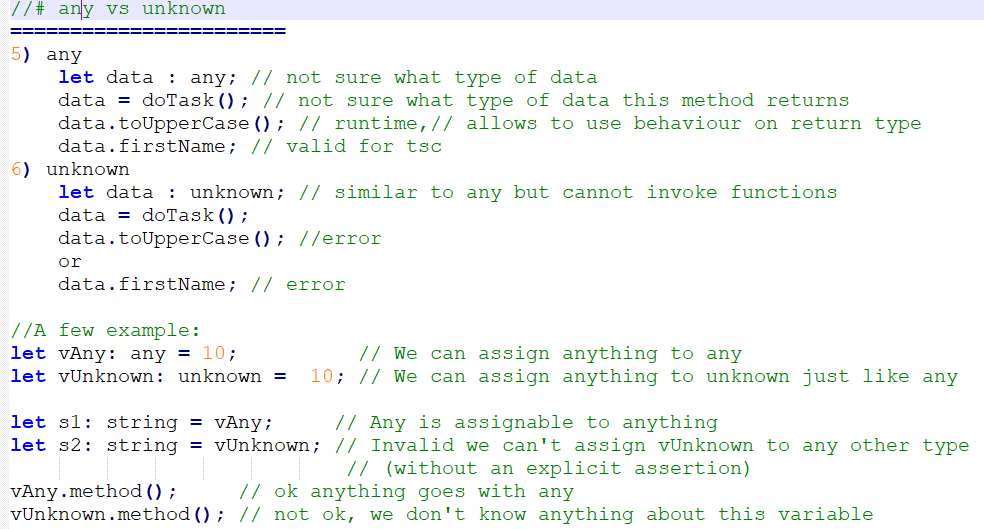
[*https://v2.angular.io/docs/ts/latest/guide/webpack.html*](https://v2.angular.io/docs/ts/latest/guide/webpack.html)

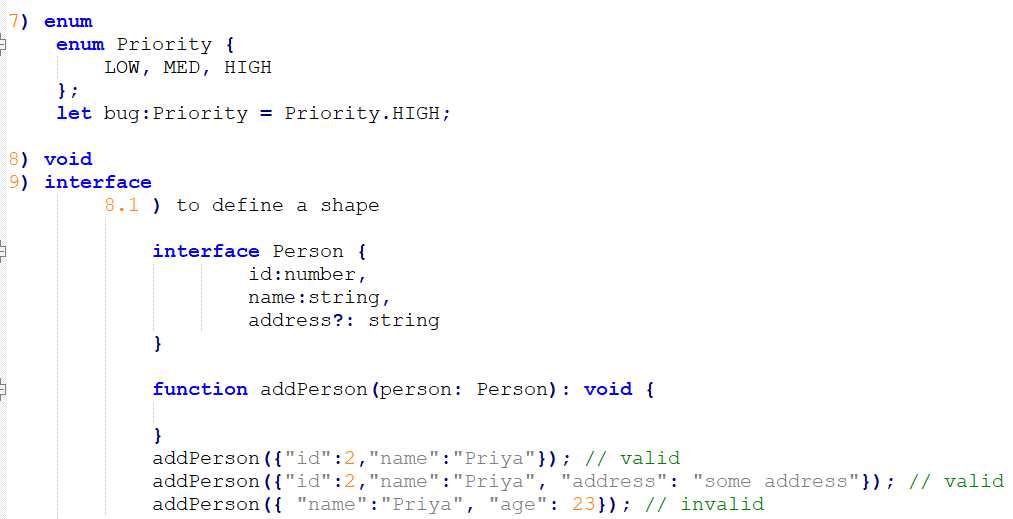
**

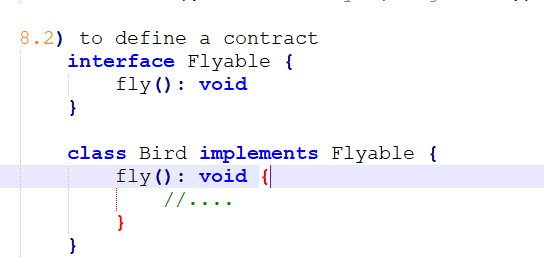
**

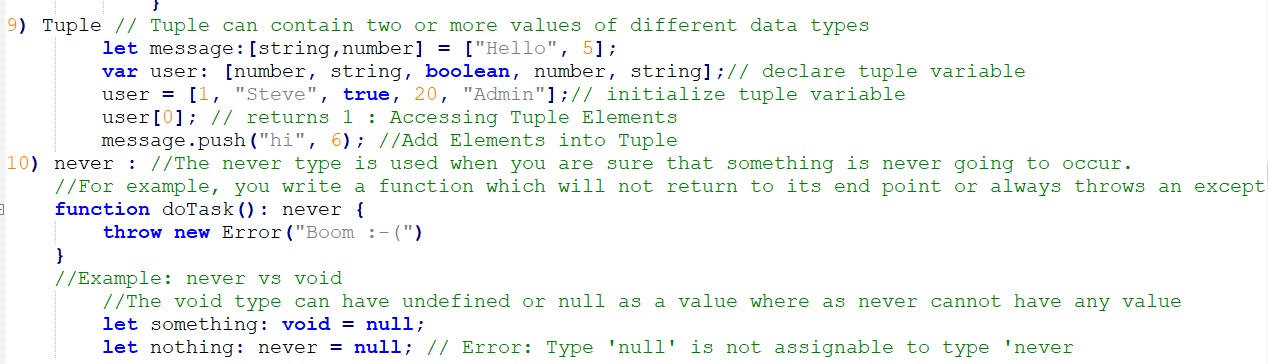
### TypeScript datatypes:

**

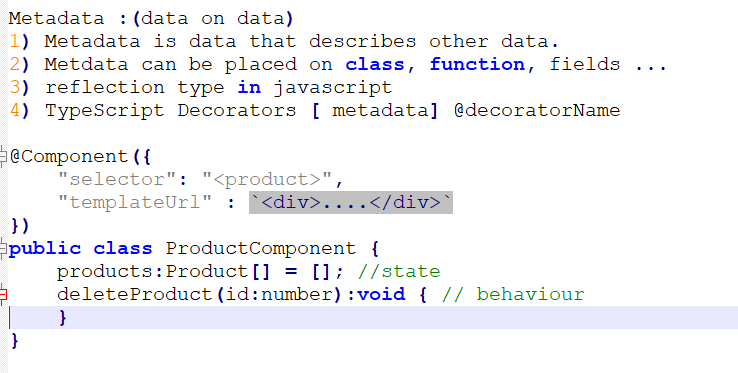
**

**

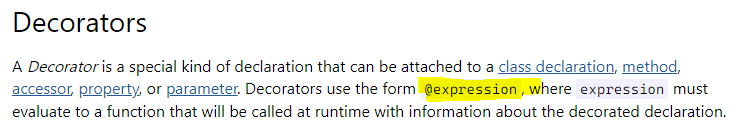
**

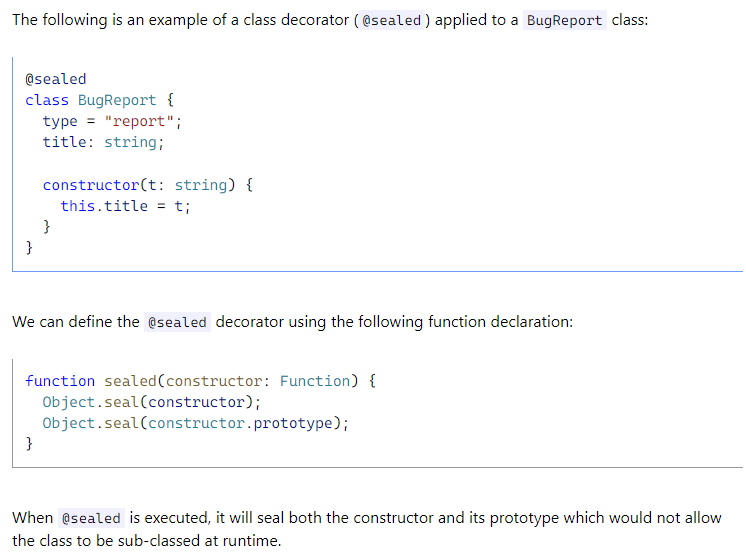
**

### Metadata

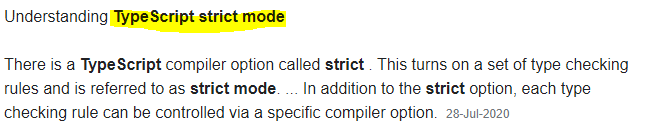
**

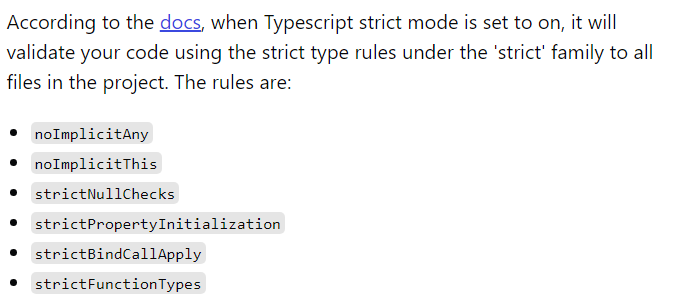
#### Decoraters :

**

**

*\*\*\* In typescript, you attach metadata by using @Component, @NgModule, @Injectable or @Directive \*\*\**

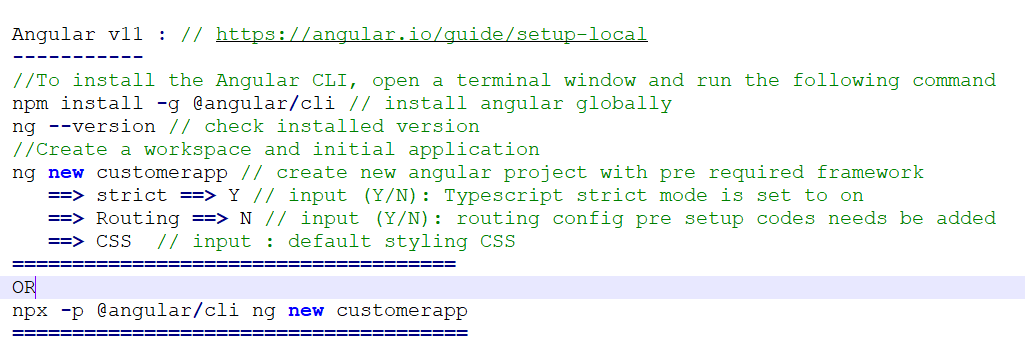
**

**

### 

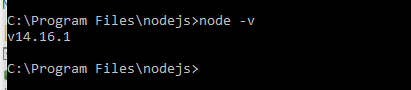
### Angular SetUp

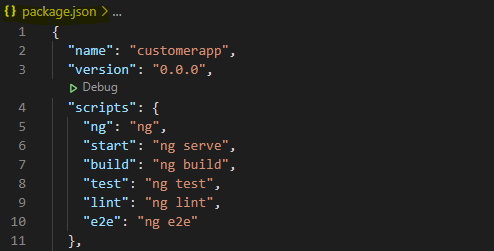
*commands*

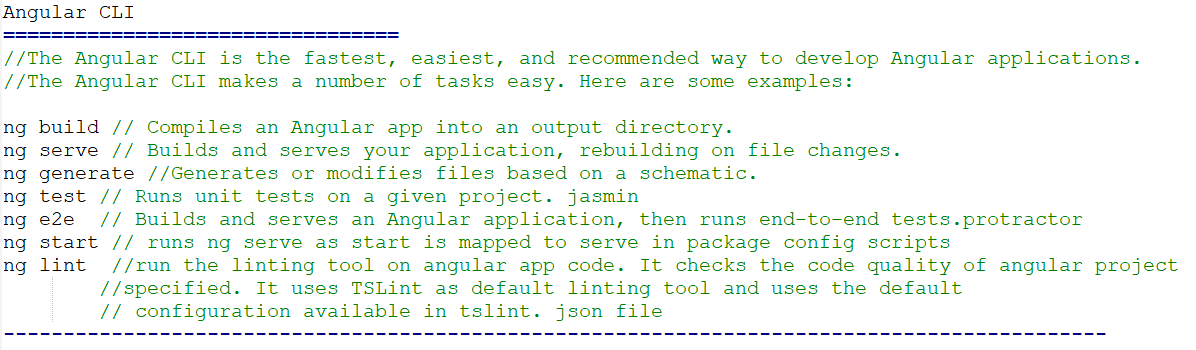
*npm *

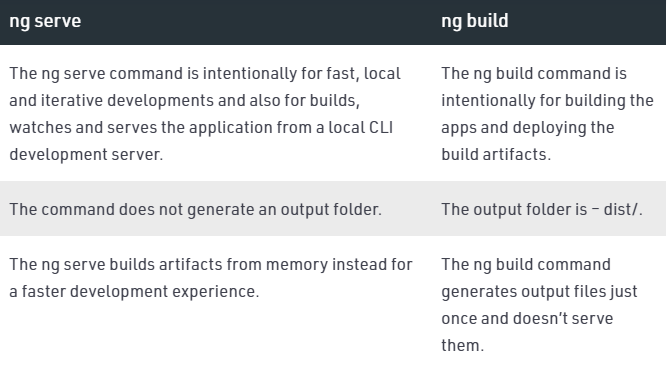
[Angular - Setting up the local environment and workspace](https://angular.io/guide/setup-local)

*Run commands in vscode terminal / in node cli*

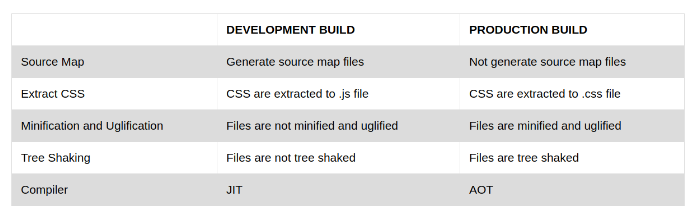
**

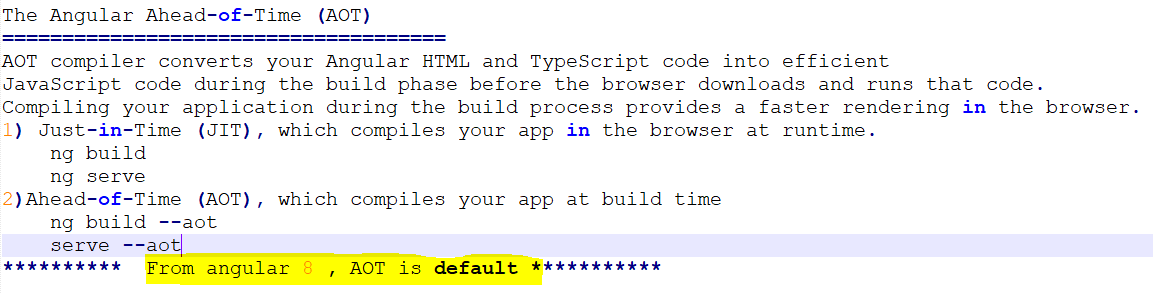


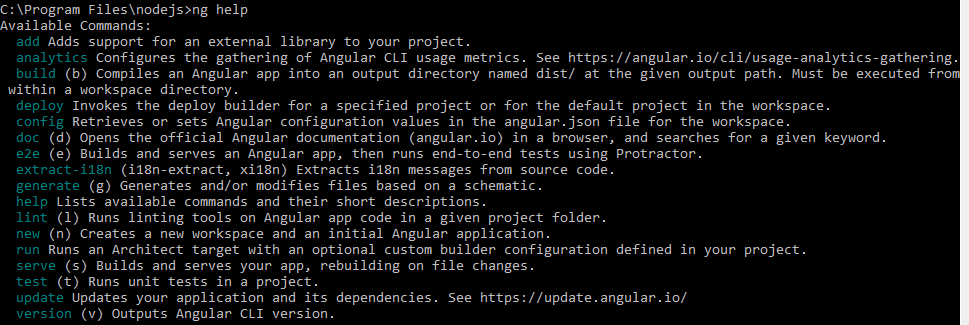


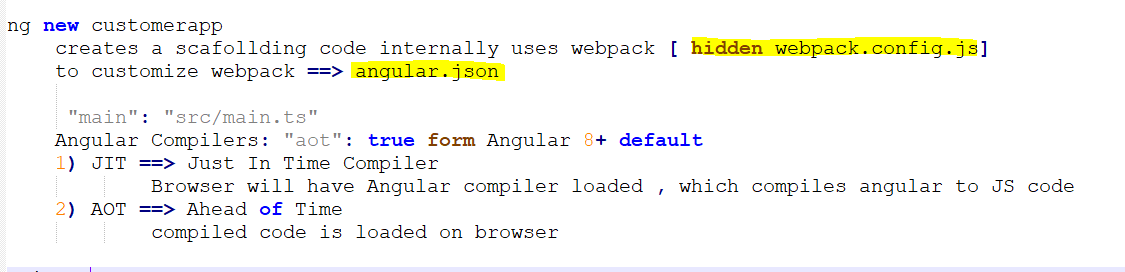


ng build --prod



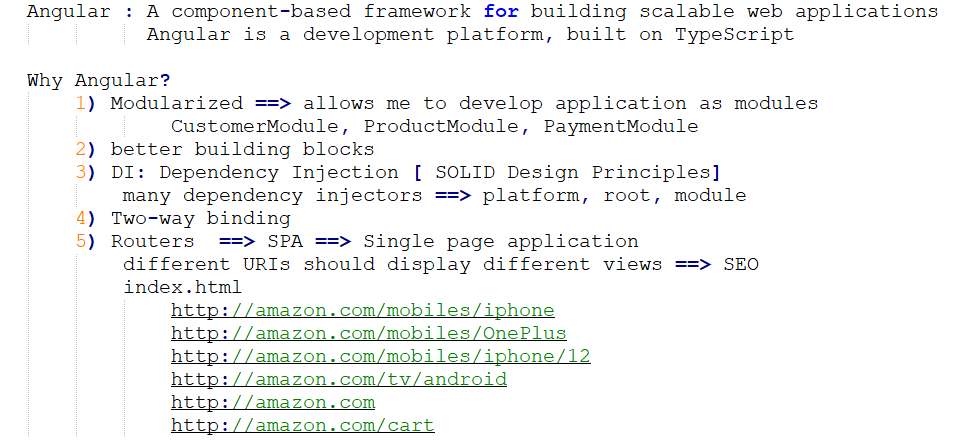


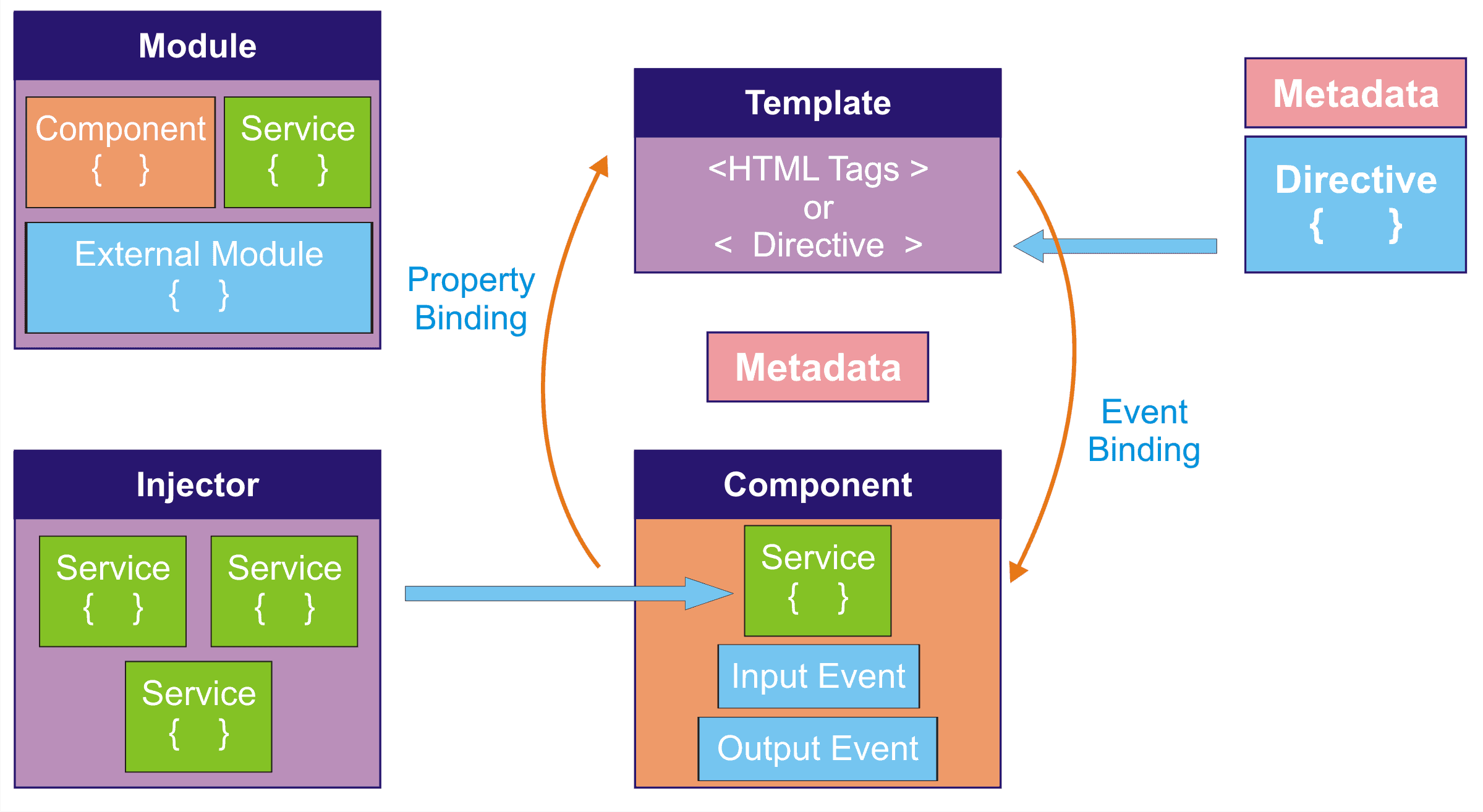


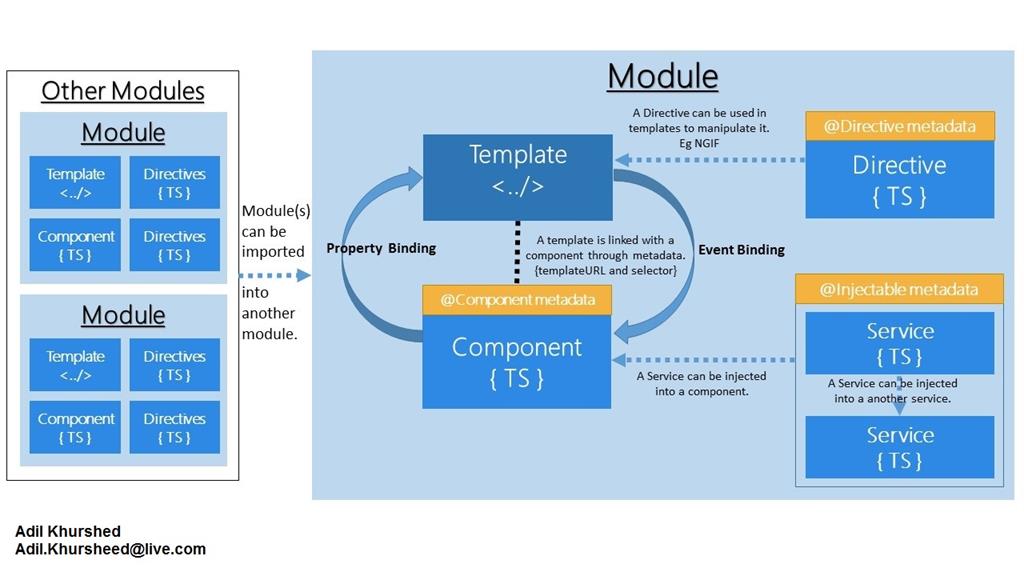


### SPA frameworks and angular

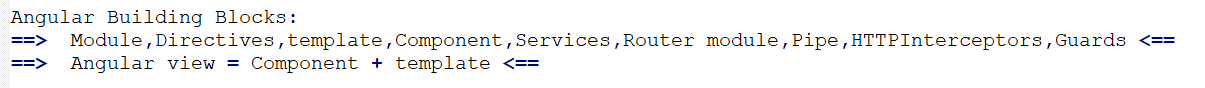
==> Complete Solution Frameworks: Backbone, Ember.js, CanJS, Angular.JS [ 1.x ] , Angular [ 2+], Vue



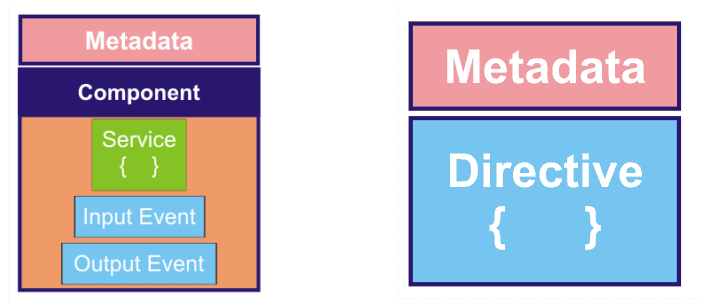
Angular building blocks



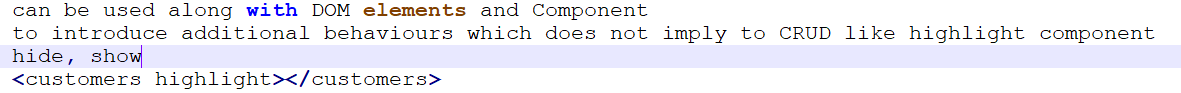
Angular Architecture is divided into different parts like Component, Template, Directive, Service, and Module.



Metadata is commonly defined as Data about Data. In particular to angular, here metadata is the Decorator.In typescript, you attach metadata by using @Component, @NgModule, @Injectable or @Directive

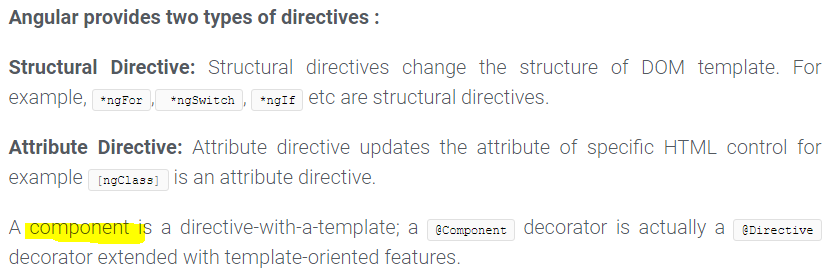




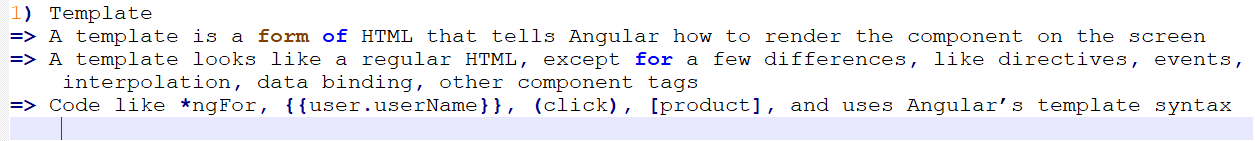


Angular Templates are dynamic. When Angular renders them, it transforms the DOM according to the instructions given by the directives. *A Directive is a class with a @Direcitve decorator*

**



### Template :



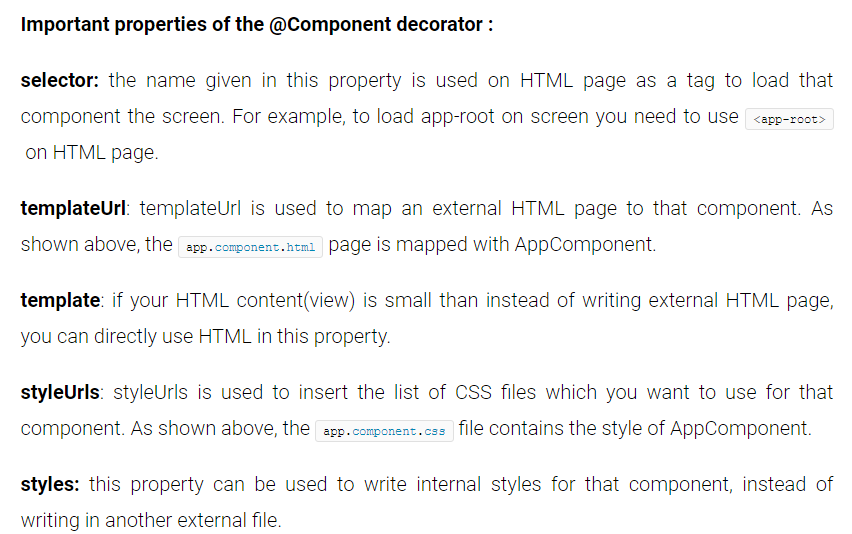


### Component



|  |  |
| --- | --- |
| Component Tree | Code Snippet of Component |

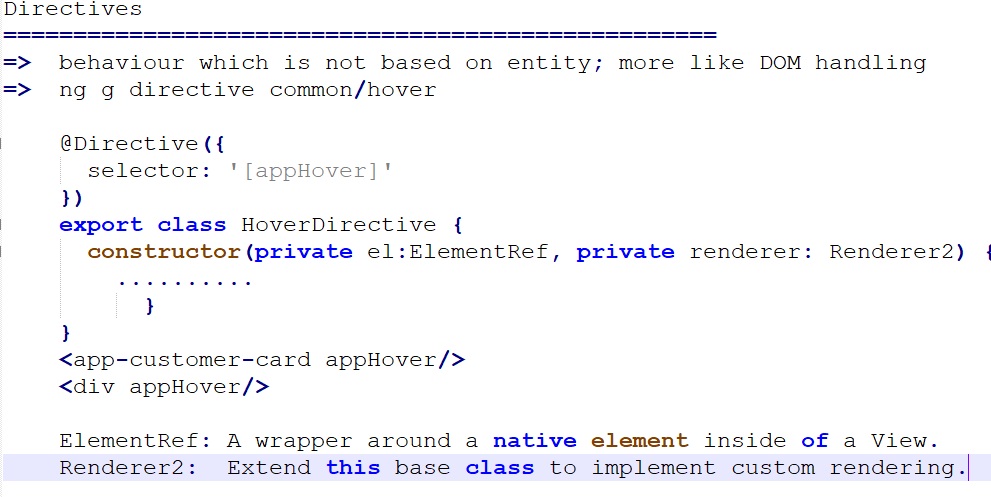


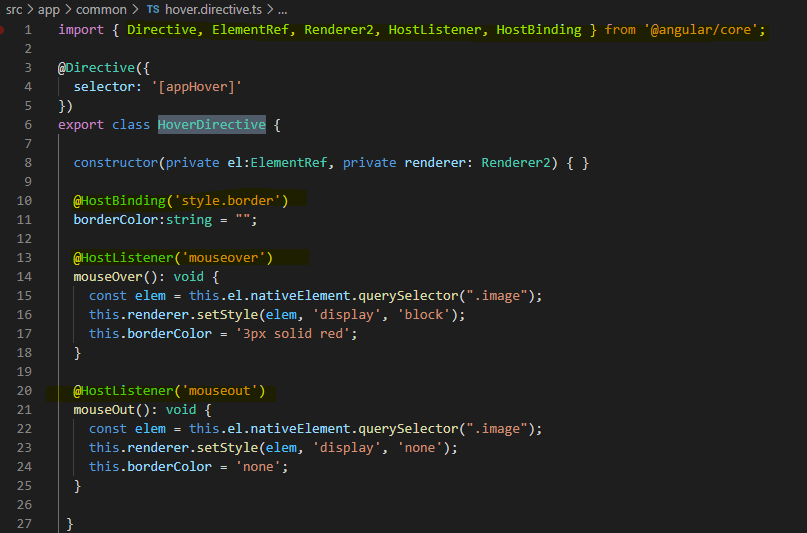


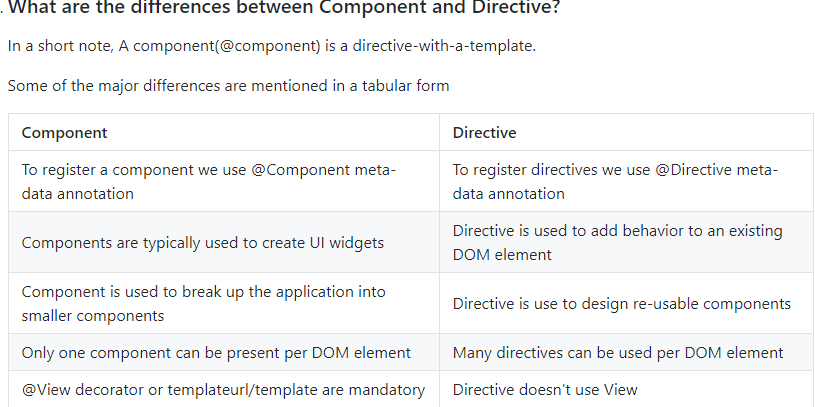
|  |  |
| --- | --- |
| Angular Architecture and Design Guidelines | Angular Architecture and Design Guidelines |
| Angular Architecture and Design Guidelines | Angular Architecture and Design Guidelines |

Angular Component Interaction heavily**relies on decorators** like ***@Input*** and **@OutPu**t decorator for Child to Parent and Parent to child component interaction.

Directive Example:

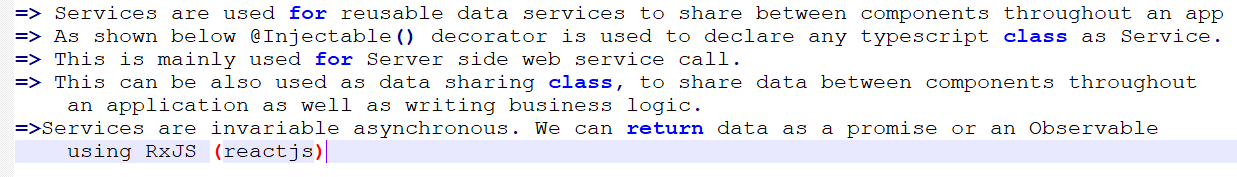


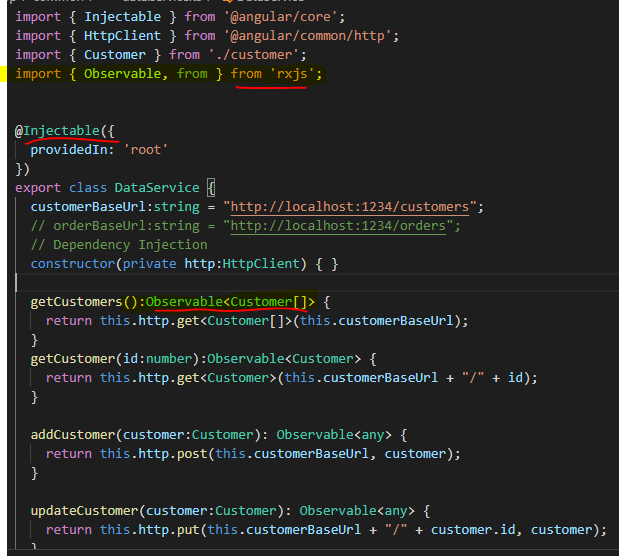




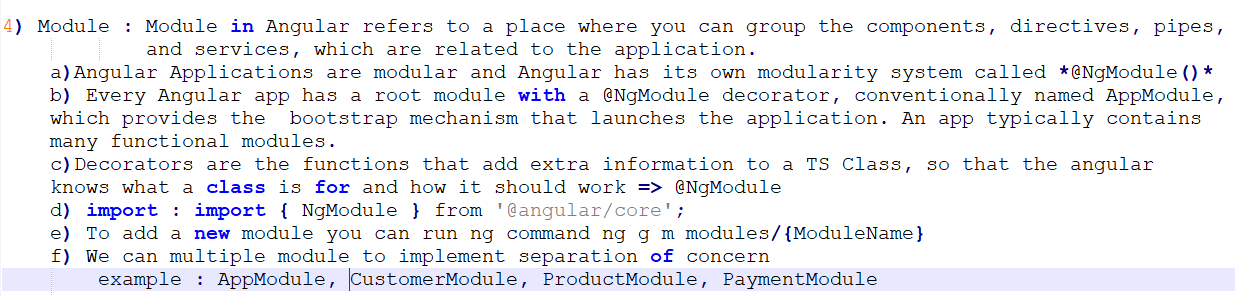


### Services

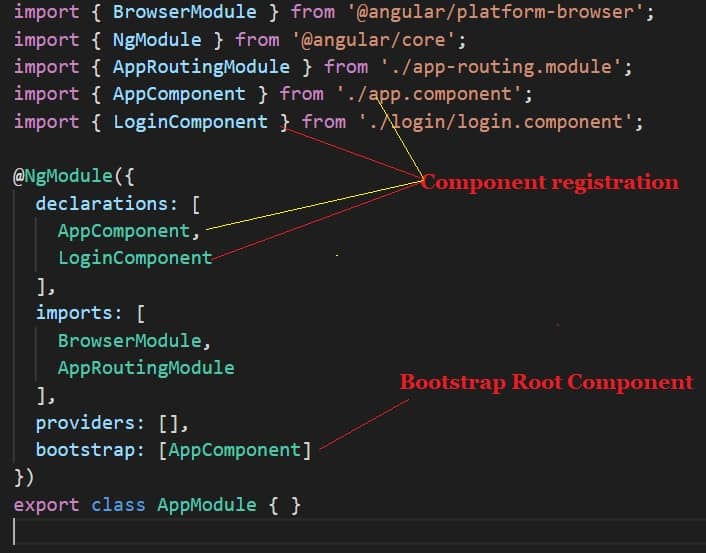
code which has business logic, API calls ==> http   
@Injectable 



### Module







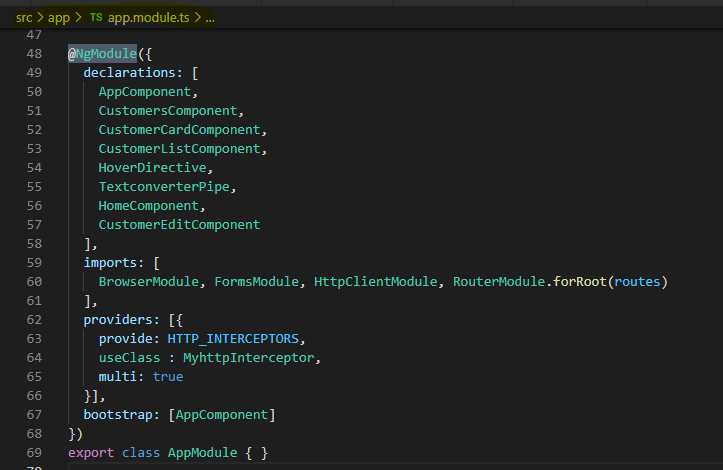
**declarations:** declaration property contains a list of the component which you define for this module. When a view is displayed on the screen of any application, it is the mixture of Component and Template (Directives + Pipes), which are constituents of declarations.

**imports:** if you want to use external modules(libraries) like FormsModule, RouterModule etc then you need to add that module name here. E

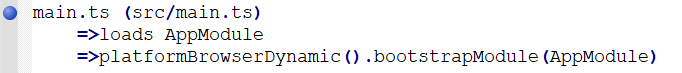
**exports:** if you want to use component or directive of this module into another module then you need to add that component or directive name here. generally, root module or AppModule does not contain export property.xternal module can be an inbuilt angular module or user-defined module.

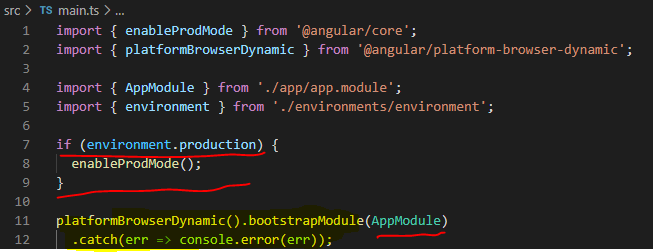
**providers:** whatever service you create in that module you need to provide it here. The injector will get to know about the service from here.

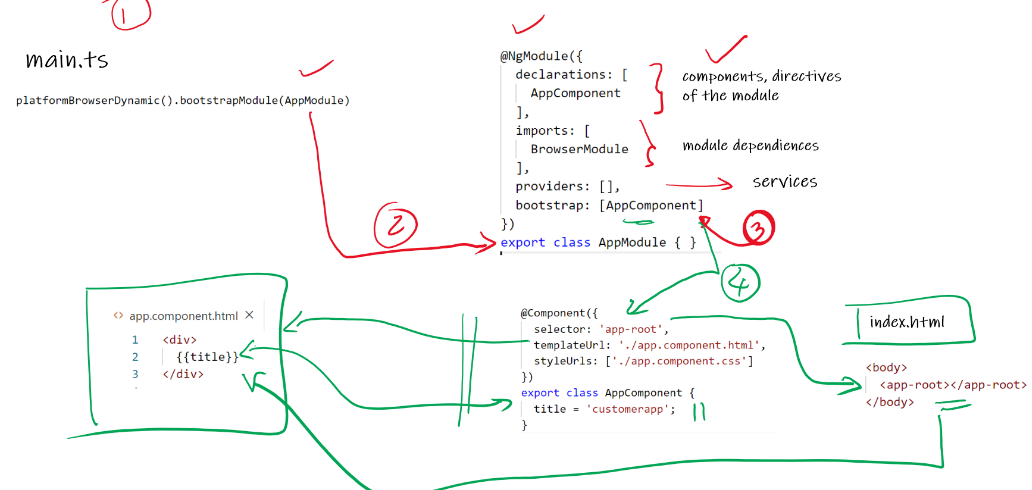
**bootstrap:** you need to provide the name of the component which you want to load when the application loaded on the browser. Generally, it is the name of the root component. Furthermore, only root module contains this bootstrap property.



### Angular Flow

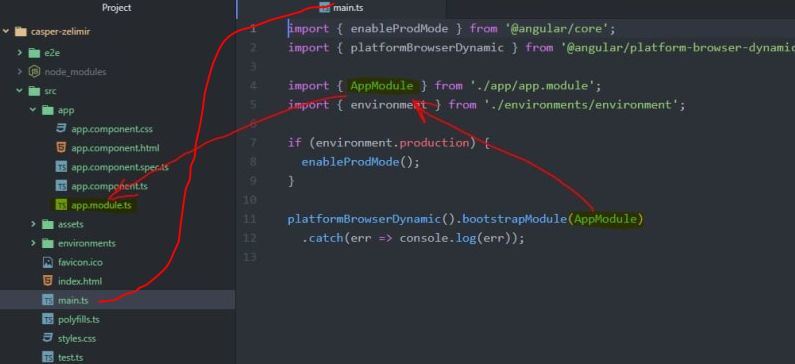




****

Which TS is used for entry point of angular application?

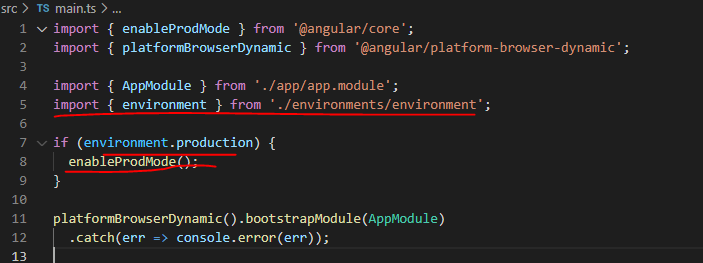
The **main.ts** file, that is the first code which gets executed. The job of **main.ts** is to bootstrap the application. It loads everything and controls the startup of the application.

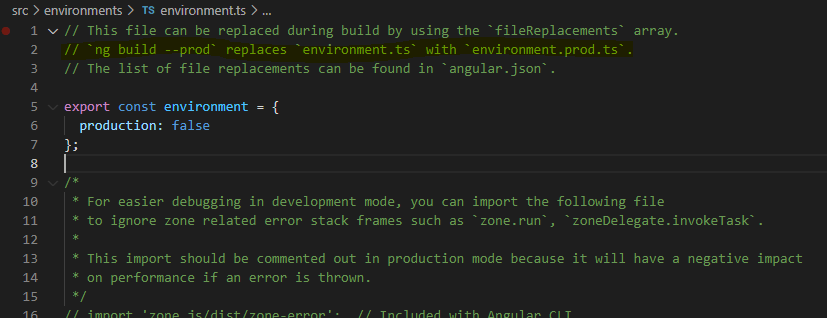


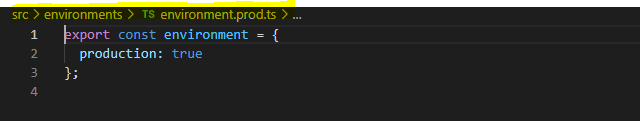
**SUMMARY**

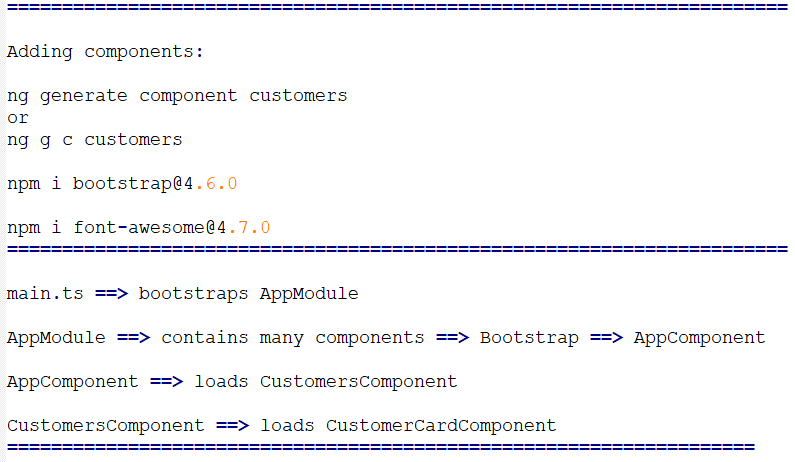
* Angular started with main.ts.
* Then we bootstrap an angular application and we pass app.module.ts as an argument. In app.module.ts we tell angular: *"There is the app component which you should know when you try to start yourself".*
* And angular now analyze this app component, reading the set up we pass there and there is *SELECTOR app-root*.
* Now, angular is enable to handle *app-root* in the index.html and knows rules for the *SELECTOR*.
* *SELECTOR* should insert the app components and have some HTML code - a template attached to him - html component.
* This is how Angular application starts.
  + main.ts file is the entry point of our web-app.It compiles the web-app and bootstraps the AppModule to run in the browser. It starts with importing the basic module which we need.
  + **platformBrowserDynamic().bootstrapModule(AppModule)**
  + This code has reference to the parent module i.e AppModule. Hence when it executes in the browser, the file that is called is index.html. index.html internally refers to the main.ts file which calls the parent module i.e AppModule.
  + When AppModule is called, it calls app.module.ts which further calls the AppComponent based on the bootstrap.
  + **bootstrap:[AppComponent]**
  + In app.component.ts, there is **selector:"app-root"** which is used in the index.html file. This will display the contents present in app.component.html.

#### Production environment in main ts file





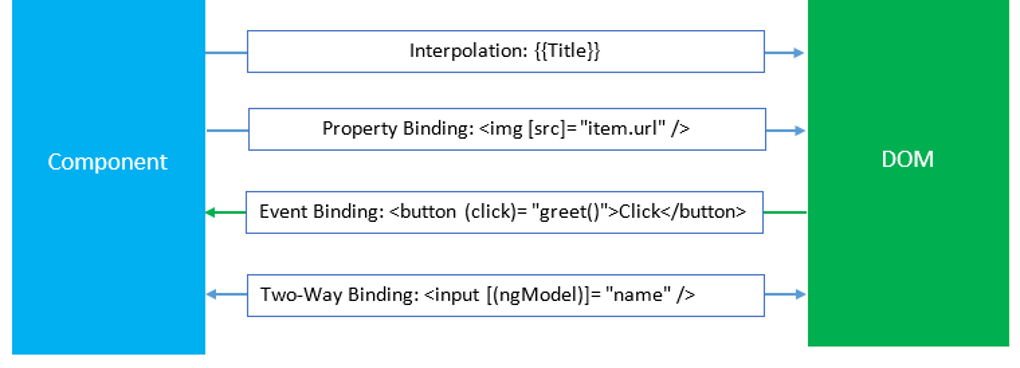


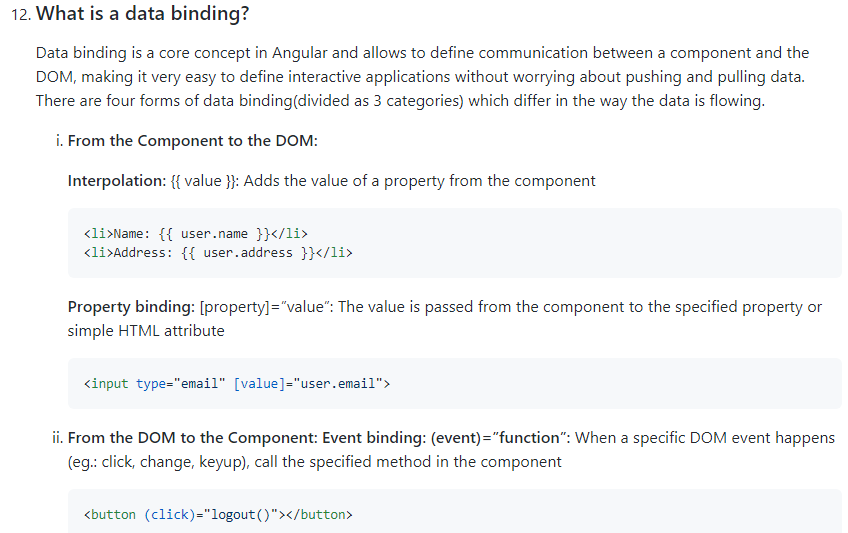


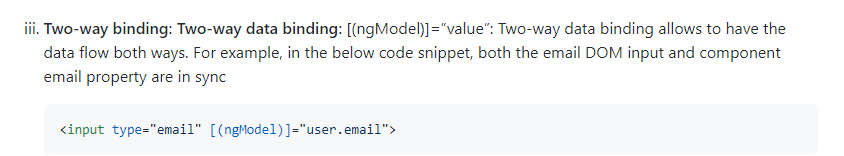
What is the use of Polyfills TS in angular?

**Polyfills** in **angular** are few lines of code which make your **application** compatible for different browsers. The code we write is mostly in ES6(New Features: Overview and Comparison) and is not compatible with IE or firefox and needs some environment setups before being able to be viewed or **used** in these browsers.

### Data Binding



****

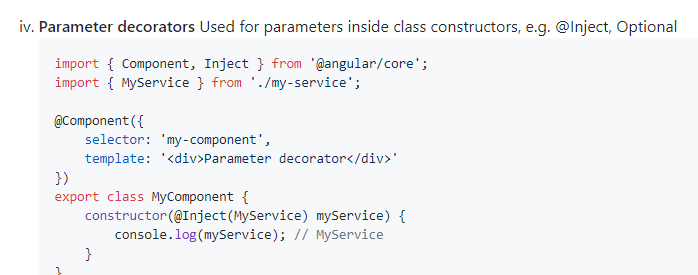
****

Meatdata



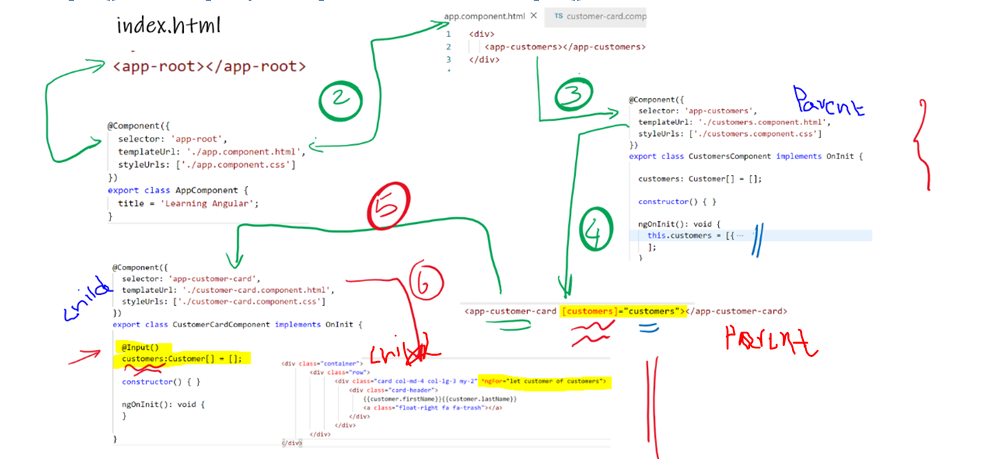






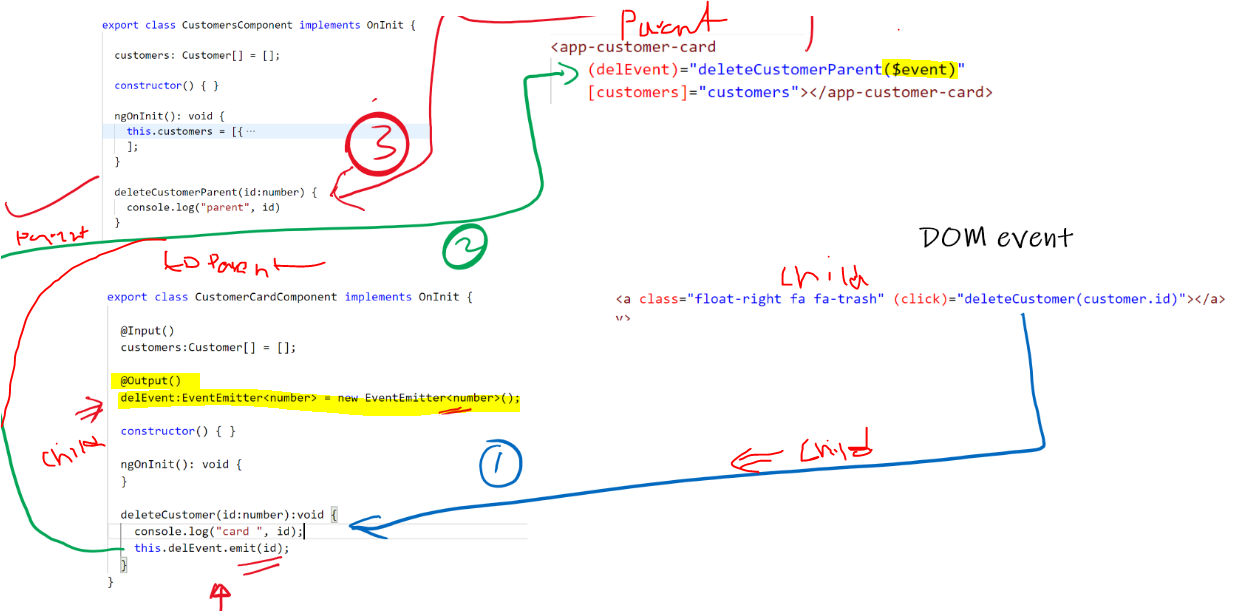
### @Input() ==> Property from parent to child, @Output() ==> EventEmitter

Customer to customer card

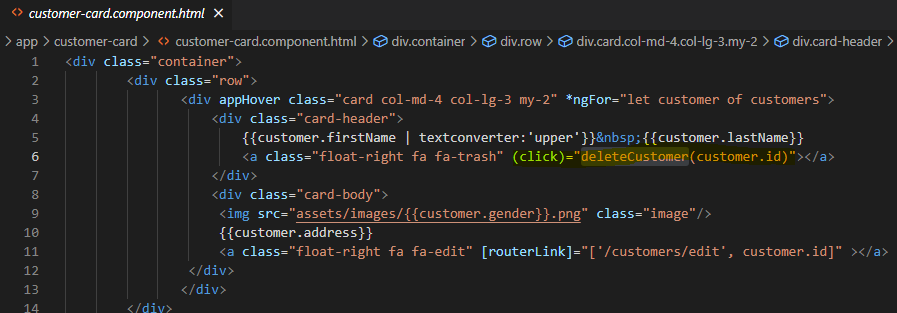


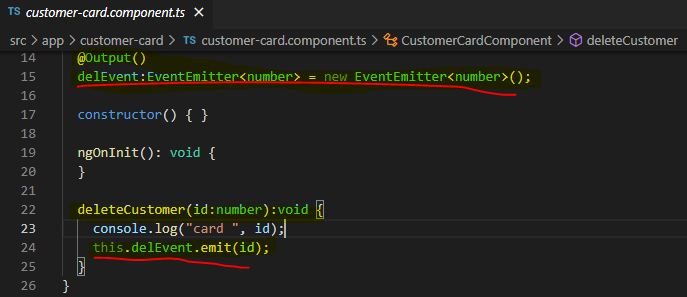


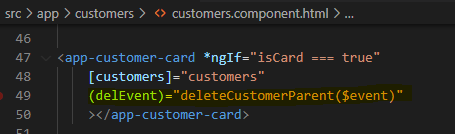
### @output

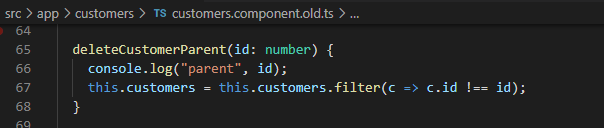


CustomerCard to Customer

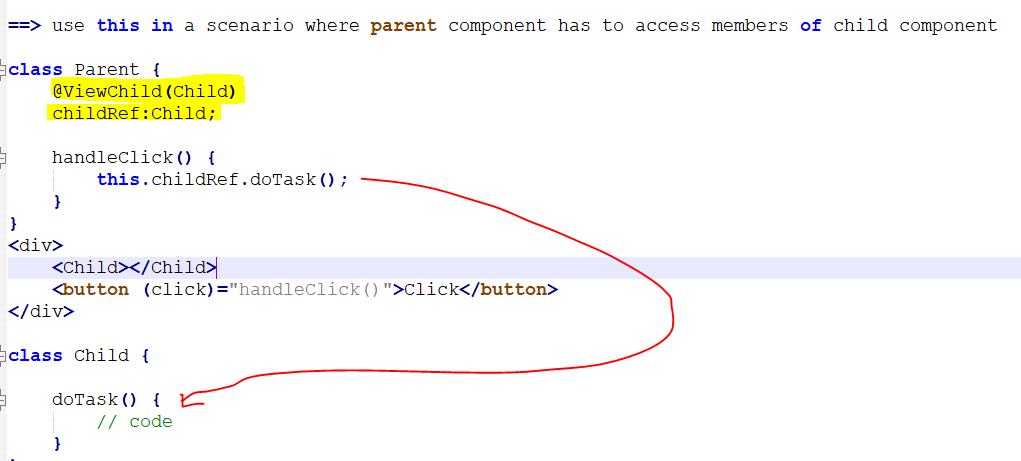


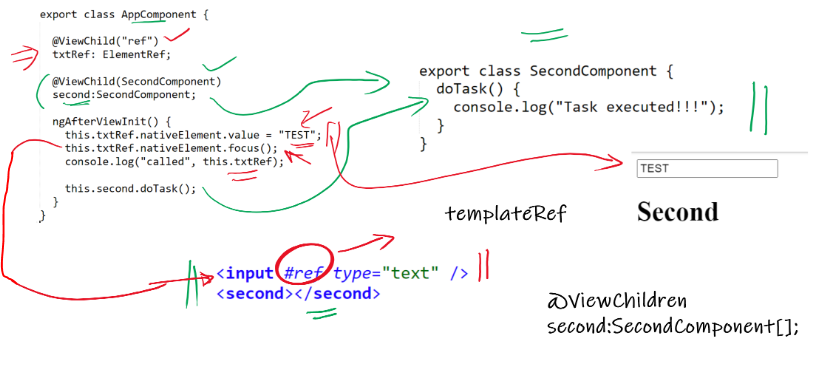


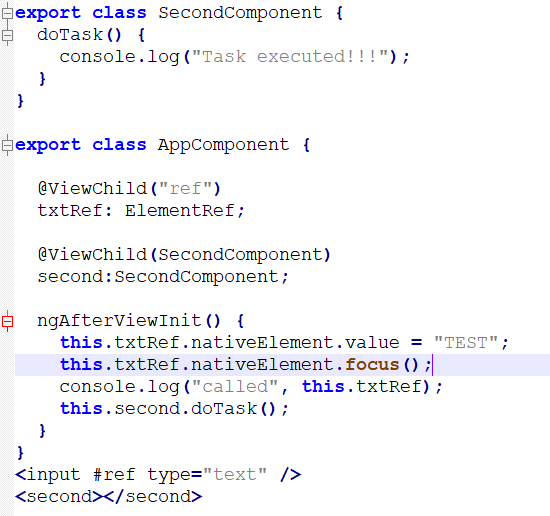




### @ViewChild

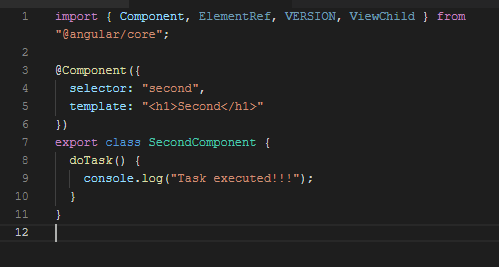




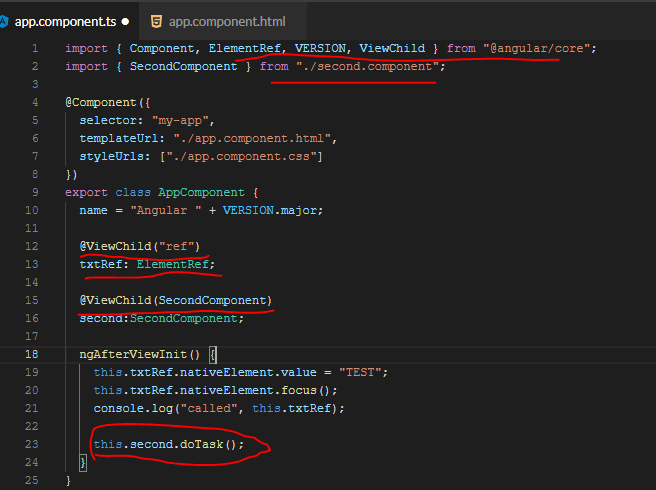


<https://stackblitz.com/edit/ng-view-child>

child component 🡺secondcomponent



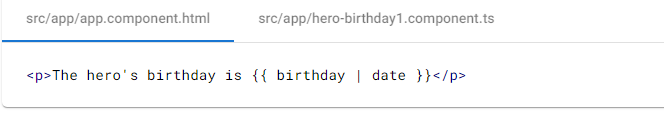
Now access second componet task in parnet component (app component)



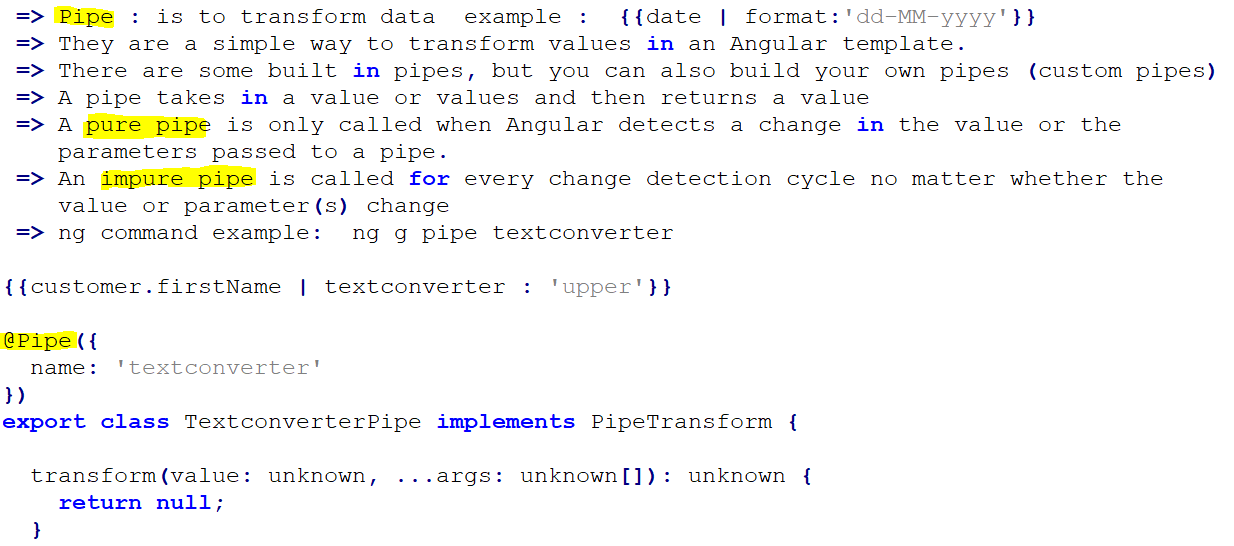
### Angular Pipes

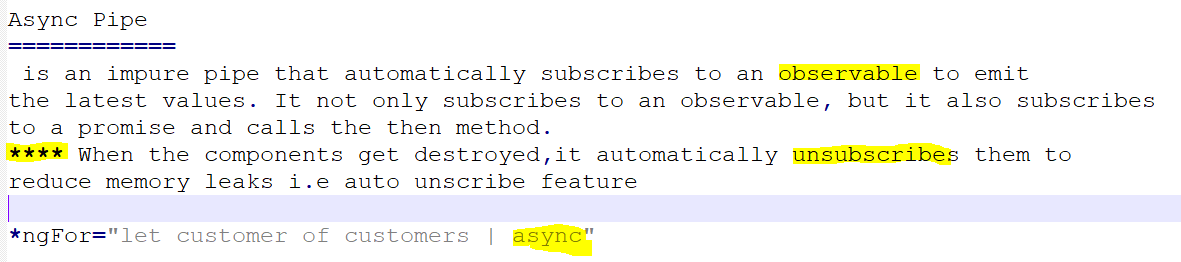
They are a simple way to transform values in an **Angular** template. There are some built in **pipes**, but you can also build your own **pipes**. A **pipe** takes in a value or values and then returns a value.

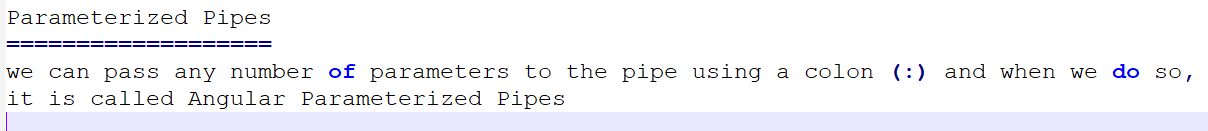
|  |  |
| --- | --- |
| Angular Pipes: Learn How to Create and Implement | Simplilearn | Angular Pipes with Real-time Examples - Dot Net Tutorials |

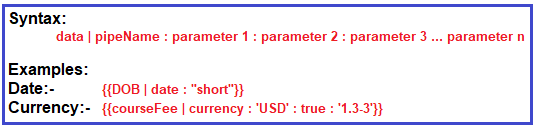


|  |  |
| --- | --- |
|  | Angular 8 Pipes - Tutorial And Example |



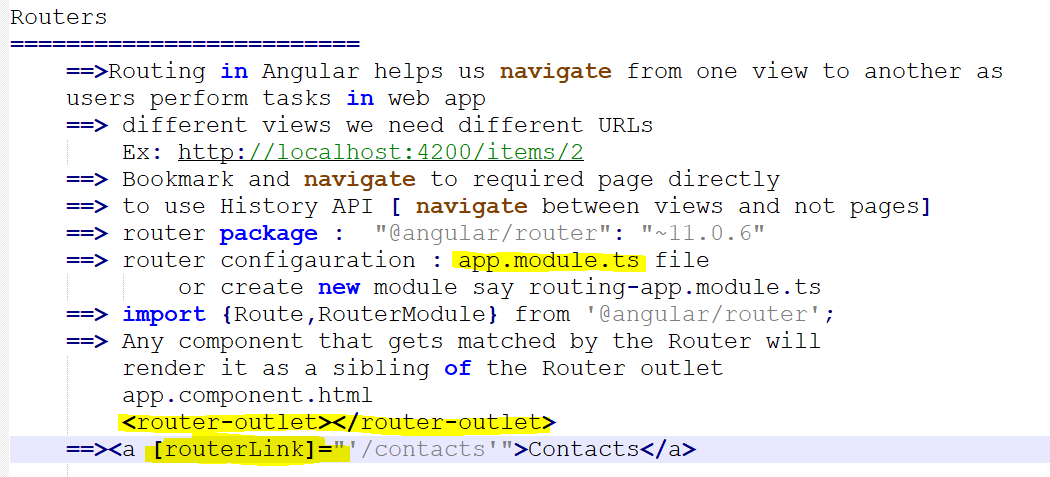


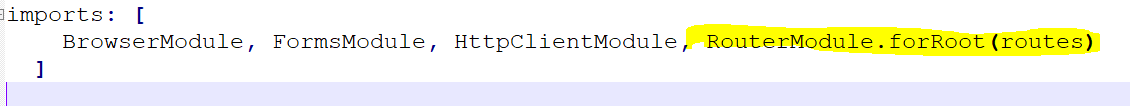




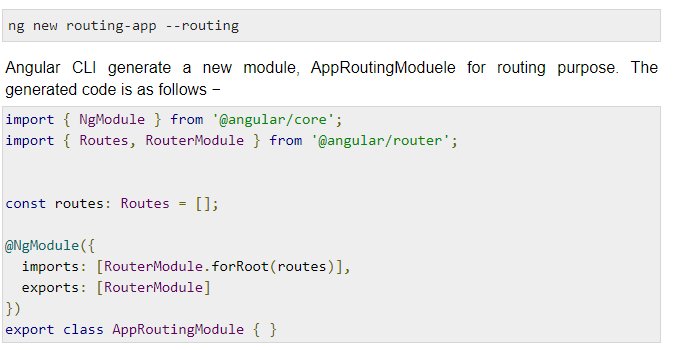


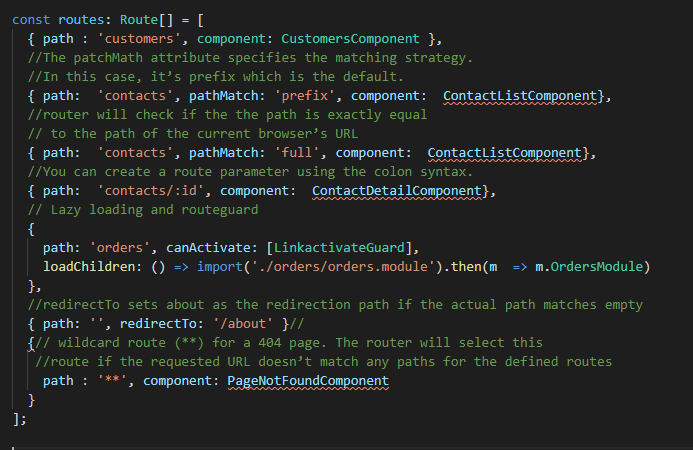
### Angular Routing

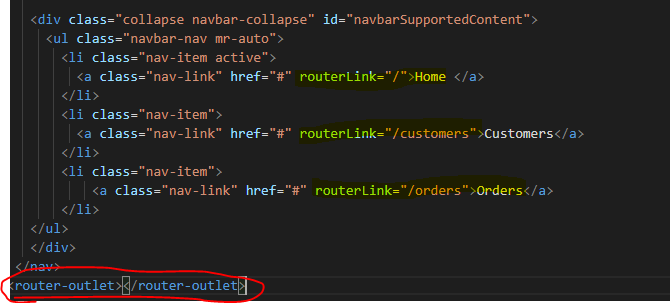




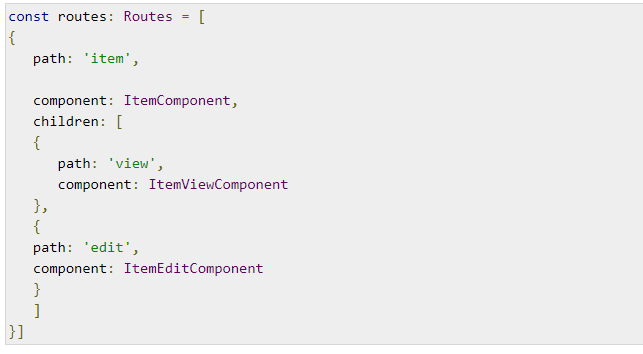
we can generate separate routing module :

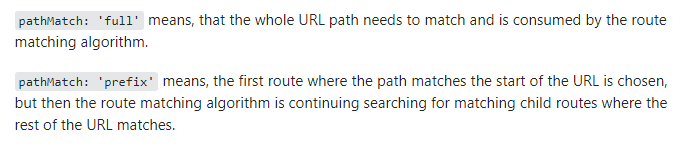


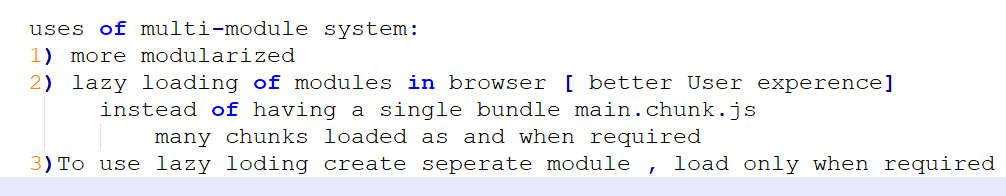




**Nested routing**



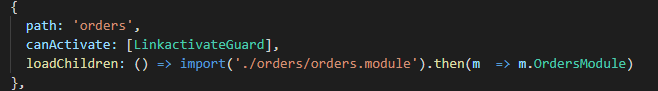
Lazy Routing/Loading modules :

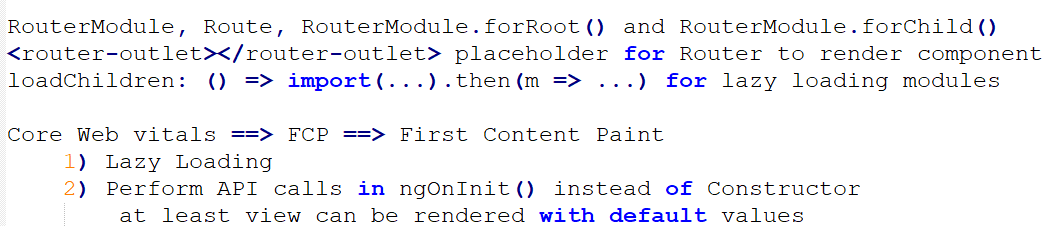


Create order module



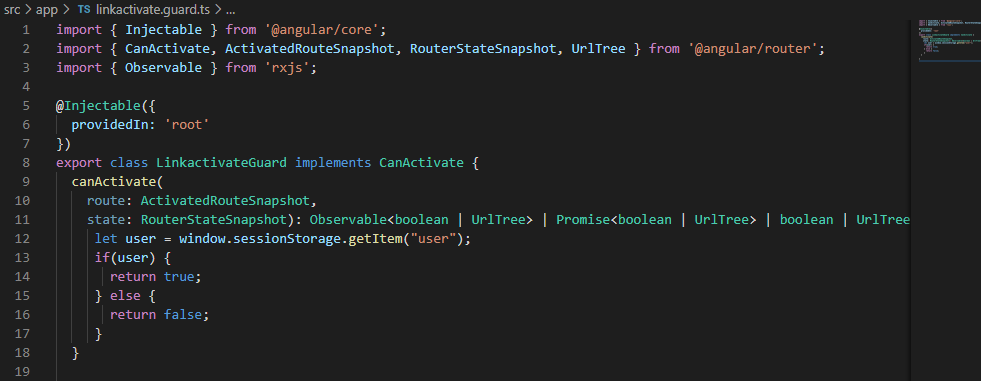
Import order module and load children on only when you navigate





#### Guard :

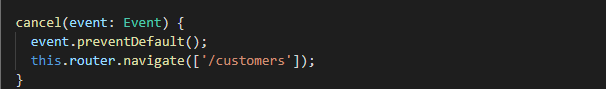








### Router navigation



In JavaScript we can use the **localSorage**or **sessionStorage**

// Store  
localStorage.setItem("email", "email@example.com");// Fetch  
document.getElementById("result").innerHTML = localStorage.getItem("email");

.

There are following way to store data into client side.

* **In Memory :** You can store data in client side computer memory.
* **Session Storage :** This maintains a separate storage area for each given origin that’s available for the duration of the page session.This session will flush when the tabs closed.
* **Local Storage :** Store data into client side browser using HTML5, Its supports only all modern browsers.Its same as session storage but persists more than even the browser is closed and reopened.

### Angular Services

*SharedService and DataService*

***Business logic / Http API calls/ communicate between components which doesn't have parent child relationship***

[The online code editor for web apps. Powered by Visual Studio Code. - StackBlitz](https://stackblitz.com/)

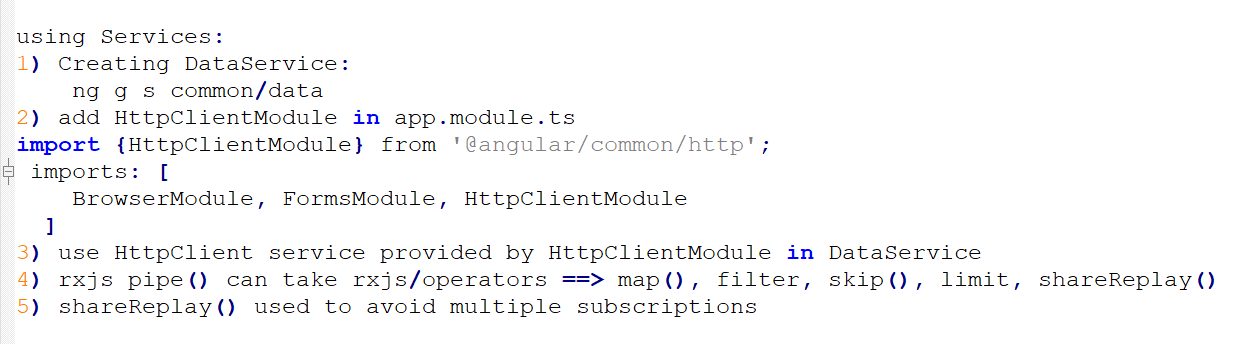
[angular-ivy-5zxwjt - StackBlitz](https://stackblitz.com/edit/angular-ivy-5zxwjt)

***----------------------------------------***

***Fake REST API creation :*** *data json should be placed in root folder*

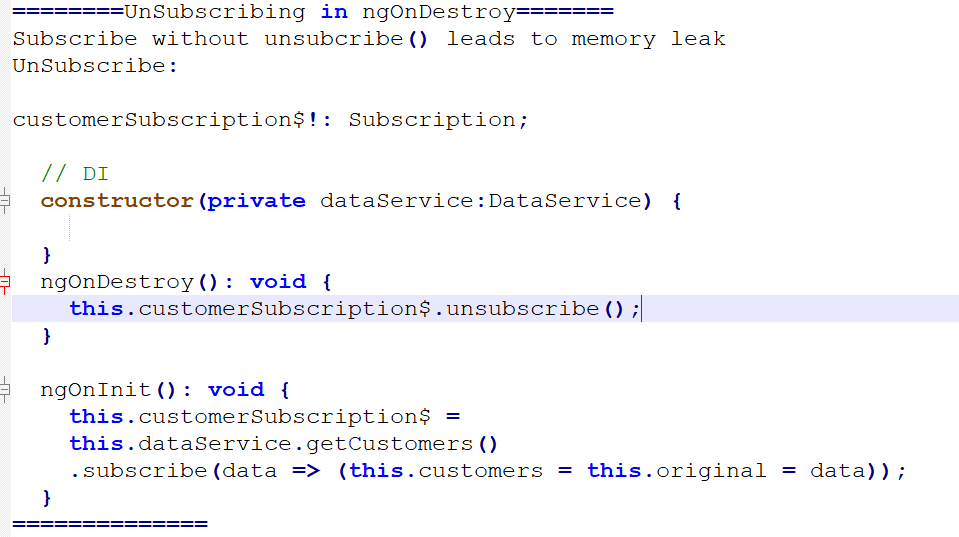
***npx json-server --watch data.json --port 1234***

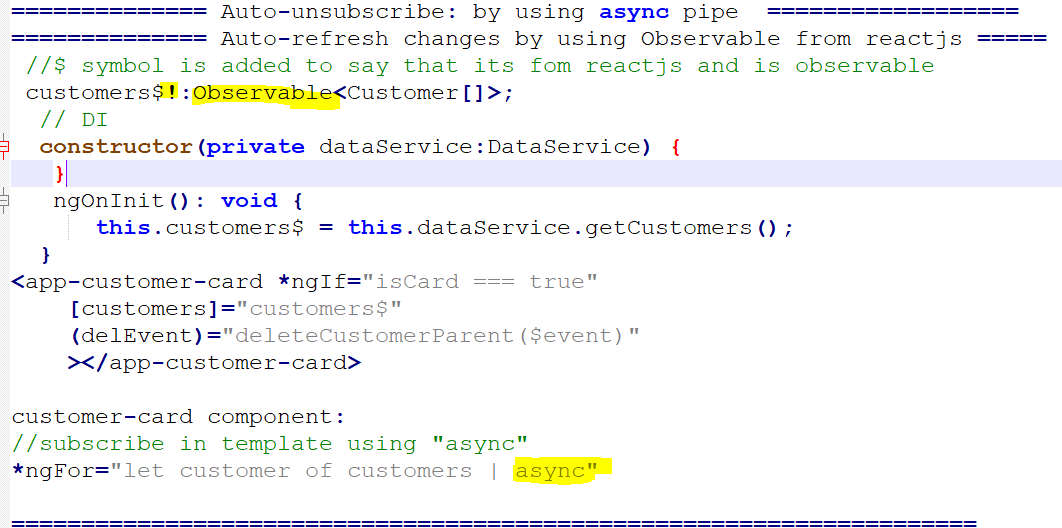
***-----------------------------------------------***

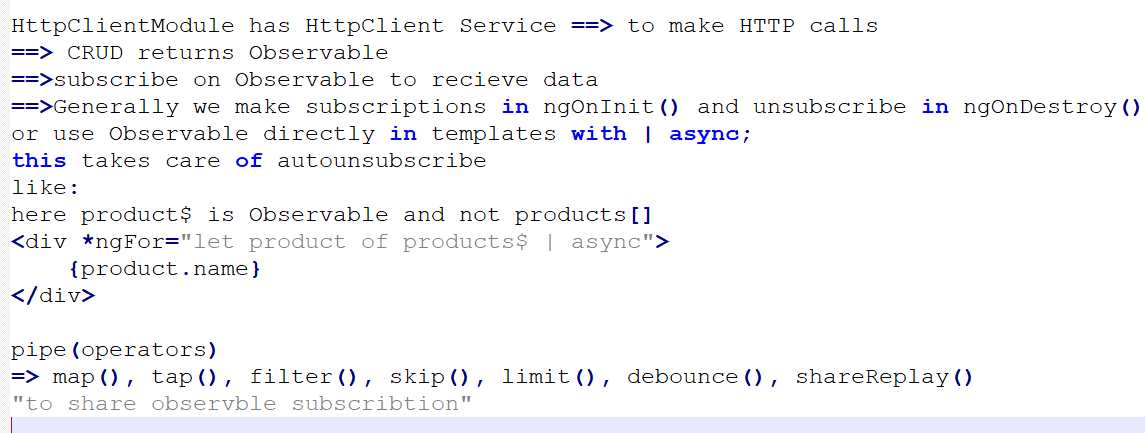
**

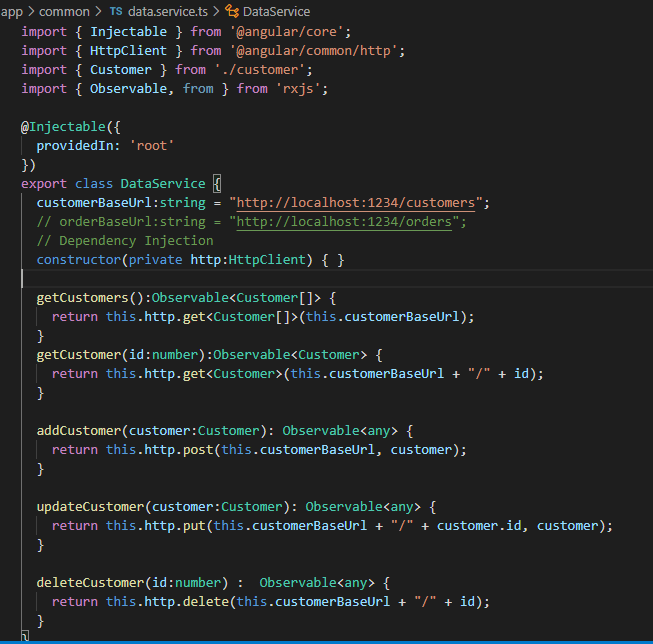
**

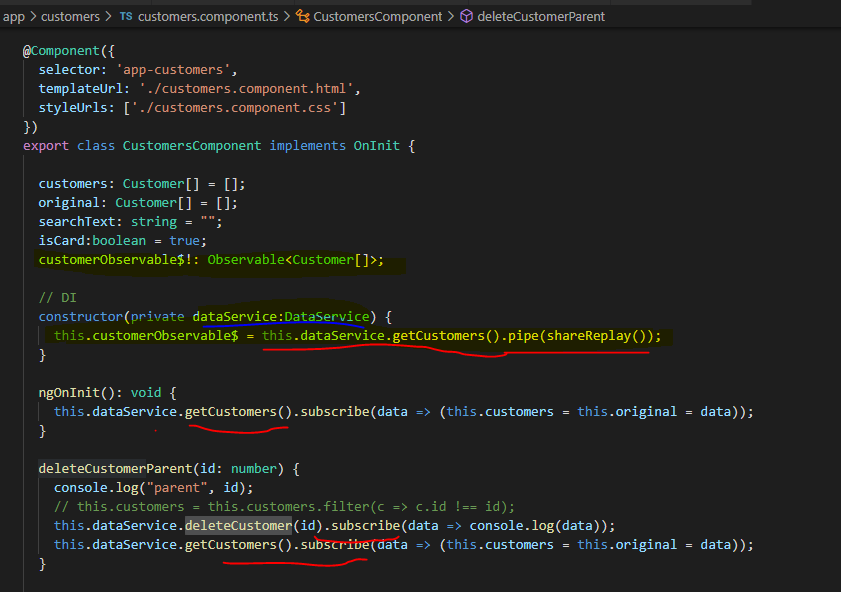
rxjs ==> ReactiveJS ==> Observable Observer Pattern

**

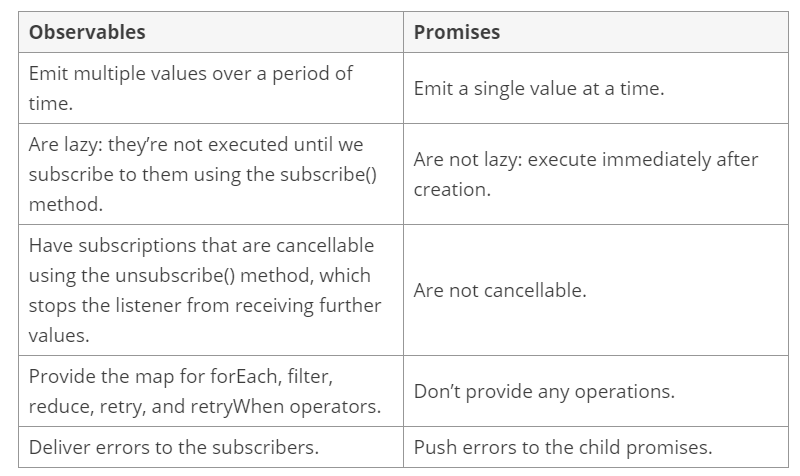
**

**

**

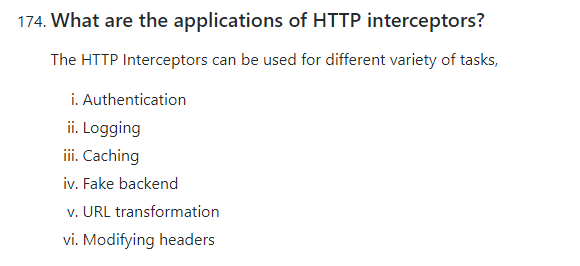
**



**

[*https://www.syncfusion.com/blogs/post/angular-promises-versus-observables.aspx#:~:text=Promises%20deal%20with%20one%20asynchronous,over%20a%20period%20of%20time.&text=Emit%20multiple%20values%20over%20a,single%20value%20at%20a%20time*](https://www.syncfusion.com/blogs/post/angular-promises-versus-observables.aspx#:~:text=Promises%20deal%20with%20one%20asynchronous,over%20a%20period%20of%20time.&text=Emit%20multiple%20values%20over%20a,single%20value%20at%20a%20time)*.*

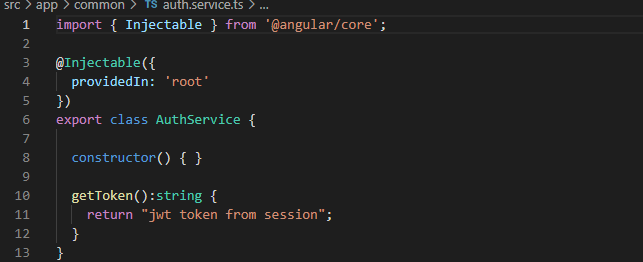
|  |  |  |
| --- | --- | --- |
| **operations** | **Observables** | **Promises** |
| Creation | |  |  | | --- | --- | | 1  2  3  4  5 | const obs = new Observable((observer) =&gt; {    observer.next(10);    }) ; | | |  |  | | --- | --- | | 1  2  3  4  5 | const promise = new Promise(() =&gt; {    resolve(10);    }); | |
| Transform | |  |  | | --- | --- | | 1 | Obs.pipe(map(value) =&gt; value \* 2); | | |  |  | | --- | --- | | 1 | promise.then((value) =&gt; value \* 2); | |
| Subscribe | |  |  | | --- | --- | | 1  2  3  4  5 | const sub = obs.subscribe((value) =&gt; {    console.log(value)    }); | | |  |  | | --- | --- | | 1  2  3  4  5 | promise.then((value) =&gt; {    console.log(value)    }); | |
| Unsubscribe | |  |  | | --- | --- | | 1 | sub.unsubscribe(); | | Can’t unsubscribe |

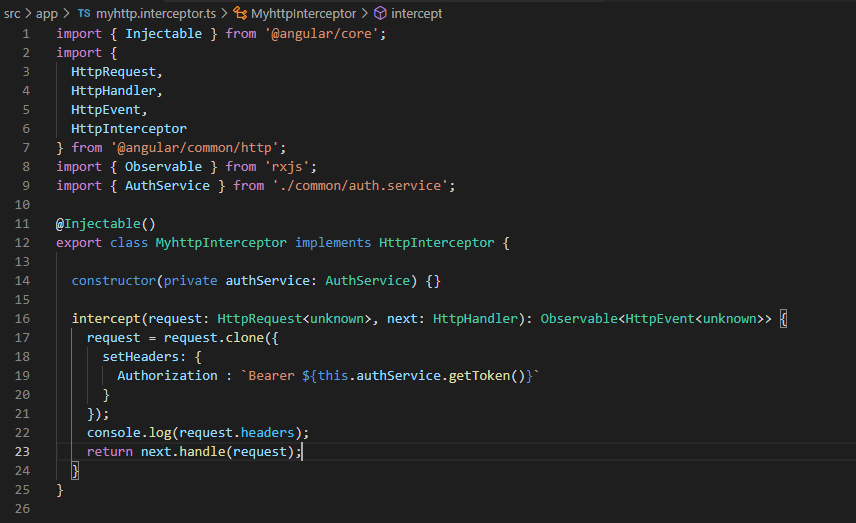
**



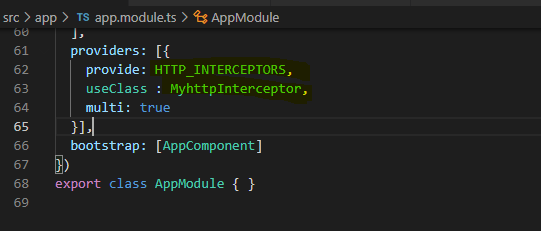
sending tokens [ JWT], encrypt





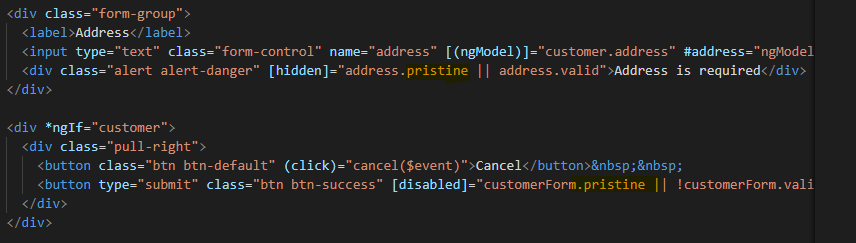


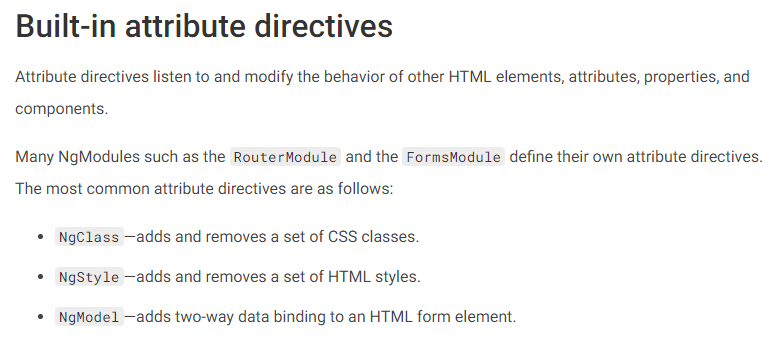


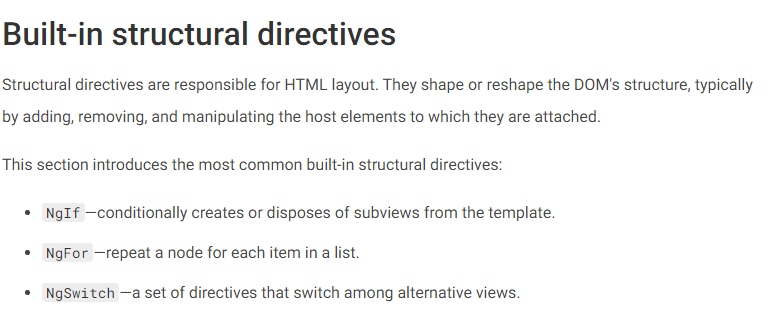


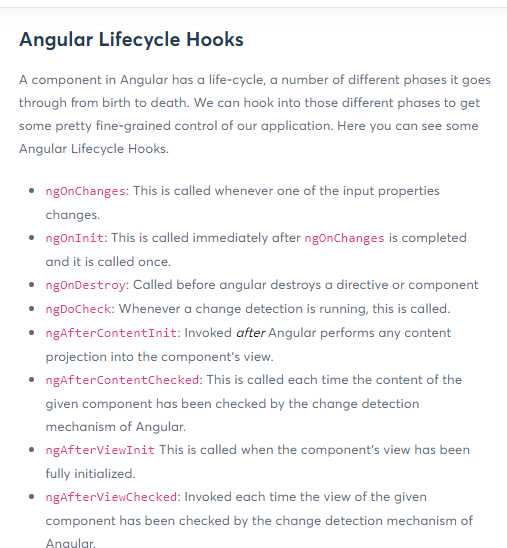
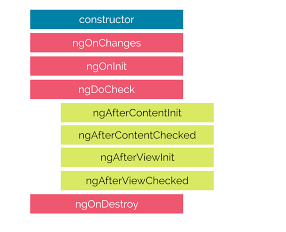
**Difference between ng-pristine and ng-dirty:**  
The main difference between both of them is that ng-dirty is used to tell that the input field is modified by the user and the ng-pristine is used to tell us that the field is untouched by the user

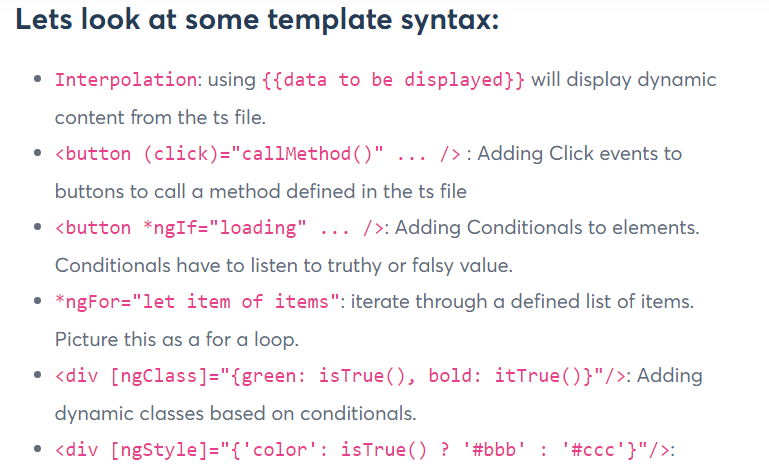
**pristine**: This property returns true if the **element's** contents have not been changed. **dirty**: This property returns true if the **element's** contents have been changed. **untouched**: This property returns true if the user has not visited the **element**. **touched**: This property returns true if the user has visited the **element**.











Cheatsheet

<https://angular.io/guide/cheatsheet>

<https://www.wrappixel.com/angular-cheat-sheet/>

npm install --D source-map-explorer

ChangeDetection

==> How angular renders the views?

uses NgZone

zone.js runs detection loop whenver a callback executes from timers, http calls

[Angular - Security](https://angular.io/guide/security)

Deployment

Cloud deployement

Docker image creation and deployment

*i -g @angular/cli@latest // install latest angular cli*

*ng --version // check angular version*

*npx -p @angular/cli ng new customerapp // create new angular application*

*npx ng g c*

*//To install the Angular CLI, open a terminal window and run the following command*

*npm install -g @angular/cli // install angular globally*

*ng --version // check installed version*

*//Create a workspace and initial application*

*ng new customerapp // create new angular project with pre required framework*

*==> strict ==> Y // input (Y/N): Typescript strict mode is set to on*

*==> Routing ==> N // input (Y/N): routing config pre setup codes needs be added*

*==> CSS // input : default styling CSS*

*npx -p @angular/cli ng new customerapp*

*ng help*

*ng build –help*

*ng g m modules/{ModuleName} //* *To add a new module*

*”ng g c components/MyNewComponent” //add component*

## UNIT and e2e(EndToEnd) TESTING

