1. **What Is Angular 4 or 5 or 6? What is new in Angular 4?**

* Angular is a most popular web development framework
* Angular is an open source framework written and maintained by angular team at Google
* Angular is written in TypeScript and so it comes with all the capabilities that typescript offers.
* Angular is a framework for building client applications in HTML

1. **What Is Angular Prerequisites?**

Before you can install Angular 4 or 5, you must need to have some prerequisites.

* You must to have Node.js installed.
* You must to have NPM (Node Package Manager) installed.

We need to setup our machine’s local environments which are the following.

* node.js >= 6.9.x
* npm >= 3.x.x
* Visual Studio Code or Atom
* And so on

1. **What Is Angular CLI? How To Updating Angular CLI?**

* The Angular CLI is a tool to initialize, develop, scaffold and maintain Angular applications. The Angular CLI will add reference to components, directives and pipes automatically in “app.module”.
* **Installation:**

npm install -g @angular/cli

* **To update Angular CLI:**

If you're using Angular CLI lesser version, uninstall angular-cli package and install new versions.

|  |
| --- |
| * npm uninstall -g angular-cli * npm uninstall --save-dev angular-cli * npm uninstall -g @angular/cli * npm cache clean * npm install -g @angular/cli@latest |

1. **What Is Architecture Overview of Angular?**

* Angular is a framework for building client applications in HTML and either JavaScript or a language like TypeScript that compiles to JavaScript.
* The framework consists of several libraries, some of them core and some optional.
* You write Angular applications 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*.
* Then you launch the app by *bootstrapping* the *root module*. Angular takes over, presenting your application content in a browser and responding to user interactions according to the instructions you've provided.



1. **How to upgrade your application from one version of angular to another version of angular?**

Run the below command to upgrade from angular version 2 to angular version 4:

npm install @angular/{common,compiler,compiler-cli,core,forms,http,platform-browser,platform-browser-dynamic,platform-server,router,animations}@latest typescript@latest --save

It will upgrade and install the following angular module to angular 4.0.0.

• @angular/common@latest  
• @angular/compiler@latest  
• @angular/compiler-cli@latest  
• @angular/core@latest  
• @angular/forms@latest  
• @angular/http@latest  
• @angular/platform-browser@latest  
• @angular/platform-browser-dynamic@latest  
• @angular/platform-server@latest  
• @angular/router@latest  
• @angular/animations@latest

Upgrade from Angular 4 to Angular 5.

Update CLI by uninstalling older version and install new CLI.

## **Preliminary Steps**

First, run through the checklist below and modify your app accordingly. Do this to avoid breaking your app after you upgrade (Please note, If you started with Angular 4 you might not have to change anything here):

1. Stop using DefaultIterableDiffer, KeyValueDiffers#factories, or IterableDiffers#factories
2. Rename your template tags to ng-template
3. Replace OpaqueTokens with InjectionTokens.
4. If you call DifferFactory.create(…) remove the ChangeDetectorRef argument.
5. Stop passing any arguments to the constructor for ErrorHandler
6. If you use ngProbeToken, make sure you import it from @angular/core instead of @angular/platform-browser
7. If you use TrackByFn, instead use TrackByFunction
8. If you import any animations services or tools from @angular/core, you should import them from @angular/animations
9. Replace ngOutletContext with ngTemplateOutletContext.
10. Replace CollectionChangeRecord with IterableChangeRecord
11. Anywhere you use Renderer, now use Renderer2
12. Switch from HttpModule and the Http service to HttpClientModule and the HttpClient service. HttpClient simplifies the default ergonomics (You don’t need to map to JSON anymore) and now supports typed return values and interceptors. Read more on [io](https://angular.io/guide/http" \t "_blank)
13. If you use DOCUMENT from @angular/platform-browser, you should start to import this from @angular/common
14. If you use preserveQueryParams, instead use queryParamsHandling
15. If you rely on the date, currency, decimal, or percent pipes, in 5 you will see minor changes to the format. For apps using locales other than en-us you will need to import it and optionally locale\_extended\_fr from @angular/common/i18n\_data/locale\_fr and registerLocaleData(local).
16. Do not rely on gendir, instead look at using skipTemplateCodeGen. Read more
17. If you created a custom form control with min and max input properties, you will need to adapt to the new behavior by renaming them or using the native validators.

After that upgrade using npm with below command.

npm install @angular/animations@'^5.0.0' @angular/common@'^5.0.0' @angular/compiler@'^5.0.0' @angular/compiler-cli@'^5.0.0' @angular/core@'^5.0.0' @angular/forms@'^5.0.0' @angular/http@'^5.0.0' @angular/platform-browser@'^5.0.0' @angular/platform-browser-dynamic@'^5.0.0' @angular/platform-server@'^5.0.0' @angular/router@'^5.0.0' typescript@2.4.2 rxjs@'^5.5.2

npm install typescript@2.4.2 --save-exact

1. **Angular versions and release dates?**

|  |  |
| --- | --- |
| 1. Version | Release Date |
| AngularJS (Version) | October 20, 2010 |
| Angular 2.0.0 | September 2014 |
| Angular 4.0.0 | March 2017 |
| Angular 5.0.0 | November 2017 |
| Future releases of angular 6 | March or April 2018 |

Each version is expected to be backward-compatible with the prior release. Google pledged to do twice-a-year upgrades.

1. **Why angular 4 and what are new in angular 4?**

One of the most important things is that Angular 4.0 doesn't change that much. In Angular 4.0, some under the hood changes, some improvements, and performance improvements have been done. According to Angular [change-log](https://github.com/angular/angular/blob/master/CHANGELOG.md#we-need-your-help) the "Angular 4.0 release is backwards compatible and will work with your existing code, but you may have use cases they haven’t anticipated"

Let's see what are the new changes made in Angular 4.0.   
 **ngIf,**In Angular 2.0, if we wanted to use ngIf with multiple conditions or specifically with else, we write, as shown below.  
  
In Angular 2.0

1. <button (click)="showPara=!showPara">Show and Hide Paragraph</button>
3. <p \*ngIf="showPara">This time show paragraph value is **true**</p>
4. <p \*ngIf="!showPara">This time show paragraph value is **false**</p>

Output  
In Angular 4.0   
NgIf syntax has been extended to support the else clause to display the template when the condition is false. In addition, the condition value can now be stored in local variable for later reuse. This is especially useful when used with the async pipe.   
  
Example

1. <button (click)="showPara=!showPara">Show and Hide Paragraph</button>
3. <p \*ngIf="showPara;else hidePara">This time show paragraph value is **true**</p>
4. <ng-template #hidePara>
5. <p>This time show paragraph value is **false**</p>
6. </ng-template>

Its output will be same, as given above.   
  
You can also store the value in a local variable and later you can use it.

1. <div \*ngIf="employeeObservable | async; else loading; let emp">
2. Hello {{emp.first}} {{emp.last}}!
3. </div>
4. <template #loading>Waiting...</template>

There is one more part, which is added that is **"**then**"**. We can check the condition and using then, we can show different templates. The syntax is given below.

1. <div \*ngIf="condition; then thenBlock else elseBlock"> ... </div>
2. <template #thenBlock>Then template</template>
3. <template #elseBlock>Else template</template>

**Email Validator**   
  
In Angular 2.0, we use an email validator, using pattern option. We provided an email pattern in pattern attribute and then we check whether the form is valid or not but in Angular 4.0, there is a validator to validate an email.  
  
In Angular 2.0

1. <form #frm="ngForm">
2. <label>Email</label>
3. <input
4. type="email"
5. ngModel
6. name="email"
7. required
8. style="width:300px"
9. pattern="^\w+([\.-]?\w+)\*@\w+([\.-]?\w+)\*(\.\w{2,3})+$"/>
10. <button [disabled]=!frm.valid>Submit</button>
11. </form>

Output  
  
   
In Angular 4.0  
  
We can use an email Directive, which is shown below.

1. <form #frm="ngForm">
2. <label>Email</label>
3. <input
4. type="email"
5. ngModel
6. name="email"
7. required
8. style="width:300px"
9. email/>
10. <button [disabled]=!frm.valid>Submit</button>
11. </form>

Its output will be the same as given above.

Pattern validator is still there, so if you want to use it, then you can use it for other types of validation.  
 **Renderer2**

We have renderer in Angular 2.0 and we use it as shown below.   
  
In Angular 2.0

1. **import** { Component,Renderer } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls: ['./app.component.css']
7. })
8. **export** **class** AppComponent {
9. constructor(**private** renderer:Renderer){}
11. onChangeBackground(element:HTMLElement){
12. **this**.renderer.setElementStyle(element,'background-color','#000');
13. **this**.renderer.setElementStyle(element,'color','#fff');
14. }
15. }

On HTML page, we use this, which is shown below.

1. <button (click)="onChangeBackground(ourElement)">Click Me</button>
2. <h1 #ourElement>Hello</h1>

Output  
  
  
In Angular 4.0  
  
Old renderer is deprecated and in Angular 4.0, we have renderer2. We use renderer in Angular 4.0, which is shown below.

1. **import** { Component,Renderer2 } from '@angular/core';
3. @Component({
4. selector: 'app-root',
5. templateUrl: './app.component.html',
6. styleUrls: ['./app.component.css']
7. })
8. **export** **class** AppComponent {
9. constructor(**private** renderer:Renderer2){}
11. onChangeBackground(element:HTMLElement){
12. **this**.renderer.setStyle(element,'background-color','#000');
13. **this**.renderer.setStyle(element,'color','#fff');
14. }
15. }

The output will be the same as you have seen above.   
  
**Angular 4.0 Supports TypeScript 2.0**Angular 2.0 supports Typescript 1 but Angular 4.0 is compitable with Typescript 2.1 and 2.2.  
  
**Animation Package**According to [Angular 4.0 Blog](http://angularjs.blogspot.in/2017/03/angular-400-now-available.html), Angular 4.0 have pulled the animations out of @angular/core and into their own package. This means that if we don’t use animations, this extra code will not end up in our production bundles. This change also allows us to more easily find the documentation and to take better advantage of autocompletion.

**"Now, how can we add Animation?"**We can add animations ourself to our main NgModule by importing BrowserAnimationsModule from @angular/platform-Browser/animations.  
  
I will describe more about this point in my upcoming article.

**Smaller and faster**  
In this release, Angular 4.0 delivers an Application, which is smaller and faster.   
  
**View Engine changes**Under the hood, some changes have been done in AOT generated code.

**8) What Is Bootstrapping in Angular 4 and 5?**

* 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).

app.module.ts -

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

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

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

**import { LoginComponent } from './login/login.component';**

**import { SignupComponent } from './signup/signup.component';**

**@NgModule({**

**declarations: [**

**AppComponent,**

**LoginComponent,**

**SignupComponent**

**],**

**imports: [**

**BrowserModule**

**],**

**providers: [],**

**bootstrap: [AppComponent]**

**})**

**export class AppModule { }**

By default Bootstrap file is created in the folder “src/main.ts” and “main.ts” file is very stable. Once you have set it up, you may never change it again and its looks like -

**import { enableProdMode } from '@angular/core';**

**import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';**

**import { AppModule } from './app/app.module';**

**import { environment } from './environments/environment';**

**if (environment.production) {**

**enableProdMode();**

**}**

**platformBrowserDynamic().bootstrapModule(AppModule).catch(err => console.log(err));**

1. **What is Modules?**

* Angular apps are modular and Angular has its own modularity system called NgModules.

1. **What is Components?**

* A component controls a patch of screen called a view.

1. **What is Templates?**

* A template is a form of HTML that tells Angular how to render the component.

1. **What is Metadata?**

* Metadata tells Angular how to process a class.
* In TypeScript, you attach metadata by using a **decorator**. Here's some metadata for HeroListComponent:

|  |
| --- |
| **@Component({**  **selector: 'app-hero-list',**  **templateUrl: './hero-list.component.html',**  **providers: [ HeroService ]**  **})**  **export class HeroListComponent implements OnInit {**  **/\* . . . \*/**  **}** |

* The @[Component](https://angular.io/api/core/Component) decorator takes a required configuration object with the information Angular needs to create and present the component and its view.

Here are a few of the most useful @[Component](https://angular.io/api/core/Component) configuration options:

* selector: CSS selector that tells Angular to create and insert an instance of this component where it finds a <app-hero-list> tag in parent HTML. For example, if an app's HTML contains <app-hero-list></app-hero-list>, then Angular inserts an instance of the HeroListComponent view between those tags.
* templateUrl: module-relative address of this component's HTML template, shown [above](https://angular.io/guide/architecture#templates).
* providers: array of **dependency injection providers** for services that the component requires. This is one way to tell Angular that the component's constructor requires a HeroService so it can get the list of heroes to display.

The metadata in the @[Component](https://angular.io/api/core/Component) tells Angular where to get the major building blocks you specify for the component.

The template, metadata, and component together describe a view.

Apply other metadata decorators in a similar fashion to guide Angular behavior. @[Injectable](https://angular.io/api/core/Injectable), @[Input](https://angular.io/api/core/Input), and @[Output](https://angular.io/api/core/Output) are a few of the more popular decorators.

1. **What is Data binding?**

* Data binding plays an important role in communication between a template and its component.



* In an Angular application, data in a component can be used to bind in its view. As mentioned earlier, the methods and properties of a component are directly accessible in the component’s view, and the data binding system provides a way to access them.
* Angular supports the following types of bindings:

1. **Property binding:** A property on an HTML element can be bound to the value of a field in the component’s class. Whenever value of the property in the component changes, it automatically updates value of the HTML property it is bound to. The following snippet shows a simple example of property binding:

|  |
| --- |
| **<a [href]="url">Click here to visit the link</a>** |
|  |

Notice the square brackets around the HTML property in the above snippet. They indicate that this field is bound to data.

1. **Event Binding:** The events of any HTML element can be bound to public methods of its component class. The method is called whenever the event is triggered by a user’s action on the page. The following snippet shows an example of event binding:

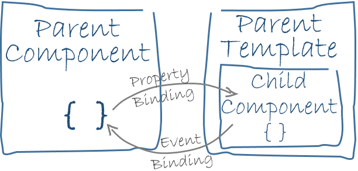
|  |
| --- |
| **<button (click)="close()">Close</button>** |
|  |
|  |

1. **Two-way Binding:** It is a derived binding that uses both property binding and the two-way binding under the hood. This binding is used to display a value and also to update the value when a new value is entered in the UI. *ngModel*is a built-in directive that supports two-way binding. The following snippet shows how it is used:

|  |
| --- |
| **<input type="text" [(ngModel)]="name" />** |

As you see, the two-way bound directive is enclosed inside [()] (called banana-in-a-box). This notation is again a mix of both property and event bindings. The parentheses around the event name indicate that it is data bound.





**14) What is Directives? What are the types of 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](https://angular.io/api/core/Directive) decorator. A component is a *directive-with-a-template*; a @[Component](https://angular.io/api/core/Component) decorator is actually a @[Directive](https://angular.io/api/core/Directive) decorator extended with template-oriented features.
* Two other kinds of directives exist: **structural and attribute directives.**
* **Structural** **directives alter layout by adding, removing, and replacing elements in DOM. Structure directive starts with \*.**
* **Attribute directives alter the appearance or behavior of an existing element. In templates they look like regular HTML attributes, hence the name.**
* The [ngModel](https://angular.io/api/forms/NgModel) directive, which implements two-way data binding, is an example of an attribute directive. [ngModel](https://angular.io/api/forms/NgModel)modifies the behavior of an existing element (typically an <input>) by setting its display value property and responding to change events.
* Directives are used to extend the HTML by creating custom HTML elements and extending the existing elements.
* Angular supports three types of directives.
* The directives used to create custom elements are called **Components**. The directives used to extend an HTML element through a new attribute are called **Attribute Directives**. And the directives that interact with the DOM and manipulate the target element are called **Structural Directives**.
* The components are created using the @Component decorator. The attribute and structural directives are created using the @Directive decorator. The components accept a template, to present it in the custom element. The directives don't have a template, as they are going to act as an add-on to an element.
* Some built in directives are below:
  1. ngFor -- use index as I to find the index
  2. \*ngIf(for large element trees) / hidden attribute(for small element trees)

To show and hide element in DOM

* 1. ngSwitchCase
  2. ngClass
  3. ngStyle
  4. Building Custom Directives

**15) What is Services?**

* Service is a broad category encompassing any value, function, or feature that your application needs.
* Almost anything can be a service. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

**16) What is Dependency injection?**

* Dependency Injection (DI) is a design pattern that solves the problem of handling dependencies needed in a code block.
* 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.
* Angular can tell which services a component needs by looking at the types of its constructor parameters. For example, the constructor of your HeroListComponent needs a HeroService:

**17) Explain important files present in project of angular JS.**

|  |  |
| --- | --- |
| **File** | **Purpose** |
| src/app/ | Angular application files go here.  Ships with the "Hello Angular" sample's AppComponent, AppModule, a component unit test (app.component.spec.ts), and the bootstrap file, main.ts.  Try the [sample application](https://angular.io/generated/live-examples/setup/stackblitz.html) / [download example](https://angular.io/generated/zips/setup/setup.zip) and the [unit test](https://angular.io/generated/live-examples/setup/quickstart-specs.stackblitz.html) / [download example](https://angular.io/generated/zips/setup/quickstart-specs.setup.zip) as live examples. |
| e2e/ | End-to-end (e2e) tests of the application, written in Jasmine and run by the[protractor](http://www.protractortest.org/) e2e test runner.  Initialized with an e2e test for the "Hello Angular" sample. |
| node\_modules/ | The npm packages installed with the npm install command. |
| .editorconfig .git/ .gitignore .travis.yml | Tooling configuration files and folders. Ignore them until you have a compelling reason to do otherwise. |
| CHANGELOG.md | The history of changes to the QuickStart repository. Delete or ignore. |
| favicon.ico | The application icon that appears in the browser tab. |
| index.html | The application host page. It loads a few essential scripts in a prescribed order. Then it boots the application, placing the root AppComponent in the custom <my-app> body tag.  The same index.html satisfies all documentation application samples. |
| karma.conf.js | Configuration for the [karma](https://karma-runner.github.io/1.0/index.html) test runner described in the [Testing](https://angular.io/guide/testing) guide. |
| karma-test-shim.js | Script to run [karma](https://karma-runner.github.io/1.0/index.html) with SystemJS as described in the [Testing](https://angular.io/guide/testing) guide. |
| non-essential-files.txt | A list of files that you can delete if you want to purge your setup of the original QuickStart Seed testing and git maintenance artifacts. See instructions in the optional [Deleting non-essential files](https://angular.io/guide/setup#non-essential) section. Do this only in the beginning to avoid accidentally deleting your own tests and git setup! |
| LICENSE | The open source MIT license to use this setup code in your application. |
| package.json | Identifies npmpackage dependencies for the project.  Contains command scripts for running the application, running tests, and more. Enter npm run for a listing. [Read more](https://github.com/angular/quickstart/blob/master/README.md#npm-scripts) about them. |
| protractor.config.js | Configuration for the [protractor](http://www.protractortest.org/) end-to-end (e2e) test runner. |
| README.md | Instruction for using this git repository in your project. Worth reading before deleting. |
| styles.css | Global styles for the application. Initialized with an <h1> style for the QuickStart demo. |
| systemjs .config.js | Tells the **SystemJS** module loader where to find modules referenced in JavaScript import statements. For example:  content\_copyimport { [Component](https://angular.io/api/core/Component) } from '@angular/core;  Don't touch this file unless you are fully versed in SystemJS configuration. |
| systemjs .config.extras.js | Optional extra SystemJS configuration. A way to add SystemJS mappings, such as for application barrels, without changing the original system.config.js. |
| tsconfig.json | Tells the TypeScript compiler how to transpile TypeScript source files into JavaScript files that run in all modern browsers. |
| tslint.json | The npm installed TypeScript linter inspects your TypeScript code and complains when you violate one of its rules.  This file defines linting rules favored by the [Angular style guide](https://angular.io/guide/styleguide) and by the authors of the documentation. |

**18) How to deploy angular App?**

* One of the tremendous benefits of the Angular CLI is the ability to create a production build of your app, along with AOT (Ahead of Time) compilation.
* In the end, the ultimate goal is to create an app that is as small as possible in file size. So, to demonstrate this, let's run the following command in the console within an Angular project setup to work with the CLI:

**ng build**

* When you run the **ng build** command, it creates a **/dist** folder. Here are the files and their associated sizes after running the above command.
* **Note**: Your file sizes will vary based on your project.

|  |
| --- |
| **vendor.bundle.js 2.2 MB**  **polyfills.bundle.js 163 KB**  **main.bundle.js 13 KB**  **inline.bundle.js 6 KB**  **styles.bundle.js 10 KB** |

* As you can see, we have a massive vendor.bundle.js file, because when you run **ng build** without specifying the production environment, it doesn't make use of uglifying and tree-shaking.
* Let's re-run the **ng build** command, but specify the **--prod** flag (for production):

**ng build –prod**

* Now let's look at the files and their associated sizes:

|  |
| --- |
| **vendor.bundle.js 352 KB // Reduced from 2.2 MB**  **polyfills.bundle.js 57 KB // Reduced from 163 KB**  **main.bundle.js 12 KB // Reduced from 13 KB**  **inline.bundle.js 2 KB // Reduced from 6 KB**  **styles.bundle.js 0 KB // Reduced from 10 KB** |

So, adding the production flag reduced the bundle from around 2.4 MB to 423 KB, which is nearly an **83% reduction**.

How does it do this? Well, a number of things are occurring when you add the **--prod** flag:

* Removes unwanted white space by minifying files.
* Uglifies files by renaming functions and variable names.
* AoT compilation, which removes the compilation process at runtime and instead performs compilation during the build process.

All of these things drastically reduce the file size of your Angular app, thus decreases load times.

**Something important**: You may have heard of **AoT** (Ahead of Time Compilation). As of March 1, 2017, when you specify the **--prod** flag as we did above, it automatically includes AoT. Previously, you had to explicitly specify an **--aot** flag.

* If your app does not contain a backend, you can simply take the contents of the **/dist** folder and upload them to your site. The app will work if you're uploading it to the root public folder, such as [**mysite.com**](http://mysite.com/), but if it's within a sub folder such as [**mysite.com/whatever**](http://mysite.com/whatever), you can specify the **--base-href** flag during the build process based on the folder structure of where the app will be placed.

**19) What is @NgModule?**

* An NgModule is a class marked by the @[NgModule](https://angular.io/api/core/NgModule) decorator.
* NgModule metadata does the following:
  1. Declares which components, directives, and pipes belong to the module.
  2. Makes some of those components, directives, and pipes public so that other module's component templates can use them.
  3. Imports other modules with the components, directives, and pipes that components in the current module need.
  4. Provides services that the other application components can use.

[NgModule](https://angular.io/api/core/NgModule) is a decorator function that takes a single metadata object whose properties describe the module. The most important properties are:

* declarations - the view classes that belong to this module. Angular has three kinds of view classes: [components](https://angular.io/guide/architecture#components), [directives](https://angular.io/guide/architecture#directives), and [pipes](https://angular.io/guide/pipes).
* exports - the subset of declarations that should be visible and usable in the component [templates](https://angular.io/guide/architecture#templates) of other modules.
* imports - other modules whose exported classes are needed by component templates declared in this module.
* providers - creators of [services](https://angular.io/guide/architecture#services) that this module contributes to the global collection of services; they become accessible in all parts of the app.
* bootstrap - the main application view, called the root component, that hosts all other app views. Only the root module should set this bootstrap property.

|  |
| --- |
| @NgModule({  imports: [ BrowserModule ],  providers: [ Logger ],  declarations: [ AppComponent ],  exports: [ AppComponent ],  bootstrap: [ AppComponent ]  }) |

**20) What are built-in modules in angular?**

* An Angular app needs at least one module that serves as the root module. As you add features to your app, you can add them in modules. The following are frequently used Angular modules with examples of some of the things they contain:

|  |  |  |
| --- | --- | --- |
| **NgModule** | **Import it from** | **Why you use it** |
| [BrowserModule](https://angular.io/api/platform-browser/BrowserModule) | @angular/platform-browser | When you want to run your app in a browser |
| [CommonModule](https://angular.io/api/common/CommonModule) | @angular/common | When you want to use [NgIf](https://angular.io/api/common/NgIf), NgFor |
| [FormsModule](https://angular.io/api/forms/FormsModule) | @angular/forms | When you build template driven forms (includes [NgModel](https://angular.io/api/forms/NgModel)) |
| [ReactiveFormsModule](https://angular.io/api/forms/ReactiveFormsModule) | @angular/forms | When building reactive forms |
| [RouterModule](https://angular.io/api/router/RouterModule) | @angular/router | For Routing and when you want to use [RouterLink](https://angular.io/api/router/RouterLink),.forRoot(), and .forChild() |
| [HttpClientModule](https://angular.io/api/common/http/HttpClientModule) | @angular/common/http | When you to talk to a server |

**21) How to Create Custom Directives/attribute directive?**

* Create the directive class file in a terminal window with this CLI command.

ng generate directive highlight

|  |
| --- |
| import { Directive, ElementRef } from '@angular/core';  @Directive({  selector: '[appHighlight]'  })  export class HighlightDirective {  constructor(el: ElementRef) {  el.nativeElement.style.backgroundColor = 'yellow';  }  } |

## Apply the attribute directive

|  |
| --- |
| <p appHighlight>Highlight me!</p> |

* The directive could be more dynamic. It could detect when the user mouses into or out of the element and respond by setting or clearing the highlight colour. Begin by adding [HostListener](https://angular.io/api/core/HostListener) to the list of imported symbols.

|  |
| --- |
| import { [Directive](https://angular.io/api/core/Directive), [ElementRef](https://angular.io/api/core/ElementRef), [HostListener](https://angular.io/api/core/HostListener) } from '@angular/core';  @HostListener('mouseenter') onMouseEnter() {  this.highlight('yellow');  }  @HostListener('mouseleave') onMouseLeave() {  this.highlight(null);  }  private highlight(color: string) {  this.el.nativeElement.style.backgroundColor = color;  } |

**22) How to use routing in angular?**

**23) What is XSS attach and how to handle it in angular?**

**24) What is AOT compilation?**

**25) What is Node.JS?**

**26) What is form in Angular and How to Validate Form in Angular 4?**

**27) How to Handle Error and Logging it?**

Two types of errors -

1.     If the backend returns an unsuccessful response like - 404, 500 and so on

2.     If something goes wrong in the client side like -network error etc.

In the both cases - We are using HttpErrorResponse and return the useful information on what went wrong in this call!

**Example –**

http.get('/api/users')

.subscribe(data => {console.log(data);}, //Successful responses call the first callback.

(err: HttpErrorResponse) => {

**if** (err.error **instanceof** Error) {

console.log('Error - ', err.error.message);

}**else** {

console.log('Error status - ${err.status}, and Error Detail - ${err.error}');

}

}

});

**How To handle and retry the failed request due to Poor Network Connection in Angular 4?**

In Angular 4, simply retry the request using RxJS operator called .retry (). It is automatically presubscribed to an Observable, thus reissuing the request when an error occurred!

**Import RxJS for using retry () method-**

**import** 'rxjs/add/operator/retry';

**HTTP Observables for retry the failed request**

http.get('/api/users')

.retry(**2**) //Retry this request up to 2 times.

.subscribe(data => {console.log(data);}, //Successful responses call the first callback.

(err: HttpErrorResponse) => {

**if** (err.error **instanceof** Error) {

console.log('Error - ', err.error.message);

}**else** {

console.log('Error status - ${err.status}, and Error Detail - ${err.error}');

}

}

});

**28) How to do unit test in angular? How to test angular application?**

**29) What is REST API? What is HttpClient vs HttpClientModule?**

* REST: Representational state transfer
* API: Application Program Interface
* It uses [HTTP](http://searchwindevelopment.techtarget.com/definition/HTTP) requests to GET, PUT, POST and DELETE data.
* REST technology is generally preferred to the more robust Simple Object Access Protocol ([SOAP](http://searchsoa.techtarget.com/definition/SOAP)) technology because REST leverages less [bandwidth](http://searchenterprisewan.techtarget.com/definition/bandwidth), making it more suitable for internet usage.
* GET – To read from the server (Read Methods)
* POST- To create new data(Write Methods)
* PUT- To update the data(Write Methods)
* DELETE- To delete the data(Write Methods)

**Use in angular**

* We need to import **HttpClientModule** from ‘@angular/common/http’ in app.module.ts
* Import ;[

**HttpClientModule**

]

* We need to import **HttpClient** from ‘@angular/common/http’ in app.component.ts
* GET :

constructor(private httpClient: HttpClient){

this.httpClient.get(‘RestApiURL’).subscribe(

(data:any[])=>{

Data contains all the values

})}

* POST: constructor(private httpClient: HttpClient){

this.httpClient.post(‘RestApiURL’,{

name:’mark’,

age: 41

}).subscribe(

(data:any[])=>{

Data contains all the values

})}

**35) How To Set Authorization Headers in GET/POST/PUT requests in Angular?**

**36) What is CORS issue? How to resolve it? What Is HttpInterceptor in Angular?**

To resolve Cross-origin resource sharing issue we use HttpInterceptor.

**How to Set Http Request Header in Angular 4 and Angular 2?**

The HTTP Interceptors are used to Set Http Headers Request in Angular 4 using the import from “@angular/common/http”. The HTTP Interceptors are available in Angular 4.x versions.

The HTTP Interceptors are not supported in Angular 2. We are creating the HttpClient Injectable class to achieve this. You can see the below examples for set http headers request with and without HTTP interceptors.

**What Is the Use of Interceptors in Angular 4?**

The Interceptors is a common used to set default headers for all responses.

**Example 1 –  For Angular 4**  
 **Set Headers Http Request Using Http Interceptors -**

The HTTP Interceptors are now available via using the new HttpClient from angular/common/http in the Angular 4.x versions.

**Steps 1 - Writing an interceptor for adds a header for every request!**

**import** {HttpEvent, HttpInterceptor, HttpHandler, HttpRequest} from '@angular/common/http';

**export** **class** AddHttpHeaderInterceptor **implements** HttpInterceptor {

intercept(req: **HttpRequest**<any>, next: **HttpHandler**): Observable<HttpEvent<any>> {

// Get the auth header from the service.

**const** authHeader = 'token\_value';

**const** clonedReq = req.clone({headers: **req.headers.set**('Authorization', authHeader)});

**return** next.handle(clonedReq);

}

}

**Steps 2 - Providing your interceptor!**

- After creating the interceptor, we need to register it using the HTTP\_INTERCEPTORS in the @NgModule()

**import** { HTTP\_INTERCEPTORS } from '@angular/common/http';

@NgModule({

providers: [{

provide: **HTTP\_INTERCEPTORS**,

useClass: **AddHttpHeaderInterceptor**,

multi: **true**,

}],

})

**export** **class** AppModule {}

**Example 2-  For Angular 2**  
 **Set Headers Http Request without HTTP Interceptors  -**

**Steps 1**– We are creating the HttpClient Injectable class.

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

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

@Injectable()

**export** **class** HttpClient {

**constructor**(**private** http: **Http**) {}

createAuthHeader(headers: **Headers**) {

headers.append('Content-Type', 'application/json');

headers.append('Authorization', 'Basic ' + btoa('user-name:password'));

}

get(url) {

**let** headers = **new** Headers();

**this**.createAuthHeader(headers);

**return** **this**.http.get(url, {headers: **headers**});

}

post(url, data) {

**let** headers = **new** Headers();

**this**.createAuthHeader(headers);

**return** **this**.http.post(url, data, {headers: **headers**});

}

put(url, data) {

**let** headers = **new** Headers();

**this**.createAuthHeader(headers);

**return** **this**.http.put(url, data, {headers: **headers**});

}

}

**Steps 2 - Injecting the HttpClient object in the Component**

**import** { HttpClient } from '../http-client';

**export** **class** userComponent {

**constructor**(http: **HttpClient**) {

**this**.http = httpClient;

}

getUsers() {

**this**.http.get(url).subscribe(data =>{console.log(data); });

}

addUsers(user) {

**this**.http.post(url, user).subscribe(data =>{console.log(data); });

}

}

# **Access-Control-Allow-Origin: Dealing with CORS Errors in Angular**

Getting this error in your Angular app?

*No ‘Access-Control-Allow-Origin’ header is present on the requested resource.*

You’ve run afoul of the **Same Origin Policy** – it says that every AJAX request must match the exact host, protocol, and port of your site. Things that might cause this:

* Hitting a server from a locally-served file (a request from file:///YourApp/index.html to [http://api.awesome.com](http://api.awesome.com/))
* Hitting an external API (a request from [http://yourapp.com](http://yourapp.com/) to [http://api.awesome.com](http://api.awesome.com/)).
* Hitting an internal API (a request from [http://yourapp.com](http://yourapp.com/) to [http://api.yourapp.com](http://api.yourapp.com/)).
* Hitting a different port on the same host (webapp is on [http://localhost:3000](http://localhost:3000/), API is [http://localhost:4000](http://localhost:4000/))
* Requesting over http from https or vice-versa (requesting [https://yourapp.com](https://yourapp.com/) from [http://yourapp.com](http://yourapp.com/))

To be clear, this is not an Angular error. It afflicts all web apps equally, and most of the fixes we’ll look at below are actually modifying the server or the browser.

### **How to fix it**

Here are a few ways to solve this problem:

##### **Best: CORS header (requires server changes)**

CORS (Cross-Origin Resource Sharing) is a way for the server to say “I will accept your request, even though you came from a different origin.” This requires cooperation from the server – so if you can’t modify the server (e.g. if you’re using an external API), this approach won’t work.

Modify the server to add the header Access-Control-Allow-Origin: \* to enable cross-origin requests from anywhere (or specify a domain instead of \*). This should solve your problem.

##### **2nd choice: Proxy Server**

If you can’t modify the server, you can run your own proxy. **And** this proxy can return the Access-Control-Allow-Origin header if it’s not at the Same Origin as your page.

Instead of sending API requests to some remote server, you’ll make requests to your proxy, which will forward them to the remote server. Here are a [few](https://github.com/gr2m/CORS-Proxy) [proxy](https://github.com/Rob--W/cors-anywhere) [options](http://crossorigin.me/).

##### **3rd choice: JSONP (requires server support)**

If CORS and the proxy server don’t work for you, JSONP may help. You essentially make a GET request with a callback parameter:

(get) <http://api.example.com/endpoint?callback=foo>

The server will wrap the JSON reply in a function call to your callback, where you can handle it:

foo({"your": "json", here: true})

There are some downsides, notably that JSONP only supports GET requests and that you still need a cooperative server.

##### **Dev-Only: Disable Same Origin**

If this is only for development or learning purposes, the easiest thing to do is to disable the Same Origin Policy in your browser. Be aware that if you do this, you’re opening your browser up to security risks. Follow these instructions:

* [Chrome](http://stackoverflow.com/questions/3102819/disable-same-origin-policy-in-chrome)
* [Firefox](http://stackoverflow.com/questions/17088609/disable-firefox-same-origin-policy)

This is more of a last resort. Modifying the server to support CORS or running a proxy are the best approaches.

### **Armed and Dangerous**

You’re all set now to tackle any Access-Control-Allow-Origin errors that come your way!

**What is Observables?**

* A sequence of items that arrive asynchronously over time.
* You can think of an observable as an array whose items arrive asynchronously over time. Observables help you manage asynchronous data, such as data coming from a backend service. Observables are used within Angular itself, including Angular’s event system and its http client service. To use observables, Angular uses a third-party library called Reactive Extensions (RxJS). Observables are a proposed feature for ES 2016, the next version of JavaScript.

**30) What is Cookie Based Authentication in Angular 4?**

**31) What is Token Based Authentication in Angular 4?**

**32) What Are Events Binding in Angular 4?**

**33) What Is Property Binding? How To achieve in Angular ?**

**34) What is data binding in angular?**

**37) How To Open Dialog Box in Angular 4?**

**38) What Is Pipes? Why Use Pipes in Angular?**

**39) What Is Angular Router? What’s New Angular 4 Router?**

**40) Explain component decorators in Angular4.**

**41) Write down important commands?**

**42) Explain the component directory structure of angular4.**

**43) Explain ngFor directive with an example.**

**44) Explain ngIf directive with an example.**

**45) Write the difference between directive and component in angular js.**

**46) What do you understand by Isolated Unit Tests?**

**47) What are DSL Animation Functions in Angular js. List them.**

**48) What is life cycle hooks?**

**49) What are class decorators?**

**50) What is Ahead-of-time (AOT) compilation? Explain its advantages and disadvantages?**

**51) What is @inputs In Angular 4?**

**52) What is @outputs In Angular 4?**

**53) What is lazy loading? How to achieve lazy lading in angular app?**

**54) What is Change detection?**

**55) What Are Event Emitters in Angular 2/4?**

**56) What is viewchild?**

**57) How to share global data across components?**

**58) what is the use of interceptors in angular 4?**

**59) what is the forRoot method?**

**60) what is the difference between `{`ngfor`}` and `{`ngforof`}` in angular 2?**

**61) how are JWTS used to authenticate angular 4 applications?**

**62) what is json web token?**

**63) Explain app.module.ts file.**

**64) Explain tsconfig.json file.**

**65) Explain package.json file.**

**66) Explain systemjs.config.json file.**

**67) What Are Differences Between Components And Directives?**

**68) What Is Npm?**

**69) What Is the Use Of Codelyzer In Angular 2 Application?**

**70) What Is Shadow Dom?**

**71) Difference Between Constructor And Ngoninit?**

**73) What is the difference between Constructor and ngOnInit in Angular ?**

**74) What you do for security of your angular application?**