**Angular 2, 4 and 5 questions.**

**Note: "PA" stands for Possible Answer (one of many valid ones).**

**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.