**Missing Angular**

1. Difference between ng-content,ng-template and ng-container?

**Ans:** ng-content is used to project content into Angular components.

Imagine If you are Passing Data from Parent to Child. Normally we will use Input property binding and we will get the data in child ts and we will bind the data. Instead of doing that ng-content we can place the content inside component tag from parent and bind your parent data and use ng-content inside child so that we can avoid input property binding

**Parent Component**

<parent>

<child>

<div class="head">{{heading}}</div>

</child>

</parent>

**child component**

<ng-content select=".head"></ng-content>

ng-container is an extremely simple directive that allows you to group elements in a template but doesn’t itself get rendered in the DOM (but what it contains does). It just becomes a comment:

It is especially helpful when you want to apply two structural directives to the same element.

ng-template is used to render HTML in a template but is never rendered directly. If you add a ng-template tag to your template, it and everything inside it will be replaced by a comment.

It can be for example used to define the else case of an \*ngIf.

|  |
| --- |
| <div> |
|  | Hello word! |
|  | <div \*ngIf="false else content"> |
|  | Shouldnt be displayed |
|  | </div> |
|  | </div> |
|  |  |
|  | <ng-template #content> |
|  | Should be displayed |
|  | </ng-template> |

To sum up, ng-content is used to display children in a template, ng-container is used as a non-rendered container to avoid having to add a span or a div, and ng-template allows you to group some content that is not rendered directly but can be used in other places of your template or you code

1. **What are the advantages with AOT?**

**Faster rendering, Fewer asynchronous requests,** **Smaller Angular framework download size,** **Detect template errors earlier**

### Can I use arrow functions in AOT?

No.

### What is the purpose of metadata json files?

The metadata.json file can be treated as a diagram of the overall structure of a decorator's metadata, represented as an abstract syntax tree(AST). During the analysis phase, the AOT collector scan the metadata recorded in the Angular decorators and outputs metadata information in .metadata.json files, one per .d.ts file.

### What is folding?

This process is used in aot compliation process. Aot have different phases and folding is used in code analysis process.

**Folding**: Non-exported members are folded while generating the code. For example,

|  |
| --- |
| let selector = 'app-root';  @Component({    selector: selector  }) |

Gets folded into:

|  |
| --- |
| @Component({    selector: 'app-root'  }) |

But not everything is foldable. The compiler can’t fold spread operator on arrays, objects created using new keywords and function calls.

### What are macros?

The AOT compiler supports macros in the form of functions or static methods that return an expression in a single return expression

### What is Non null type assertion operator?

! is [non-null assertion operator](https://www.typescriptlang.org/docs/handbook/release-notes/typescript-2-0.html#non-null-assertion-operator) (post-fix expression) - it just saying to type checker that **you're** sure that a is not null or undefined.

the operation a! produces a value of the type of a with null and undefined excluded

### What is codelyzer?

**Codelyzer** is a tool **used** for static analysis of the Angular TypeScript projects. This tool sits on the top of TSLint and checks if the Angular TypeScript projects follow the set of linting rules. It has over 50 rules to check the Angular projects. You can add this tool using the npm package manager to your project.

**Angular Animations Learn Pending**

### What is State function?

### What is Style function?

### What is the purpose of animate function?

### How do you create app shell in Angular?

An App shell is a way to render a portion of your application via a route at build time. This is useful to first paint of your application that appears quickly because the browser can render static HTML and CSS without the need to initialize JavaScript. You can achieve this using Angular CLI which generates an app shell for running server-side of your app.

ng generate appShell [options] (or)

ng g appShell [options]

### How do you run Bazel directly?

Sometimes you may want to bypass the Angular CLI builder and run Bazel directly using Bazel CLI. You can install it globally using @bazel/bazel npm package. i.e, Bazel CLI is available under @bazel/bazel package. After you can apply the below common commands,

bazel build [targets] // Compile the default output artifacts of the given targets.

### What is lazy loading?

Lazy loading is one of the most useful concepts of Angular Routing. It helps us to download the web pages in chunks instead of downloading everything in a big bundle. It is used for lazy loading by asynchronously loading the feature module for routing whenever required using the property loadChildren.

### What is Angular Material?

Angular Material is a collection of Material Design components for Angular framework following the Material Design spec. You can apply Material Design very easily using Angular Material. The installation can be done through npm or yarn,

npm install --save @angular/material @angular/cdk @angular/animations

(OR)

yarn add @angular/material @angular/cdk @angular/animations

### How to use polyfills in Angular application?

The Angular CLI provides support for polyfills officially. When you create a new project with the ng new command, a src/polyfills.ts configuration file is created as part of your project folder. This file includes the mandatory and many of the optional polyfills as JavaScript import statements. Let's categorize the polyfills,

1. **Mandatory polyfills:** These are installed automatically when you create your project with ng new command and the respective import statements enabled in 'src/polyfills.ts' file.
2. **Optional polyfills:** You need to install its npm package and then create import statement in 'src/polyfills.ts' file. For example, first you need to install below npm package for adding web animations (optional) polyfill. bash npm install --save web-animations-js and create import statement in polyfill file. javascript import 'web-animations-js';

### What is Angular security model for preventing XSS attacks?

Angular treats all values as untrusted by default. i.e, Angular sanitizes and escapes untrusted values When a value is inserted into the DOM from a template, via property, attribute, style, class binding, or interpolation.

### What is Sanitization? Is angular supports it?

**Sanitization** is the inspection of an untrusted value, turning it into a value that's safe to insert into the DOM. Yes, Angular suppports sanitization. It sanitizes untrusted values for HTML, styles, and URLs but sanitizing resource URLs isn't possible because they contain arbitrary code.

### What is the difference between interpolated content and innerHTML?

The main difference between interpolated and innerHTML code is the behavior of code interpreted. Interpolated content is always escaped i.e, HTML isn't interpreted and the browser displays angle brackets in the element's text content. Where as in innerHTML binding, the content is interpreted i.e, the browser will convert < and > characters as HTMLEntities. For example, the usage in template would be as below,

### Is angular prevents http level vulnerabilities?

Angular has built-in support for preventing http level vulnerabilities such as as cross-site request forgery (CSRF or XSRF) and cross-site script inclusion (XSSI). Even though these vulnerabilities need to be mitigated on server-side, Angular provides helpers to make the integration easier on the client side.

1. HttpClient supports a token mechanism used to prevent XSRF attacks
2. HttpClient library recognizes the convention of prefixed JSON responses(which non-executable js code with ")]}',\n" characters) and automatically strips the string ")]}',\n" from all responses before further parsing

### Is multiple interceptors supported in Angular?

Yes, Angular supports multiple interceptors at a time. You could define multiple interceptors in providers property:

providers: [

{ provide: HTTP\_INTERCEPTORS, useClass: MyFirstInterceptor, multi: true },

{ provide: HTTP\_INTERCEPTORS, useClass: MySecondInterceptor, multi: true }

],

The interceptors will be called in the order in which they were provided. i.e, MyFirstInterceptor will be called first in the above interceptors configuration.

### What is an angular library?

A library is a collection of components, services, directives, etc. that can be shared across different Angular projects.

More precisely, in programming, the Library is a collection of precompiled routines that a program can use.

An Angular library is an Angular project that differs from an app in that it cannot run on its own. A library must be imported and used in an angular app.

So, an Angular library is a Sharable code which provides Reusable functionality.

# Why we need a Library?

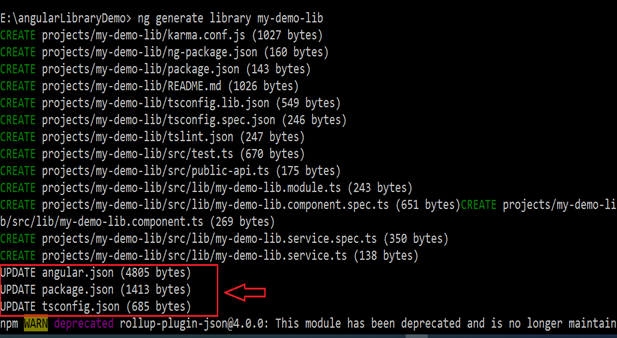
Now the main question is why we need this library? Do we need this? Well yes,

Sometimes components, services, etc could apply to multiple angular projects. So to re-use this code easily across different apps, this concept of creating our Angular library was introduced.

These libraries can be used locally in your workspace, or you can publish them as **npm packages** to share with other projects or with other Angular developers across the globe.

# Creating a Library:

ng generate library <library-name>



Now have a look at the **public-api.ts** file inside the src folder.

This is the **entry point** for our library project. Whatever the component, Services, Directives, etc you want to expose to the outside world needs to export here. By doing this anyone can use these by importing it.

**Using a Library in Local Angular Application:**

To use this library anywhere we first need to BUILD it. We cannot use a library without building a **dist** folder. So for this open your terminal and write the following command:

Syntax: ng build <library-name>

This command will **create a dist folder** and inside this folder, our compiled library is stored.

Now we can use this compiled library in any angular application.

As well as do not forget to export it to the **public-api.ts** file also.



**For more info:**

[**https://medium.com/@jinalshah999/how-to-build-and-publish-angular-library-a333d6ff12c0#:~:text=An%20Angular%20library%20is%20an,code%20which%20provides%20Reusable%20functionality**](https://medium.com/@jinalshah999/how-to-build-and-publish-angular-library-a333d6ff12c0#:~:text=An%20Angular%20library%20is%20an,code%20which%20provides%20Reusable%20functionality)**.**

### What is the reason for No provider for HTTP exception?

This exception is due to missing HttpClientModule in your module. You just need to import in module as below

### What is safe navigation operator?

The **Safe Navigation Operator** is also known as the "**Elvis Operator**". This **operator** is very useful to protect against null and undefined values in property paths. This **operator** allows us to **navigate** an object path in situations when we are not aware whether a path exists or not. It is very useful to prevent null-reference exceptions.

1. <div> Hi, This Text will generate error!!! {{data.firstName}} </div>

<div> Hi, This Text will not generating error Console!!! {{data?.firstName}} </div>

**Non-empty assertion operator**Will not prevent null or undefined. It simply tells TypeScript's type checker to do "strict null detection" for specific property expressions.

### What are the list of template expression operators?

1. Pipe operator
2. Safe navigation operator
3. Non-null assertion operator

### Is it necessary for bootstrapped component to be entry component?

Yes, the bootstrapped component needs to be an entry component. This is because the bootstrapping process is an imperative process.

### What is a routed entry component?

The components referenced in router configuration are called as routed entry components. This routed entry component defined in a route definition as below,

const routes: Routes = [

{

path: '',

component: TodoListComponent // router entry component

}

];

### Do I still need to use entryComponents array in Angular9?

No. In previous angular releases, the entryComponents array of ngModule decorator is used to tell the compiler which components would be created and inserted dynamically in the view. In Angular9, this is not required anymore with Ivy.

### Give few examples for NgModules?

The Angular core libraries and third-party libraries are available as NgModules.

1. Angular libraries such as FormsModule, HttpClientModule, and RouterModule are NgModules.
2. Many third-party libraries such as Material Design, Ionic, and AngularFire2 are NgModules.

### What are feature modules?

Feature modules are NgModules, which are used for the purpose of organizing code. The feature module can be created with Angular CLI using the below command in the root directory, // for lazy loading

ng generate module MyCustomFeature //

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

import { CommonModule } from '@angular/common';

@NgModule({

imports: [

CommonModule

],

declarations: []

})

export class MyCustomFeature { }

### What happens if browserModule used in feature module?

If you do import BrowserModule into a lazy loaded feature module, Angular returns an error telling you to use CommonModule instead. Because BrowserModule’s providers are for the entire app so it should only be in the root module, not in feature module. Whereas Feature modules only need the common directives in CommonModule.

### How do you restrict provider scope to a module?

It is possible to restrict service provider scope to a specific module instead making available to entire application. There are two possible ways to do it.

1. **Using providedIn in service:**
2. import { Injectable } from '@angular/core';
3. import { SomeModule } from './some.module';
4. @Injectable({
5. providedIn: SomeModule,
6. })
7. export class SomeService {

}

### How do you provide a singleton service?

There are two possible ways to provide a singleton service.

1. Set the providedIn property of the @Injectable() to "root". This is the preferred way(starting from Angular 6.0) of creating a singleton service since it makes your services tree-shakable.
2. import { Injectable } from '@angular/core';
3. @Injectable({
4. providedIn: 'root',
5. })
6. export class MyService {

}

### Can I share services using modules?

No, it is not recommended to share services by importing module. i.e Import modules when you want to use directives, pipes, and components only. The best approach to get a hold of shared services is through 'Angular dependency injection' because importing a module will result in a new service instance.

### How do you create displayBlock components?

By default, Angular CLI creates components in an inline displayed mode(i.e, display:inline). But it is possible to create components with display: block style using displayBlock option,

ng generate component my-component --displayBlock

### What is a zone context?

Execution Context is an abstract concept that holds information about the environment within the current code being executed. A zone provides an execution context that persists across asynchronous operations is called as zone context. For example, the zone context will be same in both outside and inside setTimeout callback function,

zone.run(() => {

// outside zone

expect(zoneThis).toBe(zone);

setTimeout(function() {

// the same outside zone exist here

expect(zoneThis).toBe(zone);

});

});

The current zone is retrieved through Zone.current.

### What are the lifecycle hooks of a zone?

**onScheduleTask:** This hook triggers when a new asynchronous task is scheduled. For example, when you call setTimeout()

**onInvokeTask:** This hook triggers when an asynchronous task is about to execute. For example, when the callback of setTimeout() is about to execute.

**onHasTask:** This hook triggers when the status of one kind of task inside a zone changes from stable(no tasks in the zone) to unstable(a new task is scheduled in the zone) or from unstable to stable.

**onInvoke:** This hook triggers when a synchronous function is going to execute in the zone

### What are the methods of NgZone used to control change detection?

NgZone service provides a run() method that allows you to execute a function inside the angular zone. This function is used to execute third party APIs which are not handled by Zone and trigger change detection automatically at the correct time.

Whereas runOutsideAngular() method is used when you don't want to trigger change detection.

### What are template driven forms?

Template driven forms are model-driven forms where you write the logic, validations, controls etc, in the template part of the code using directives. They are suitable for simple scenarios and uses two-way binding with [(ngModel)] syntax.

### Can you give an example of built-in validators?

In reactive forms, you can use built-in validator like required and minlength on your input form controls. For example, the registration form can have these validators on name input field

### How do you verify the model changes in forms?

You can add a getter property(let's say, diagnostic) inside component to return a JSON representation of the model during the development. This is useful to verify whether the values are really flowing from the input box to the model and vice versa or not.

get diagnostic() { return JSON.stringify(this.model); }

### How do you update specific properties of a form model?

You can also use setValue method to update properties.

### How to set ngFor and ngIf on the same element?

In this case, You need to use either ng-container or ng-template. Let's say if you try to loop over the items only when the items are available, the below code throws an error in the browser

# [**Component host property**](https://stackoverflow.com/questions/41161088/angular2-component-host-property)

The host property is used to bind properties, attributes, and events to that particular class component, using a set of key-value pairs.

* property '[propName]':'expr'
* attribute '[attr.attrName]':'expr'
* event (event)="someFunction(event);otherExpr",
* style [style.width]="booleanExpr"
* class [class.className]="booleanExpr" binding.
* class [class]="expr" where expr is a string with space separated classes

I have added class to host tag.

Like following:

* Component
* @Component({
* selector: 'mytag',
* templateUrl: './layout.template.html',
* host: {
* 'class' : 'myclass1 myclass2 myclass3'
* }
* })
* export class MyTagComponent {
* View code

<mytag></mytag>

* Result

<mytag class="myclass1 myclass2 myclass3"></mytag>

What is the use of angular CDK?

The **Angular** Component Dev Kit (**CDK**) is a library of predefined behaviors included in **Angular** Material, a UI component library for **Angular** developers. It contains reusable logic for many features that are **used** as common building blocks for the interfaces of **Angular** applications

The overlay package provides a way to open floating panels on the screen.

@import '~@angular/cdk/overlay-prebuilt.css';

Features of cdk

Clipboard

Drag and Drop

Platform : A set of utilities that gather information about the current platform and the different features it supports.

Portal : The portals package provides a flexible system for rendering dynamic content into an application.

Scroll

Table

## ***What is the purpose of metadata json files?***

The metadata.json file can be treated as a diagram of the overall structure of a decorator metadata, represented as an abstract syntax tree(AST). During the analysis phase, the AOT collector scan the metadata recorded in the Angular decorators and outputs metadata information in .metadata.json files, one per .d.ts file.

## ***What is hammerjs in angular?***

**HammerJS** gives us access to mobile gesture events that are not normally found in the browser, including tap, swipe, pan, pinch, press, and rotate.

**Babel** is a JavaScript transpiler that converts edge JavaScript into plain old ES5 JavaScript that can run in any browser (even the old ones). It makes available all the syntactical sugar that was added to JavaScript with the new ES6 specification, including classes, fat arrows and multiline strings.

Does angular use Babel?

**angular**-cli uses **es6** modules as output format of TypeScript: In tsconfig. json: ... If I understand it correctly, many projects **use babel** to get **es6** modules working, but I have not found any reference to **babel** inside **angular**-cli.

## ***How to optimize loading large data in angular?***

**AOT, Tree-shaking**, **Uglify,** **Lazy loading,** **Ivy Render Engine,** **RxJS**

## ***How to transfer components to custom elements?***

Transforming components to custom elements involves **two** major steps,

1. **Build custom element class** Angular provides the createCustomElement() function for converting an Angular component (along with its dependencies) to a custom element. The conversion process implements NgElementConstructor interface, and creates a constructor class which is used to produce a self-bootstrapping instance of Angular component.
2. **Register element class with browser** It uses customElements.define() JS function, to register the configured constructor and its associated custom-element tag with the browser CustomElementRegistry. When the browser encounters the tag for the registered element, it uses the constructor to create a custom-element instance.

## ***How to create multiple router-outlets?***

<router-outlet></router-outlet>

<router-outlet name="sidebar"></router-outlet>

{

path: "",

component: SidebarComponent,

outlet: "sidebar"

## ***What are the difference between****@Inject()****and****@Injectable()****?***

The @Injectable() decorator aims to actually set some metadata about which dependencies to inject into the constructor of the associated class. It is a class decorator that does not require parameters. Without this decorator no dependency will be injected.

@Injectable()

export class SomeService {

constructor(private http:Http) {

}

}

The @Inject decorator must be used at the level of constructor parameters to specify metadata regarding elements to inject. Without it, the type of parameters is used (obj:SomeType is equivalent to @Inject(SomeType) obj).

@Injectable()

export class SomeService {

constructor(@Inject(Http) private http:Http, @Inject('sometoken') obj) {

}

}

## ***How do you listen for events in a component?***

HostListener or via elementRef.nativeElement

## ***What is Self and Host Decorator in Angular?***

**@Host()**: The @Host() decorator tells DI to look for a dependency in any injector until it reaches the host.

**@Self()**: The @Self() decorator tells DI to look for a dependency only from itself, so it will not walk up the tree.

[What is the difference between formControlName and FormControl?](https://github.com/learning-zone/angular-interview-questions#q-what-is-the-difference-between-formcontrolname-and-formcontrol)

I believe you missed an important point: [formGroup] directive in the second example. formControlName is used together with [formGroup] to save your form multiple dot navigations. For example:

<div>

<input type="text" [formControl]="myForm.controls.firstName"/>

<input type="text" [formControl]="myForm.controls.lastName"/>

<input type="text" [formControl]="myForm.controls.email"/>

<input type="text" [formControl]="myForm.controls.title"/>

</div>

Is equivalent to:

<div [formGroup]="myForm">

<input type="text" formControlName="firstName"/>

<input type="text" formControlName="lastName"/>

<input type="text" formControlName="email"/>

<input type="text" formControlName="title"/>

</div>

NGForm:

ut using **ngForm** inside your template directly you are using the strategy **Template-Driven**.

Simply we can say

**Reactive form** can be used in the following situation

* Complex forms with more number of fields.
* Multiple complex validation are there. Custom validations are required
* Require JSON structure to be send with the values in the form.

We can get entire form in a structured way by using "form.value"

If we have 4 fields *First Name, Last Name, Email, Phone Number* in reactive form.

HTML code will be

<form [formGroup]="form">

First Name <input formControlName="firstName">

Last Name <input formControlName="lastName">

Email <input formControlName="email">

Phone Number <input formControlName="phoneNumber">

</form>

We can get the values from the form like below

{

"firstName": "FName",

"lastName": "LName",

"email": "test@123.com",

"phoneNumber": "123"

}

by calling form.value, where form is FormGroup Variable that we created.

**Template Driven Form** : It can be used when using simple forms. Like login page. With the two way data binding. We can simply assign value to variable from ui and vice versa.

Simple example is if we are givng two way binding for the below input.

<input [(ngModel)]="username">

We can simply display the value that user is giving in the UI.

<p>Hello {{username}}!</p>

## ***How to cache an observable data?***

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

import { Http, Response } from '@angular/http';

import { Observable, ReplaySubject } from 'rxjs';

@Injectable()

export class CachedService {

data$: Observable<Response> = this.dataSubject.asObservable();

private dataSubject = new ReplaySubject<Response>(1);

constructor(private http: Http) { }

fetch() {

this.http.get(...).subscribe(res => this.dataSubject.next(res));

}

}

This will make an HTTP call when the fetch method is called, and any subscribers to service.data$ ($ as a suffix basically represents a stream of values) will get the response from the ReplaySubject. As it replays earlier values, any subscribers who join after the HTTP call resolves will still get the previous response.

If we want to trigger an update, we can just call service.fetch() to kick off a new HTTP call and all subscribers will be updated once the new response arrives.

@Component({ ... })

export class SomeComponent implements OnInit {

constructor(private service: CachedService) { }

ngOnInit() {

this.service.fetch();

this.service.data$.subscribe(...);

}

}

## ***What is RxJS BehaviorSubject, ReplaySubject and AsyncSubject in angular?***

**BehaviorSubject**: It has the characteristic that it stores the **current** value. This means that we can always directly get the last emitted value from the BehaviorSubject. We can either get the value by accessing the **.value** property on the BehaviorSubject or we can subscribe to it.

**ReplaySubject**: It can send **old** values to new subscribers. It however has the extra characteristic that it can record a part of the observable execution and therefore store multiple old values and **replay** them to new subscribers.

When creating the ReplaySubject we can specify how much values want to store and for how long want to store them.

**AsyncSubject**: It is a Subject variant where only the last value of the Observable execution is sent to its subscribers, and only when the execution completes.

## ***What is the difference between RouterModule.forRoot() and RouterModule.forChild()?***

* **RouterModule.forRoot(ROUTES)**: forRoot creates a module that contains all the directives, the given routes, and the router service itself.
* **RouterModule.forChild(ROUTES)**: forChild creates a module that contains all the directives and the given routes, but does not include the router service.
* **forRoot()**: service register to entire application
* **forChild()**: service register to particular child component

## ***What is auxiliary routes in angular?***

Angular supports the concept of auxiliary routes, which allow to set up and navigate multiple independent routes in a single app. Each component has one primary route and zero or more auxiliary outlets. Auxiliary outlets must have unique name within a component.

To define the auxiliary route we must first add a named router outlet where contents for the auxiliary route are to be rendered.  
Example:

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

@Component({

selector: 'app',

template: `

<nav>

<a [routerLink]="['/component-one']">Component One</a>

<a [routerLink]="['/component-two']">Component Two</a>

<a [routerLink]="[{ outlets: { 'sidebar': ['component-aux'] } }]">Component Aux</a>

</nav>

<div style="color: green; margin-top: 1rem;">Outlet:</div>

<div style="border: 2px solid green; padding: 1rem;">

<router-outlet></router-outlet>

</div>

<div style="color: green; margin-top: 1rem;">Sidebar Outlet:</div>

<div style="border: 2px solid blue; padding: 1rem;">

<router-outlet name="sidebar"></router-outlet>

</div>

`

})

export class AppComponent { }

{ path: 'component-aux', component: ComponentAux, outlet: 'sidebar' },

## ***How can you cancel a router navigation?***

canDeactivate guard

## ***What does forwardRef do?***

Allows to refer to references which are not yet defined.

For instance, forwardRef is used when the token which we need to refer to for the purposes of DI is declared, but not yet defined. It is also used when the token which we use when creating a query is not yet defined.  
Example:

@Component({

// ....

providers: [

{ provide: SourceComponent, useExisting: forwardRef(() => TargetComponent) }

// ^^ not defined yet

]

})

export class TargetComponent extends SourceComponent implements OnInit {

// ^^ declared here lower in source code

}

## ***How will you intercept http to inject header to each http call?***

equest = request.clone({headers: request.headers.set('Content-Type', 'application/json')});

[What is the difference between switchMap, concatMap and mergeMap?](https://github.com/learning-zone/angular-interview-questions#q-what-is-the-difference-between-switchmap-concatmap-and-mergemap)

The map operator is the most common of all. For each value that the Observable emits you can apply a function in which you can modify the data. The return value will, behind the scenes, be reemitted as an Observable again so you can keep using it in your stream. It works pretty much the same as how you would use it with Arrays

MergeMap essentially is a combination of mergeAll and map. MergeAll takes care of subscribing to the ‘inner’ Observable so that we no longer have to Subscribe two times as mergeAll merges the value of the ‘inner’ Observable into the ‘outer’ Observable. This could look like this:

SwitchMap has similar behaviour in that it will also subscribe to the inner Observable for you. However switchMap is a combination of switchAll and map. SwitchAll cancels the previous subscription and subscribes to the new one

# ConcatMap

The last example is concatMap. As you might expect, concatMap also subscribes to the inner Observable for you. But unlike switchMap, that unsubscribes from the current Observable if a new Observable comes in, concatMap will not subscribe to the next Observable until the current one completes. The benefit of this is that the order in which the Observables are emitting is maintained.

# Conclusion

Mapping data to the format you need is a common task. RxJS comes with a few very neat operators that help you get the job done. To recap: map is for mapping ‘normal’ values to whatever format you need it to be. The return value will be wrapped in an Observable again, so you can keep using it in your data stream. When you have to deal with an ‘inner’ Observable it’s easier to use mergeMap, switchMap or concatMap. Use mergeMap if you simply want to flatten the data into one Observable, use switchMap if you need to flatten the data into one Observable but only need the latest value and use concatMap if you need to flatten the data into one Observable and the order is important to you.

**Difference Between MergeMap and ForkJoin**

ForkJoin will return an array of responses when all requests have finished.   
  
ForkJoin operator faces some problems with delayed requests too. The order will be preserved but if one request is delayed all the others have to wait for its resolution.  
  
Another issue with forkJoin() is related to the way it handles failed requests. If any of the executed requests fails it will fail for the whole collection. Instead of items we will receive first encountered exception.  
  
MergeMap() executes requests in parallel and it is fault tolerant so we still display most of the posts even if some of the requests fail. Order is not maintained in MergeMap.

<https://medium.com/@luukgruijs/understanding-rxjs-map-mergemap-switchmap-and-concatmap-833fc1fb09ff#:~:text=Use%20mergeMap%20if%20you%20simply,order%20is%20important%20to%20you>.

## ***What is the difference between an observable and a subject?***

A RxJS Subject is a special type of Observable that allows values to be multicasted to many Observers. While plain Observables are unicast (each subscribed Observer owns an independent execution of the Observable), Subjects are multicast.

A Subject is like an Observable, but can multicast to many Observers. Subjects are like EventEmitters: they maintain a registry of many listeners.

## ***What is transclusion in angular?***

Transclusion is a way to let you define a fixed view template, and at the same time allow you to define a slot for dynamic content by using ng-content tag.

## ***What is Traceur Compiler?***

Traceur compiler takes classes, generators, and other features from ECMAScript edition 6 (ES6) and compiles it into JavaScript ES5 that runs on the browser. This means developers can use the code from a future version that has more features and encourages design patterns.

## ***What are rxjs lettable operators?***

* do -> tap
* catch -> catchError
* switch -> switchAll
* finally -> finalize

## ***What selector force a style down through the child component tree into all the child component views?***

Component styles normally apply only to the HTML in the components own template.

Use the ::ng-deep shadow-piercing descendant combinator to force a style down through the child component tree into all the child component views. The ::ng-deep combinator works to any depth of nested components, and it applies to both the view children and content children of the component.

**Example**

The following example targets all <h3> elements, from the host element down through this component to all of its child elements in the DOM.

**parent.component.html**

Parent content.

<app-child></app-child>

**parent.component.css**

::ng-deep h3 {

font-style: italic;

}

**child.component.html**

<h3>Child title</h3>

Child content.

Flex layout angular:

Angular Flex Layout provides a sophisticated layout API using Flexbox CSS + mediaQuery. This module provides Angular developers with component layout features using a custom Layout API, mediaQuery observables, and injected DOM flexbox-2016 CSS stylings.

npm i -s @angular/flex-layout @angular/cdk

|  |  |  |
| --- | --- | --- |
| [fxLayout](https://github.com/angular/flex-layout/wiki/fxLayout-API) | <direction> [wrap] | row | column | row-reverse | column-reverse |
| [fxLayoutAlign](https://github.com/angular/flex-layout/wiki/fxLayoutAlign-API) | <main-axis> <cross-axis> | main-axis: start | center | end | space-around | space-between | space-evenly; cross-axis: start | center | end | stretch | space-between | space-around | baseline |
| [fxLayoutGap](https://github.com/angular/flex-layout/wiki/fxLayoutGap-API) | % | px | vw | vh |  |
| [gdAlignColumns](https://github.com/angular/flex-layout/wiki/gdAlignColumns-API) | <main-axis> <cross-axis> | main-axis: start | center | end | space-around | space-between | space-evenly | stretch; cross-axis: start | center | end | stretch |
| [gdAlignRows](https://github.com/angular/flex-layout/wiki/gdAlignRows-API) | <main-axis> <cross-axis> | main-axis: start | center | end | space-around | space-between | space-evenly | stretch; cross-axis: start | center | end | stretch |
| [gdAreas](https://github.com/angular/flex-layout/wiki/gdAreas-API) | names separated by |, e.g. gdAreas="area1 | area2 | area3" | gdInline for inline-grid |
| [gdAuto](https://github.com/angular/flex-layout/wiki/gdAuto-API) | row | column | dense | row dense | column dense | gdInline for inline-grid |
| [gdColumns](https://github.com/angular/flex-layout/wiki/gdColumns-API) | any valid input for grid-template-columns | gdInline for inline-grid ! at the end means grid-auto-columns |
| [gdRows](https://github.com/angular/flex-layout/wiki/gdRows-API) | any valid input grid-template-rows | gdInline for inline-grid ! at the end means grid-auto-rows |
| [gdGap](https://github.com/angular/flex-layout/wiki/gdGap-API) | % | px | vw | vh | gdInline for inline-grid |

These directives affect the flow and layout children elements in the container

#### API for DOM elements:

| **HTML** | **Allowed values** |
| --- | --- |
| [fxFlex](https://github.com/angular/flex-layout/wiki/fxFlex-API) | "" | px | % | vw | vh | <grow> <shrink> <basis>, |
| [fxFlexOrder](https://github.com/angular/flex-layout/wiki/fxFlexOrder-API) | int |
| [fxFlexOffset](https://github.com/angular/flex-layout/wiki/fxFlexOffset-API) | % | px | vw | vh |
| [fxFlexAlign](https://github.com/angular/flex-layout/wiki/fxFlexAlign-API) | start | baseline | center | end |
| [fxFlexFill, fxFill](https://github.com/angular/flex-layout/wiki/fxFlexFill-API) |  |
| [gdArea](https://github.com/angular/flex-layout/wiki/gdArea-API) | string name for the area as defined in gdAreas |
| [gdColumn](https://github.com/angular/flex-layout/wiki/gdColumn-API) | any valid value for grid-column |
| [gdRow](https://github.com/angular/flex-layout/wiki/gdRow-API) | any valid value for grid-row |
| [gdGridAlign](https://github.com/angular/flex-layout/wiki/gdGridAlign-API) | <row-axis> <cross-axis> |

How to create custom element in angular ?

There are five main reasons to use Angular elements.

## **Embedding Angular Components Into Non-Angular Applications**

With Angular Elements, we can make our components truly reusable. Meaning, you can use Angular Components in other frameworks and libraries, such as React, Vue, and Ember!

1. create a component for example button
2. decalare component into entry component
3. import create custom element api in app.component file
4. const btn = createcustomelement(componentName,{injector})

createcustomelement.define(‘custom-elemnt’, btn)

Then, go to the polyfills.ts file and import the following:

import '@webcomponents/webcomponentsjs/custom-elements-es5-adapter.js';

"package": "jscat ./dist/angular-custom-elements/runtime.js ./dist/angular-custom-elements/polyfills.js ./dist/angular-custom-elements/scripts.js ./dist/angular-custom-elements/main.js > custom-button-element.js",

function add(a, b) {

return a + b;

}

console.log(add.call(null, 1, 2)); // 3

console.log(add.apply(null, [1, 2])); // 3

**canActivate** is used to prevent unauthorized users from accessing certain routes. [See docs](https://angular.io/guide/router#canactivate-requiring-authentication) for more info.

**canLoad** is used to prevent the application from loading entire modules lazily if the user is not authorized to do so.

For ES5 and ES6

<https://blog.alexdevero.com/javascript-arrays/>

Observable Exaample:

// Create a new Observable

        const sqnc = new Observable(countOnetoTen);

        // Execute the Observable and print the

        // result of each notification

        // next() is a call to countOnetoTen method

        // to get the next value from the observable

        sqnc.subscribe({

            next(num) { console.log(num); }

        });

        // This function runs when subscribe()

        // is called

        function countOnetoTen(observer) {

            for(var i = 1; i <= 10; i++) {

                // Calls the next observable

                // notification

                observer.next(i);

            }

            // Unsubscribe after completing

            // the sequence

            return {unsubscribe(){}};

        }

# **Inheritance in JavaScript**

JavaScript inheritance is done through prototypes. You do not define anything with a class keyword, but you make a function that's used as a constructor to build new objects (with the new keyword ).

function person(name) {

this.name = name;

}

person.prototype.getName = function() {

return this.name;

}

var john = new person('john');

alert( john.getName() );

difference-between-regular-functions-and-arrow-functions

Unlike **regular functions**, arrow functions do not have their own this.

Arguments objects are not available in arrow functions, but are available in regular functions.

let user = {

    show(){

        console.log(arguments);

    }

};

user.show(1, 2, 3);

multithreading in js?

**JavaScript** does not support **multi-threading** because the **JavaScript** interpreter in the browser is a single thread

httpBackened: If you use the $http service in your Angular app, then you'll want to make use of the $httpBackend service for unit testing. The $httpBackend service allows you to mock up http requests that your Angular app is making. Instead of hitting an actual server, $httpBackend specifies expected requests and the values they should return.

JavaScript does not support Function Overloading.

Controller to controller communication: using $broadcast and $on

https://www.freecodecamp.org/news/how-to-create-angular-6-custom-elements-web-components-c88814dc6e0a/

**Missing Node**

### What is node JS

### Nodejs is event driven, single threaded and uses the concept of events and callbacks Every call is async in nodejs. In event driven there is generally a main loop that listens the events and then trigger a callback when any of the event is detected. node js is single threaded and on cpu is not handle all the app, so we need to use child procees using child\_process module single thread means only one command at a time.

### What is Event Loop?

### Evnet loop is basically allows to perform non blocking i/o operations. Follows the six phases timers, i/o callbacks, idle, poll, check, close call backs

### What is REPL?

### Repl: Read Eval Print Loop Read − Reads user's input, parses the input into JavaScript data-structure, and stores in memory. Eval − Takes and evaluates the data structure. Print − Prints the result. Loop − Loops the above command until the user presses ctrl-c twice.

### What is aunthetication and authorization?

### Authentication means confirming your own identity while authorization means granting access to the system.

### What is Buffer and Streams?

### The Buffer class was introduced as part of the Node.js API to make it possible to manipulate or interact with streams of binary data. Stream in Node.js simply means a sequence of data being moved from one point to the other over time. The whole concept is, you have a huge amount of data to process, but you don’t need to wait for all the data to be available before you start processing it. Basically, this big data is broken down and sent in chunks. So from the original definition of a buffer (“streams of binary data… in the context of… file system”) this simply means binary data being moved in the file system. For example, moving the texts stored in file1.txt to file2.txt.

### What is the Use of CommonJS?

### CommonJS is basically used for organized the code and uses the modules and requier Modules.exports are the fundamental building blocks of the code structure. The module system allows you to organize your code, hide information and only expose the public interface of a component using module.exports. Every time you use the require call, you are loading another module.

### What is Cluster?

### Cluster” was introduced to scale an application execution on multiple processor cores by creating worker processes. Worker processes share a single port, therefore, requests are routed through a shared port. Examples PM2, RabbitMq

What is Error first Callback?

Error-first callbacks are used to pass error and data. The first argument to these functions is an error object and the subsequent arguments represent associated data. So, you can check the first argument var post = new Post({title: 'test'}); post.save(function(err,savedInstance)

Fs Systems in nodejs

// handle async code fs.readFile('DATA', 'utf8', function(err, contents) { console.log(contents); }); var contents = fs.readFileSync('DATA', 'utf8'); // handle sync code fs.readFileStream('DATA', 'utf8'); // handle large group of data and use less memory