Angular

Q1. What is the purpose of the ViewChild decorator in this component class?

```
@Component({
    ...
    template: '<p #bio></p>'
})
export class UserDetailsComponent {
    @ViewChild('bio') bio;
}
```

- [x] It provides access from within the component class to the ElementRef object for the `<p>` tag that has the bio template reference variable in the component's template view.
- [ ] It indicates that the `<p>` tag be rendered as a child of the parent view that uses this component.
- [ ] It makes the `<p>` tag in the template support content projection.
- [ ] It makes the `<p>` tag visible in the final render. If the #bio was used in the template and the @ViewChild was not used in the class, then Angular would automatically hide the `<p>` tag that has #bio on it.

DigitalOcean - viewchild-access-component

Q2. What method is used to wire up a FormControl to a native DOM input element in reactive forms?

- [ ] Add the string name given to the FormControl to an attribute named controls on the element to indicate what fields it should include.
- [ ] Use the square bracket binding syntax around the value attribute on the DOM element and set that equal to an instance of the FormControl.
- [x] Use the formControlName directive and set the value equal to the string name given to the FormControl.
- [ ] Use the string name given to the FormControl as the value for the DOM element id attribute.

Angular.io - Reactive Form Groups

Q3. What is the difference between the `paramMap` and the `queryParamMap` on the `ActivatedRoute` class?

- [ ] The paramMap is an object literal of the parameters in a route's URL path. The queryParamMap is an Observable of those same parameters.
- [ ] The paramMap is an Observable that contains the parameter values that are part of a route's URL path. The queryParamMap is a method that takes in an array of keys and is used to find specific parameters in the paramMap.
- [ ] paramMap is the legacy name from Angular 3. The new name is queryParamMap.
- [x] Both are Observables containing values from the requested route's URL string. The paramMap contains the parameter values that are in the URL path and the queryParamMap contains the URL query parameters.

StackOverflow

Q4. Based on the following usage of the async pipe, and assuming the users class field is an Observable, how many subscriptions to the users Observable are being made?

```
<h2>Names</h2>
<div *ngFor="let user of users | async">{{ user.name }}</div>
<h2>Ages</h2>
<div *ngFor="let user of users | async">{{ user.age }}</div>
<h2>Genders</h2>
<div *ngFor="let user of users | async">{{ user.gender }}</div>
```

- [ ] None. The async pipe does not subscribe automatically.
- [ ] None. The template syntax is not correct.
- [x] Three. There is one for each async pipe.

- [ ] One. The async pipe caches Observables by type internally.

UltimateCourses

Q5. How can you use the HttpClient to send a POST request to an endpoint from within an addOrder function in this OrderService?

```
export class OrderService {
  constructor(private httpClient: HttpClient) {}

  addOrder(order: Order) {
    // Missing line
  }
}
```

- [ ] this.httpClient.url(this.orderUrl).post(order);
- [ ] this.httpClient.send(this.orderUrl, order);
- [ ] this.httpClient.post(this.orderUrl, order);
- [x] this.httpClient.post(this.orderUrl, order).subscribe();

Angular.io - Sending data to server

Q6. What is the RouterModule.forRoot method used for?

- [ ] Registering any providers that you intend to use in routed components.
- [x] Registering route definitions at the root application level.
- [ ] Indicating that Angular should cheer on your routes to be successful.
- [ ] Declaring that you intend to use routing only at the root level.

O'REILLY

Q7. Which DOM elements will this component metadata selector match on?

```
@Component({
    selector: 'app-user-card',
    . . .
})
```

- [ ] Any element with the attribute app-user-card, such as `<div app-user-card></div>`.
- [ ] The first instance of `<app-user-card></app-user-card>`.
- [x] All instances of `<app-user-card></app-user-card>`.
- [ ] All instances of `<user-card></user-card>`.

Angular.io - Component Metadata

Q8. What is the correct template syntax for using the built-in ngFor structural directive to render out a list of productNames?

- [ ] A

```
<ul>
  <li [ngFor]="let productName of productNames">{{ productName }}</li>
</ul>
```

- [ ] B

```
<ul>
  <li ngFor="let productName of productNames">{{ productName }}</li>
</ul>
```

- [x] C

```
<ul>
  <li *ngFor="let productName of productNames">{{ productName }}</li>
</ul>
```

- [ ] D

```
<ul>
  <? for productName in productNames { ?>
  <li>{{ productName }}</li>
  <? } ?>
</ul>
```

Angular.io- Structural Directives

Q9. What are the two component decorator metadata properties used to set up CSS styles for a component?

- [ ] viewEncapsulation and viewEncapsulationFiles.
- [ ] There is only one and it is the property named css.
- [ ] css and cssUrl.
- [x] styles and styleUrls.

Angular.io - Component Styles

Q10. With the following component class, what template syntax would you use in the template to display the value of the title class field?

```
@Component({
  selector: 'app-title-card',
  template: '',
})
class TitleCardComponent {
  title = 'User Data';
}
```

- [ ] `{{ 'title' }}`
- [x] `{{ title }}`
- [ ] `[title]`
- [ ] A class field cannot be displayed in a template via the template syntax.

Angular.io - String Interpolation or Text Interpolation

Q11. What is the purpose of the valueChanges method on a FormControl?

- [ ] It is used to configure what values are allowed for the control.
- [ ] It is used to change the value of a control to a new value. You would call that method and pass in the new value for the form field. It even supports passing in an array of values that can be set over time.
- [ ] It returns a Boolean based on if the value of the control is different from the value with which it was initialized.
- [x] It is an observable that emits every time the value of the control changes, so you can react to new values and make logic decisions at that time.

Angular.io - Displaying a from control value

Q12. What directive is used to link an `<a>` tag to routing?

- [ ] routeTo
- [x] routerLink
- [ ] routePath
- [ ] appLink

Angular.io - RouterLink

Q13. What is the Output decorator used for in this component class?

```
@Component({
    selector: 'app-shopping-cart',
    . . .
})
export class ShoppingCartComponent {
    @Output() itemTotalChanged = new EventEmitter();
}
```

- [ ] It makes the `itemTotalChanged` class field public.
- [ ] It provides a way to bind values to the `itemTotalChanged` class field, like so: `<app-shopping-cart [itemTotalChanged]="newTotal"></app-shopping-cart>`.
- [x] It provides a way to bind events to the `itemTotalChanged` class field, like so: `<app-shopping-cart (itemTotalChanged)="logNewTotal($event)"></app-shopping-cart>`.
- [ ] It is simply a way to put a comment in front of a class field for documentation.

Angular.io - Sending data to parent component

Q14. What is the difference between these two markup examples for conditionally handling display?

```
<div *ngIf="isVisible">Active</div>
<div [hidden]="!isVisible">Active</div>
```

- [ ] The `ngIf` is shorthand for the other example. When Angular processes that directive, it writes a div element to the DOM with the hidden property.
- [ ] They are fundamentally the same.
- [x] The `ngIf` directive does not render the div in the DOM if the expression is false. The `hidden` property usage hides the div content in the browser viewport, but the div is still in the DOM.
- [ ] The `ngIf` is valid, but the use of the `hidden` property is wrong and will throw an error.

StackOverflow

Q15. How can you disable the submit button when the form has errors in this template-driven forms example?

```
<form #userForm="ngForm">
  <input type="text" ngModel name="firstName" required />
```

```
  <input type="text" ngModel name="lastName" required />
  <button (click)="submit(userForm.value)">Save</button>
</form>
```

- [ ] A

```
<button (click)="submit(userForm.value)" disable="userForm.invalid">Save</butt
```

- [x] B

```
<button (click)="submit(userForm.value)" [disabled]="userForm.invalid">Save</b
```

- [ ] C

```
<button (click)="submit(userForm.value)" [ngForm.disabled]="userForm.valid">Sa
```

- [ ] D

```
<button (click)="submit(userForm.value)" *ngIf="userForm.valid">Save</button>
```

Angular.io - Submit the form with ngSubmit

Q16. You want to see what files would be generated by creating a new contact-card component. Which command would you use?

- [x] ng generate component contact-card --dry-run
- [ ] ng generate component contact-card --no-files
- [ ] ng generate component component --dry
- [ ] ng generate component --exclude

Angular.io - ng generate options

Q17. Based on the following component, what template syntax would you use to bind the TitleCardComponent's titleText field to the h1 element title property?

```
@Component({
  selector: 'app-title-card',
```

```
  template: '<h1 title="User Data"> {{titleText}}</h1>',
})
export class TitleCardComponent {
  titleText = 'User Data';
}
```

- [ ] `<h1 data-title="titleText">{{ titleText }}</h1>`
- [ ] `<h1 title="titleText">{{ titleText }}</h1>`
- [x] `<h1 [title]="titleText">{{ titleText }}</h1>`
- [ ] `<h1 titleText>{{ titleText }}</h1>`

Angular.io - String Interpolation

Q18. What are Angular lifecycle hooks?

- [ ] loggers for tracking the health of an Angular app
- [ ] providers that can be used to track the instances of components
- [ ] built-in pipes that can be used in templates for DOM events
- [x] reserved named methods for components and directives that Angular will call during set times in its execution, and can be used to tap into those lifecycle moments

Angular.io - Lifecycle hooks

Q19. Pick the best description for this template syntax code:

```
<span>Boss: {{job?.bossName}} </span>
```

- [ ] The ? is shorthand for the async pipe. The job value must be an Observable.
- [x] It is using the safe navigation operator (?) on the job field. If the job field is undefined, the access to the bossName will be ignored and no error will occur.
- [ ] There is an error in the template syntax. The ? is not valid here.
- [ ] It is diplaying the job value if it has one; otherwise it is displaying the bossName.

StackOverflow

Q20. How would you configure a route definition for a UserDetailComponent that supports the URL path user/23 (where 23 represents the id of the requested user)?

- [x] `{ path: 'user/:id', component: UserDetailComponent }`
- [ ] `{ url: 'user/:id', routedComponent: UserDetailComponent }`
- [ ] `{ routedPath: 'user/:id', component: UserDetailComponent }`

- [ ] `{ destination: new UserDetailComponent(), route: 'user/:id' }`

CodeCraft - Parameterised Routes

Q21. What are the HostListener decorators and the HostBinding decorator doing in this directive?

```
@Directive({
  selector: '[appCallout]',
})
export class CalloutDirective {
  @HostBinding('style.font-weight') fontWeight = 'normal';

  @HostListener('mouseenter')
  onMouseEnter() {
    this.fontWeight = 'bold';
  }

  @HostListener('mouseleave')
  onMouseLeave() {
    this.fontWeight = 'normal';
  }
}
```

- [x] They are setting the CalloutDirective.fontWeight field based on whether or not the mouse is over the DOM element. The HostListener then sets the font-weight CSS property to the fontWeight value.
- [ ] They are setting up the directive to check the DOM element that it is on. If it has event bindings added for mouse enter and leave it will use this code. Otherwise nothing will happen.
- [ ] This is an incorrect use of HostListener and HostBinding. The HostListener and HostBinding decorators do not do anything on directives; they work only when used on components.
- [ ] If the DOM element that this directive is placed on has the CSS property font-weight set on it, the mouseenter and mouseleave events will get raised.

DigitalOcean

Q22. What Angular template syntax can you use on this template-driven form field to access the field value and check for validation within the template markup?

```
<input type="text" ngModel name="firstName" required minlength="4" />
<span *ngIf="">Invalid field data</span>
```

- [x] You can make use of a template reference variable and the exportAs feature that the ngModel directive has.
- [ ] You can use the ngModel directive in combination with the input field name.
- [ ] You can use a template reference variable for the HTML input element and then check the valid property off of that.
- [ ] It is not possible to get access to the field value with template-driven forms. You must use reactive forms for that.

1. Angular.io -Show and hide validation error
2. Medium

Q23. What is the value type that will be stored in the headerText template reference variable in this markup?

```
<h1 #headerText>User List</h1>
```

- [x] an Angular ElementRef, a wrapper around a native element
- [ ] the inner text of the `<h1>` element
- [ ] a header component class
- [ ] the native DOM element type of HTMLHeadingElement

Pluralsight - Template reference variable

Q24. What is the difference, if any, of the resulting code logic based on these two provider configurations?

```
[{ provide: FormattedLogger, useClass: Logger }][{ provide: FormattedLogger, u
```

- [ ] They are the same. Both will result in a new instance of Logger that is bound to the FormattedLogger token.
- [x] The useClass syntax tells the injector to make a new instance of Logger and bind that instance to the FormattedLogger token. The useExisting syntax refers to an already existing object instance declared as Logger.
- [ ] Both of them are wrong. A strong type connot be used for useClass or useExisting.
- [ ] They are the same. Both will result in the FormattedLogger token being an alias for the instance of Logger.

1. Angular.io - Dependency Providers
2. TektutorialHub

Q25. What is the purpose of the data property (seen in the example below) in a route configuration?

```
{
    path: 'customers',
    component: CustomerListComponent,
    data: { accountSection: true }
}
```

- [ ] a key/value mapping for setting @Input values on the routed component instance
- [x] a way to include static, read-only data associated with the route that can be retrieved from the ActivatedRoute
- [ ] a property on the route that can be used to load dynamic data for the route
- [ ] an object that will get auto-injected into the routed component's constructor.

1. TektutorialsHub
2. StackOverflow

Q26. How does the built-in `ngIf` structural directive change the rendered DOM based on this template syntax?

```
@Component({
  selector: 'app-product',
  template: '<div *ngIf="product">{{ product.name }}</div>',
})
export class ProductComponent {
  @Input() product;
}
```

- [ ] The `<div>` acts as a placeholder. If the product class field is "truthy," the `<div>` will get replaced by just the `product.name` value; if not, then nothing will get rendered.
- [ ] The `<div>` will always be rendered, and if the product field is "truthy," the `<div>` element will contain the `product.name` value; otherwise it will render the `<div>` element with no value in it.
- [ ] It produces an error, since ngIf is not a built-in structural directive.
- [x] If the product class field is "truthy," then the rendered DOM will include the `<div>` with the value of the `product.name` field. If it is not "truthy,' the rendered DOM will not contain the `<div>` element.

Reference (angular.io)

Q27. What does this code accomplish?

```
@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  bootstrap: [AppComponent],
})
export class AppModule {}

platformBrowserDynamic().bootstrapModule(AppModule);
```

- [ ] It executes a unit test for an NgModule.
- [ ] It provides a way to code the document structure of an Angular application. The @NgModule is a form of inline code commenting that gets ignored by the TypeScript compiler but will show up with special formatting in code editor applications.
- [ ] It declares an Angular module named AppModule and makes it available for lazy loading throughout the application.
- [x] It declares an Angular module named AppModule that contains a bootstrapped component named AppComponent. Then it registers that module with Angular, so the app can start up.

Angular.io - The basic NgModule

Q28. Which choice best describes what the *resolve* property does in this route configuration?

```
{
   path: ':id',
   component: UserComponent,
   resolve: {
     user: UserResolverService
   }
}
```

- [x] Prior to loading the *UserComponent*, the router will subscribe to the *Observable* returned by a *resolve* method in the *UserResolverService*. This technique can be used to get preloaded data for a *route*.
- [ ] After the *route* is done resolving, and the component is loaded and rendered, the *UserResolverService* will have a method named *user* run that will clean up any open data connections.
- [ ] There is an error. The correct property name is *onResolve*.
- [ ] The *UserComponent* will have a parameter in its constructor for *user*, and the *router* will handle injecting in a value for that from a call to a *user* method in the

UserResolverService.

angular.io

Q29. What is the purpose of the ContentChildren decorator in this component class?

```
@Component({
 . . .
 template: '<ng-content></ng-content>'
})
export class TabsListComponent {
 @ContentChildren(TabComponent) tabs;
}
```

- [ ] If any *TabsComponent* elements are added to the *TabsListComponent* template, they will get put into the element at runtime.
- [ ] It creates *TabComponent* components in the *TabsListComponent* template when a *TabsListComponent* is instantiated.
- [x] It provides access from within the component class to any *TabComponent* components that were content projected into the for this component.
- [ ] It restricts the allowed elements that can be put into a *TabsListComponent* element to allow only *TabComponent* elements.

betterprogramming.pub

Q30. In order for Angular to process components in an application, where do the component types need to be registered?

- [ ] within a script tag in the index.html file
- [ ] in an NgModule decorator metadata tag named *components*
- [ ] No registration is needed simply include the component files in an app directory.
- [x] in an NgModule decorator metadata property named *declarations*

angular.io

Q31. What is the purpose of the `fixture.detectChanges()` call in this unit test?

```
TestBed.configureTestingModule({
  declarations: [UserCardComponent],
});
let fixture = TestBed.createComponent(UserCardComponent);

fixture.detectChanges();
```

```
expect(fixture.nativeElement.querySelector('h1').textContent).toContain(
  fixture.componentInstance.title,
);
```

- [ ] It tracks any potential UI changes and will fail the unit test if any are made.
- [ ] It is used to ensure component template stability across multiple unit tests in the entire test suite.
- [x] It forces Angular to perform change detection, which will render the *UserCardComponent* before you can validate its template.
- [ ] It is used to log change-detection events to the console during unit test runs.

angular.io

Q32. What will the URL segment look like based on the following call to the `Router.navigate` method when goToUser is passed the value 15?

```
export class ToolsComponent {
  constructor(private router: Router) {}
  goToUser(id: number) {
    this.router.navigate(['user', id]);
  }
}
```

- [x] /user/15
- [ ] /user?id=15
- [ ] /user:15
- [ ] /user;id=15

angular.io

Q33. When a service is provided for root and is also added to the provider's configuration for a lazy-loaded module, what instance of that service does the injector provide to constructors in the lazy-loaded module?

- [x] A new instance of that service is created when the module is lazy loaded.
- [ ] Providing a service of the same type at a lazy-loaded module level is not allowed.
- [ ] If an instance of the service has not been created at the root level yet. it will create one there and then use it.
- [ ] A single instance of that service is always instantiated at root and is the only one ever used, including within lazy modules.

Q34. What is the HostBinding decorator doing in this directive?

```
@Directive({
  selector: ' [appHighlight] ',
})
export class HighlightDirective {
  @HostBinding('class.highlighted') highlight = true;
}
```

- [x] It is adding the CSS class named highlighted to any DOM element that has the appHighlight directive on it.
- [ ] HostBinding does not do anything on directives, only on components.
- [ ] It is specifying if the host element gets the highlighted class added to its class attribute, then the directive class field highlight will get set to true; and if it is not added on the host it will get set to false.
- [ ] It is creating an inline style on the host element with a CSS property named highlight set to true.

StackOverflow

Q35. In reactive forms, what Angular form class type is used on the native DOM element to wire it up?

- [ ] `FormArray`
- [ ] `FormControl`
- [x] `FormGroup`
- [ ] `all of these answers`

Q36. Assuming the username FormControl has been configured with a minLength validator, how can you set up an error display in the following reactive forms markup for the username field?

```
<form [formGroup]="form">
  <input type="text" formControlName= "username">
  ...
</form>
```

- [ ] A

```
    <span *ngIf="username.minLength.invalid">
        Username length is not valid
```

```
    </span>
```

- [ ] B

```
<input type="text" formControlName="username" [showMinLength]="true">
```

- [ ] C

```
    <span *ngIf="form.get('username').getError('minLength') as minLengthError"
      Username must be at least {{ minLengthError.requiredLength }} characters
    </span>
```

- [x] D

```
<input type="text" formControlName="username" #userName="ngModer">
    <span *ngIf="userName.errors.minlength">
      Username must be at least {{ userName.errors.minlength.requiredLength }}
    </span>
```

Codecraft

Q37. How does the emulated view encapsulation mode handle CSS for a component?

- [ ] It renders the CSS exactly how you wrote it without any changes.
- [ ] It makes use of shadow DOM markup and CSS.
- [x] It creates unique attributes for DOM elements and scopes the CSS selectors you write to those attribute ids.
- [ ] It renders all of the CSS rules you write as inline CSS on all of the DOM elements you use them on in the template.

Angular.io

Q38. With the following TestBed setup, what can be used to access the rendered DOM for the UserCardComponent?

```
TestBed.configureTestingModule({
  declarations: [UserCardComponent],
});
let fixture = TestBed.createComponent(UserCardComponent);
```

- [ ] `fixture.componentTemplate`
- [ ] `fixture.getComponentHtml()`
- [x] `fixture.nativeElement`
- [ ] `fixture.componentInstance.template`

1. StackOverflow
2. Angular.io

Q39. Given these two components, what will get rendered to the DOM based on the markup usage?

```
@Component({
 selector: 'app-card',
 template: '<h1>Data Card</h1><ng-content></ng-content>'
})
export class CardComponent { }

@Component({
 selector: 'app-bio',
 template: '<ng-content></ng-content>.
})
export class BioComponent { }

// markup usage:
<app-card><app-bio>Been around for four years.</app-bio></app-card>
```

- [x] A

```
 <app-card>
  <h1>Data Card</hl>
  <app-bio>
   Been around for four years.
  </app-bio>
 </app-card>
```

- [ ] B

```
<h1>Data Card</h1>
 <app-bio>
  Been around for four years.
 </app-bio>
```

- [ ] C

```
<app-card>
  <h1>Data Card</hl>
  <ng-content></ng-content>
  <app-bio>
   Been around for four years.
   <ng-content></ng-content>
  </app-bio>
</app-card>
```

- [ ] D

```
<app-card>
  <h1>Data Card</hl>
</app-card>
```

Q40. Given the app-title-card component in the code below, what DOM will the app-user-card component render?

```
@Component({
   selector: 'app-user-card',
   template: '<app-title-card></app-title-card><p>Jenny Smith</p>'
})

@Component({
   selector: 'app-title-card',
   template: '<h1>User Data</hl>'
})

// usage of user card component in parent component html
<app-user-card></app-user-card>
```

- [x] A

```
<app-user-card>
  <app-title-card>
    <h1>User Data</h1>
  </app-title-card>
  <p>Jenny Smith</p>
</app-user-card>
```

- [ ] B

```
<h1>User Data</h1>
<p>Jenny Smith<p>
```

- [ ] C

```
<app-user-card>
  <app-title-card></app-title-card>
</app-user-card>
```

- [ ] D

```
<div app-user-card>
  <h1 app-title-card>User Data</h1>
  <p>Jenny Smith</p>
</div>
```

Q41. Pick the matching code for the custom provider registration that the @Inject () decorator is looking for:

```
constructor(@Inject('Logger') private logger) { }
```

- [ ] A

```
providers: [Logger];
```

- [x] B

```
providers: [{ provide: 'Logger', useClass: Logger }];
```

- [ ] C

```
@Injectable({
    providedIn: 'root'
})
```

- [ ] D

```
providers: [{ provide: 'Logger' }];
```

1. StackOverflow
2. TektutorialHub
3. Angular.io - Dependency Injection In Action

Q42. Which choice best describes the following usage of the HttpClient.get method in the getsettings class method?

```
export class SettingsService {
    constructor(private httpClient: HttpClient) { }
    ...

getSettings()
{
    return this.httpClient.get<Settings>(this.settingsUrl)
        .pipe(
            retry(3)
        );
}}
```

- ☐ The RxJs pipe method is an alias for the subscribe method, so a call to `getSettings` will execute the get query. The retry operator is used to tell the pipe call to retry the get query three times.
- ☐ It will produce an error at runtime because the pipe method is not available off of the `Httpclient.get` call.
- ☐ Every single call to the getSettings method will result in the Httpclient making three total get requests to the settingsUrl, which is not ideal because there will always be two extra calls that are not needed. The retry operator should not be used in this manner.
- ☑ When the result of the getSettings method is subscribed to, the HTTP GET call will be made; if it fails, it will be retried up to three times before it gives up and returns an error.

1. learnrxjs.io
2. dev.to

Q43. When a service requires some setup to initialize its default state through a method, how can you make sure that said method is invoked before the service gets injected anywhere?

- [ ] Put the logic of that service method into the service constructor instead.
- [x] Use a factory provider at the root AppModule level that depends on the service to call that service method.
- [ ] it is not possible to do it at application start; you can do it only at a component level.
- [ ] Instantiate an instance of the service at the global level (window scope) and then call that method.

1. Angular.io
2. Stackoverflow

Q44. What statement best describes this usage of the TestBed?

```
const spy = jasmine.createSpyObj('DataService', ['getUsersFromApi']);
TestBed.configureTestingModule({
  providers: [UserService, { provide: DataService, useValue: spy }],
});
const userService = TestBed.get(UserService);
```

- [ ] The TestBed is required anytime you want to make use of a spy object in a unit test for an Angular provider.
- [ ] The TestBed is being used to test a component's view.
- [x] The TestBed scaffolds an NgModule with two providers and handles any dependency injection. If any Angular class requests the DataService in its constructor, the TestBed will inject spy in that constructor.
- [ ] The TestBed is configuring the test runner to tell it to only execute tests for the two providers listed in its providers array.
  - `All other tests be ignored, including tests that assert results against one of these providers and a non-defined provider. Although it will work when multiple providers in this configuration are asserted against in a single test.`

Q45. What is the primary difference between a component and a directive?

- [ ] A component uses a selector metadata property and a directive does not.
- [ ] A directive can be used for adding custom events to the DOM and a component cannot.
- [x] A component has a template and a directive does not.
- [ ] A directive can target only native DOM elements.

StackOverflow

Q46. What could you add to this directive class to allow the truncate length to be set during directive usage in markup?

```
@Directive({
    selector: '[appTruncate]'
})
export class TruncateDirective {
    . . .
}

// example of desired usage:
<p [appTruncate]="10">Some very long text here</p>
```

- [x] `@Input() appTruncate: number;`
- [ ] `@Output() appTruncate;`
- [ ] `constructor(maxLength: number) { }`
- [ ] `Nothing. The directive selector cannot be used to pass in values to the directive.`

1. Angular.io
2. StackOverflow

Q47. How can you pass query parameters to this `HttpClient.get` request?

```
export class OrderService {
  constructor(private httpClient: HttpClient) {}

  getOrdersByYear(year: number): Observable<Order[]> {
    return this.httpClient.get<Order[]>(this.ordersUrl);
  }
}
```

- [ ] A `return this.httpClient.get<Order[]>(this.ordersUrl, {'year': year})`
- [ ] B `return this.httpClient.get<Order[]>(this.ordersUrl, year)`
- [x] C

```
const options = { params: new HttpParams().set('year', year) };
return this.httpClient.get<Order[]>(this.ordersUrl, options);
```

- [ ] D

```
getOrdersByYear(year: number): Observable<Order[]> {
    return this.httpClient.addParam('year', year).get<Order[]>(this.ordersUrl,
}
```

1. StackOverflow
2. TektutorialHub

Q48. Assuming the `DataService` has been registered in the providers for the application, which answer best describes what happens based on this component's constructor?

```
@Component({
    ...
})
export class OrderHistoryComponent {
    constructor(private dataService: DataService) {}
    ...
}
```

- [ ] It is declaring that the `OrderHistoryComponent` will have its own version of a `DataService` and that it should never use any existing instances. The `DataService` would need to be instantiated within the class as a private field for this code to be complete and working.
- [x] When Angular creates a new instance of the `OrderHistoryComponent`, the injector will provide an instance of a `DataService` class to the component constructor's first argument. The constructor's `dataService` parameter will be used to set a private instance field with the same name on the instance.
- [ ] It provides a way to do component testing only; the constructor has no usage in the actual run of the Angular application.
- [ ] It enables the custom element that the component targets to have a custom property named `dataService` that can be used to bind an existing `DataService` instance to.

1. StackOverflow
2. Angular.io - Dependency Injection

Q49. Finish this markup using the `ngIf` directive to implement an else case that will display the text "User is not active":

```
<div *ngIf="userIsActive; else inactive">
  Currently active!
```

```
</div>
```

- [ ] A

```
<div #inactive>User is not active.</div>
```

- [ ] B

```
<div *ngIf="inactive">
  User is not active.
</div>
```

- [ ] C

```
<ng-template #else="inactive">
  <div>User is not active.</div>
</ng-template>
```

- [x] D

```
<ng-template #inactive>
  <div>User is not active.</div>
</ng-template>
```

Angular.io

Q50. What is the correct syntax for a route definition to lazy load a feature module?

- [ ] A

```
{
    path: 'users',
    lazy: './users/users.module#UsersModule'
}
```

- [x] B

```
{
    path: 'users',
```

```
    loadChildren: () => import('./users/users.module').then(m => m.UserModule)
}
```

- C

```
{
    path: 'users',
    loadChildren: './users/users.module#UsersModule'
}
```

- D

```
{
    path: 'users',
    module: UsersModule
}
```

Angular.io - Lazy Loading Modules

Q51. Describe how the validation is set up and configured in this reactive forms example:

```
export class UserFormControl implements OnInit {
    ...
    ngOnInit() {
        this.form = this.formBuilder.group({
            username: this.formBuilder.control('',
                [Validators.required, Validators.minLength(5), this.unique]),
        )};
    }
    unique(control: FormControl) {
        return control.value !== 'admin' ? null: {notUnique: true};
    }
}
```

- The `FormControl` for `username` is getting configured to exclude three validators from the validators that it is allowed to use.
- The `FormControl` for `username` is getting configured to allow three possible validators to be used: `required`, `maxLength`, and a custom one named `unique`. To enable these `validators`, a validator directive would need to be put on the form fields in the markup.

- [ ] Validation cannot be set up this way in reactive forms.
- [x] The `FormControl` for `username` is getting configured with three validators: the `required` and `minLength` validators that come from Angular, and a custom validator function named `unique` that checks for the value not equal to the string `admin`.

1. Angular.io - Form Validation
2. Angular University Blog

Q52. What does the Injectable decorator do on this service class?

```
@Injectable({
    providedIn: 'root'
)}
export class DataService { }
```

- [ ] It registers a provider for the service that is available only at the root module level, not to any children modules.
- [x] It registers a provider for the service in the root application injector, making a single instance of it available throughout the application.
- [ ] It makes it so the service can be injected only in the bootstrapped component for the application.
- [ ] It sets up a compile time rule that allows you to put the service type only in the providers metadata property of the root NgModule.

Angular.io

Q53. Describe the usage of this code

```
export interface AppSettings {
  title: string;
  version: number;
}
export const APP_SETTINGS = new InjectionToken<AppSettings>('app.settings');
```

- [ ] The InjectionToken is adding an instance of the AppSettings to the root provider via the InjectionToken constructor call, making it automatically available to all NgModules, services and components throughout the Angular application without the need to inject it anywhere.
- [x] The InjectionToken is used to create a provider token for a non-class dependency. An Object literal can be provider as a value for the APP_SETTINGS dependency provider type that can then be injected into components, services, etc ..

- [ ] The InjectionToken is used to create a dynamic decorator for the AppSettings that can be used on constructor parameters via an @AppSettings decorator.
- [ ] This code has an error since you cannot use a TypeScript interface for the generic type on the InjectionToken

Q54. For the following template-driven forms example, what argument can be passed to the submit method in the click event to submit the data for the form?

```
<form #form="ngForm">
        <input type="text" ngModel="firstName">
        <input type="text" ngModel="lastName">
        <button (click)="submit()">Save</button>
</form>
```

- [x] submit(form.value)
- [ ] submit($event)
- [ ] submit(ngForm.value)
- [ ] submit(FirstName, lastName)

Q55. What is the purpose of the `preloadingStrategy` property configuration in this router code?

```
RouterModule.forRoot(
  ...{
    preloadingStrategy: PreloadAllModules,
  },
);
```

- [ ] It enables the option to flag individual routes for preloading.
- [ ] It preloads all dependencies for routes, creating instances of services when the app first starts up
- [ ] It ensures all modules get built into a single app module bundle file.
- [x] It configures the router to immediately load all routes that have a loadChildren property(routes that are typically loaded when requested)

References:

- Angular Router, PreloadAllModules
- Route preloading in Angular
- Preloading strategy

- Custom preloading strategy
- Preloading strategy, save loading time

Q56. What is an alternative way to write this markup to bind the value of the class field `userName` to the `h1` element title property?

```
<h1 [title]="userName">Current user is {{ userName }}</h1>
```

- [ ] title="userName"
- [x] title="{{ userName }}"
- [ ] title="{{ 'userName' }}"
- [ ] The only way to do it is by using the square brackets.

Q57. What is the `async` pipe doing in this example?

```
@Component({
  selector: 'app-users',
  template: '<div *ngFor="let user of users | async">{{ user.name }}</div>',
})
export class UsersComponent implements OnInit {
  users;
  constructor(private httpClient: HttpClient) {}
  ngOnInit(): void {
    this.users = this.httpClient.get<{ name: string }>('users');
  }
}
```

- [ ] It is doing nothing, since the async pipe cannot be used in an `ngFor` statement.
- [ ] It is configuring the `ngFor` iteration to support multiple lists of users at the same time.
- [x] It is subscribing to the observable returned from the `HttpClient.get` method and unwrapping the returned value so it can be iterated over in the `ngFor`.
- [ ] It is allowing all of the users in the `users` field to be rendered concurrently to the DOM.

Q58. How would you make use of this directive in markup based on its selector value

```
@Directive({
        selector: '[appTruncate]'
})
export class TruncateDirective{
```

```
        . . .
}
```

- [ ] `html <p data-directive="appTruncate">Some long text </p>`
- [x] `html <p appTruncate>Some long text</p>`
- [ ] `html <p app-truncate>Some long text</p>`
- [ ] `html <app-truncate>Some long text</app-truncate>`

Q59. What lifecycle hook can be used on a component to monitor all changes to @Input values on that component?

- [ ] ngOnInit
- [ ] ngChanges
- [ ] ngAfterInputChange
- [x] ngOnChanges

How to detect when an @Input() value changes in Angular?

Q60. What would be an example template syntax usage of this custom pipe?

```
@Pipe({ name: 'truncate' })
export class TruncatePipe implements PipeTransform {
  transform(value: string, maxLength: number, showEllipsis: boolean) {
    const newValue = maxLength ? value.substr(0, maxLength) : value;
    return showEllipsis ? '${newValue}...' : newValue;
  }
}
```

- [ ] `{{ 'some long text' | truncate:10 }}`
- [x] `'some long text' | truncate:10:true`
- [ ] `{{ 'some long text' | truncate }}`
- [ ] all of these answers

[How do I call an Angular 2 pipe with multiple arguments?]
(https://stackoverflow.com/questions/36816788/how-do-i-call-an-angular-2-pipe-with-multiple-arguments)

Q61. Which Angular CLI command would you run to generate a UsersComponent and add it to the SharedModule (in file shared.module.ts in your application)?

- [ ] ng generate component --newModule=shared
- [x] ng generate component users --module=shared

- [ ] ng generate component users --shared
- [ ] ng generate component --add=shared

Q62. How can you rewrite this markup so the div container is not needed in the final DOM render

```
<div *ngIf="location">
        <h1>{{ location.name }}</h1>
        <p>{{ location.description }}</p>
</div>
```

- [ ] A

```
<div *ngIf="location">
        <h1>{{ location.name }}</h1>
        <p>{{ location.description }}</p>
{{ endNgIf }}
```

- [ ] B

```
<ng-template *ngIf="location">
        <h1>{{ location.name }}</h1>
        <p>{{ location.description }}</p>
</ng-template>
```

- [ ] C

```
<div *ngIf="location" [display]=" ' hidden' ">
        <h1>{{ location.name }}</h1>
        <p>{{ location.description }}</p>
</div>
```

- [x] D

```
<ng-container *ngIf="location">
        <h1>{{ location.name }}</h1>
        <p>{{ location.description }}</p>
</ng-container>
```

Q63. Describe the usage of this code:

```
export interface AppSettings {
  title: string;
  version: number;
}
```

- [ ] This code as an error since you cannot use a TypeScript interface for the generic type on the InjectionToken.
- [ ] The InjectionToken is used to create a provider token for a non-class dependency. An object lieteral can be provided as a value for the APP_SETTINGS dependency provider type that can then be injected into components, services, etc.
- [ ] The InjectionToken is adding an instance of the AppSettings to the roote provider via the the InjectionToken constructor call, making it auto available to all NgModules, services, and componentts throughtout the Angular application without the need to inject it anywhere.
- [x] The InjectionToken is used to create a dynamic decorator for the AppSettings that can be sed on constructor parameters via an @AppSettings decorator.

Q64. What Angular utilities, if any, are required to unit test a service with no constructor dependencies?

- [ ] By.css() helper method is needed
- [ ] A text fixture is required to run the service for the unit test.
- [ ] None. A service can be instantiated and unit tested on its own.
- [x] The TestBed class iss needed to instantiate the service.

Angular unit tests - recheck answers

Q65. What is the difference between the CanActivate and the CanLoad route guards?

- [ ] CanActivate is used to check access. CanLoad is used to preload data for the route.
- [x] CanLoad is used at app startup to allow or deny routes to be added to the route table. CanActivate is used to manage access to routes at the time they are requested.
- [ ] CanActivate and CanLoad do the exact same thing.
- [ ] CanLoad prevents an entire NgModule from being delivered and loaded. CanActivate stops routing to a component in that NgModule, but that module is still loaded.

CanActivate vs Canload CanActivate prevents access on routes, CanLoad prevents lazy loading.

Q66. What is the outlet property used for in this router definition object?

```
{
        path: 'document',
        component: DocumentComponent,
        outlet: 'document-box'
}
```

- [ ] it will locate all instancess of in the DOM and inser a DocumentComponent element into them on route navigation.
- [ ] It declares that the DocumentComponent can be used as a child to a < element in addition ot being routed to.
- [x] It us used to target a element with the name attribute matching the string value as the location for the DocumentComponent to be rendered when routed to.
- [ ] It is a source of power for the router. (definitely not the answer :P)

Angular-outlet - recheck answer

Q67. In this template syntax, every time the items property is changed (added to, removed from, etc.), the ngFor structural directive re-runs its logic for all DOM elements in the loop. What syntax can be used to make this more performant?

```
<div *ngFor="let item of items">
  {{ item.id }} - {{ item.name }}
</div>
```

- [ ] `*ngFor="let item of items; let uniqueItem"`
- [ ] `*ngFor="let item of items.distinct()"`
- [ ] `*ngFor="let item of items: let i = index"`
- [x] `*ngFor="let item of items; trackBy: trackById"`

StackOverflow - How to use `trackBy` with `ngFor`

Q68. What does this Angular CLI command do?

```
ng build --configuration=production --progress=false
```

- [ ] It builds the Angular application, setting the build configuration to the "production" target specified in the angular.json file, and logs progress output to the console.
- [ ] It builds the Angular application, setting the build configuration to the "production" target specified in the angular.json file, and watching files for changes.

- [ ] It builds the Angular application, setting the build configuration to the "production" target specified in the angular.json file, and disables watching files for changes.
- [x] It builds the Angular application, setting the build configuration to the "production" target specified in the angular.json file, and prevents progress output to the console.

Angular documentation - `ng build`

Q69. Service classes can be registered as providers via which decorators?

- [ ] @Injectable, @NgModule, @Component, and @Directive.
- [x] @Injectable only.
- [ ] @Injectable and @NgModule only.
- [ ] @Service and @NgModule only.

Q70. What is the Input decorator used for in this component class?

```
@Component({
        selector:'app-product-name',
        ...
})
export class ProductNameComponent {
        @Input() productName: string
}
```

- [ ] It is used simply to put a comment in front of a class field for documentation.
- [x] It provides a way to bind values to the productName field by using the component selector.
- [ ] It autogenerates an `html <input type='text' id='productName'>` Dom element in the component template.
- [ ] It provides a way to bind values to the productName instance field,just like native DOM element property bindings. Angular documentation - `Input()`

Q71. Which route guard can be used to mediate navigation to a route?

- [x] all of these answers.
- [ ] CanDeactivate.
- [ ] CanLoad
- [ ] CanActivate. Angular documentation - `Input()`

Q72. How can you configure the injector to use an existing object for a token instead of having it instantiate a class instance?

- [x] Use the `useValue` provider configuration and set that equal to an existing object or an object literal.
- [ ] It is not possible. Providers can be configured only with class types.
- [ ] Simply add the object instance or literal to the providers array.
- [ ] Make use of the `asValue` provider configuration property, setting it to true.

Configuring dependency providers

Q73. Based on this route definition, what can be injected into UserDetailComponent constructor to get ahold of the id route parameter?

```
{path: 'user/:id', component: UserDetailComponent }
```

- [x] ActivatedRoute
- [ ] CurrentRoute
- [ ] UrlPath
- [ ] @Inject('id')

Common Routing Tasks

Q74. With the following reactive form markup, what would you add to wire up a call to an onSubmit class method?

```
<form [formGroup]="form">
  <input type="text" formControlName="username" />
  <button type="submit" [disabled]="form. invalid">Submit</button>
</form>
```

- [ ] neither of these answers
- [ ] Add (click)="onSubmit()" to the element.
- [x] Add (ngSubmit)="onSubmit ()" to the element.
- [ ] both of these answers

Angular - Forms

Q75. What is the expected DOM code for this usage of the ngClass attribute directive when isActive is true?

```
<div [ngClass]="{ ‘active-item': isActive }">Item One</div>
```

- [ ] `<div active-item>Item One</div>`
- [x] `<div class="active-item">Item One</div>`
- [ ] `<div class="is-active">Item One</div>`
- [ ] `<div class="active-item isActive">Item One</div>`

Angular - NgClass

Q76. Which answer best explains the usage of ngModel in this template code?

```
<input [(ngModel)]="user.name" />
```

- [ ] It is conditionally displaying the input element if the user.name property has a value.
- [x] It is the two-way data binding syntax. The input element value property will be bound to the user.name property, and the value change event for the form element will update the user.name property value.
- [ ] There is a typo in the code. It should have only the square brackets.
- [ ] It is binding the value of the user.name property to the input element's val property to set its initial value.

Angular - NgModel

Q77. What method is used to wire up a FormControl to a native DOM input element in reactive forms?

- [ ] Use the string name given to the FormControl as the value for the DOM element id attribute.
- [x] Use the formControlName directive and set the value equal to the string name given to the FormControl.
- [ ] Add the string name given to the FormControl to an attribute named controls on the element to indicate what fields it should include.
- [ ] Use the square bracket binding syntax around the value attribute on the DOM element and set that equal to an instance of the FormControl.

Angular - Reactive Forms

Q78. What other template syntax (replacing the ngClass directive) can be used to add or remove the CSS class names in this markup?

```
<span [ngClass]="{ 'active': isActive, 'can-toggle': canToggle }"> Employed </
```

- [ ]

```
<span class="{{ isActive ? 'is-active' : '' }}" class="{{ canToggle ? 'can-tog
  Employed
</span>
```

- [x]

```
<span [class.active]="isActive" [class.can-toggle]="canToggle"> Employed </spa
```

- [ ]

```
<span [styles.class.active]="isActive" [styles.class.can-toggle]="canToggle">
```

- [ ]

```
<span [css.class.active]="isActive" [css.class.can-toggle]="canToggle"> Employ
```

Q79. In this directive decorator example, what is the purpose of the multi property in the provider object literal?

```
@Directive({
  selector: '[customValidator]',
  providers: [
    {
      provide: NG_VALIDATORS,
      useExisting: CustomValidatorDirective,
      multi: true,
    },
  ],
})
export class CustomValidatorDirective implements Validator {}
```

- [ ] It indicates that the CustomValidatorDirective can be used on multiple form element types.
- [ ] It allows for multiple instances of the CustomValidatorDirective to be instantiated. Without multi, the CustomValidatorDirective would be a singleton for the entire app.

- [x] It allows the registration of different providers for the single NG_VALIDATORS token. In this case, it is adding the CustomValidatorDirective to the list of form validators available.
- [ ] It indicates that there will be multiple classes that handle the logic implementation for the custom validator.

StackOverflow

Q80. Which Angular CLI command would you use to run your unit tests in a process that reruns your test suite on file changes?

- [ ] ng test --single-run=false
- [ ] ng test --watch-files
- [ ] ng test --progress
- [x] ng test

Q81. What is the most common use for the ngOnDestory lifecle hook?

- [ ] Remove dome elements from the components's view
- [ ] All of theses answers
- [ ] Delete any injected serviced the
- [x] Unsubscribe from obervables and detach

Q82. What NgModule decorator metadata property is leverage to allow other ....?

- [ ] public
- [ ] experts
- [ ] Shared
- [x] declarations