#### General Questions:

* What are some of the reasons you would choose to use Angular in your project?
* What did you like about working with Angular?
* How do you keep your Angular code more readable and maintainable?
* What does testable code mean to you in context of Angular?
* What does reusable code mean to you in context of Angular?

#### Animations Questions:

* How do you define transition between two states in Angular?
* How do you define a wildcard state?

#### Architecture Questions:

* What is a good use case for ngrx/store?
* What is a good use case for ngrx/entity?
* Can you talk about a bug related to a race condition, how to solve it and how to test it?
* What is the difference between a smart/container component and dumb/presentational component? What is a good use case example? What are the advantages?

#### API Questions:

* What does this code do:

@HostBinding('class.valid') isValid;

<div \*ngIf='someObservableData | async as data; else loading'>{{data}}</div><ng-template #loading> Loading Data...</ng-template>

* Why would you use renderer methods instead of using native element methods?
* How would you control size of an element on resize of the window in a component?
* What would be a good use for NgZone service?
* What are the bootstrap options for NgZone? Why would you use them? (Angular 5+)
* How would you protect a component being activated through the router?
* How would you insert an embedded view from a prepared TemplateRef?
* What is the difference between @ViewChild() and @ContentChild()

#### Template Syntax Questions:

* How can you add an active class to a selected element in a list component?
* What is a template variable. How would you use it?
* What is the difference of using a property binding verses a function binding on a template?
* What happens if you subscribe to a data source multiple times with async pipe?
* what is the difference between ng-content, ng-container and ng- template?
* When you create a data-binding in Angular, are you working with attributes or properties? What is the difference anyway?
* When can you omit the brackets in template binding?

#### Component Questions:

* What is the minimum definition of a component?
* What is the difference between a component and a directive?
* How do components communicate with each other?
* How do you create two way data binding in Angular?
* How would you create a component to display error messages throughout your application?
* What does a lean component mean to you?

#### Component Interaction & State Management Questions:

* How would you pass data from a parent component to a child component?
* How would you pass data from a child component to a parent component?
* Which components will be notified when an event is emitted?
* Tell me about the different ways how you would get data to your components from a service and talk about why would you use one way vs the other?
* How would you use cached data?

#### Forms Questions:

* When do you use template driven vs model driven forms? Why?
* How do you submit a form?
* What's the difference between NgForm, FormGroup, and FormControl? How do they work together?
* What's the advantage of using FormBuilder?
* How do you add form validation to a form built with FormBuilder?
* What's the difference between dirty, touched, and pristine on a form element?
* How can you access validation errors in the template to display error messages?
* What is async validation and how is it done?

#### NgModules Questions:

* What is the purpose of NgModule?
* How do you decide to create a new NgModule?
* What are the attributes that you can define in an NgModule annotation?
* What is the difference between a module's forRoot() and forChild() methods and why do you need it?
* What would you have in a shared module?
* What would you not put shared module?
* What module would you put a singleton service whose instance will be shared throughout the application (e.g. ExceptionService andLoggerService)?
* What is the purpose of exports in a NgModule?
* What is the difference between exports and declerations in NgModule?
* Why is it bad if SharedModule provides a service to a lazy loaded module?

#### Services Questions:

* What is the use case of services?
* How are the services injected to your application?
* How do you unit test a service with a dependency?
* Why is it a bad idea to create a new service in a component like the one below?

let service = new DataService();

#### Structural Directives Questions:

* What is a structural directive?
* How do you identify a structural directive in html?
* When creating your own structural directives, how would you decide on hiding or removing an element? What would be the advantages or disadvantages of choosing one method rather than the other?

#### Style Guide Questions:

* What are some of the Angular Style Guide suggestions you follow on your code? Why?
* Is it important to have a style guide? Why/not?

#### Styling Questions:

* How would you select a custom component to style it.
* What pseudo-class selector targets styles in the element that hosts the component?
* How would you select all the child components' elements?
* How would you select a css class in any ancestor of the component host element, all the way up to the document root?
* What selector force a style down through the child component tree into all the child component views?
* What does :host-context() pseudo-class selector targets?
* What does the following css do?

:host-context(.theme-light) h2 { background-color: red;}

#### Lifecycle Hooks Questions:

* What is the possible order of lifecycle hooks.
* When will ngOnInit be called?
* How would you make use of ngOnInit()?
* What would you consider a thing you should be careful doing on ngOnInit()?
* What is the difference between ngOnInit() and constructor() of a component?
* What is a good use case for ngOnChanges()?

#### Observables RxJS Questions:

* What is the difference between an observable and a promise?
* What is the difference between an observable and a subject?
* What are some of the angular apis that are using observables?
* How would you cache an observable data?
* How would you implement a multiple api calls that needs to happen in order using rxjs?
* What is the difference between switchMap, concatMap and mergeMap?
* How would you make sure an api call that needs to be called only once but with multiple conditions. Example: if you need to get some data in multiple routes but, once you get it, you can reuse it in the routes that needs it, therefor no need to make another call to your backend apis.
* How would you implement a [brush behavior](https://bl.ocks.org/mbostock/34f08d5e11952a80609169b7917d4172) using rxjs?
* How would you implement a color picker with rxjs?
* If you need to respond to two different Observable/Subject with one callback function, how would you do it?(ex: if you need to change the url through route parameters and with prev/next buttons).
* What is the difference between scan() vs reduce() ?

#### Performance Questions:

* What are some of the things that you pay attention to, to make sure your angular application is performant?
* What tools would you use to find a performance issue in your code?
* What tools have you used to improve the performance of your application?
* What are some ways you may improve your website's scrolling performance?
* Explain the difference between layout, painting and compositing.
* Have you seen Jeff Cross's NgCruise talk on performance?

#### Pipes Questions:

* What is a pure pipe?
* What is an async pipe?
* What kind of data can be used with async pipe?
* How do you create a custom pipe?
* How does async pipe prevents memory leeks?
* What is the difference between pure and impure pipes?

#### Router Questions:

* What is the difference between RouterModule.forRoot() vs RouterModule.forChild()? Why is it important?
* How does loadChildren property work?
* Do you need a Routing Module? Why/not?
* When does a lazy loaded module is loaded?
* Below link doesn't work. Why? How do I fix it?

<div routerLink='product.id'></div>

* Can you explain the difference between ActivatedRoute and RouterState?
* How do you debug router?
* Why do we need route guards?
* What is a RouterOutlet?

#### Security Questions:

#### Testing Questions:

* What are some of the different tests types you can write?
* How do you mock a service to inject in an integration test?
* How do you mock a module in an integration test?
* How do you test a component that has a dependency to an async service?
* What is the difference between 'async()' and 'fakeAsync()'?

#### TypeScript Questions:

* Why do you need type definitions?
* How would you define a custom type?
* What is the difference between an Interface and a Class?
* First line below gives compile error, second line doesn't. Why?

someService.someMethod(x);someService['someMethod'](x);

* What are Discriminated union types?
* How do you define Object of Objects type in typescript?
* How can you capture the 'type' the user provides (e.g. number), so that we can use that information later.

#### JavaScript Questions:

* Explain the difference between var, let and const key words.
* Could you make sure a const value is garbage collected?
* Explain Object.assign and possible use cases.
* Explain Object.freeze and possible use cases.
* Explain the code below. How many times the createVal function is called?

function createVal(){ return Math.random();};function fun( val = createVal()){ // Do something with val...}fun();fun(5);

* What is the spread operator doing in this function call? Seriously!

doStuff(...args);

#### Coding Questions:

* What would these components render?

import { Component, ContentChildren, Directive, Input, QueryList } from '@angular/core';

@Directive({selector: 'pane'})export class Pane { @Input() id: string;}

@Component({ selector: 'tab', template: ` <div>panes: {{serializedPanes}}</div> `})

export class Tab {

@ContentChildren(Pane) panes: QueryList<Pane>; get serializedPanes(): string { return this.panes ? this.panes.map(p => p.id).join(', ') : ''; }

}

@Component({ selector: 'example-app', template: ` <tab> <pane id="1"></pane> <pane id="2"></pane> <pane id="3" \*ngIf="shouldShow"></pane> </tab> <button (click)="show()">Show 3</button> `,})

export class ContentChildrenComp { shouldShow = false; show() { this.shouldShow = true; }}

#### Fun Questions:

* What's a cool project that you've recently worked on?
* What are some things you like about the developer tools you use?
* Who inspires you in the angular community?
* Do you have any pet projects? What kind?
* How did you design the architecture of your project?
* What's your favorite feature of Angular?
* What is your least favorite thing about Angular? (Please share your thoughts by making a pull request to [angularFeelings](https://github.com/Yonet/Angular-Interview-Questions/blob/master/angularFeelings.md))
* How do you like your coffee?
* If you could decide on a new feature for angular, what would it be?

## Novice Level

### Familiarity of Basic Terminology

1. What are the differences between AngularJS (angular 1.x) and Angular (Angular 2.x and beyond)?
2. What is a component? Why would you use it?
3. What is the minimum definition of a component?
4. What is a module, and what does it contain?
5. What is a service, and when will you use it?
6. What is a promise? Explain it laymen's terms.
7. What are the lifecycle hooks for components and directives?
8. What are pipes? Give me an example.
9. What are the differences between reactive forms and template driven forms?
10. What is a dumb, or presentation, component? What are the benefits of using dumb components?

### Ability to Build Simple App

1. How do components communicate with each other?
2. How would you use http to load data from server?
3. How do you create routes?
4. How can you get the current state of a route?
5. How do you create two-way data binding?
6. How do you load external modules?
7. How would you display form validation errors?
8. Which lifecycle hook would you use to unsubscribe an observable?
9. How are services injected to your application?
10. How would you create route parameters and access them from a component?

### Basic Concepts

1. Why would you use Angular instead of another framework, e.g., React?
2. What is the difference between an observable and a promise?
3. What is the difference between a component and a directive?
4. Why would you use typescript aka benefits of typescript?
5. Why different life cycle hooks are needed for a component/directive?
6. Why does angular use rxjs?
7. What is the purpose of using zone.js?
8. What is the difference between ngOnInit() and the constructor() of a component?
9. When will ngOnInit() be called? How would you make use of ngOnInit()?
10. What are the benefits of using formBuilder?

## Intermediate Level

To be in the intermediate level, you have to build at least one medium sized angular app. You have to have familiarity with routing, https, different built process, unit test, etc. here are the questions you could expect.

### Essential Terminology Questions

1. How will you protect a route for authorized user only?
2. What is a custom pipe and how will you use it?
3. What is a structural directive?
4. What is the difference between RouterModule.forRoot() vs RouterModule.forChild()? Why is it important?
5. What is the difference between a module's forRoot() and forChild() methods and why do you need it?
6. What's the difference between dirty, touched, and pristine on a form element?
7. What is an async pipe? What kind of data can be used with async pipe?
8. What is injectable? Give me some example.
9. What is a pure pipe?
10. How will you create two-way data binding in Angular?

### Comfortability to Build Medium Size App Questions

1. How do components communicate with each other?
2. How do you decide to create a new NgModule?
3. How will you inject custom header in your http call?
4. How do you identify a structural directive in html?
5. How would you select a custom component to style it?
6. How would you select all the child components' elements?
7. How would you cache an observable data?
8. How would you save data from a form control?
9. How Event Emitters works in Angular?
10. How do you mock a service to inject in a unit test?

### Core Concepts Understandability Questions

1. Tell me about feature module and shared module?
2. What would you not put in a shared module?
3. Why angular uses decorator?
4. What is async validation and how is it done?
5. Why do you need type definitions?
6. Which components will be notified when an event is emitted?
7. Why would you export from ngModule?
8. Why is it bad if SharedModule provides a service to a lazy loaded module?
9. Can you explain the difference between ActivatedRoute and RouterState?
10. Which service will you put in the module and why?

## Expert Level

### Performance and Edge case Related Terminology

1. What is a factory Component?
2. What is lazy loading and why will you use it?
3. What is Ahead of time (AOT) compilation and why will you use it?
4. What are some of the Angular Style Guide suggestions you follow on your code? Why?
5. What is wildcard state?
6. How do you put animation between two states?
7. What would be a good use for NgZone service?
8. How would you protect a component being activated through the router?
9. How would you insert an embedded view from a prepared TemplateRef?
10. What is attribute directive and why will you use it?

### Master a Large App

1. How will you intercept http to inject header to each http call?
2. How would you create a component to display error messages throughout your application?
3. How will you parallelize multiple observable call?
4. How will you put one async call before another?
5. How can you use web worker in angular app?
6. What tools would you use to find a performance issue in your code?
7. What are some ways you may improve your website's scrolling performance?
8. Explain the difference between layout, painting and compositing.
9. How can you cancel a router navigation?
10. How would you animate routing?
11. How would you cancel a promise on which you are waiting?

### Rockstar and Fighter for Angular Questions

1. When does a lazy loaded module is loaded?
2. Why angular uses url segment?
3. How will you make angular app secure?
4. How will you localize numbers currencies and dates?
5. What is the best way to use translation in your app?
6. How will you setup different environment build differently for your app?
7. How will you use scss or css preprocessing with your application?
8. How will you optimize image/svg in your angular app?
9. How would you make sure an api call that needs to be called only once but with multiple conditions? Example: if you need to get some data in multiple routes but, once you get it, you can reuse it in the routes that needs it, therefor no need to make another call to your backend apis.
10. If you need to respond to two different Observable/Subject with one callback function, how would you do it? (ex: if you need to change the url through route parameters and with prev/next buttons).

##### Basic concepts

What is Angular?

What is the difference between AngularJS and Angular?

What should be the directory structure of the components of any Angular application and why?What is MVVM and what is the difference before MVC?

##### Angular template syntax

What is interpolation in Angular?

What methods of using patterns in Angular do you know?

What is the difference between structural and attribute directive, what are the built-in directives?

What are the ng-template, ng-container, ng-content directives for?

##### Angular development enviroments

What is a directive and how to create your own?

What is directive, component, module, service, pipe in Angular and what are they for?

Tell us about the main parameters @NgModule, @Component, @Directive, @Injectable, @PipeWhat are dynamic components and how can they be used in Angular?

How to apply animation to components?

##### Angular render lifecycle and core environments

Explain the Angular application's bootstrap mechanism in the browser?

How do components interact in Angular (describe the components view)?

What is the life cycle of the components?

What is the Shadow DOM and how to work with it in Angular?

What is Data Binding and what problems do you know about it?

How do you use one-way and two-way data binding?

What are the advantages and disadvantages of a Regular DOM (Angular) over a Virtual DOM (React)?

What is ngZone?How to update the view if your data model is updated outside the 'zone'?

What is EventEmmiter and how to subscribe to events?

What is Change Detection, how does Change Detection Mechanism work?

What are the strategies for detecting changes?

How many change detectors can there be in an entire application?

The main difference between the constructor and ngOnInit?

##### Angular data flow

What is Dependency Injection?

What is Singleton Service and why is it used in Angular?

How can I define my ErrorHandler, Logging, Cache handler in Angular?

What is an Observable?

What is the difference between Observable and Promise?

What is the difference between Observable and BehaviorSubject / Subject?

What is the difference between switchMap, mergeMap, concatMap?

##### Angular with Backend integrations

How can the backend APIs interact, which is required for proxying requests?

What are HTTP Interceptors?How to use Json Web Tokens for authentication when developing on Angular?

How are XSS and CSRF attacks processed in Angular?

##### Angular routing

What is routing and how to create it in Angular?

What is the life cycle of the Angular Router?

What is lazy loading (Lazy-loading) and what is it used for?

What is the difference between Routing and Navigation?

How to load data before the route is activated?

##### Angular Forms (also big ui enterprise)

What are FormGroup and FormControl and what are they used for?

What are reactive forms in Angular?How to apply validation for simple and complex forms?

##### Build environments

What is the difference between Angular CLI and Webpack Development Environment?

What are JIT and AOT, what are their differences and what are the areas of application?

##### Test development

What is Unit testing, integration, e2e testing (End-to-End) and how is it used in Angular?

What is Karma, Jest, Jasmine (why are they used together when developing on Angular)?How to test input parameters and popup events of components?

**Angular 2 Interview Questions**

What is Angular 2?

AngularJS is a framework to build large scale and high performance web application while keeping them as easy-to-maintain. Following are the features of AngularJS framework.

* **Components** − The earlier version of Angular had a focus of Controllers but now has changed the focus to having components over controllers. Components help to build the applications into many modules. This helps in better maintaining the application over a period of time.
* **TypeScript** − The newer version of Angular is based on TypeScript. This is a superset of JavaScript and is maintained by Microsoft.
* **Services** − Services are a set of code that can be shared by different components of an application. So for example if you had a data component that picked data from a database, you could have it as a shared service that could be used across multiple applications.

What are the key components of Angular 2?

Angular 2 has the following components −

* **Modules** − This is used to break up the application into logical pieces of code. Each piece of code or module is designed to perform a single task.
* **Component** − This can be used to bring the modules together.
* **Templates** − This is used to define the views of an Angular JS application.
* **Metadata** − This can be used to add more data to an Angular JS class.
* **Service** − This is used to create components which can be shared across the entire application.

Explain Modules in Angular 2.

Modules are used in Angular JS to put logical boundaries in your application. Hence, instead of coding everything into one application, you can instead build everything into separate modules to separate the functionality of your application. A module is made up of the following parts −

* **Bootstrap array** − This is used to tell Angular JS which components need to be loaded so that its functionality can be accessed in the application. Once you include the component in the bootstrap array, you need to declare them so that they can be used across other components in the Angular JS application.
* **Export array** − This is used to export components, directives, and pipes which can then be used in other modules.
* **Import array** − Just like the export array, the import array can be used to import the functionality from other Angular JS modules.

Explain Components in Angular 2.

Each application consists of Components. Each component is a logical boundary of functionality for the application. You need to have layered services, which are used to share the functionality across components.Following is the anatomy of a Component. A component consists of −

* **Class** − This is like a C or Java class which consists of properties and methods.
* **Metadata** − This is used to decorate the class and extend the functionality of the class.
* **Template** − This is used to define the HTML view which is displayed in the application.

What are Angular 2 directives? Explain with examples.

A directive is a custom HTML element that is used to extend the power of HTML. Angular 2 has the following directives that get called as part of the BrowserModule module.

* **ngIf** −
* The **ngif** element is used to add elements to the HTML code if it evaluates to true, else it will not add the elements to the HTML code.

### Syntax

\*ngIf = 'expression'

If the expression evaluates to true then the corresponding gets added, else the elements are not added.

* **ngFor** −
* The **ngFor** element is used to elements based on the condition of the For loop.

### Syntax

\*ngFor = 'let variable of variablelist'

The variable is a temporary variable to display the values in the **variablelist**.

How will you handle errors in Angular 2 applications?

Angular 2 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.

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.

What is CLI?

Command Line Interface (CLI) can be used to create our Angular JS application. It also helps in creating a unit and end-to-end tests for the application.

What is Dependency Injection? Explain with example.

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.

**Step 1** − Create a separate class which has the injectable decorator. The injectable decorator allows the functionality of this class to be injected and used in any Angular JS module.

@Injectable() export class classname { }

**Step 2** − Next in your appComponent module or the module in which you want to use the service, you need to define it as a provider in the @Component decorator.

@Component ({ providers : [classname] })

Explain tsconfig.json file.

This file is used to give the options about TypeScript used for the Angular JS project.

{ "compilerOptions": { "target": "es5", "module": "commonjs", "moduleResolution": "node", "sourceMap": true, "emitDecoratorMetadata": true, "experimentalDecorators": true, "lib": [ "es2015", "dom" ], "noImplicitAny": true, "suppressImplicitAnyIndexErrors": true } }

Following are some key points to note about the above code.

* The target for the compilation is es5 and that is because most browsers can only understand ES5 typescript.
* The sourceMap option is used to generate Map files, which are useful when debugging. Hence, during development it is good to keep this option as true.
* The "emitDecoratorMetadata": true and "experimentalDecorators": true is required for Angular JS decorators. If not in place, Angular JS application will not compile.

Explain package.json file.

This file contains information about Angular 2 project. Following are the typical settings in the file.

{ "name": "angular-quickstart", "version": "1.0.0", "description": "QuickStart package.json from the documentation, supplemented with testing support", "scripts": { "build": "tsc -p src/", "build:watch": "tsc -p src/ -w", "build:e2e": "tsc -p e2e/", "serve": "lite-server -c=bs-config.json", "serve:e2e": "lite-server -c=bs-config.e2e.json", "prestart": "npm run build", "start": "concurrently \"npm run build:watch\" \"npm run serve\"", "pree2e": "npm run build:e2e", "e2e": "concurrently \"npm run serve:e2e\" \"npm run protractor\" --killothers --success first", "preprotractor": "webdriver-manager update", "protractor": "protractor protractor.config.js", "pretest": "npm run build", "test": "concurrently \"npm run build:watch\" \"karma start karma.conf.js\"", "pretest:once": "npm run build", "test:once": "karma start karma.conf.js --single-run", "lint": "tslint ./src/\*\*/\*.ts -t verbose" }, "keywords": [], "author": "", "license": "MIT", "dependencies": { "@angular/common": "<2.4.0", "@angular/compiler": "<2.4.0", "@angular/core": "<2.4.0", "@angular/forms": "<2.4.0", "@angular/http": "<2.4.0", "@angular/platform-browser": "<2.4.0", "@angular/platform-browser-dynamic": "<2.4.0", "@angular/router": "<3.4.0", "angular-in-memory-web-api": <0.2.4", "systemjs": "0.19.40", "core-js": "^2.4.1", "rxjs": "5.0.1", "zone.js": "^0.7.4" }, "devDependencies": { "concurrently": "^3.2.0", "lite-server": "^2.2.2", "typescript": "<2.0.10", "canonical-path": "0.0.2", "tslint": "^3.15.1", "lodash": "^4.16.4", "jasmine-core": "<2.4.1", "karma": "^1.3.0", "karma-chrome-launcher": "^2.0.0", "karma-cli": "^1.0.1", "karma-jasmine": "^1.0.2", "karma-jasmine-html-reporter": "^0.2.2", "protractor": <4.0.14", "rimraf": "^2.5.4", "@types/node": "^6.0.46", "@types/jasmine": "2.5.36" }, "repository": {} }

Some key points to note about the above code −

* There are two types of dependencies, first is the dependencies and then there are dev dependencies. The dev ones are required during the development process and the others are needed to run the application.
* The "build:watch": "tsc -p src/ -w" command is used to compile the typescript in the background by looking for changes in the typescript files.

Explain systemjs.config.json file.

This file contains the system files required for Angular JS application. This loads all the necessary script files without the need to add a script tag to the html pages. The typical files will have the following code.

/\*\* \* System configuration for Angular samples \* Adjust as necessary for your application needs. \*/ (function (global) { System.config({ paths: { // paths serve as alias 'npm:': 'node\_modules/' }, // map tells the System loader where to look for things map: { // our app is within the app folder app: 'app', // angular bundles '@angular/core': '[npm:@angular/core/bundles/core.umd.js](mailto:npm:@angular/core/bundles/core.umd.js)', '@angular/common': '[npm:@angular/common/bundles/common.umd.js](mailto:npm:@angular/common/bundles/common.umd.js)', '@angular/compiler': '[npm:@angular/compiler/bundles/compiler.umd.js](mailto:npm:@angular/compiler/bundles/compiler.umd.js)', '@angular/platform-browser': '[npm:@angular/platformbrowser/bundles/platform-browser.umd.js](mailto:npm:@angular/platformbrowser/bundles/platform-browser.umd.js)', '@angular/platform-browser-dynamic': '[npm:@angular/platform-browserdynamic/bundles/platform-browser-dynamic.umd.js](mailto:npm:@angular/platform-browserdynamic/bundles/platform-browser-dynamic.umd.js)', '@angular/http': '[npm:@angular/http/bundles/http.umd.js](mailto:npm:@angular/http/bundles/http.umd.js)', '@angular/router': '[npm:@angular/router/bundles/router.umd.js](mailto:npm:@angular/router/bundles/router.umd.js)', '@angular/forms': '[npm:@angular/forms/bundles/forms.umd.js](mailto:npm:@angular/forms/bundles/forms.umd.js)', // other libraries 'rxjs': 'npm:rxjs', 'angular-in-memory-web-api': 'npm:angular-in-memory-web-api/bundles/inmemory-web-api.umd.js' }, // packages tells the System loader how to load when no filename and/or no extension packages: { app: { defaultExtension: 'js' }, rxjs: { defaultExtension: 'js' } } }); })(this);

Some key points to note about the above code −

* 'npm:': 'node\_modules/' tells the location in our project where all the npm modules are located.
* The mapping of app: 'app' tells the folder where all our applications files are loaded.

Explain app.module.ts file.

The following code will be present in the **app.module.ts** file.

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 { }

Let's go through each line of the code in detail.

* The import statement is used to import functionality from the existing modules. Thus, the first 3 statements are used to import the NgModule, BrowserModule and AppComponent modules into this module.
* The NgModule decorator is used to later on define the imports, declarations, and bootstrapping options.
* The BrowserModule is required by default for any web based angular application.
* The bootstrap option tells Angular which Component to bootstrap in the application.

How will you convert an input to all lowercase?

lowercase filter is used to convert the input to all lowercase.

In below example, we've added lowercase filter to an expression using pipe character. Here we've added lowercase filter to print student name in all lowercase letters.

<div> The name of this Tutorial is {{TutorialName}}  
 The first Topic is {{appList[0] | lowercase}}  
 The second Topic is {{appList[1] | lowercase}}  
 The third Topic is {{appList[2]| lowercase}}  
 </div>

How will you convert an input to all uppercase?

uppercase filter is used to convert the input to all uppercase.

In below example, we've added uppercase filter to an expression using pipe character. Here we've added uppercase filter to print student name in all uppercase letters.

<div> The name of this Tutorial is {{TutorialName}}  
 The first Topic is {{appList[0] | uppercase}}  
 The second Topic is {{appList[1] | uppercase}}  
 The third Topic is {{appList[2]| uppercase}}  
 </div>

How will you get a substring from a string?

slice filter is used to slice a piece of data from the input string.

In below example, we've added slice filter to an expression using pipe character. Here property value will be sliced based on the start and end positions.

<div> The name of this Tutorial is {{TutorialName}}  
 The first Topic is {{appList[0] | slice:1:2}}  
 The second Topic is {{appList[1] | slice:1:3}}  
 The third Topic is {{appList[2]| slice:2:3}}  
 </div>

How will you convert a string into a date?

date filter is used to convert the input string to date format.

In below example, we've added date filter to an expression using pipe character. Here property value will be converted to date format.

<div> The date of this Tutorial is {{newdate | date:"MM/dd/yy"}}  
</div>

How will you convert a string into a currency?

currency filter is used to convert the input string to currency format.

In below example, we've added currency filter to an expression using pipe character. Here property value will be converted to currency format.

<div> The currency of this Tutorial is {{newValue | currency}}  
</div>

How will you convert a string into a percentage?

percent filter is used to convert the input string to percentage format.

In below example, we've added percent filter to an expression using pipe character. Here property value will be converted to percentage format.

<div> The percentage of this Tutorial is {{newValue | percent}}  
</div>

When ngOnChanges event get called in Angular 2 Application Lifecycle?

When the value of a data bound property changes, then this method is called.

When ngOnInit event get called in Angular 2 Application Lifecycle?

This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.

When ngDoCheck event get called in Angular 2 Application Lifecycle?

This is for the detection and to act on changes that Angular can't or won't detect on its own.

When ngAfterContentInit event get called in Angular 2 Application Lifecycle?

This is called in response after Angular projects external content into the component's view.

When ngAfterContentChecked event get called in Angular 2 Application Lifecycle?

This is called in response after Angular checks the content projected into the component.

When ngAfterViewInit event get called in Angular 2 Application Lifecycle?

This is called in response after Angular initializes the component's views and child views.

When ngAfterViewChecked event get called in Angular 2 Application Lifecycle?

This is called in response after Angular checks the component's views and child views.

When ngOnDestroy event get called in Angular 2 Application Lifecycle?

This is the cleanup phase just before Angular destroys the directive/component.

## Template Syntax Questions

**What is a template reference variable, and how would you use it?**

A: A variable (defined using a #) in a component template, which can be referenced in other parts of the template. For example, a template variable for a form can then be referenced by other elements of the form.

**What are the possible ways to bind component properties to an associated template?**

A: Interpolation binding, one way binding, two way binding.

**What's the difference between binding a value with or without square brackets, i.e.: <input attr="something" /> vs <input [attr]="something" />?**

A: The square brackets will cause "something" to be evaluated as an expression, as opposed to just be passed in as a literal string.

**What does the ngFor template syntax look like?**

A: example:<ul><li \*ngFor="let val of values">{{val}}</li></ul>

**What does the pipe syntax look like in Angular templates?**

A: example: <div>{{ value | my-pipe : option }}</div>

**What does an interpolated string look like in a template?**

A: example: <div title="Hello {{username}}">...</div>

**What is <ng-container>?**

A: A grouping element that does not interfere with styles or layout (it's analogous to curly braces in JavaScript).

**What is <ng-template>?**

A: It's an Angular element for rendering HTML when using structural directives. The ng-template itself does not render to anything but a comment directly.

## Component/Directive Questions

**What is the minimum definition of a component?**

A: A class with a Component decorator specifying a template.

**What is the difference between a component and a template?**

A: A component is a directive with a template (representing a view).

**What are different kinds of directives supported in Angular 2?**

A: Structural, component, and attribute directives.

**How do components communicate with each other?**

A: Various ways - for example: Input/Output properties, services, ViewChild/ViewContent.

**How do you create two way data binding in Angular 2.0?**

A: By using the two-way binding syntax [()] (along with ngModel, if you're doing this in the context of a form control element).

**How would you create a component to display error messages throughout your application?**

A: Implement your own ErrorHandler and configure it in the list of providers for the modules you want it used in.

**How would you support logging in your Angular app?**

PA: One way would be to use angular2-logger, which is a package inspired by log4j.

**How would you use the ngClass directive?**

A: For example: <div [ngClass]="{firstCondition: 'class1', secondCondition: 'class2'}">...</div>

**How do you resolve a template URL relative to a Component class?**

A: By specifying the moduleId to be module.id in the Component decorator. (Note: while this is still needed when using SystemJS, it is not necessary anymore when using Webpack module bundler for Angular 2 projects.)

**What are dynamic components?**

A: Components that are added at runtime (i.e. not fixed). For example, an ad banner component that is determined at runtime.

**What is ComponentFactoryResolver used for?**

A: A class that is used to create dynamic components - it produces a ComponentFactory for a particular component which can then be loaded into view via a createComponent on ViewContainerRef.

## NgModules Questions:

**What is the purpose of NgModule?**

A: It's to give Angular information on a particular module’s contents, through decorator properties like: declarations, imports, exports, providers, etc.

**How do you decide to create a new NgModule?**

A: Typically for a nontrivial feature in an application, that will involve a collection of related components and services.

**What are the attributes that you can define in an NgModule annotation?**

A: Declarations, imports, exports, providers, bootstrap

**What is the difference between a module's forRoot() and forChild() methods and why do you need it?**

A: forRoot and forChild are conventional names for methods that deliver different import values to root and feature modules.

**What would you have in a shared module?**

A: Common components, directives, and pipes used in other modules in your application.

**What would you not put shared module?**

A: Services that should not have multiple instances created for the application.

**What module would you put a singleton service whose instance will be shared throughout the application (e.g. ExceptionService andLoggerService)?**

A: Root Module

**What is the purpose of exports in an NgModule?**

A: Provide components, directives, pipes to other modules for their usage.

**Why is it (potentially) bad if SharedModule provides a service to a lazy loaded module?**

A: You will have two instances of the service in your application, which is often not what you want.

**Can we import a module twice?**

A: Yes, and the latest import will be what is used.

**Can you re-export classes and modules?**

A: Yes.

**What kind of classes can you import in an angular module?**

A: Components, pipes, directives

**What is the providers property used for in a module's NgModule metadata?**

A: To provide a list of service cerators that this module contributes to the global collection of services.

**What is bootstrapping in Angular?**

A: The mechanism that launches the application in Angular, loading the root module (typically called AppModule) which loads one or more bootstrapped components into the application's DOM.

## Services Questions:

**What is the use case of services?**

A: One very common use case is providing data to components, often by fetching it from a server. Though there’s no real definition of service from Angular point of view – it could do almost anything (e.g., logging is another common use case, among many).

**How are the services injected to your application?**

A: Via Angular’s DI (Dependency Injection) mechanism

**Why is it a bad idea to create a new service in a component like the one below?**

let service = new DataService();

A: The object may not be created with its needed dependencies.

**How to make sure that a single instance of a service will be used in an entire application?**

A: Provide it in the root module.

**Why do we need provider aliases? And how do you create one?**

A: To substitute an alternative implementation for a provider. Can create like so: { provide: LoggerService, useClass: DateLoggerService }

## Lifecycle Hooks Questions:

**What is the possible order of lifecycle hooks in Angular?**

A: ngOnChanges, ngOnInit, ngDoCheck, ngAfterContentInit, ngAfterContentChecked, ngAfterViewInit, ngAfterViewChecked, ngOnDestroy.

**When will ngOnInit be called?**

A: Called once, after the first ngOnChanges.

**How would you make use of onNgInit()?**

PA: Fetch initial component data (e.g. from server).

**What would you consider a thing you should be careful doing on onNgInit()?**

A: You cannot expect the component's children's data-bound properties to have been checked at this point.

**What is the difference between onNgInit() and constructor() of a component?**

A: A directive’s data-bound input properties are not set until after construction, so that’s one difference.

## Pipes Questions:

**What is a pure pipe?**

A: A pipe that is only executed when Angular detects a pure change to the input value (e.g. new primitive object or new object reference).

**What is an impure pipe?**

A: A pipe that is executed during every component change detection cycle (i.e., often – every keystroke, mouse move).

**What is an async pipe?**

A: An impure pipe that accepts a promise or observable as input and eventually returns emitted values.

**What kind of data can be used with async pipe?**

A: Stateful, asynchronous

**What types of pipes are supported in Angular 2?**

A: Pure and impure pipes (async pipes are a kind of impure pipe).

## Routing Questions:

**What is the difference between RouterModule.forRoot() vs RouterModule.forChild()? Why is it important?**

A: forRoot is a convention for configuring app-wide Router service with routes, whereas forChild is for configuring the routes of lazy-loaded modules.

**How does loadChildren property work?**

A: The Router calls it to dynamically load lazy loaded modules for particular routes.

**When does a lazy loaded module get loaded?**

A: When its related route is first requested.

**How would you use a Route Guard?**

A: You would implement CanActivate or CanDeactivate and specify that guard class in the route path you’re guarding.

**What are some different types of RouteGuards?**

A: CanActivate, CanDeactivate, CanLoad, Resolve, etc.

**How would you intercept 404 errors in Angular 2?**

A: Can provide a final wildcard path like so: { path: ‘\*\*’, component: PageNotFoundComponent }

**This link doesn't work. Why? How do I fix it?** <div routerLink='product.id'></div>

A: <a [routerLink]=”[’product.id’]”>{{product.id}}</a>

**What do route guards return?**

A: boolean or a Promise/Observable resolving to boolean value.

**What is <router-outlet> for?**

A: Specifies the place where routes are mounted in the application.

## Styling Questions:

**How would you select a custom component to style it?**

A: Using the :host pseudo-class selector in your component's styles.

**How do you reference the host of a component?**

A: Let DI inject an ElementRef into the constructor of your component.

**What pseudo-class selector targets styles in the element that hosts the component?**

A: The :host pseudo class selector.

**How would you select all the child components' elements?**

A: With the @ViewChildren decorator, like for example:

@ViewChildren(ChildDirective) viewChildren: QueryList<ChildDirective>;

**How would you select a css class in any ancestor of the component host element, all the way up to the document root?**

A: Using the :host-context() pseudo-class selector.

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

A: Use the /deep/ selector along with :host pseudo-class selector.

**What does :host-context() pseudo-class selector target?**

A: The :host-context() selector looks for a CSS class in any ancestor of the component host element, up to the document root.

**What does the following css do?** :host-context(.theme-light) h2 { background-color: red; }

A: Will change this component’s background-color to red if the context of the host has the .theme-light class applied.

## Forms Questions:

**When do you use template driven vs model driven forms? Why?**

A: Template driven forms make more sense for simpler forms, at least in terms of validation. Model driven or Reactive forms lend themselves to easier testing of the validation logic, so if that’s complex, Reactive forms make more sense. There’s also the issue of asynchronous (template driven forms) vs. synchronous (model driven).

**How do you submit a form?**

PA: use the ngSubmit event binding like so: <form (ngSubmit)="onSubmit()" …>

**What's the difference between NgForm, FormGroup, and FormControl? How do they work together?**

A: FormGroup tracks the value and validity state of a group of AbstractControl instances. FormControl does the same for an individual control. NgForm is a directive that Angular automatically attaches to each <form> tag. It has its own ‘valid’ property which is true only if every contained control is valid.

**What's the advantage of using FormBuilder?**

A: Reduces repetition and clutter by handling details of control creation for you.

**How do you add form validation to a form built with FormBuilder?**

A: pass in Validator objects along with the FormControl objects...

**What's the difference between dirty, touched, and pristine on a form element?**

A: Dirty means it contains user data, touched means the user has at least done something with a particular control (perhaps just literally ‘touched’ it by giving it focus?), and pristine means the control has not been touched at all by the user.

**How can you access validation errors in the template to display error messages?**

PA: Use formErrors

**What is async validation and how is it done?**

A: Verifying some field using some asynchronous call (perhaps a server call)… return a Promise<ValidationResult> from your validator. When creating a FormControl object, you can pass an asynchronous validator into the constructor (e.g. new FormControl(‘value’, syncValidator, asyncValidator)).

**What is patchValue used for?**

A: Setting a form value (one or more fields with an object) bypassing validation.

## Observables Questions

**What are observables?**

A: They provide a declarative way of message passing between publishers and subscribers in an application. They typically produce one or more values over time, which are subscribed to by observers. They provide some advantages over promises.

**What is RxJS?**

A: RxJS stands for Reactive Extensions for JavaScript, and is a reactive programming library centered around observables and operators making it easier to write complex asynchronous code.

**How do observables differ from promises?**

A: Observables are declarative; promises execute immediately upon creation. Observables can provide many values, whereas promises provide one value. Observables are cancellable, whereas promiess aren't. Error handling also differs between them.

**What are some advantages to using observables?**

A: Observables are cancellable; they come with powerful transformative functions (especially when using RxJS) to make asynchronous coding easier.

**Does an observable compute anything if it has no calls to subscribe?**

A: No, it will not.

## Animations Questions

**How do you define transition between two states?**

PA: Using the transition and animate function in an animations block like so: animations: [transition('inactive => active'), animate('100 ms ease-in')]

**How do you define a wildcare state?**

A: Using the asterisk - example: transition('\* => active'), animate('100ms ease-in'))

## Architecture / Framework Questions:

**What are some of the top level building blocks of the Angular framework?**

A: Services, Templates, Modules, Components, Providers, etc.

**What is AOT?**

A: Ahead of time compilation.

**What is Reactive programming and how does it relate to Angular?**

A: It's an "asynchronous programming paradigm concerned with data streams and the propagation of change."

**What are some differences between Angular 2 and 4?**

A: Improvements in AOT, allowing the "else" clause in ngIf, support for TypeScript 2.1, breaking out the animations package...

**What are some security related features built in to the Angular framework?**

A: Sanitation, to prevent cross site scripting. Built-in support in the HttpClient to prevent cross-site request forgery.

**How can you bypass sanitation in Angular and why would you do so?**

A: To inject known safe code, you can bypass sanitation (e.g. to embed an iframe).

**What is a good use case for ngrx/store?**

A: Complex application state management requirements, involving asynchronous requests to update state.

**What is Redux and how does it relate to an Angular app?**

A: It's a way to manage application state and improve maintainability of asynchronicity in your application by providing a single source of truth for the application state, and a unidirectional flow of data change in the application. ngrx/store is one implementation of Redux principles.

**What would be a good use case for having your own routing module?**

A: An application whose requirements imply having many routes, and potentially route guards, and child routes.

**Where would you configure TypeScript specific compiler options for your project?**

A: In the tsconfig.json file.A

**What is the tslint.json file used for?**

A: Linting the TypeScript code (making sure it conforms to certain standards / conventions).

**What are some changes introduced in Angular 4?**

A: Introduction of the else clause in ngIf; splitting out of animation package; support for TypeScript 2.1; improvements around AOT.

**What are some changes introduced in Angular 5?**

A: New version of HttpClient; build optimizer; Universal State Transfer API (allows sharing state of app between server and client easily); support for TypeScript 2.3.

## API Questions:

**What does this line do?** @HostBinding('[class.valid]') isValid;

A: Applies the css class ‘valid’ to whatever is using this directive conditionally based on the value of isValid.

**Why would you use renderer methods instead of using native element methods?**

A: Potentially if you’re rendering to something besides the browser, e.g. rendering native elements on a mobile device, or server side rendering (?).

**What is the point of calling renderer.invokeElementMethod(rendererEl, methodName)?**

A: To invoke a method on a particular element but avoid direct DOM access (so we don’t tie our code just to the browser).

**How would you control size of an element on resize of the window in a component?**

A: @HostListener('window:resize', ['$event']) onResize(event: any) { this.calculateBodyHeight(); }

**What would be a good use for NgZone service?**

A: Running an asynchronous process outside of Angular change detection.

**How would you protect a component being activated through the router?**

A: Route Guards

**How would you insert an embedded view from a prepared TemplateRef?**

PA: `viewContainerRef.createEmbeddedView(templateRef);

## Testing Questions

**What is Protractor?**

A: E2E (end-to-end) testing framework.

**What is Karma?**

A: Unit test testing library.

**What are spec files?**

A: Jasmine unit test files.

**What is TestBed?**

A: Class to create testable component fixtures.

**What does detectChanges to in Angular jasmine tests?**

A: propagates changes to the DOM by running Angular change detection.

**Why would you use a spy in a test?**

A: To verify a particular value was returned or a method was called, for example when calling a service.

### What is Angular Framework?

Angular is a **typeScript-based open-source** front-end platform that makes it easy to build applications with in web/mobile/desktop. The major features of this framework such as declarative templates, dependency injection, end to end tooling, and many more other features are used to ease the development.

### What is the difference between AngularJS and Angular?

Angular is a completely revived component-based framework in which an application is a tree of individual components.

Some of the major difference in tabular form

|  |  |
| --- | --- |
| **AngularJS** | **Angular** |
| It is based on MVC architecture | This is based on Service/Controller |
| This uses use JavaScript to build the application | Introduced the typescript to write the application |
| Based on controllers concept | This is a component based UI approach |
| Not a mobile friendly framework | Developed considering mobile platform |
| Difficulty in SEO friendly application development | Ease to create SEO friendly applications |

### What is typescript?

TypeScript is a typed superset of JavaScript created by Microsoft that adds optional types, classes, async/await, and many other features, and compiles to plain JavaScript. Angular built entirely in TypeScript and used as a primary language. You can install it globally as

npm install -g typescript

Let's see a simple example of typescript usage,

function greeter(person: string) { return "Hello, " + person;}let user = "Sudheer";document.body.innerHTML = greeter(user);

The greeter method allows only string type as argument.

### Write a pictorial diagram of Angular architecture?

The main building blocks of an Angular application is shown in the below diagram 

### What are the key components of Angular?

Angular has the below key components,

* 1. **Component:** These are the basic building blocks of angular application to control HTML views.
  2. **Modules:** An angular module is set of angular basic building blocks like component, directives, services etc. An application is divided into logical pieces and each piece of code is called as "module" which perform a single task.
  3. **Templates:** This represent the views of an Angular application.
  4. **Services:** It is used to create components which can be shared across the entire application.
  5. **Metadata:** This can be used to add more data to an Angular class.

### What are directives?

Directives add behaviour to an existing DOM element or an existing component instance.

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

Now this directive extends HTML element behavior with a yellow background as below

<p myHighlight>Highlight me!</p>

### What are components?

Components are the most basic UI building block of an Angular app which formed a tree of Angular components. These components are subset of directives. Unlike directives, components always have a template and only one component can be instantiated per an element in a template. Let's see a simple example of Angular component

import { Component } from '@angular/core';@Component ({ selector: 'my-app', template: ` <div> <h1>{{title}}</h1> <div>Learn Angular6 with examples</div> </div> `,})export class AppComponent { title: string = 'Welcome to Angular world';}

### What are the differences between Component and Directive?

In a short note, A component(@component) is a directive-with-a-template.

Some of the major differences are mentioned in a tabular form

|  |  |
| --- | --- |
| **Component** | **Directive** |
| To register a component we use @Component meta-data annotation | To register directives we use @Directive meta-data annotation |
| Components are typically used to create UI widgets | Directive is used to add behavior to an existing DOM element |
| Component is used to break up the application into smaller components | Directive is use to design re-usable components |
| Only one component can be present per DOM element | Many directives can be used per DOM element |
| @View decorator or templateurl/template are mandatory | Directive doesn't use View |

### What is a template?

A template is a HTML view where you can display data by binding controls to properties of an Angular component. You can store your component's template in one of two places. You can define it inline using the template property, or you can define the template in a separate HTML file and link to it in the component metadata using the @Component decorator's templateUrl property. **Using inline template with template syntax,**

import { Component } from '@angular/core';@Component ({ selector: 'my-app', template: ' <div> <h1>{{title}}</h1> <div>Learn Angular</div> </div> '})export class AppComponent { title: string = 'Hello World';}

**Using separate template file such as app.component.html**

import { Component } from '@angular/core';@Component ({ selector: 'my-app', templateUrl: 'app/app.component.html'})export class AppComponent { title: string = 'Hello World';}

### What is a module?

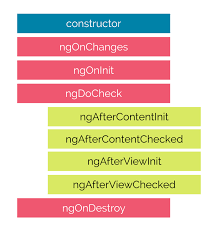
Modules are logical boundaries in your application and the application is divided into separate modules to separate the functionality of your application. Lets take an example of **app.module.ts** root module declared with **@NgModule**decorator as below,

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 { }

The NgModule decorator has three options

* 1. The imports option is used to import other dependent modules. The BrowserModule is required by default for any web based angular application
  2. The declarations option is used to define components in the respective module
  3. The bootstrap option tells Angular which Component to bootstrap in the application

### What are lifecycle hooks available?

Angular application goes through an entire set of processes or has a lifecycle right from its initiation to the end of the application. The representation of lifecycle in pictorial representation as follows, 

The description of each lifecycle method is as below,

* 1. **ngOnChanges:** When the value of a data bound property changes, then this method is called.
  2. **ngOnInit:** This is called whenever the initialization of the directive/component after Angular first displays the data-bound properties happens.
  3. **ngDoCheck:** This is for the detection and to act on changes that Angular can't or won't detect on its own.
  4. **ngAfterContentInit:** This is called in response after Angular projects external content into the component's view.
  5. **ngAfterContentChecked:** This is called in response after Angular checks the content projected into the component.
  6. **ngAfterViewInit:** This is called in response after Angular initializes the component's views and child views.
  7. **ngAfterViewChecked:** This is called in response after Angular checks the component's views and child views.
  8. **ngOnDestroy:** This is the cleanup phase just before Angular destroys the directive/component.

### What is a data binding?

Data binding is a core concept in Angular and allows to define communication between a component and the DOM, making it very easy to define interactive applications without worrying about pushing and pulling data. There are four forms of data binding(divided as 3 categories) which differ in the way the data is flowing.

**From the Component to the DOM:** **Interpolation:** {{ value }}: Adds the value of a property from the component

<li>Name: {{ user.name }}</li><li>Address: {{ user.address }}</li>

**Property binding:** [property]=”value”: The value is passed from the component to the specified property or simple HTML attribute

<input type="email" [value]="user.email">

**From the DOM to the Component:** **Event binding: (event)=”function”:** When a specific DOM event happens (eg.: click, change, keyup), call the specified method in the component

<button (click)="logout()"></button>

**Two-way binding:** **Two-way data binding:** [(ngModel)]=”value”: Two-way data binding allows to have the data flow both ways. For example, in the below code snippet, both the email DOM input and component email property are in sync

<input type="email" [(ngModel)]="user.email">

### What is metadata?

Metadata is used to decorate a class so that it can configure the expected behavior of the class. The metadata is represented by decorators

**Class decorators**, e.g. @Component and @NgModule

import { NgModule, Component } from '@angular/core';@Component({ selector: 'my-component', template: '<div>Class decorator</div>',})export class MyComponent { constructor() { console.log('Hey I am a component!'); }}@NgModule({ imports: [], declarations: [],})export class MyModule { constructor() { console.log('Hey I am a module!'); }}

**Property decorators** Used for properties inside classes, e.g. @Input and @Output

import { Component, Input } from '@angular/core';@Component({ selector: 'my-component', template: '<div>Property decorator</div>'})export class MyComponent { @Input() title: string;}

**Method decorators** Used for methods inside classes, e.g. @HostListener

import { Component, HostListener } from '@angular/core';@Component({ selector: 'my-component', template: '<div>Method decorator</div>'})export class MyComponent { @HostListener('click', ['$event']) onHostClick(event: Event) { // clicked, `event` available }}

**Parameter decorators** Used for parameters inside class constructors, e.g. @Inject

import { Component, Inject } from '@angular/core';import { MyService } from './my-service';@Component({ selector: 'my-component', template: '<div>Parameter decorator</div>'})export class MyComponent { constructor(@Inject(MyService) myService) { console.log(myService); // MyService }}

### What is angular CLI?

Angular CLI(**Command Line Interface**) is a command line interface to scaffold and build angular apps using nodejs style (commonJs) modules. You need to install using below npm command,

npm install @angular/cli@latest

Below are the list of few commands, which will come handy while creating angular projects

**Creating New Project:** ng new

**Generating Components, Directives & Services:** ng generate/g The different types of commands would be,

* + ng generate class my-new-class: add a class to your application
  + ng generate component my-new-component: add a component to your application
  + ng generate directive my-new-directive: add a directive to your application
  + ng generate enum my-new-enum: add an enum to your application
  + ng generate module my-new-module: add a module to your application
  + ng generate pipe my-new-pipe: add a pipe to your application
  + ng generate service my-new-service: add a service to your application

**Running the Project:** ng serve

### What is the difference between constructor and ngOnInit?

TypeScript classes has a default method called constructor which is normally used for the initialization purpose. Whereas ngOnInit method is specific to Angular, especially used to define Angular bindings. Even though constructor getting called first, it is preferred to move all of your Angular bindings to ngOnInit method. In order to use ngOnInit, you need to implement OnInit interface as below,

export class App implements OnInit{ constructor(){ //called first time before the ngOnInit() } ngOnInit(){ //called after the constructor and called after the first ngOnChanges() }}

### What is a service?

A service is used when a common functionality needs to be provided to various modules. Services allow for greater separation of concerns for your application and better modularity by allowing you to extract common functionality out of components. Let's create a repoService which can be used across components,

import { Injectable } from '@angular/core';import { Http } from '@angular/http';@Injectable() // The Injectable decorator is required for dependency injection to workexport class RepoService{ constructor(private http: Http){ } fetchAll(){ return this.http.get('https://api.github.com/repositories').map(res => res.json()); }}

The above service uses Http service as a dependency.

### What is dependency injection in Angular?

Dependency injection (DI), is an important application design pattern in which a class asks for dependencies from external sources rather than creating them itself. Angular comes with its own dependency injection framework for resolving dependencies( services or objects that a class needs to perform its function).So you can have your services depend on other services throughout your application.

### How is Dependency Hierarchy formed?

### What is the purpose of async pipe?

The AsyncPipe subscribes to an observable or promise and returns the latest value it has emitted. When a new value is emitted, the pipe marks the component to be checked for changes. Let's take a time observable which continuously updates the view for every 2 seconds with the current time.

@Component({ selector: 'async-observable-pipe', template: `<div><code>observable|async</code>: Time: {{ time | async }}</div>`})export class AsyncObservablePipeComponent { time = new Observable(observer => setInterval(() => observer.next(new Date().toString()), 2000) );}

### What is the option to choose between inline and external template file?

You can store your component's template in one of two places. You can define it inline using the **template** property, or you can define the template in a separate HTML file and link to it in the component metadata using the **@Component**decorator's **templateUrl** property. The choice between inline and separate HTML is a matter of taste, circumstances, and organization policy. But normally we use inline template for small portion of code and external template file for bigger views. By default, the Angular CLI generates components with a template file. But you can override that with the below command,

ng generate component hero -it

### What is the purpose of ngFor directive?

We use Angular ngFor directive in the template to display each item in the list. For example, here we iterate over list of users,

<li \*ngFor="let user of users"> {{ user }}</li>

The user variable in the ngFor double-quoted instruction is a **template input variable**

### What is the purpose of ngIf directive?

Sometimes an app needs to display a view or a portion of a view only under specific circumstances. The Angular ngIf directive inserts or removes an element based on a truthy/falsy condition. Let's take an example to display a message if the user age is more than 18,

<p \*ngIf="user.age > 18">You are not eligible for student pass!</p>

**Note:** Angular isn't showing and hiding the message. It is adding and removing the paragraph element from the DOM. That improves performance, especially in the larger projects with many data bindings.

### What happens if you use script tag inside template?

Angular recognizes the value as unsafe and automatically sanitizes it, which removes the **<script>** tag but keeps safe content such as the text content of the <script> tag. This way it eliminates the risk of script injection attacks. If you still use it then it will be ignored and a warning appears in the browser console. Let's take an example of innerHtml property binding which causes XSS vulnerability,

export class InnerHtmlBindingComponent { // For example, a user/attacker-controlled value from a URL. htmlSnippet = 'Template <script>alert("0wned")</script> <b>Syntax</b>';}

### What is interpolation?

Interpolation is a special syntax that Angular converts into property binding. It’s a convenient alternative to property binding. It is represented by double curly braces({{}}). The text between the braces is often the name of a component property. Angular replaces that name with the string value of the corresponding component property. Let's take an example,

<h3> {{title}} <img src="{{url}}" style="height:30px"></h3>

In the example above, Angular evaluates the title and url properties and fills in the blanks, first displaying a bold application title and then a URL.

### What are template expressions?

A template expression produces a value similar to any Javascript expression. Angular executes the expression and assigns it to a property of a binding target; the target might be an HTML element, a component, or a directive. In the property binding, a template expression appears in quotes to the right of the = symbol as in [property]="expression". In interpolation syntax, the template expression is surrounded by double curly braces. For example, in the below interpolation, the template expression is {{username}},

<h3>{{username}}, welcome to Angular</h3>

The below javascript expressions are prohibited in template expression

assignments (=, +=, -=, ...)

new

chaining expressions with ; or ,

increment and decrement operators (++ and --)

### What are template statements?

A template statement responds to an event raised by a binding target such as an element, component, or directive. The template statements appear in quotes to the right of the = symbol like **(event)="statement"**. Let's take an example of button click event's statement

<button (click)="editProfile()">Edit Profile</button>

In the above expression, editProfile is a template statement. The below JavaScript syntax expressions are not allowed.

new

increment and decrement operators, ++ and --

operator assignment, such as += and -=

the bitwise operators | and &

the template expression operators

### How do you categorize data binding types?

Binding types can be grouped into three categories distinguished by the direction of data flow. They are listed as below,

From the source-to-view

From view-to-source

View-to-source-to-view

The possible binding syntax can be tabularized as below,

| Data direction | Syntax | Type | |---- | --------- | From the source-to-view(One-way) | 1. {{expression}} 2. [target]="expression" 3. bind-target="expression" | Interpolation, Property, Attribute, Class, Style| | From view-to-source(One-way) | 1. (target)="statement" 2. on-target="statement" | Event | | View-to-source-to-view(Two-way)| 1. [(target)]="expression" 2. bindon-target="expression"| Two-way |

### What are the new features of Angular 2?

Angular 2 is a platform that encompasses a wide range of capabilities. Some new features were added in Angular 2 which includes:

* + Universal server rendering- It is the library which is used to make building universal apps a smooth experience. It is an important feature of Angular 2.
  + A mobile toolkit- It provides all the mobile toolkit and techniques to build high-performance mobile applications. The web applications which are developed using the mobile toolkit can be loaded on any device with or without internet connection which is a great advantage.
  + A command line interface-it can generate components, routes, services, and pipes with the help of commands.
  + Data binding- data binding has been improved in Angular 2. So, whatever DOM element property you need to bind, you just wrap it in square brackets. E.g.-<img[src]='product.image' />
  + Modular- various modules have been removed from angular’s core, which has resulted in better performance.
  + Modern- Angular 2 has been targeted as modern browsers in which various hacks that make angular harder to develop have been removed.

### How do you define transition between two states in angular?

Transitions between two states take place so that we can build simple animations between two states driven by a model attribute. Transition basically means navigating from the current state to a new state. In angular, the transition is an animation-specific function which is used in angular’s animation DSL language. Transition declares the sequence of animation steps that will be executed when the entered value is satisfied. A function is provided an argument for a transition and it will be executed each time a state change occurs. In this, if the function is true, then the animation will run else it won’t get executed.

These animation transitions are placed within the animation triggers. The transition depends upon what the animation was in the previous state and what it will become in the next state. In other words, if a transition is defined that matches the old/current state criteria then the associated animation will be triggered.

Syntax:

function transition (stateChangeExpr: string,steps: AnimationMetadata | AnimationMetadata []):AnimationTransitionMetadata;

### How to declare a component in Angular 2?

Components in Angular 2 are simply directives that are always associated with a direct template. Angular 2 components have an extremely well defined life-cycle. When working with angular components, we can make use of interfaces which allows us to implement functionality for different times in a components lifecycle. A component must belong to an NgModule in order for it to be usable by another component or application. Components can even control their runtime behaviour by implementing various Life-cycle hooks.

##### Declaration of component:

@component ({selector: 'great', template: 'hello {{name}}!'})Class greet{Name: string = 'world';}

Components always have a template and only one component can be instantiated per an element in a template. When a component is instantiated, angular creates a change detector, which is responsible for propagating the component’s building.

### What is the difference between observable and promises?

The differences between observable and promises are:

* + Observable is a more powerful way of handling HTTP asynchronous requests. Whereas, A promise handles a single event when an asynchronous operation completes or fails.
  + An observable is like a stream which allows passing zero or more events where the callback is called for each event. Whereas, A promise eventually calls the success or failed callback even when you don’t need the notification or the result it provides anymore.
  + Observable works with multiple values for a particular time. Whereas, Promises works with and even returns a single value at a time.
  + Observables can be canceled. Whereas, Promises cannot be canceled.
  + Observable supports map, filter, reduce and similar operators. Whereas, Promises have more readable codes with try/catch and async/await.
  + In observable, one operator ‘retry’ can be used to retry whenever needed. Whereas, Promises cannot be retried. A promise should have access to the original function that returned the promise in order to have a retry capability.

### List the differences between Angular 2 components vs. directives.

Apart from components, directives are also used in Angular 2 which allows us to attach behavior to elements in DOM. There are certain differences between the components and directives in Angular 2. They are:

* + In Angular 2, a component is a directive with a view whereas a directive is a decorator with no view. Components are the specific type of directive that allows us to utilize web component functionality throughout our application. Whereas, Directive is the mechanism by which we attach behavior to elements.
  + A component is used to break up the application into smaller components. Whereas, Directive is used to design the re-usable components.
  + Components can be used to define pipes. Whereas, We cannot define pipes using directives.
  + Components can be present per DOM element. Whereas, Directive is used to add behavior to an existing DOM element.

### What is ECMAScript?

ECMAScript is a standard for scripting languages. It is a subset of Javascript. Languages such as ActionScript, JavaScript use ECMAScript as its core. ECMA stands for European Computer Manufacturer’s Association. Coders commonly use ECMAScript for client-side scripting on the World Wide Web. It is also used for server applications and services. It includes structured, dynamic, functional, and prototype-based features. The ECMAScript was developed by Brendan Eich of Netscape. The ECMAScript is standardized by the ECMA international standards organization in the ECMA-262 and ECMA-402 specifications. It is a programming language which is designed specifically for acting on an existing entity or system. It provides the rules, details, and guidelines that a scripting language must observe to be considered ECMAScript compliant.

### What is Traceur Compiler?

Traceur is a compiler which takes ECMAScript and compiles it down to regular Javascript that runs in your browser. Traceur can be used in several ways like- typing or pasting the ES6 code into the read-eval-print-loop page, or by including traceur in the web page and compiling ES6 code content on the fly, or many other ways. Even traceur is written in ES6, compiled to ES5. The main goal of a traceur compiler is to inform the designs of Javascript features and allows us to write the code in a better manner. Nowadays, traceur compilers are broadly used in Angular 2 platform. It also supports transpilling and type checking via type annotations.

### List the modern browsers supported by Angular 2.

Angular supports most of the recent browsers some of which are:

* + Google Chrome
  + Firefox
  + Edge
  + IE for versions 9-11
  + Safari
  + iOS 7.1
  + Android 4.1
  + IE Mobile

### When to use Ngoninit and constructor in Angular 2?

Constructors are used for initializing class members and also for dependency injection. Ngonlnit is used for the initialization work. Both of these methods are called when the component is created. It is really important that we should know, when to and how to use them. These are used for providing the best structure for your component’s code. A constructor method is a pre-defined method in the constructor class which is only called when the class is instantiated. It is also used for properly initializing the fields. The constructor in Angular 2 is used to create a new instance of the class. Ngonlnit is the class we import when we implement the constructor in order to use it in a class. The method used in this case is ngOnlnit(). This method helps in initializing the directive or the component after the data-bound properties are displayed and the directive or components input is set.

### How to cache an observable data in Angular 2?

Caching of an observable data is done with the help of “observable.cache”. We can use caching in order to cache the response in the memory and then, on the next subscription, instead of requesting the remote server again. This operator is used at the end of the string. Caching is important for the performance, especially on bandwidth restricted devices and slow networks. You should have a good understanding of caching while working with promises but while translating it to observable, it is a bit difficult. Therefore, when interacting with observables, we typically set up a subscription on the consumer side and react to values coming through the pipe. We can easily add caching to the observables by adding publishReplay(1) and refCount.

### List out the differences between ActivatedRoute and RouterState, with reference to Angular 2.

Here are some of the differences between ActivatedRoute and RouterState with reference to Angular 2:-

* + ActivatedRoute consists of the information about a route associated with a component loaded in an outlet. Whereas, RouterState represents the state in which the writer actually is.
  + We need ActivatedRouteSnapchat to traverse all the activated routes. Whereas, during a navigation, after redirects have been applied, the router creates a RouterStateSnapshot.
  + ActivatedRouteSnapshot has old data. When route changes, ActivateRouteSnapshot has data from previous route. Whereas, the RouterState cares about application components, or, to be more specific, about their arrangements.

### What would you have in a shared module in Angular 2?

Shared module is used to import the services in both eager and lazy loaded module. We all know that lazy loaded modules create their own branch on the dependency injection tree. Shared module consists of the services that are registered by the angular in the root app injector. For this, we need not import it in the lazy module because lazy loaded modules already have access to the services defined at the root. Components, pipes and directives are also defined in the shared module. Other modules that import the shared module can use it in their templates. This means that the modules can be imported normally in the lazy loaded module. The shared module contains the code that will be used across the applications and featured modules. It also consists of the common template components. “Dumb components” should also be present in the shared module. It typically consists of some common angular modules too. When you are importing the shared module, you will also need to import the module with its providers, because there is no app module in the test.

### What do you mean by a structural directive in Angular 2?

Structural directives are used to manipulate DOM in angular. Structural directives are responsible for HTML layout. By adding, removing, or manipulating LMNs in angular, they shape or reshape the structure of DOM. This structural directive is applied to a host element with the help of other directives. The directives then do whatever it is supposed to do with that host element and its descendants. Structural directives can be easily recognized. It can also delay the instantiation of a component or an element. It can also be used for cosmetic effect or manually handling the timing of the loading of components. Structural directives are bound to a template. The two most common structural directives are “ngIf” and “ngFor”. The process occurring in a structural directive is dynamic.

### What do you understand by a template variable? How is it used?

A template in Angular 2 is used to instantiate embedded views. A template variable can be accessed in two ways. Either by placing a directive on an element and have the template variable for this embedded view injected into the constructor of the directive using the template variable token, or you can query for the template variable from a component or a directive via the query. A template variable in Angular 2 is a reference to a DOM element or directive within a template. Template variables are used to access the values of DOM element properties. It is declared with the help of “#” and “ref-“as a prefix. For example: – #myVar and ref-myVar. Template variable names cannot be made duplicate as in this way, it might give unpredictable values. The scope of a reference variable is the entire template. It can be used anywhere inside a template. In Angular 2, a component needs to have a view and to define a view, a template variable is used. It allows us to express data and property binding, event binding and template concerns.

### Explain the concept of lazy loading in Angular 2.

Lazy loading is a module which is used to decrease the start-up time. When lazy is used, then our system application does not need to load everything at once. It only needs to load what the user expects to see when the application first loads. The modules which are lazily loaded will only be loaded when the user navigates to their routes. Lazy loading improves the performance of our system applications. It keeps the initial payload small and these smaller payloads lead to faster download speeds. It helps in lowering the resource cost, especially on mobile networks. If a user doesn’t visit a section of the application, they won’t ever download those resources. The concept of lazy loading in angular requires us to format the application in a certain way. All the assets that are to be lazy loaded should be added to its own module. Lazy loading is setup in the main routing file. Lazy loading overcomes the problem of slow loading of applications in their own way which hence improves the loading time of the application.

Lazy loading can be done only in four steps:–

* + Update your route file
  + Install angular-router-loader and add the loader to your webpack configuration file.
  + Define the lazy routes
  + Import the routes to the module.

### What is the difference between constructor and ngOnlnit in Angular js?

The comprehensive comparison that taps into components initialization process is given below:-

* + ngonInit is just a method in a class which structurally is not different to any other method in a class. Whereas, a constructor is a completely different thing. It will be called when an instance of a class is created.
  + A class constructor in angular is used to inject dependencies, which is called constructor injection pattern. Whereas, when ngOnInit is called, it has finished creating a component DOM, injected all required dependencies through constructor and processed input bindings.
  + A constructor is a default method of the class that is executed when the class is instantiated. Whereas, ngOnInit is a life cycle hook called by Angular 2 to indicate that angular is done creating the component.
  + ngOnInit relies on the binding of the component. Whereas, it is not the case when a constructor is used.

### What is the meaning of component lifecycle in Angular 2?

The component lifecycle hooks overview the life cycle sequence and the interfaces. Angular manages the life cycle of a component. Angular creates it, renders it. It can also create and render its children. It also checks when its data-bound properties change. It can even destroy it before removing it from the DOM. The life cycle hook offered by angular provides the visibility into these key life moments and the ability to act when they occur. The components go through an entire set of processes or life cycle right from its initiation to the end of the application.

There are a number of lifecycle hooks which are listed below:–

* + ngOnChanges
  + ngOnInit
  + ngDoCheck
  + ngAfterContentInit
  + ngAfterContentChecked
  + ngAfterViewInit
  + ngAfterViewChecked
  + ngOnDestroy

### What is the use of ngForTrackBy directive?

For iterating over a collection in Angular 2, the ngFor directive is used which instantiates a template once per item from the collection. If a data needs to be changed at some point in the collection, then a problem occurs because angular cannot keep a track of items in the collection and has no knowledge of the items which were added or deleted. This results in the deletion of all the DOM elements that are associated with the data and are again created. If the collection is big, then it becomes more complicated because a lot of DOM manipulation occurs which are expensive. So, to solve this problem, a trackBy function is used which takes the index and the current item as arguments and returns the unique identifier for this item.

### List the key components of Angular 2?

The Angular 2 comprises of the following key components:

* + Module – This is used to break the application into the logical pieces of the program code and each piece of code or module is designed to perform a single and unique task.
  + Component – This is used to bring the modules together.
  + Templates – This is used to define the Views of an Angular JS application.
  + Metadata – This is used to add more data to an Angular JS application.
  + Service – This component is used to develop the components, which can be used to share in the entire application.

### What is a template in Angular 2?

The template in Angular 2 is used to define the views of the AngularJS Application.

### How will you convert a string into a percentage?

To convert a string into a percentage format, a percent filter is used.

### Explain component specific hooks?

Below are few component specific hooks in Angular2.

* + ngafterContentinit: It initializes the component content
  + ngAfterConctentChecked: It checks the binding of the external content.
  + ngafterViewinit: It creates the component view.
  + ngAfterviewChecked: It checks the bindings of the component’s view.

### What is CLI?

CLI is the acronym of Command Line Interface, which can be used to create the Angular JS application.Using CLI, you can also create a unit and end-to-end tests for the Angular application.

### What is AOT compilation?

AOT stands for Ahead of Time.It is the compilation in which Angular compiles the components and templates to JavaScript and HTML while developing.

### What are Event emitters?

An Event emitter is a class defined in core module that can be used by components and directives to emit custom events.

### What is Angular @ RouteParams?

The RouteParams are used to map the given URL’s based on the route URLs and they become optional parameters for that route.

### Explain Angular 2 hidden property?

The hidden property in Angular 2 is a special case.

* + The property is more powerful and is used to bind any property of the elements.
  + It is considered the closest cousin of ngshow and nghide.
  + It sets the display property “display: none”.

### Why are decorators used in Angular 2?

In Angular 2, decorators are used as an identifier of class or type of the object that is created by the TypeScript.The Angular 2 identifies the class below decorator call as the definition of the class and extends the decorator specific properties with the class definition.

### Explain host decorator in Angular 2?

The host decorators in Angular 2 bind the properties of components with UI element values.The properties inside a component class definition which are decorated with @HostBinding are accessed in a template from the assigned property that is @HostBinding()title=’Our title'( whatever the title is).

### What are Pipes in Angular 2?

Pipes in Angular 2 are used in templates in order to convert them into a content that is user-friendly and readable one within the interpolation braces that is {{release| date}}, here the symbol “|” denotes the pipe.

How can you handle errors in Angular 2 Applications?

The Angular 2 Applications provide with the option of error handling.The errors in Angular 2 can be handled by including the ReactJS catch library and later using the catch function.

* + The catch function, which is used after adding the catch library contains the link to the Error handler function.
  + And in this error, handler function, the errors are sent to the error console, and also the errors are thrown back to continue the execution.
  + So, whenever an error occurs it will be redirected to the error console of the web.

### Can you automate porting Angular 1 code to Angular 2?

No, currently there is not any such tool available that ports the Angular 1 code to the Angular 2 code.  
In the process of porting, the Angular 1 code to Angular 2, the side by side manual conversion of Angular 1 directives to the Angular 2 components takes place because they are two different frameworks and hence requires different approaches to solve the same problem.

Which module does is required for every Angular 2 app?

AppModule is required for every Angular 2 app.**36.**

### What is .angular-cli.json. Where can I find it.

angular-cli.json is used to configure a project in angular2. You can find it in the root folder of your angular2 Project.

Is Angular Modules and ES modules are the same?

No, Both are different.

### Explain component decorators in Angular4.

A decorator is the core concept when developing an angular framework with version 2 and above. It may become a core language feature for JavaScript soon. In angular 4, decorators are used extensively and are also used to compile a code. There are 4 different types of decorators:

* + Class decorators
  + Property decorators
  + Method decorators
  + Parameter decorators
* A decorator is a function that is invoked with a prefixed “@” symbol and is immediately followed by a class, parameter, method, or property. A decorator returns the same thing which was given as an input but in an augmented form.

### Write the CLI command to generate a component in Angular4.

Components are just simple classes which are declared as components with the help of component decorators.  
It becomes easy to create an application which already works, with the help of angular CLI commands. “Ng generate” is used to generate components, routes, services, and pipes. Simple test shells are also created with the help of this CLI command. For generating a component in angular4 with the help of CLI command, you need to follow the following syntax-

* + ng generate component component\_name;
* It generates the component and adds the component to module declarations.

### Explain the component directory structure of angular4.

Here are the elements which are present in the component directory structure anf modules:-

* + module.ts- in this, the angular module is declared. @NgModule decorator is used which initializes the different aspects of angular applications. AppComponent is also declared in it.
  + components.ts- it simply defines the components in angular and this is the place where the app-root sector is also defined. A title attribute is also declared in the component.
  + component.html- it is the template file of the application which represents the visual parts of our components.

### Explain ngFor directive with an example.

The ngFor directive instantiates a template for every element of the given iterator. The different local variables of the ngFor directive can be used in iterations. The ngFor directive can even be used with the HTML elements. It also performs various changes in DOM. Several exported values can be aliased to local variables with the help of ngFor directive. It allows us to build data presentation lists and tables in our HTML templates. Here’s an example of ngFor directive with the help of HTML:

<tr \*ngFor="hero of heroes"><td>({hero.name})</td></tr>

### Explain property binding or one way binding in angular js.

Basically, property binding means passing data from the component class and setting the value of a given element in the view. It is a one-way binding in which the data is passed from component to the class. It allows us to control the element property values from component to class. Property binding in angular can take place by three ways:

* + Interpolation can be used to define a value, as long as the value being defined is a string.
  + Wrapping brackets around the element property and binding it to the component property is the most common type of property binding.
  + The third way is by adding “bind” before the element property.

### Explain ngIf directive with an example.

The ngIf is a built-in template directive which is used to add or remove some parts of DOM. This addition or removal depends on the expression being true or false.

If the expression is evaluated to false, then the ngIf directive removes the HTML element. If the expression is evaluated to be true, then the element gets added to the DOM.

Syntax- \*ngIf=”<condition>”

Example-<ul \*ngFor="let person of people"\*ngIf="person.age < 30"><li>{{person.name}}</li></ul>

### Write the difference between directive and component in angular js.

In angular js, there are differences between the meta-data annotations. Some of the differences are:

* + A directive is used to add behavior to an existing element. Whereas, a component is used to create a component with attached behavior.
  + “@directive” is used to create a directive. Whereas, “@component” is used to create a component.
  + A directive is used to attach different behaviors to an existing DOM element. Whereas, with the help of component, we can break our application into smaller components.
  + A directive is used to create reusable behavior. Whereas, a component is used to create reusable components.
  + A directive does not require a view. Whereas, a component needs a view via @view.

### What do you understand by Isolated Unit Tests?

As the name implies, unit test is all about testing individual units of code. In order to answer some questions, isolating the unit of code under test is really important. When we do this, we are not forced into creating related pieces such as DOM elements for sorting. With the help of isolated unit tests, it becomes easier to implement everything. To simulate the requests, dependency injections are also provided. The individual sort function can be tested in isolation. And not only the sort function, any function can be tested in isolation.

### What is a routing in angular js?

ngRoute module is used when you want to navigate through different pages of your application but you also want your application to be a single page application. This ngRoute module navigates through different pages of your application without reloading the entire application. The angular js route module should be included to make your application ready for routing. The ngRoute is added as a dependency in the application. The routing engine captures the specific URL requested by the user and renders the view based on the defined routing rules.

### What do you understand by services with reference to angular js?

Services in angular js are used to organize and share code across your application. These are the suitable objects which are wired together with the help of dependency injection. The angular js services are lazily instantiated. The service is only instantiated by angular js only when the application component depends on it. In angular js, new services can be made or can even be used in other built-in services. Over 30 built-in services are present in angular js.

### What are DSL Animation Functions in Angular js. List them.

DSL Animation functions are used for crafting animations in Angular js. Below are list of DSL Animation functions in angular js.

* + trigger()
  + state()
  + transition()
  + group()
  + sequence()
  + style()
  + animate()
  + keyframes()

### How Angular 4 is different from Angular 2?

### What is Isolated Unit Tests?

### How to include an external css in Angular 4?

To include external CSS in Angular 4, open your .angular-cli.json and add your css file path in styles array.

What is difference between Structural and Attribute directives?

### In which directory all external modules and files are stored in Angular 4

Angular4 stores all external modules and files in node\_modules directory.

### What Is Angular 5? Whats new in it?

### Explain Bootstrapping in AngularJs?

### Explain service workers in Angular js?

### Explain how component routing works in Angular. How to define routes and sub-routes in Angular 5?

### Explain NPM?

NPM stands for node package manager. It is used for installing dependencies for javascript packages.

### What is Angular CLI? List the command to install Angular CLI?

Angular CLI is Command Line Interface for Angular that runs Webpack.You can use npm install -g @angular/cli command to install angular CLI.

### How to create a new project in angular js using CLI.

After installing Angular CLI run ng new project-name command to create a new Angular project.

### What are Decorators?

Decorators are functions that adds metadata to class members and functions. It was proposed in ES2016 and implemented in Typescript.

### List the types of Data Binding supported by Angular 5?

Angular 5 supports four types of Data Binding They are

* + String Interpolation
  + Property Binding
  + Event Binding
  + Two-way-binding

### How to run Angular 5 application locally during development?

ng serve command is used to run Angular5 application locally during development. To start development server on specific port ng serve -p aPortNumber command is used.

### What an Angular 5 component made of ? How do you generate a new component?

Angular5 component is made of a Component decorator and a component definition of a class. ng generate component component\_name command is used to generate a component in Angular5.

### How do we import a module in Angular5 ?

Simply use below syntax to import a module in Angular5.

import { ModuleName } from 'someWhere';

### Explain $event in Angular5?

In Angular5 $event is a reserved keyword that represents the data emitted by an event (event data).It is commonly used as a parameter for event based methods.

### What do double curly brackets are used in Angular5?

double curly brackets like {{}} are used form data interpolation in Angular5.

### What is \*ngFor directive used for?

\*ngFor directive is used for Iterating over a list of items and for Generating a new DOM element for each one.

### Explain Webpack?

Webpack is module bundler Bundler for Angular2 or above. It bundles, minifies and transpiles an Angular application.

### What is transpiling?

Transpiling is a process of converting code from one language to another. In Angular, Traceur compiler is used for converting TypeScript to JavaScript so that browsers can understand.

### Explain component life cycle in Angular?

In Angular component life cycle in Angular goes through the following stages.

* + Create
  + Render
  + Create and render children
  + Check for bound data changes and re-render
  + Destroy

### Explain NgModule?

NgModule is a decorator function in Angular that takes a single metadata object whose properties describe the module.

### List some new features comes with Angular6

Latest Key features of Angular 6

* + Added support for creating Custom Elements based on Angular Components.
  + Animations: expose element and params within transition matchers.
  + Bazel: change ng\_package rule to APF v6
  + singleline, multiline and jsdoc comments are now supported
  + compiler-cli : add resource inlining to ngc
  + support for TypeScript 2.7
  + Require node 8 as runtime engine

### Explain Ivy Renderer in Angular?

### What is Bazel and Closure Compiler in Angular6?

### Explain Service Worker in reference to Angular JS

### Write command to generate a component in specific module in Angular cli?

You can generate a component in specific module in AngularJs by running below commnad on CLI.

ng g component component\_name --module=module\_name

### How to generate a module in Angular?

In order to generate a module in Angular, cd to the current project directory and below command.

ng g module module\_name

### Explain Zone in Angular?

### How to update or upgrade angular cli version

In order to upgrade angular-cli package that was installed globally in your system, you need to run following commands

npm uninstall -g angular-clinpm cache clean or npm cache verify (if npm > 5)npm install -g @angular/cli@latest

Instead of upgrading global version of angular-cli you can also upgrade the local version for specific project y running following commands:

rm -rf node\_modulesnpm uninstall --save-dev angular-clinpm install --save-dev @angular/cli@latestnpm install

### List some Inbuilt Pipes available in Angular

Below are the list of few Pipes available in Angular Js

* + DatePipe
  + CurrencyPipe
  + AsyncPipe
  + DecimalPipe
  + TitleCasePipe
  + JsonPipe
  + SlicePipe
  + PercentPipe
  + UpperCasePipe
  + LowerCasePipe

### 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.

### 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. icon

### 

### 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. icon

### 

### 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 the 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 an OOP approach and is made available 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 a 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 CoffeeScript, Dart(Google).
* Finally, it is like, TS makes life easier when compared to JS.

Aaa

TypeScript

### What is Typescript? How is it different from javascript?

Typescript is a free and open-source programming language which is designed and developed by Microsoft. It was designed by Anders Hejlsberg at Microsoft. Typescript is a superset of JavaScript which primarily provides optional static typing, classes and interfaces. Typescript is compiled to provide clean and simple JavaScript code which runs on any browser. It allows JavaScript developers for using highly productive development tools and practices like static checking and code refactoring.

Differences between Typescript and JavaScript are

* + [JavaScript](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/) don’t support Es6 while Typescript supports .
  + JavaScript build up reusable components by using unctions and prototype-based inheritance while Typescript supports Classes that allow programmer to think in more object oriented way .
  + JavaScript don’t have any interfaces while Typescript has interfaces.
  + There is no static typing in JavaScript whereas there is static typing in Typescript.
  + JavaScript has no optional parameter feature while Typescript has optional parameter feature.

### List some features of Typescript?

Features of Typescript are:-

* + Typescript can be compiled to all major versions of Javascript(ES3,ES5,ES6,ES7).
  + Typescript can be used for cross –browser development and is an open source project.
  + Typescript is a superset of JavaScript that provides typed nature to your code.
  + It is used to avoid the disadvantages of JavaScript like Typescript is designed to handle the needs of complex programs.
  + Typescript was debuted as a beta programming language on October 1, 2012 and since then has gone through many versions with improved capabilities.
  + Another key feature of Typescript is in version Typescript 2.6 that covers error suppression comments.
  + Typescript is more predictable and is generally easier to debug.

### List some benefits of using Typescript?

Following are some benefits of using Typescript

* + One of the biggest advantages of Typescript is its code completion and intelligence.
  + It provides the benefits of optional static typing .Here Typescript provides types that can be added to variables, functions, properties etc.
  + Typescript has the ability to compile down to a version of JavaScript that runs on all browsers.
  + Typescript tries to extend JavaScript. Compiler generates JavaScript.
  + Typescript is a backward compatible version of JavaScript that compiles to pure JavaScript which makes their writing easier and faster.
  + Typescript is purely object oriented programming which means it is a programming paradigm based on the concepts of objects.
  + Most important advantage is it offers a “compiler” that can convert to JavaScript equivalent code. And it has a concept of namespace defined by a “module”.

### Who developed Typescript and what is the current stable version of Typescript?

Typescript was developed by Anders Hejlsberg in 2012, who is the lead architect of C# and the creator of Delphi and Turbo Pascal. Typescript is a free and open source programming language maintained by Microsoft. It is a superset of JavaScript. It was developed for the development of large application. The current stable version of this programming language is 2.7.0-beta-20171212 which was released on December 12, 2017. Typescript compiles to simple JavaScript code which runs on any browser that supports ECMAScript 3 framework. It offers support for the latest and evolving JavaScript features.

### Tell the minimum requirements for installing Typescript. Also, mention the steps involved in it.

Installing Typescript with the help of node and npm is recommended. Here, npm is used to install all the libraries and tools. Typescript should be installed globally usingnpm install –g typescript

It installs a command line code “tsc” which will further be used to compile your Typescript code. Make sure that you check the version of Typescript installed on your system.

Following steps are involved for installing Typescript on your system:

* + Download and run the .msi installer for node.
  + Enter the command “node –v” to check if the installation was successful.
  + Type the following command in the terminal window to install Typescript:npm install –g typescript

### List the built-in types in Typescript.

These are also called the primitive types in Typescript. These are of 5 types: –

* + Number type: it is used to represent number type values and represents double precision floating point values.
  + Syntax- var variable\_name: number;
  + String type: it represents a sequence of characters stored as Unicode UTF-16 code. It is the same as JavaScript primitive type.
  + Syntax- var variable\_name: string;
  + Boolean type: in Typescript, it is used to represent a logical value. When we use the Boolean type, we get output only in true or false. It is also the same as JavaScript primitive type.
  + Syntax- var variable\_name: bool;
  + Null type: it represents a null literal and it is not possible to directly reference the null type value itself.
  + Syntax- var variable\_name:number=null;
  + Undefined type: it is the type of undefined literal. This type of built-in type is the sub-type of all the types.
  + Syntax- var variable\_name:number=undefined;

### What are variables in Typescript? How to create a variable in Typescript?

A variable is a named space in the memory which stores a particular value. While declaring a variable in Typescript, certain rules should be followed like-

* + A variable name should contain alphabets and numeric digits.
  + It cannot contain spaces and special characters except underscore (\_) and dollar ($) sign.
  + A variable name cannot begin with a digit.

You should always keep in mind, that a variable should be declared before being used. Variables in Typescript are declared by placing a “var” keyword prior to the variable name. A variable can be declared using 4 methods: –

* + Declare its type and value in one statement.Syntax- var variable\_name:string = value;
  + Declare its type but no value.Syntax- var variable\_name:string;
  + Declare its value but no type.Syntax- var variable\_name = value;
  + Declare neither value nor type.Syntax- var variable\_name;

### What do you mean by interfaces? Explain them with reference to Typescript.

An interface is a syntactical contract which defines the syntax of any entity. Interfaces in Typescript define properties, methods, and events which are the members of the interface. It contains only the declaration of the members. It is responsible for deriving classes to define the members. It often helps in providing a standard structure that the deriving classes would follow. For declaring an interface, we make use of the “interface” keyword.

Syntax-

interface interface\_name{Statements;}

Interfaces need not to be converted to JavaScript for execution. They have zero runtime JavaScript impact. Interfaces can define both the kind of keys which an array uses and the type of entry it contains.

### How to compile a Typescript file?

A Typescript file is used to write more manageable and complex code in AppBuilder but before that, they need to be compiled to JavaScript. Here are the steps which are followed while compiling a Typescript file to JavaScript: –

* + It should be firstly verified that the Typescript engine has been enabled or not. In the title bar, click your username and select options.
  + In the project navigator, select and right-click the TS files that are to be compiled.
  + Choose compile to JavaScript.
  + Add a <script> reference for the JS file in the HTML code if needed.
  + To compile a typescript file via command line tsc <TypeScript File Name> command is used.

### What do you understand by classes in Typescript? List some features of classes.

We all know that Typescript is a type of object-oriented JavaScript and supports object-oriented programming features like- classes, interfaces, etc. Here, a class in terms of object-oriented programming is a blueprint for creating objects. A class is used to encapsulate data for the object. A built-in support is provided by Typescript for this feature. The “class” keyword is used to declare a class in Typescript.  
Example class Greeter { greeting: string; constructor (message: string) { this.greeting = message; } greet() { return "Hello, " + this.greeting; }}

Some features of a class are-

* + Encapsulation i.e. creation of self-contained modules that bind processing functions to the data, takes place.
  + Classes are created in hierarchies and with the help of inheritance; the features of one class can be used in the other.
  + Polymorphism takes place.
  + Abstraction

### Is Native Javascript supports modules?

Currently, Modules are not supported by Native JavaScript.To create and work with modules in Javascript you require an external like CommonJS.

### How to compile multiple Typescript files into a single file?

Using below command we can compile multiple typescript files into a single filetsc --outFile outputfile.ts file1.ts file2.ts file3.ts ... filen.ts

### How to Call Base Class Constructor from Child Class in TypeScript?

super() function is used to called parent or base class constructor from Child Class.

### What are Closures in Javascript?

Closures are nothing but a statefull functions.

A closure is an inner function that has access to outer function’s variables in addition to it’s own variable and global variables.In simple term a closure is a function inside a function.Closures using Lexical scoping in which function have access to global and outer variables.Below is a sample example of Javascript closure.

function employee() { var employee\_dept = 'IT'; return { getDept: function () { return employee\_dept; }, setDept: function (new\_dept) { employee\_dept = new\_name; } }​}var emp1 = employee (); // In this juncture, the employee\_dept outer function has returned.​mjID.getDept(); // ITmjID.setDept('Account'); // Changes the outer function's variable​mjID.getDept(); //outputs Account

Closures prevents your important variable to be modified accidently.It can be only modified by itself.

### List types of scopes available in Javascript?

There are two types of scopes are available in Javascript, they are

* + Global Scope
  + Local Scope

### What are Modules in Typescript?

Modules are powerful way to share code between files.By using modules effectively in your project you can keep you file size small and dependencies clear.

Modules are executed within their own scope, not in the global scope; this means that variables, functions, classes, etc. declared in a module are not visible outside the module unless they are explicitly exported using one of the export forms.

Creating a Module

module module\_name{class xyz{export foo(x, y){ return x\*y; }}}

### What is namespace in Typescript? How to declare a namespace in Typescript?

Internal Modules are known as namespaces in Typescript.Namespaces are used to maintain the legacy code of typescript interally. A namespace is simply a way to logically group related classes or interfaces in a wrapper.  
Synatax for creating namespace in Typescriptnamespace YourNameSpace { export class Class1 { } export class Class2 { } }

### Explain Decorators in Typescript? List type of Decorators available in Typescript?

Decorators are function that supports annotating or modifying classes its members.Its allow way to add both annotations and a meta-programming syntax for class declarations and members. Decorators are an experimental feature of Typescript and maybe change in future releases.

You can enable Decorators using the command line or editing your tsconfig.json

Enabling Decorators in TypeScript via command line

tsc --target ES5 --experimentalDecorators

### What are Mixins?

In Javascript Mixins are another way of building up classes from reusable components is to build them by combining simpler partial classes.

### What is default visibility for properties/methods in Typescript classes?

Public is default visibility for properties/methods in Typescript classes

JavaScript

### Explain what is Javascript? List some data types supported by Javascript?

#### Javascript

Javascript is an object-oriented computer programming language commonly used to create interactive effects within web browsers.It is first used by the Netscape browser, that provides access to the HTML document object model (DOM), provides access to the browser object model (BOM). Javascript syntax looks a lot like java, c or c++ syntax.

Below is the list of data types supported by Javascript:-

* + Undefined
  + Null
  + Boolean
  + String
  + Symbol
  + Number
  + Object

### What close() does in Javascript?

In Javascript close() method is used to close the current window. You must write window.close() to ensure that this command is associated with a window object and not some other JavaScript object.

### What is the difference between let and var?

Both var and let are used for variable/ method declaration in javascript but the main difference between let and var is that var is function scoped whereas let is block scoped.

### Explain Closures in JavaScript?

Closures are the combination of lexical environment and function within which the function was declared. This allows JavaScript programmers to write better, more creative, concise and expressive codes. The closure will consist of all the local variables that were in-scope when the closure was created.

Sure, closures appear to be complex and beyond the scope, but after you read this article, closures will be much more easy to understand and more simple for your everyday [JavaScript](https://www.onlineinterviewquestions.com/advanced-javascript-interview-questions/) programming tasks. JavaScript is a very function-oriented language it gives the user freedom to use functions as the wish of the programmer.

### Explain JavaScript Event Delegation Model?

In JavaScript, there is some cool stuff that makes it the best of all. One of them is Delegation Model. When capturing and bubbling, allow functions to implement one single handler to many elements at one particular time then that is called event delegation. Event delegation allows you to add event listeners to one parent instead of specified nodes. That particular listener analyzes bubbled events to find a match on the child elements. Many people think it to be complicated but in reality, it is very simple if one starts understanding it.

### Describe negative infinity in JavaScript?

NEGATIVE\_INFINITY property represents negative infinity and is a number in javascript, which is derived by ‘dividing negative number by zero’. It can be better understood as a number that is lower than any other number. Its properties are as follows:  
– A number of objects need not to be created to access this static property.  
– The value of negative infinity is the same as the negative value of the infinity property of the global object.

The values behave differently than the mathematical infinity:

* + Any positive value, including POSITIVE\_INFINITY, multiplied by NEGATIVE\_INFINITY is NEGATIVE\_INFINITY.
  + Any negative value, including NEGATIVE\_INFINITY, multiplied by NEGATIVE\_INFINITY is POSITIVE\_INFINITY.
  + Zero multiplied by NEGATIVE\_INFINITY is NaN.
  + NaN multiplied by NEGATIVE\_INFINITY is NaN.
  + NEGATIVE\_INFINITY, divided by any negative value except NEGATIVE\_INFINITY, is POSITIVE\_INFINITY.
  + NEGATIVE\_INFINITY, divided by any positive value except POSITIVE\_INFINITY, is NEGATIVE\_INFINITY.
  + NEGATIVE\_INFINITY, divided by either NEGATIVE\_INFINITY or POSITIVE\_INFINITY, is NaN.
  + Any number divided by NEGATIVE\_INFINITY is zero.

### Explain function hoisting in JavaScript?

JavaScript’s default behavior that allows moving declarations to the top is called Hoisting. The 2 ways of creating functions in JavaScript are Function Declaration and Function Expression. Let’s find out more about these:

#### *Function Declaration*

A function with the specific parameters is known as function declarations. To create a variable in JavaScript is called declarations.

e.g:

hoisted(); // logs "foo"function hoisted() { console.log('foo');}

Function Expression

When a function is created by using an expression it is called function expression.

e.g:

notHoisted(); // TypeError: notHoisted is not a functionvar notHoisted = function() { console.log('bar');};

### What is the use of let & const?

In modern javascript let & const are different ways of creating variables.Earlier in javascript, we use the var keyword for creating variables. let & const keyword is introduced in version ES6 with the vision of creating two different types of variables in javascript one is immutable and other is mutable.  
const: It is used to create an immutable variable. Immutable variables are variables whose value is never changed in the complete life cycle of the program.  
let: let is used to create a mutable variable. Mutable variables are normal variables like var that can be changed any number of time.

### Explain Arrow functions?

An arrow function is a consise and short way to write function expressions in Es6 or above.A rrow functions cannot be used as constructors and also does not supports this, arguments, super, or new.target keywords. It is best suited for non-method functions. In general an arrow function looks like const function\_name= ()=>{}

const greet=()=>{console.log('hello');} greet();

### What are exports and imports?

Imports and exports help us to write modular javascript code. Using Imports and exports we can split our code into multiple files. Imports allow taking only some specific variables or methods of a file. We can import methods or variables that are exported by a module. See the below example for more detail.

//index.js import name,age from './person'; console.log(name); console.log(age); //person.js let name ='Sharad', occupation='developer', age =26; export { name, age};

### What is difference between module.exports and export?

The module is a plain JavaScript object with an exports property. Exports is a plain JavaScript variable that happens to be set to module.exports. At the end of your file, node.js will basically ‘return’ module.exports to the require function. A simplified way to view a JS file in Node could be this:

var module = { exports: {} };var exports = module.exports;// your codereturn module.exports;

If you set a property on exports, like exports.a = 9;, that will set module.exports.a as well because objects are passed around as references in JavaScript, which means that if you set multiple variables to the same object, they are all the same object; so then exports and module.exports are the same objects.  
But if you set exports to something new, it will no longer be set to module.exports, so exports and module.exports are no longer the same objects.

### How to import all exports of a file as an object.

import \* as object name from ‘./file.js’ is used to import all exported members as an object. You can simply access the exported variables or methods using dot (.) operator of the object.

Example:

objectname.member1;objectname.member2;objectname.memberfunc();

### Explain “use strict” ?

“use strict” is a javascript directive that is introduced in Es5. The purpose of using “use strict” directive is to enforce the code is executed in strict mode. In strict mode we can’t use a variable without declaring it. “use strict” is ignored by earlier versions of Javascript.

### In Javascript are calculations with fractional numbers guaranteed to be precise?

NO, calculations with fractional numbers are not guaranteed to be precise in Javascript

### List the comparison operators supported by Javascript?

Javascript supports below comparison operators

* + > Greater than
  + < Less than
  + <= Less than or equal to
  + >= Greater than or equal to
  + == Equal to
  + != Not Equal to
  + === Equal to with datatype check
  + !== Not equal to with datatype check

### How do you declare variables in Javascript?

In Javascript variable are declared using the var keyword.A variable must begin with A letter, $ or \_.

eg. var myVar=”Online Interview Questions”;

PS: All variables in Javascript are Case sensitive.

### What will happen if an infinite while loop is run in Javascript?

The program will crash the browser.

### List HTML DOM mouse events?

HTML DOM mouse events

* + onclick
  + ondblclick
  + mousemove
  + mousedown
  + mouseover
  + mouseout
  + mouseup

### How to get the last index of a string in Javascript?

string.length-1 is used to get the last index of a string in Javascript

Example Usage:-

var myString="JavascriptQuestions";console.log(myString.length-1);

### How to get the primitive value of a string in Javascript?

In Javascript valueOf() method is used to get the primitive value of a string.

Example Usage:

var myVar= "Hi!"console.log(myVar.valueOf())

### What are the primitive data types in JavaScript?

A primitive is a basic data type that’s not built out of other data types. It can only represent one single value. All primitives are built-in data types by necessity, (the compiler has to know about them,) but not all built-in data types are primitives.

In JavaScript there are 5 primitive data types are available they are undefined, null, boolean, string and number are available.Everything else in Javascript is an object.

### Explain Event bubbling and Event Capturing in JavaScript?

Event Capture and Bubbling: In HTML DOM API there are two ways of event propagation and determines the order in which event will be received. The two ways are Event Bubbling and Event Capturing. The first method event bubbling directs the event to its intended target, and the second is called event capture in which the event goes down to the element.

### Event Capture

The capture procedure is rarely used but when it’s used it proves to be very helpful. This process is also called ‘trickling’. In this process, the event is captured first by the outermost element and then propagated to the innermost element. For example:

<div><ul><li></li></ul></div>

From the above example, suppose the click event did occur in the ‘li’ element, in that case capturing event it will be first handled ‘div’, then ‘ul’ and at last the target element will be hit that is ‘li’

### Event Bubbling

Bubbling just works like the bubbles, the event gets handled by the innermost element and then propagated to the outer element.

<div> <ul><li></li></ul></div>

From the above example, suppose the click event did occur in the ‘li’ element in bubbling model the event will be handled first by ‘li’ then by ‘ul’ and at last by ‘div’ element.

### What does the instanceof operator do?

In Javascript instanceof operator checks whether the object is an instance of a class or not:

Example Usage

Square.prototype = new Square();console.log(sq instanceof Square); // true

### What is Javascript BOM?

BOM stands for “Browser Object Modal” that allows Javascript to ‘talk’ to the browser, no standards, modern browsers implement similar BOMS – window, screen, location, history, navigator, timing, cookies.

### What are different types of Popup boxes available in Javascript?

In Javascript there are 3 types of Popup Boxes are available, they are

* + Alert
  + Confirm
  + Prompt

### How can you create an array in Javascript?

There are 3 different ways to create an array in Javascript. They are

* + By array literal  
    usage: var myArray=[value1,value2...valueN];
  + By creating instance of Array  
    usage:var myArray=new Array();
  + By using an Array constructor  
    usage:var myArray=new Array('value1','value2',...,'valueN');

### What is the ‘Strict’ mode in JavaScript and how can it be enabled?

Strict mode is a way to introduce better error-checking into your code. When you use strict mode, you cannot, for example, use implicitly declared variables, or assign a value to a read-only property, or add a property to an object that is not extensible.

You can enable strict mode by adding “use strict”; at the beginning of a file, a program, or a function. This kind of declaration is known as a directive prologue. The scope of a strict mode declaration depends on its context. If it is declared in a global context (outside the scope of a function), all the code in the program is in strict mode. If it is declared in a function, all the code in the function is in strict mode.

### How to calculate Fibonacci numbers in JavaScript?

#### Calculating Fibonacci series in JavaScript

Fibonacci numbers are a sequence of numbers where each value is the sum of the previous two, starting with 0 and 1. The first few values are 0, 1, 1, 2, 3, 5, 8, 13 ,…,

function fib(n) {var a=0, b=1;for (var i=0; i < n; i++) {var temp = a+b; a = b;b = temp;}return a;}

### What is the difference between the substr() and substring() functions in JavaScript?

#### Difference between the substr() and substring() functions in JavaScript.

The substr() function has the form substr(startIndex,length). It returns the substring from startIndex and returns ‘length’ number of characters.

var s = "hello";( s.substr(1,4) == "ello" ) // true

The substring() function has the form substring(startIndex,endIndex). It returns the substring from startIndex up to endIndex – 1.

var s = "hello";( s.substring(1,4) == "ell" ) // true

### What are different types of Inheritence? Which Inheritance is followed in Javascript.

There are two types of Inherientence in OOPS Classic and Prototypical Inheritance. Javascript follows Prototypical Inheritance.

### What is output of undefined \* 2 in Javascript?

nan is output of undefined \* 2.

### How to add/remove properties to object dynamically in Javascript?

You can add a property to an object using object.property\_name =value , delete object.property\_name is used to delete a property.

Example:

let user = new Object(); // adding a property user.name='Anil'; user.age =25; console.log(user); delete user.age; console.log(user);

### How to convert Javascript date to ISO standard?

toISOString() method is used to convert javascript date to ISO standard. It converts JavaScript Date object into a string, using the ISO standard.

Usage:

var date = new Date();var n = date.toISOString();console.log(n);// YYYY-MM-DDTHH:mm:ss.sssZ

### How to get inner Html of an element in JavaScript?

InnerHTML property of HTML DOM is used to get inner Html of an element in JavaScript.

Example Usage:

This is inner Element

<script type="text/javascript">var inner= document.getElementById("inner").innerHTML ;console.log(inner); // This is inner Elementdocument.getElementById("inner").innerHTML = "Html changed!";var inner= document.getElementById("inner").innerHTML ;console.log(inner); // Html changed!</script>

### How to clone an object in Javascript?

Object.assign() method is used for cloning an object in Javascript.Here is sample usage

var x = {myProp: "value"};var y = Object.assign({}, x);

### List different ways of empty an array in Javascript?

In Javascript, there are many ways to empty an array in Javascript, below we have listed 4 major

* + By assigning an empty array.
  + var arr1 =[1,4,5,6];arr1=[];
  + By assigning array length to 0.
  + var arr2 =[1,4,5,6];arr2.length=0;
  + By poping the elements of the array.
  + var arr2 =[1,4,5,6];while(arr.length > 0) { arr.pop();}
  + By using .splice() .
  + var arr =[1,4,5,6];arr.splice(0,arr.length)

### How to get an element by class in JavaScript ?

document.getElementsByClassName() method is used in Javascript to get an element with a class name.

|  |  |
| --- | --- |
| **getElementsByClassName()** |  |
| Method Name | getElementsByClassName |
| Syntax | document.getElementsByClassName('className') |
| Parameter | String (name of class) |
| Output | Array of HTMLCollection that have inputted className |

### Explain Typecasting in Javascript?

In Programming whenever we need to convert a variable from one data type to another Typecasting is used. In Javascript, we can do this via library functions. There are basically 3 typecasts are available in Javascript Programming, they are:

* + Boolean(value): Casts the inputted value to a Boolean
  + Number(value): Casts the inputted value to an Integer or Floating point Number.
  + String(value) : Casts the inputted value value a string

### How to encode and decode a URL in JavaScript?

encodeURI() function is used to encode an URL in Javascript.It takes a url string as parameter and return encoded string. Note: encodeURI() did not encode characters like / ? : @ & = + $ #, if you have to encode these characters too please use encodeURIComponent(). Usage:

var uri = "my profile.php?name=sammer&occupation=pāntiNG";var encoded\_uri = encodeURI(uri);

decodeURI() function is used to decode an URL in Javascript.It takes a encoded url string as parameter and return decoded string. Usage:

var uri = "my profile.php?name=sammer&occupation=pāntiNG";var encoded\_uri = encodeURI(uri);decodeURI(encoded\_uri);

### How to you change the title of the page by JavaScript?

You can change the title of a webpage using setting the title property of the document object.

Example usage

document.title="My New Title";

### What is difference between deep and shallow object coping in JavaScript?

### List some Unit Testing Frameworks JavaScript

Below is the list of few most Popular Javascript Unit Testing Frameworks:

* + Unit.js
  + Jasmine
  + Karma
  + Chai
  + AVA
  + Mocha
  + JSUnit
  + QUnit
  + Jest

### How to add a new property in existing function JavaScript?

### Explain JavaScript Accessors ?

JavaScript Accessors

### List few difference between primitive and non primitive JavaScript data types?

### Explain higher-order functions in JavaScript?

### Explain few difference between null, undefined or undeclared JavaScript variable?

### How host objects are different from native objects in JavaScript?

### What is difference between var x =1; and x=1;?

### Explain spread operator in JavaScript?

### How to call a function in every x seconds in JavaScript?

### Explain Promise in JavaScript?

### What is difference between Array.splice() and Array.slice() method in JavaScript?

### Is JavaScript multi-threaded or single-threaded?

JavaScript is single-threaded.

### Explain JavaScript Debounce Function?

### List some Design patterns in JavaScript?

### What is console.time() and console.timeEnd()? What is its syntax, and why is it used?

### What are different types of Scope Chain available in JavaScript?

### How to remove duplicate values from a JavaScript array?

We can use array.indexOf method to check a value exists or not. See below example to remove duplicate values.

let duplicates = ['delhi','kanpur','kanpur','goa','delhi','new york'];

function removeDuplicatesValues(arr){ let unique\_array = []; for(let i = 0;i < arr.length; i++){ if(unique\_array.indexOf(arr[i]) == -1){ unique\_array.push(arr[i]) } } return unique\_array}console.log(removeDuplicatesValues(duplicates));

### How to redirect a page to another page in Javascript?

### Is it possible to do 301 redirects in Javascript ?

JavaScript entirely runs on the client machine. 301 is response code that is sent by the server as a response. So it is not possible to do 301 Redirects In JavaScript.

### Write a program to reverse a string in pure JavaScript?

### Write program to remove duplicate in an array ?

### List few Difference between JAVA and JavaScript?

### Explain MUL function in Javascript?

### List few advantages of using JavaScript?

Few advantage og Javascript

* + Javascript is executed on user's computer, the meaning is that whatever you do in Javascript will not add any processing strain on the server. and that's why it is called as the client-side programming language. And this feature makes your sites responsive for the end user and less expensive for you in terms of server traffic.
  + With the help of Javascript, you can create highly responsive interfaces which will improve the user experience and provide dynamic functionality, without waiting for the server to show another page.
  + If you want to make online systems available offline and sync automatically once the computer goes online, then Javascript is the best technology you can use. you can do this using the right browser add-ons (Such as Google or Yahoo Browser Plus).
  + Content loading and changing it dynamically. Using Ajax in Javascript you can load content into the document if and when the user needs it, without reloading the entire page.
  + Using the Principles of unobtrusive JavaScript(defensive Scripting), JavaScript can test for what is possible in your browser and react accordingly.

Javascript is an object-oriented programming language that supports the creation of both client and server side applications. It is dynamic, weakly typed, prototype-based and multi-paradigm high-level programming language.