## 1. What is Angular 5?

Angular is a framework for building client applications in HTML and either JavaScript or a language like a TypeScript that compiles to JavaScript.It is best suited for Single Page Application(SPA).

## 2.Why Angular?

* User Experience similar to a Desktop Application
* Productivity and Tooling
* Performance
* Community
* Full-featured Framework
  + Validation, Routing, DI…
* Platform for Targeting Native Mobile, not just Web Browsers
* TypeScript
  + Angular 2 was written in TypeScript, a superset of JavaScript that implements many new ES2016+ features.
* Modular
  + Hence angular have clean and clear structure.
* Unit Testing
  + Easy to Unit Test and easy to maintain the code.

## 3. What’s New in Angular 5?

The **Angular 5 Contains** bunch of new features, performance improvements and a lot of bug fixes and also some surprises to Angular lovers.

* Make AOT the default
* Watch mode
* Type checking in templates
* More flexible metadata
* Remove \*.ngfactory.ts files
* Better error messages
* Smooth upgrades
* Tree-Shakeable components
* Hybrid Upgrade Application
* Performance Improvements

## 4. What’s New in Angular 4?

Angular 4 contains some additional Enhancement and Improvement. Consider the following enhancements.

* Smaller & Faster Apps
* View Engine Size Reduce
* Animation Package
* NgIf and ngFor Improvement
* Template
* NgIf with Else
* Use of AS keyword
* Pipes
* HTTP Request Simplified
* Apps Testing Simplified
* Introduce Meta Tags
* Added some Forms Validators Attributes
* Added Compare Select Options
* Enhancement in Router
* Added Optional Parameter
* Improvement Internationalization

## 5. What Is Angular CLI?

The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications.Using CLI , you can create a UNIT and END-TO-END test of the Angular application.

## 6. What Is Bootstrapping in Angular?

main.ts is the entry point of your application, compiles the application with just-in-time and bootstrap the application.The Bootstrap is the **root** AppComponent that Angular creates and inserts into the “**index.html**” host web page.The bootstrapping process creates the components listed in the bootstrap array and inserts each one into the browser (**DOM**).

The bootstrapping process sets up the execution environment, digs the root AppComponent out of the module’s bootstrap array, creates an instance of the component and inserts it within the element tag identified by the component ’s.selector

## 7. What Is Architecture Overview of Angular?

[](http://www.techbloginterview.com/wp-content/uploads/2018/01/overview.png)

### Modules

* Angular apps are modular in nature.
* The angular application is nothing but collections of individual modules.
* Angular has its own modularity system called NgModules.
* Every Angular app has at least one NgModule class, [the root module](https://angular.io/guide/bootstrapping), conventionally named.AppModule
* An NgModule, whether a root or feature, is a class with a@[NgModule](https://angular.io/api/core/NgModule) decorator.

### Components

* A component controls a patch of the screen called a view.
* You define a component’s application logic—what it does to support the view—inside a class. The class interacts with the view through an API of properties and methods.
* Angular creates, updates, and destroys components as the user moves through the application.
* At least one component should be there called Root Component(app.component.ts)

### Metadata

* Components have @component decorator , contains selector , template,templateUrl , style , styleUrls , providers.
* selector: CSS selector that tells Angular to create and insert an instance of this component where it finds a <hero-list>tag in parent HTML. For example, if an app’s HTML contains,<hero-list></hero-list> then Angular inserts an instance of the view HeroListComponent between those tags.
* templateUrl: module-relative address of this component’s HTML template, shown above.
* providers: an array of **dependency injection providers** for services that the component requires.

### Templates

* You define a component’s view with its companion **template**. A template is a form of HTML that tells Angular how to render the component.
* A template looks like regular HTML

### Data binding

Angular supports **data binding**, a mechanism for coordinating parts of a template with parts of a component. Add binding markup to the template HTML to tell Angular how to connect both sides.

### Directives

Angular templates are dynamic. When Angular renders them, it transforms the DOM according to the instructions given by **directives**.

A directive is a class with a @Directive decorator.

### Services

A class contains the Business Logic.

## 8. How To Angular 5 generate Component, Directive, Pipe, Service, Class, and Module?

| **Scaffold** | **Usage** |
| --- | --- |
| [Component](https://github.com/angular/angular-cli/wiki/generate-component) | ng g component my-new-component |
| [Directive](https://github.com/angular/angular-cli/wiki/generate-directive) | ng g directive my-new-directive |
| [Pipe](https://github.com/angular/angular-cli/wiki/generate-pipe) | ng g pipe my-new-pipe |
| [Service](https://github.com/angular/angular-cli/wiki/generate-service) | ng g service my-new-service |
| [Class](https://github.com/angular/angular-cli/wiki/generate-class) | ng g class my-new-class |
| [Guard](https://github.com/angular/angular-cli/wiki/generate-guard) | ng g guard my-new-guard |
| [Interface](https://github.com/angular/angular-cli/wiki/generate-interface) | ng g interface my-new-interface |
| [Enum](https://github.com/angular/angular-cli/wiki/generate-enum) | ng g enum my-new-enum |
| [Module](https://github.com/angular/angular-cli/wiki/generate-module) | ng g module my-module |

## 9. What Is the Angular Compiler? Why we need Compilation in Angular?

The **Angular** compiler converts our applications code (**TypeScript**) into **JavaScript** code + HTML before browser downloads and runs that code.

The Angular offers two ways to compile our application code-

1. **Just-in-Time** (**JIT**) – JIT compiles our app in the browser at runtime (compiles before running).
2. **Ahead-of-Time** (**AOT**) – AOT compiles our app at build-time (compiles while running).

## 10. What Is the difference between JIT compiler and AOT compiler?

**JIT (Just-in-Time) –**

1. JIT compiles our app in the browser at runtime.
2. Compiles before running
3. Each file compiled separately
4. No need to build after changing our app code and it automatically reflects the changes in your browser page
5. Highly secure
6. Very suitable for local development

**AOT (Ahead-of-Time) –**

1. AOT compiles our app code at build time.
2. Compiles while running
3. Compiled by the machine itself, via the command line (Faster)
4. All code compiled together, inlining HTML/CSS in the scripts
5. Highly secure
6. Very suitable for production builds

## 11.Explain the life cycle hooks of Angular 5 application

Angular 5 component/directive has lifecycle events, managed by @angular/core. It creates the component, renders it, creates and renders its children, processes changes when its data-bound properties change, and then destroys it before removing its template from the DOM. Angular provides a set of lifecycle hooks(special events) which can be tapped into this lifecycle and perform operations when required. The constructor executes prior to all lifecycle events. Each interface has a single hook method prefixed with ng. For example, ngOnint interface has Oninit method that must be implemented in the component.

Some of the events are applicable for both component/directives while few are specific to components.

* **ngOnChanges**: Responds when angular sets its data-bound property which receives the current and previous object values.
* **ngOnInit**: Initializes the component/directive after first ngOnChange triggers. This is most frequently used method to retrieve the data for the template from a back-end service.
* **ngDoCheck**: Detect and act upon changes occurring outside Angular context. It is called when every change detection run.
* **ngOnDestroy**: Cleanup just before Angular destroys the directive/component. Unsubscribe observables and detach event handlers to avoid memory leaks.

**Component-specific hooks:**

* **ngAfterContentInit**: Component content has been initialized
* **ngAfterContentChecked**: After Angular checks the bindings of the external content that it projected into its view.
* **ngAfterViewInit**: After Angular creates the component’s view.
* **ngAfterViewChecked**: After Angular checks the bindings of the component’s view.

## 12. What is Lazy Loading and How to enable Lazy Loading?

Most of the enterprise application contains various modules for specific business cases. Bundling whole application code and loading will be huge performance impact at initial call. Lazy lading enables us to load only the module user is interacting and keep the rest to be loaded at runtime on demand.

Lazy loading speeds up the application initial load time by splitting the code into multiple bundles and loading them on demand.

Every Angular application must have one main module say AppModule. The code should be splitted into various child modules (NgModule) based on the application business case.

Plunkr Example:

1. We don’t require to import or declare lazily loading module in root module.
2. Add the route to top-level routing (app.routing.ts) and set loadChildren. loadChildren takes an absolute path from root folder followed by #{ModuleName}. RouterModule.forRoot() takes routes array and configures the router.
3. Import module specific routing in the child module.
4. In the child module routing, specify a path as empty string ‘ ‘, the empty path. RouterModule.forChild again takes routes array for the child module components to load and configure router for child.
5. Then, export const routing: ModuleWithProviders **= RouterModule.forChild**(routes);

## 13. What are the core differences between Observables and Promises?

A nice answer is taken from stack overflow:

A Promise handles a **single event** when an async operation completes or fails.

Note: There are Promise libraries out there that support cancellation, but ES6 Promise doesn’t so far.

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event. Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable, it doesn’t matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case. Observable also has the advantage over Promise to be **cancelable**. If the result of an HTTP request to a server or some other expensive async operation isn’t needed anymore, the Subscription of an Observable allows to cancel the subscription, while a Promise will eventually call the success or failed callback even when you don’t need the notification or the result it provides anymore. Observable provides **operators** like map, forEach, reduce, … similar to an array. There are also powerful operators like retry(), or replay(), … that are often quite handy.

**Promises vs Observables**

* Promises:
  1. returns a single value
  2. not cancellable
* Observables:
  1. works with multiple values over time
  2. cancellable
  3. supports map, filter, reduce and similar operators
  4. proposed feature for ES 2016
  5. use Reactive Extensions (RxJS)
  6. an array whose items arrive asynchronously over time

## 14. What are differences between Constructors and OnInit?

**Constructors:-**

1.      The **constructor** is a default method runs when a component is being constructed.

2.      The constructor is a typescript **feature** and it is used only for a class **instantiations**.

3.      The constructor called first time before the **ngOnInit**().

**ngOnInit**:-

1.      The **ngOnInit** event is an Angular 5 life-cycle event method that is called after the first ngOnChanges and the ngOnInit method is use to parameters defined with @**Input** otherwise the constructor is **OK**.

2.      The **ngOnInit** is called after the constructor and ngOnInit is called after the first ngOnChanges.

3.      The **ngOnChanges** is called when an input or output binding value changes.

## 15. What are Event Emitters and how it works in Angular 5?

Angular 5 doesn’t have bi-directional digest cycle, unlike angular 1. In angular 5, any change occurred in the component always gets propagated from the current component to all its children in the hierarchy. If the change from one component needs to be reflected to any of its parent component in a hierarchy, we can emit the event by using Event Emitter API.

In short, EventEmitter is class defined in @angular/core module which can be used by components and directives to emit custom events.

@output() somethingChanged = new EventEmitter();

We use somethingChanged.emit(value) method to emit the event. This is usually done in setter when the value is being changed in the class.

This event emit can be subscribed by any component of the module by using subscribe method.

myObj.somethingChanged.subscribe(val) => this.myLocalMethod(val));

## 16. Explain local reference variables, ViewChild, and ContentChild.

Local template variables in angular5 are used to refer HTML elements and use their properties to access siblings or children.

Let’s consider you have an input field named username.

<input type="text" required ... />

This HTMLInputField can be made available to the template using # symbol with a variable name say username.

<input type="text" #username required ... />

Now, this HTMLInputElement can be accessed from anywhere in the current template, for example, checking validation and showing appropriate message based on the validation rule. But, username HTML reference is not accessible in the component/directive.

To access this in the component, angular 5 provides @ViewChild decorator which accepts the local reference variable.

@ViewChild('username') username: HTMLInputElement;

ViewChild an element can be read after the view is initialized (ngAfterViewInit).

ContentChild is used to query the reference of the DOM within ng-content. Content Child are set before the ngAfterContentInitlifecycle hook.

## 17. Explain tsconfig.json file.

The tsconfig.json file corresponds to the configuration of the TypeScript compiler (tsc).

{

"compilerOptions": {

"target": "es5",

"module": "commonjs",

"moduleResolution": "node",

"sourceMap": true,

"emitDecoratorMetadata": true,

"experimentalDecorators": true,

"lib": [ "es2015", "dom" ],

"noImplicitAny": true,

"suppressImplicitAnyIndexErrors": true

}

}

* **target**: the language used for the compiled output
* **module**: the module manager used in the compiled output. system is for SystemJS, commonjs for CommonJS.
* **moduleResolution**: the strategy used to resolve module declaration files (.d.ts files). With the node approach, they are loaded from the node\_modules folder like a module (require('module-name'))
* **sourceMap**: generate or not source map files to debug directly your application TypeScript files in the browser,
* **emitDecoratorMetadata**: emit or not design-type metadata for decorated declarations in a source,
* **experimentalDecorators**: enables or not experimental support for ES7 decorators,
* **removeComments**: remove comments or not
* **noImplicitAny**: allow or not the use of variables/parameters without types (implicit)

## 18. Explain package.json file.

All npm packages contain a file, usually in the project root, called package.json – this file holds various metadata relevant to the project.

* This file is used to give information to npm that allows it to identify the project as well as handle the project’s dependencies.
* It can also contain other metadata such as a project description, the version of the project in a particular distribution, license information, even configuration data – all of which can be vital to both npm and to the end users of the package.
* The package.json file is normally located at the root directory of a Node.js project.

## 19. Explain systemjs.config.json file.

system.config.js is the one which allows to load modules(node modules) compiled using the TypeScript compiler.map refers to the name of modules to JS file that contains the JavaScript code.

## 20. Explain app.module.ts file.

This is root module that tells Angular how to assemble the application. Every Angular app has a root module class.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule({

imports: [ BrowserModule ],

declarations: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

@NgModule — takes a metadata object that tells Angular how to compile and launch the application.

Imports — the BrowserModule that this and every application needs to run in a browser.

Declarations — the application’s component.

Bootstrap — this is the root component tells which component to run first.

## 21. What is Dependency Injection?

Dependency injection is the ability to add the functionality of components at runtime. Let’s take a look at an example and the steps used to implement dependency injection.

## 22. How will you handle errors in Angular 2 applications?

Angular 5 applications have the option of error handling. This is done by including the ReactJS catch library and then using the catch function.

* The catch function contains a link to the Error Handler function.
* In the error handler function, we send the error to the console. We also throw the error back to the main program so that the execution can continue.
* Now, whenever you get an error it will be redirected to the error console of the browser.

## 23. What is routing?

Routing helps in directing users to different pages based on the option they choose on the main page. Hence, based on the option they choose, the required Angular Component will be rendered to the user.

## 24. What is @Inputs in Angular5?

@Input decorator binds a property within one component (child component) to receive a value from another component (parent component). This is one-way communication from parent to child. The component property should be annotated with a @Input decorator to act as input property. A component can receive a value from another component using component property binding.It can be annotated at any type of property such as number, string, array or user-defined class. To use an alias for the binding property name we need to assign an alias name as.@Input(alias)

## 25. What is @Outputs in Angular5?

@Output decorator binds a property of a component to send data from one component (child component) to calling component (parent component). This is one-way communication from child to the parent component. @Output binds a property of the type of angular EventEmitter class. This property name becomes custom event name for calling component. @Output the decorator can also alias the property name as @Output(alias) and now this alias name will be used in custom event binding in calling component.

## 26. What is the difference between Angular 5 components and directives?

Components and Directives are allowing us to attach behavior to elements in DOM. There are certain differences.

* A component is a directive with a view whereas a directive is a decorator with no view.
* A component is used to break up the application into smaller components whereas directive is used to design the re-usable components.
* A component can use pipes whereas directives can’t.
* A component can be present per DOM element whereas directive is used to add behavior to an existing DOM.

## 27. What is the difference between ActivatedRoute and RouterState in Angular 5?

ActivateRoute and RouterState both refer to Routing in Angular 5.

* ActivatedRoute consists of the information about a route associated with the component loaded in outlet whereas RouterState represents the state.
* ActivatedRouteSnapshort has old data When Route changes, ActivateRouteSnapshort has data from the previous route whereas the RouteState care about the arrangements and application components.
* ActivatedRouteSnapchat to traverse all the activated routes whereas RouterState maintains the states.

## 28. What are Pipes in Angular 5?

Pipes in Angular 5 are used in templates in order to convert them into a content that is user-friendly and readable one within the interpolation({{}}).

“|” denotes the pipe

## 29. Why decorators are used in Angular 5?

Decorators are used as an identifier of class or type of the objects that are created by the Typescript. Decorators specify the properties of the class.

## 30. How can you handle errors in Angular 5 application?

The error in Angular 5 can be handled by

* Try – Catch functionality
* Observable Errors
* Promise Errors

### **1. What’s new in Angular 5?**

Certain tools are optimized in the new version of [Angular](https://www.greycampus.com/angularjs-training-instructor-led), let us see what the tools are:

* Angular 5 supports Typescript version 2.4
* Angular 5 supports RxJS 5.5 which has new features like Pipeable Operators
* A build tool to make the js bundles (files) lighter
* Ahead of Time (AOT) is updated to be on by default
* Events like ActivationStart and ActivationEnd are introduced in Router

### **2. Name the building blocks of Angular.**

The Angular application is made using the following:

Modules

Component

Template

Directives

Data Binding

Services

Dependency Injection

Routing

### **3. What is Transpiling in Angular?**

Transpiling is the process of converting the typescript into javascript (using Traceur, a JS compiler). Though typescript is used to write code in the Angular applications, the code is internally transpiled into javascript.

### **4. Which of the Angular life cycle component execution happens when a data-bound input value updates?**

ngOnChanges is the life cycle hook that gets executed whenever a change happens to the data that was bound to an input.

### **5. Differentiate between Components and Directives in Angular 5.**

Components break up the application into smaller parts; whereas, Directives add behavior to an existing DOM element.

### **6. What is the use of @Input and @Output?**

When it comes to the communication of Angular Components, which are in Parent-Child Relationship; we use @Input in Child Component when we are passing data from Parent to Child Component and @Output is used in Child Component to receive an event from Child to Parent Component.

Learn more about [***Top 5 Skills That Make You A Sure Shot Programmer***](https://www.greycampus.com/blog/programming/top-five-skills-that-make-you-a-sure-shot-programmer)

### **7. What is ng-content Directive?**

The HTML elements like p (paragraph) or h1 (heading) have some content between the tags. For example, <p>this is a paragraph</p> and <h1>this is a heading</h1>. Now, similar to this, what if we want to have some custom text or content between the angular tags like  <app-tax>some tax-related content</app-tax> This will not work the way it worked for HTML elements.  Now, in such cases, the <ng-content> tag directive is used.

### **8. What does a router.navigate do?**

When we want to route to a component we use router.navigate.  Syntax: this.router.navigate([‘/component\_name’]);

### **9. What is ViewEncapsulation?**

ViewEncapsulation decides whether the styles defined in a component can affect the entire application or not. There are three ways to do this in Angular:

Emulated: styles from other HTML spread to the component.

Native: styles from other HTML do not spread to the component.

None: styles defined in a component are visible to all components.

### **10. What are Services in Angular and what command is used to create a service?**

Services help us in not repeating the code. With the creation of services, we can use the same code from different components. Here is the command to create a service in angular, ng g service User (a UserService is created when this command is used).

### **11. What is Dependency Injection in Angular 4?**

When a component is dependent on another component the dependency is injected/provided during runtime.

### **12. What is Routing in Angular 5?**

Routing helps a user in navigating to different pages using links.

### **13. How to handle Events in Angular 5?**

Any activity (button click, mouse click, mouse hover, mouse move, etc) of a user on a frontend/web screen is termed as an event. Such events are passed from the view (.HTML) page to a typescript component (.ts).

### **14. What is a RouterOutlet?**

RouterOutlet is a substitution for templates rendering the components. In other words, it represents or renders the components on a template at a particular location.

### **15. Explain the usage of {{}}?**

The set of brackets {{}} when used with an HTML tag, represent data from a component. For example, on a HTML page which has <h1>{{variableName}}</h1>, here the ‘variableName’ is actually typescript (component) data representing its value on the template; i.e., HTML. This entire concept is called String Interpolation.

You may also like: [***Other Programming workshops***](https://www.greycampus.com/programming)

### **16. In how many ways the Data Binding can be done?**

Data Binding happens between the HTML (template) and typescript (component). Data binding can be done in 3 ways:

(i) Property Binding (ii) Event Binding (iii) Two-Way Data Binding.

### **17. What is the sequence of Angular Lifecycle Hooks?**

OnChange()  -  OnInit()  -  DoCheck()  -  AfterContentInit()  -  AfterContentChecked()  -  AfterViewInit()  -  AfterViewChecked()  -  OnDestroy().

### **18. What is the purpose of using package.json in the angular project?**

With the existence of package.json, it will be easy to manage the dependencies of the project. If we are using typescript in the angular project then we can mention the typescript package and version of typescript in package.json.

### **19. How is SPA (Single Page Application) technology different from the traditional web technology?**

In traditional web technology, the client requests for a web page (HTML/JSP/asp) and the server sends the resource (or HTML page), and the client again requests for another page and the server responds with another resource. The problem here is a lot of time is consumed in the requesting/responding or due to a lot of reloading. Whereas, in the SPA technology, we maintain only one page (index.HTML) even though the URL keeps on changing.

### **20. What is Component in Angular Terminology?**

A web page in Angular has many components involved in it. A Component is basically a block in which the data can be displayed on HTML using some logic usually written in typescript.

### **21. What are ngModel and how do we represent it?**

ngModel is a directive which can be applied on a text field. This a two-way data binding. ngModel is represented by [()]

### **22. What does a Subscribe method do in Angular 4?**

It is a method which is subscribed to an observable. Whenever the subscribe method is called, an independent execution of the observable happens.

### **23. Differentiate between Observables and Promises.**

Observables are lazy, which means nothing happens until a subscription is made. Whereas Promises are eager; which means as soon as a promise is created, the execution takes place. Observable is a stream in which passing of zero or more events is possible and the callback is called for each event. Whereas, promise handles a single event.

### **24. What is an AsyncPipe in Angular?**

When an observable or promise returns something, we use a temporary property to hold the content. Later, we bind the same content to the template. With the usage of AsyncPipe, the promise or observable can be directly used in a template and a temporary property is not required.

### **25. Explain Authentication and Authorization.**

Authentication: The user login credentials are passed to an authenticate API (on the server). On the server side validation of the credentials happens and a JSON Web Token (JWT) is returned. JWT is a JSON object that has some information or attributes about the current user.  Once the JWT is given to the client, the client or the user will be identified with that JWT.

Authorization: After logging in successfully, the authenticated or genuine user does not have access to everything. The user is not authorized to access someone else’s data,  he/she is authorized to access some data.

### **26. What is AOT Compilation?**

Every angular application gets compiled internally. The angular compiler takes javascript code, compiles it and produces javascript code again. Ahead-of-Time Compilation does not happen every time or for every user, as is the case with Just-In-Time (JIT) Compilation.

### **27. What is Redux?**

It is a library which helps us maintain the state of the application. Redux is not required in applications that are simple with the simple data flow, it is used in Single Page Applications that have complex data flow.

### **28. What are Pipes?**

This feature is used to change the output on the template; something like changing the string into uppercase and displaying it on the template. It can also change Date format accordingly.

### **29. Differentiate between ng-Class and ng-Style.**

In ng-Class, loading of CSS class is possible; whereas, in ng-Style we can set the CSS style.

### **30. Why Typescript with Angular?**

Typescript is a superset of Javascript. Earlier, Javascript was the only client side language supported by all browsers. But, the problem with Javascript is, it is not a pure Object Oriented Programming Language. The code written in JS without following patterns like Prototype Pattern, becomes messy and finally leading to difficulties in maintainability and reusability. Instead of learning concepts (like patterns) to maintain code, programmers prefer to maintain the code in a OOP approach and is made avilable with a programming language like Typescript was thus developed by Microsoft in a way that it can work as Javascript and also offer what javascript cannot ie;

* pure OOPS as Typescript offers concepts like Generics, Interfaces and Types (a Static Typed Language) which makes it is easier to catch incorrect data types passing to variables.
* TS provides flexibility to programmers experienced in java, .net as it offers encapsulation through classes and interfaces.
* JS version ES5 offers features like Constructor Function, Dynamic Types, Prototypes. The next version of Javascript ie ES6 introduced new feature like Class keyword but not supported by many browsers.
* TS offers Arrow Functions (=>) which is an ES6 feature not supported by many browsers directly but when used in TS, gets compiled into JS ES5 and runs in any browser.
* TS is not the only alternative to JS, we have CoffeScript, Dart(Google).
* Finally, it is like, TS makes life easier when compared to JS.

1. Angular 4 vs. Angular 2

There is no path breaking difference between angular 2 and angular 4. Angular 4 is simply the next version of Angular 2. The underlying concepts are same. If you know Angular 2, you can easily switch to Angular 4.  
  
Angular 4 is **backward compatible** with Angular 2 for most Applications.   
  
There are some under the hood changes to reduce the **size of the AOT(Ahead-of-Time)** compiler generated code. Migrating to Angular 4 may reduce production bundle.  
  
Angular 4 is **faster** than Angular 2. The apps developed in Angular 4 are five times faster and smaller as compared to Angular 2.  
  
**TypeScript 2.1 and 2.2 compatiblity**. Before Angular 4, only TypeScript 1.8 was supported. So with Angular 4, we have all new features of Typescript available.  
  
Unlike Angular 2, the **animations** have been pulled out of @angular/core and are moved into their package in the Angular 4. If you don't use animations, this extra code will not end up in the production bundle.  
  
We can now use new if/else style syntax with **\*ngif structural directive**. In Angular 2, it was not possible to use "else" statement with nglf, but now it has been made possible in Angular 4.   
  
Angular 4 has introduced a new **titlecase pipe**. It changes the first letter of each word into uppercase  
  
"**As**" keyword is the new addition to the template syntax to simplify the "let" syntax  
  
Integration of **Angular Universal** - The integration of Angular Universal permits developers to run Angular on a server.  
  
The **template tag is now deprecated**: you should use the "ng-template" tag instead as Angular has its own template tag: ng-template now.

2. Why Angular 4? What’s New in Angular 4?

**Router ParamMap**  
  
Starting from version 4, it is possible to use **paramMap** to get the route- and query-parameter from a related route. The use of **Map** brings advantages in terms of type security.   
  
The old had an unsecure type (type params = {[key: string]: any}) and the value could have all possible types.   
  
The new way provides either a string, or an array of strings related to the used method (paramMap.get(): string and paramMap.getAll(): string[])  
  
**Animations**  
  
Earlier all the functions of animations were the part of @angular/core module, which means the code were always included in the bundle even if animations were not used.  
  
In Angular 4, Animations are to be provided in the module BrowserAnimationsModule from @angular/platform-browser/animations. This avoids creating bundles with unnecessary large sizes.     
  
**ngif**  
  
We can now use new if/else style syntax with \*ngif structural directive.  
  
**NgComponentOutlet**  
  
To build and produce components dynamically at runtime involved relatively much programming work. With the introduction of \*ngComponentOutlet-Directive in Angular 4, it is possible to build dynamic components in a declarative way.    
  
**TypeScript 2.1/2.2**  
  
We have the support of most latest TypeScript versions in Angular 4 which helps in improving the speed of ngc-Compiler.  
  
**Angular Universal**  
  
With Angular Universal, it is possible to render Angular applications on the web server. With that, websites can be optimized better for search engines as JavaScript is no longer necessary for initially rendering the page content.

3. What is the use of Interceptors?

Interceptors are used to intercept and/or mutate outgoing requests or incoming responses. It can be really useful for features like caching and logging.  
  
Interceptors can be used on multiple scenarios, i.e. setting the Origin for each outgoing request, adding authentication token to every request etc.

4. What is Angular?

Angular is a framework for building client applications in HTML, CSS and Javascript(or language like Typescript which compiles into Javascript).

5. Why do we need Angular?

We can develop application using Javascript and Jquery. But as the application become complex, code in Javascript and Jquery become difficult to maintain. We then require to structure the application code properly by incorporating object oriented features. That is why a framework like Angular has been evolved to make web application development and maintenance faster and easier. The application in Javascript is hard to test. Applications build in Angular are easily testable.

* Angular gives our application a clean object oriented structure that is easy to understand and easy to maintain.
* Angular come with a lot of utility code that can be reused in various applications.
* Applications build in Angular are easily testable.

6. What is Node js?

It is basically a runtime environment for executing Javascript code outside the browser.

7. Angular CLI

Angular CLI stands for Command-line Interface.

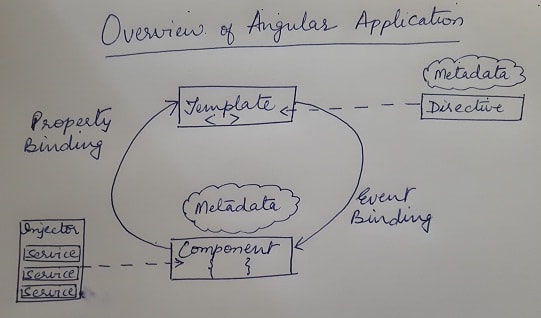
7. What is TypeScript?

TypeScript is a superset of Javascript. So any valid Javascript code is also a TypeScript code. TypeScript has many additional features that Javascript doesn't offer:

* **Strong Type** - You can specify variable type at the time of declaring a variable which makes code easier to maintain and catching errors become easier. Although this feature is optional.
* **Object Oriented features** - TypeScript brings many object oriented features which have been missing in Javascript like classes, interfaces, constructor, access modifier, properties etc.
* In TypeScript, We can catch **error** at compile time instead of runtime
* With TypeScript, We also get **intellisense** in the code editor.

Browser don't understand TypeScript code, TypeScript compiles into JavaScript at the time when we build the application.

8. Architecture of Angular Apps

Angular applications are designed by composing HTML templates with Angularized markup, writing component classes to manage those templates, adding application logic in services, and boxing components and services in modules.  


9. Building blocks of Angular Apps

**Components** - Angular embraces component based architecture which allows us to work with smaller and maintainable piece of code that can be reused at several places. Each Angular App has one or more components. A component controls a patch of screen called a view. It encapsulates Data, HTML template and Logic for a view (area of the screen that the users see).   
  
Every application has a root component that we call as App component.  
  
**Modules** - It is container for group of related components, i.e. in an employee module, we can have components for displaying employees details. Every Angular app has at least one NgModule class, the root module, conventionally named AppModule.  
  
**Templates** - You define a component's view with its companion template. A template is a form of HTML that tells Angular how to render the component.  
  
**Metadata** - Metadata tells Angular how to process a class.  
  
**Example**  
  
In fact, CourseListComponent really is just a class. It's not a component until you tell Angular about it.  
  
To tell Angular that CourseListComponent is a component, attach metadata to the class.  
  
In TypeScript, you attach metadata by using a decorator. Here's some metadata for CourseListComponent:

@Component({  
  selector:    'app-course-list',  
  templateUrl: './course-list.component.html',  
  providers:  [ courseService ]  
})  
  
export class CourseListComponent implements OnInit {  
/\* . . . \*/  
}

**Directives** - A directive is a class with a @Directive decorator. A component is a directive-with-a-template; a @Component decorator is actually a @Directive decorator extended with template-oriented features.  
  
Directives either alter the layout structure (for example, ngSwitch) or modify aspects of DOM elements and components (for example, ngStyle and ngClass).  
  
Of course, you can also write your own directives. Components such as HeroListComponent are one kind of custom directive.  
  
**Services** - A service is typically a class with a narrow, well-defined purpose. Components are big consumers of services. Component classes should be lean. They don't fetch data from the server, validate user input, or log directly to the console. They delegate such tasks to services. A component's job is to enable the user experience and nothing more. It mediates between the view (rendered by the template) and the application logic.   
  
Angular factor your application logic into services and make those services available to components through dependency injection.  
  
**Dependency injection** is a way to supply a new instance of a class with the fully-formed dependencies it requires. Most dependencies are services. Angular uses dependency injection to provide new components with the services they need.

10. Steps to follow in order to use component.

* Create a Component
* Register it in a module
* Add an element in an HTML markup

Angular 2/5 - Communicating Between Components with Observable & Subject

Jun 25 2018 - Updated to **Angular 5.2.11**

For an updated version built with **Angular 6** check out [Angular 6 - Communicating Between Components](http://jasonwatmore.com/post/2018/06/25/angular-6-communicating-between-components-with-observable-subject).

This is a quick post to show an example of something that got me stuck for a little while - how to communicate between components in Angular 2/5.

The solution is to use an Observable and a Subject (which is a type of observable), I won't go too much into the details about how observables work here, but in a nutshell there are two methods that we're interested in: **Observable.subscribe()** and **Subject.next()**.

### Observable.subscribe()

The observable subscribe method is used to subscribe to messages that are sent to an observable.

### Subject.next()

The subject next method is used to send messages to an observable which are then sent to all subscribers of that observable.

# [**What is the difference between ActivatedRoute and ActivatedRouteSnapshot in**](https://stackoverflow.com/questions/46050849/what-is-the-difference-between-activatedroute-and-activatedroutesnapshot-in-angu)

Since ActivatedRoute [can be reused](https://stackoverflow.com/questions/48155804/angular-routing-instance-creation-vs-instance-activation/48157720#48157720), ActivatedRouteSnapshot is an immutable object representing **a particular version** of ActivatedRoute. It exposes all the same properties as ActivatedRoute as plain values, while ActivatedRoute exposes them as observables.

Here is the comment in the implementation:

export class ActivatedRoute {

/\*\* The current snapshot of this route \*/

snapshot: ActivatedRouteSnapshot;

If a router reuses a component and doesn't create a new activated route, you will have two versions of ActivatedRouteSnapshot for the same ActivatedRoute. Suppose you have the following routing configuration:

path: /segment1/:id,

component: AComponent

Now you navigate to:

/segment1/1

You will have the param in the activatedRoute.snapshot.params.id as 1.

Now you navigate to:

/segment1/2

You will have the param in the activatedRoute.snapshot.params.id as 2.

You can see it by implementing the following:

export class AComponent {

constructor(r: ActivatedRoute) {

r.url.subscribe((u) => {

console.log(r.snapshot.params.id);

});

### RouterState and RouterStateSnapshot

During a navigation, after redirects have been applied, the router creates a RouterStateSnapshot. What is RouterStateSnapshot, and how is it different from RouterState?

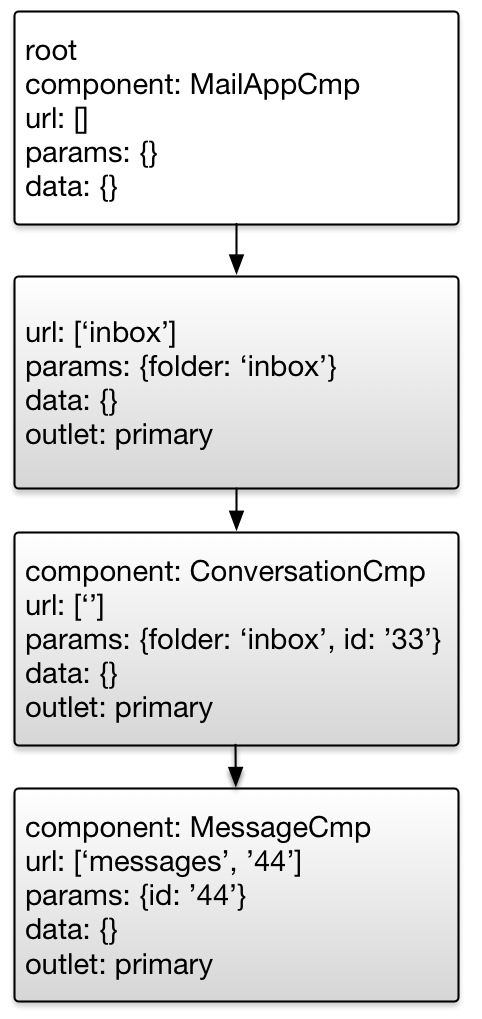
**RouteStateSnapshot is an immutable data structure representing the state of the router at a particular moment in time.** Any time a component is added or removed or parameter is updated, a new snapshot is created.

**RouterState is similar to RouteStateSnapshot, except that it represents the state of the router changing over time.**

### RouterStateSnapshot

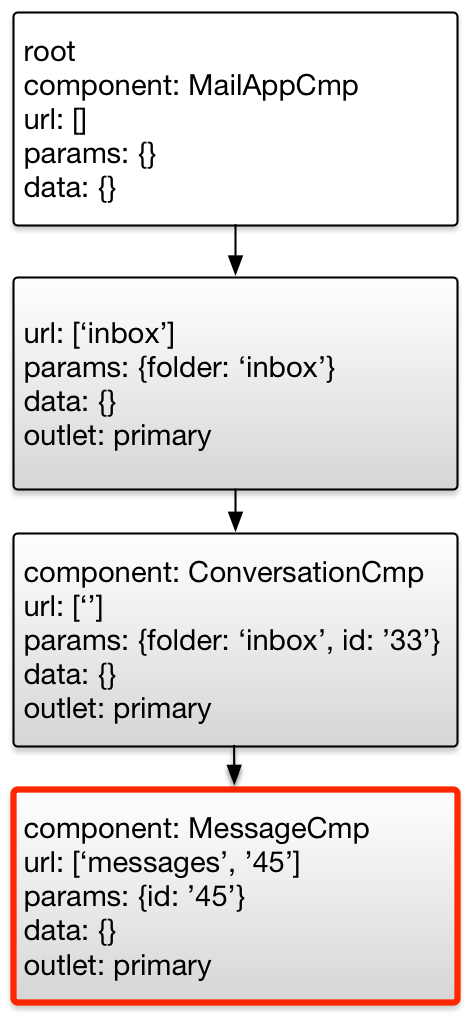
As you can see RouterStateSnapshot is a tree of activated route snapshots. Every node in this tree knows about the “consumed” URL segments, the extracted parameters, and the resolved data. To make it clearer, let’s look at this example:

When we are navigating to “/inbox/33/messages/44”, the router will look at the URL and will construct the following RouterStateSnapshot:



After that the router will instantiate ConversationCmp with MessageCmp in it.

Now imagine we are navigating to a different URL: “/inbox/33/messages/45”, which will result in the following snapshot:



**To avoid unnecessary DOM modifications, the router will reuse the components when the parameters of the corresponding routes change.**In this example, the id parameter of the message component has changed from 44 to 45. This means that we cannot just inject an ActivatedRouteSnapshot into MessageCmp because the snapshot will always have the id parameter set to 44, i.e., it will get stale.

The router state snapshot represents the state of the application at a moment in time, hence the name ‘snapshot’. But components can stay active for hours, and the data they show can change. So having only snapshots won’t cut it — we need a data structure that allows us to deal with changes.

### Introducing RouterState!

RouterState and ActivatedRoute are similar to their snapshot counterparts except that they expose all the values as observables, which are great for dealing with values changing over time.

**Any component instantiated by the router can inject its ActivatedRoute.**

If we navigate from “/inbox/33/messages/44” to “/inbox/33/messages/45”, the data observable will emit a new set of data with the new message object, and the component will display Message 45.

#### Accessing Snapshots

The router exposes parameters and data as observables, which is convenient most of the time, but not always. Sometimes what we want is a snapshot of the state that we can examine at once.

### ActivatedRoute

**ActivatedRoute provides access to the url, params, data, queryParams, and fragment observables.** We will look at each of them in detail, but first let’s examine the relationships between them.

**URL changes are the source of any changes in a route.**And it has to be this way as the user has the ability to modify the location directly.

**Any time the URL changes, the router derives a new set of parameters from it**: the router takes the positional parameters (e.g., ‘:id’) of the matched URL segments and the matrix parameters of the last matched URL segment and combines those. This operation is pure: the URL has to change for the parameters to change. Or in other words, the same URL will always result in the same set of parameters.

**Next, the router invokes the route’s data resolvers and combines the result with the provided static data.** Since data resolvers are arbitrary functions, the router cannot guarantee that you will get the same object when given the same URL. Even more, often this cannot be the case! The URL contains the id of a resource, which is fixed, and data resolvers fetch the content of that resource, which often varies over time.

Finally, the activated route provides the queryParams and fragment observables. In opposite to other observables, that are scoped to a particular route, query parameters and fragment are shared across multiple routes.

#### URL

Given the following:

And navigating first to “/inbox/33/messages/44” and then to “/inbox/33/messages/45”, we will see:

url [{path: ‘messages’, params: {}}, {path: ‘44’, params: {}}]  
url [{path: ‘messages’, params: {}}, {path: ‘45’, params: {}}]

We do not often listen to URL changes as those are too low level. One use case where it can be practical is when a component is activated by a wildcard route. Since in this case the array of URL segments is not fixed, it might be useful to examine it to show different data to the user.

#### Params

Given the following:

And when navigating first to “/inbox/33/messages;a=1/44;b=1” and then to “/inbox/33/messages;a=2/45;b=2”, we will see

params {id: ‘44’, b: ‘1’}  
params {id: ‘45’, b: ‘2’}

First thing to note is that the id parameter is a string (when dealing with URLs we always work with strings). Second, the route gets only the matrix parameters of its last URL segment. That is why the ‘a’ parameter is not present.

#### Data

Let’s tweak the configuration from above to see how the data observable works.

Where MessageResolver is defined as follows:

**The data property is used for passing a fixed object to an activated route.**It does not change throughout the lifetime of the application. **The resolve property is used for dynamic data.**

Note that in the configuration above the line “message: MessageResolver” does not tell the router to instantiate the resolver. It instructs the router to fetch one using dependency injection. This means that you have to register “MessageResolver” in the list of providers somewhere.

Once the router has fetched the resolver, it will call the ‘resolve’ method on it. The method can return a promise, an observable, or any other object. If the return value is a promise or an observable, the router will wait for that promise or observable to complete before proceeding with the activation.

The resolver does not have to be a class implementing the `Resolve` interface. It can also be a function:

The router combines the resolved and static data into a single property, which you can access, as follows:

When navigating first to “/inbox/33/message/44” and then to “/inbox/33/messages/45”, we will see

data {allowReplyAll: true, message: {id: 44, title: ‘Rx Rocks’, …}}  
data {allowReplyAll: true, message: {id: 45, title: ‘Angular Rocks’, …}}

#### Query Params and Fragment

In opposite to other observables, that are scoped to a particular route, query parameters and fragment are shared across multiple routes.

# **RouterOutlet**

DIRECTIVE

Acts as a placeholder that Angular dynamically fills based on the current router state.

1. @[Directive](https://angular.io/api/core/Directive)({ selector: '[router-outlet](https://angular.io/api/router/RouterOutlet)', [exportAs](https://angular.io/api/core/Directive#exportAs): 'outlet' })
2. class [RouterOutlet](https://angular.io/api/router/RouterOutlet) implements [OnDestroy](https://angular.io/api/core/OnDestroy), [OnInit](https://angular.io/api/core/OnInit) {
3. [activateEvents: new EventEmitter<any>()](https://angular.io/api/router/RouterOutlet#activateEvents)
4. [deactivateEvents: new EventEmitter<any>()](https://angular.io/api/router/RouterOutlet#deactivateEvents)
5. [get isActivated: boolean](https://angular.io/api/router/RouterOutlet#isActivated)
6. [get component: Object](https://angular.io/api/router/RouterOutlet#component)
7. [get activatedRoute: ActivatedRoute](https://angular.io/api/router/RouterOutlet#activatedRoute)
8. [get activatedRouteData: Data](https://angular.io/api/router/RouterOutlet#activatedRouteData)
9. [ngOnDestroy(): void](https://angular.io/api/router/RouterOutlet#ngOnDestroy)
10. [ngOnInit(): void](https://angular.io/api/router/RouterOutlet#ngOnInit)
11. [detach(): ComponentRef<any>](https://angular.io/api/router/RouterOutlet#detach)
12. [attach(ref: ComponentRef<any>, activatedRoute: ActivatedRoute)](https://angular.io/api/router/RouterOutlet#attach)
13. [deactivate(): void](https://angular.io/api/router/RouterOutlet#deactivate)
14. [activateWith(activatedRoute: ActivatedRoute, resolver: ComponentFactoryResolver | null)](https://angular.io/api/router/RouterOutlet#activateWith)
15. }

## Selectors

[router-outlet](https://angular.io/api/router/RouterOutlet)

## Outputs

activate bound to [RouterOutlet.activateEvents](https://angular.io/api/router/RouterOutlet" \l "activateEvents)

deactivate bound to [RouterOutlet.deactivateEvents](https://angular.io/api/router/RouterOutlet" \l "deactivateEvents)

## Exported as

outlet

## Description

content\_copy<[router-outlet](https://angular.io/api/router/RouterOutlet)></[router-outlet](https://angular.io/api/router/RouterOutlet)>

<[router-outlet](https://angular.io/api/router/RouterOutlet) name='left'></[router-outlet](https://angular.io/api/router/RouterOutlet)>

<[router-outlet](https://angular.io/api/router/RouterOutlet) name='right'></[router-outlet](https://angular.io/api/router/RouterOutlet)>

A router outlet will emit an activate event any time a new component is being instantiated, and a deactivate event when it is being destroyed.

content\_copy<[router-outlet](https://angular.io/api/router/RouterOutlet)

(activate)='onActivate($event)'

(deactivate)='onDeactivate($event)'></[router-outlet](https://angular.io/api/router/RouterOutlet)>

## Properties

| **Property** | **Type** | **Description** |
| --- | --- | --- |
| activateEvents | new [EventEmitter](https://angular.io/api/core/EventEmitter)<any>() |  |
| deactivateEvents | new [EventEmitter](https://angular.io/api/core/EventEmitter)<any>() |  |
| isActivated | boolean | *Read-only.* |
| component | Object | *Read-only.* |
| activatedRoute | [ActivatedRoute](https://angular.io/api/router/ActivatedRoute) | *Read-only.* |
| activatedRouteData | [Data](https://angular.io/api/router/Data) | *Read-only.* |

## Methods

| ngOnDestroy() [mode\_edit](https://github.com/angular/angular/edit/master/packages/router/src/directives/router_outlet.ts?message=docs(router)%3A%20describe%20your%20change...#L55-L56)[code](https://github.com/angular/angular/tree/6.1.7/packages/router/src/directives/router_outlet.ts#L55-L56) |
| --- |
| ngOnDestroy(): void Parameters There are no parameters. Returns void |

| ngOnInit() [mode\_edit](https://github.com/angular/angular/edit/master/packages/router/src/directives/router_outlet.ts?message=docs(router)%3A%20describe%20your%20change...#L57-L73)[code](https://github.com/angular/angular/tree/6.1.7/packages/router/src/directives/router_outlet.ts#L57-L73) |
| --- |
| ngOnInit(): void Parameters There are no parameters. Returns void |

| detach() [mode\_edit](https://github.com/angular/angular/edit/master/packages/router/src/directives/router_outlet.ts?message=docs(router)%3A%20describe%20your%20change...#L93-L104)[code](https://github.com/angular/angular/tree/6.1.7/packages/router/src/directives/router_outlet.ts#L93-L104) |
| --- |
| Called when the [RouteReuseStrategy](https://angular.io/api/router/RouteReuseStrategy) instructs to detach the subtree |
| detach(): [ComponentRef](https://angular.io/api/core/ComponentRef)<any> Parameters There are no parameters. Returns [ComponentRef](https://angular.io/api/core/ComponentRef)<any> |

| attach() [mode\_edit](https://github.com/angular/angular/edit/master/packages/router/src/directives/router_outlet.ts?message=docs(router)%3A%20describe%20your%20change...#L105-L113)[code](https://github.com/angular/angular/tree/6.1.7/packages/router/src/directives/router_outlet.ts#L105-L113) |
| --- |
| Called when the [RouteReuseStrategy](https://angular.io/api/router/RouteReuseStrategy) instructs to re-attach a previously detached subtree |
| attach(ref: [ComponentRef](https://angular.io/api/core/ComponentRef)<any>, [activatedRoute](https://angular.io/api/router/RouterOutlet#activatedRoute): [ActivatedRoute](https://angular.io/api/router/ActivatedRoute)) Parameters  |  |  | | --- | --- | | ref | Type: [ComponentRef](https://angular.io/api/core/ComponentRef). | | activatedRoute | Type: [ActivatedRoute](https://angular.io/api/router/ActivatedRoute). | |

| deactivate() [mode\_edit](https://github.com/angular/angular/edit/master/packages/router/src/directives/router_outlet.ts?message=docs(router)%3A%20describe%20your%20change...#L114-L123)[code](https://github.com/angular/angular/tree/6.1.7/packages/router/src/directives/router_outlet.ts#L114-L123) |
| --- |
| deactivate(): void Parameters There are no parameters. Returns void |

| activateWith() [mode\_edit](https://github.com/angular/angular/edit/master/packages/router/src/directives/router_outlet.ts?message=docs(router)%3A%20describe%20your%20change...#L124-L141)[code](https://github.com/angular/angular/tree/6.1.7/packages/router/src/directives/router_outlet.ts#L124-L141) |
| --- |
| activateWith([activatedRoute](https://angular.io/api/router/RouterOutlet" \l "activatedRoute): [ActivatedRoute](https://angular.io/api/router/ActivatedRoute), [resolver](https://angular.io/api/router/OutletContext#resolver): [ComponentFactoryResolver](https://angular.io/api/core/ComponentFactoryResolver) | null) Parameters  |  |  | | --- | --- | | activatedRoute | Type: [ActivatedRoute](https://angular.io/api/router/ActivatedRoute). | | resolver | Type: [ComponentFactoryResolver](https://angular.io/api/core/ComponentFactoryResolver) | null. | |