Angular Test

1. Which decorator lets you inject a service registered with an Injection Token?
   1. Import
   2. Inject
   3. Input
   4. Injectable
2. What are decorators used for?
   1. To give a DOM node additional functionality
   2. To create smaller components
   3. To create sub components
3. What is the best way to prevent a user from leaving a component based on specific conditions?
   1. Create a route guard and use the "canDeactivate" property of the component's route.
   2. Check the condition inside links and buttons that would navigate away from the component.
   3. Create a route resolver and use the "resolve" property of the component's route.
   4. Check the condition in the component's ngCanDeactivate method.
4. What is one way you can pass data from a parent component to a child component?
   1. You import the child component into the parent component.
   2. You add an @Input() property on the child component and bind to it from the parent component.
   3. You import the parent component into the child component.
   4. You add an @Output() property on the parent component and bind to it from the child component.
5. What's the best way to add component-specific styles to a component?
   1. Add the styles to a css file imported by your index.html.
   2. Specify the styles or styleUrls inside the component meta-data config.
   3. Add the styles inline inside your component template html.
   4. Reference an external stylesheet from inside your component template html.
6. What is one advantage of RxJS over Promises?
   1. They are simpler.
   2. They are more performant.
   3. They can handle a stream of data instead of just one value.
7. What is the default token in Angular’s Dependency Injector?
   1. A string
   2. A class
   3. The Token class
8. What does the command ng new my-dream-app do?
   1. Generates a new app named my-dream-app
   2. This is an invalid command.
   3. Displays the files it would generate, but doesn't actually generate an app
9. Which ng new flag instructs the Angular CLI to generate an NgModule with routing?
   1. --module-routing
   2. --add all
   3. --module
   4. --routing
10. In which typed language can you write Angular 2?
    1. TypeScript
    2. JavaScript (ES5)
    3. JavaScript (ES6/ES2015)
    4. C#
    5. CoffeeScript
11. You have a bug in your template binding and cannot remember the name of your character model's properties. How can you expose them most easily to discover the true names?
    1. Using the json Pipe

{{ characters | json }}

* 1. Searching StackOverflow
  2. Using the flux capacitor

{{ flux(characters) }}

* 1. Using the json property

{{ characters.json }}

* 1. Using the debug function

{{ characters.debug() }}

1. What do we use with Angular 2 to reference code in other files?
   1. TypeScript interfaces
   2. ES6 modules with export and import
   3. TypeScript buckets
   4. The same angular.module construct as Angular 1
   5. ES6 desctructuring
2. What does the Angular 2 @Component decorator's selector property define?
   1. The location of the Component's HTML Template
   2. The location of the Component's HTML CSS
   3. The name of the HTML element that represents the Component
   4. The name of the class that represents the Component
   5. The name of the HTML element that represents the Router
3. How do you bind an event in Angular 2?
   1. Use ngBindEvent to bind to a Component's function
   2. Wrap a HTML element's event with square brackets and set it to a Component's property
   3. Wrap a HTML element's event with parentheses and set it to a Component's function
   4. Use ngModel to bind to a Component's function
4. Which lifecycle hook would you use if you wanted to get data for a component when it started?
   1. ngOnInit
   2. ngOnActivate
   3. ngOnStart
   4. ngOnBegin
5. What will an Async Pipe do when applied in a template?
   1. Call a synchronous function asynchronously
   2. Filter data based on a filter condition
   3. Call http.get directly
   4. Subscribe to an Observable or a Promise and display its data results
6. What would you use in Angular 2 to describe routes?
   1. The Routes class
   2. The @RouteProviderWhen decorator
   3. The route.when API
   4. The @Component decorator
7. What property of the Routes do we set to identify which module to load, when lazy loading?
   1. loadChildren
   2. moduleToLoad
   3. path
   4. childRoute
   5. Loading
8. There are several conditions where you should consider using Angular for your front end framework. Which of the following is one of those conditions?
   1. You use a typed language on the backend
   2. You can't use cookies
   3. You use a non-relational database
   4. You use a relational database
9. What is the most important feature of TypeScript to learn when learning Angular?
   1. How to create function return signatures
   2. How to use interfaces
   3. How to use generics
   4. How to decorate identifiers with types
10. The CLI can do which of the following?
    1. display code for debugging
    2. create a GitHub pull request
    3. linting your code
    4. integrate with electron
11. Which way does "One-Way Data Flow" work?
    1. From templates to components
    2. From child up to parent
    3. From parent down to children
    4. Across each child, then each parent, then each grandparent, etc.
12. Dependency Injection in Angular is closely tied to which technology?
    1. Routing
    2. Change Detection
    3. TypeScript
    4. Webpack
13. What do directives do?
    1. Group visual display with functionality
    2. Format display of data
    3. Provide different views based on the URL
    4. Add new capabilities to HTML elements
14. Angular supports which of the following features?
    1. No custom template syntax
    2. No need for a build tool
    3. Integrated components & templates
    4. Lazy Loading
15. When building a form using the **Reactive approach**, how is the form model created?
    1. You set the ngForm directive on the form element in the template.
    2. You build it in the component class.
    3. Angular automatically creates it for you.
    4. You set the ngModel directive on the form element in the template.
16. Which is the correct syntax for watching the 'notification' FormControl and logging its value?
    1. this.customerForm.get('notification').valueChanges.subscribe(  
        value => console.log(value)  
       );
    2. this.customerForm.get('notification').valueChanges(  
        value => console.log(value)  
       );
    3. this.customerForm.get('notification').subscribe(  
        value => console.log(value)  
       );
17. In the following code, where could a default value be specified?

this.customerForm = new FormGroup({  
 firstName: new FormControl(),  
 sendCatalog: new FormControl()  
});

1. As part of the key:

sendCatalog(true): new FormControl()

1. As a second argument:

sendCatalog: new FormControl(), true

1. In the FormControl constructor:

sendCatalog: new FormControl(true)

1. As a second element of an array:

sendCatalog: [new FormControl(), true]

1. What is the purpose of the formControlName directive?
   1. To associate an input element with its FormControl within a FormGroup.
   2. To bind the form to the form model defined in the component class.
   3. To create a FormControl in the template with the defined name.
2. Why build a data access service?
   1. It keeps the data access logic out of the components and makes it more reusable across components.
   2. It greatly improves performance of the http requests.
   3. Angular requires each application to have a custom data access service to use Angular's built-in Http service.
3. What does this code do?

this.route.paramMap.subscribe(  
 params => this.id = +params.get('id')  
);

1. Adds the parameter to the URL path segment
2. Subscribes to changes in the route
3. Subscribes to changes to the route parameter
4. Which is the correct syntax for adding multiple validators to a FormControl?
   1. firstName: ['', [Validators.required, Validators.minLength(3)]]
   2. firstName: {'', [Validators.required, Validators.minLength(3)]}
   3. firstName: ['', Validators.required, Validators.minLength(3)]
5. What do the letters in the acronym CRUD represent?
   1. Caller, reader, user, and dealer
   2. Call, random, update, and decimal
   3. Create, read, update, and delete
   4. Create, RAM, uint, and downtime
   5. Carrier, read, uptime, and delete
6. What HTTP verb is used when retrieving data from a REST service?
   1. PUT
   2. DELETE
   3. POST
   4. GET
   5. PATCH
7. What Angular class is used to make HTTP requests?
   1. GetReq
   2. HttpClient
   3. HttpSender
   4. Interceptor
   5. ClientRequest
8. What type do most methods on HttpClient return?
   1. RESTRequest
   2. Interceptor
   3. Observable
   4. CacheObject
   5. POST
9. What is the name of the interface that Angular interceptors implement?
   1. HandleRequest
   2. HttpCatch
   3. RequestResponse
   4. InterceptCall
   5. HttpInterceptor
10. In what order do interceptors execute?
    1. The order is indeterminate
    2. Alphabetical order by class name
    3. The order in which they were provided in a module
    4. Reverse alphabetical order by class name
    5. The calling controller determines the order
11. What type of cache do interceptors allow you to build?
    1. Browser-specific caches
    2. Server-side caches
    3. Document caches
    4. Application-specific client-side caches
    5. SQL caches
12. Where are dependencies typically injected in an Angular application?
    1. destructor
    2. tsconfig file
    3. constructor
    4. forms
    5. import statement
13. What part of an injected dependency must match a provider token?
    1. imported NgModule name
    2. data type of the constructor parameter
    3. data type of the component
    4. factory function name
    5. component property name
14. What is the best resource for researching available Angular services and extension points?
    1. Codeplex
    2. Angular source code
    3. Angular XML documentation
    4. Angular API documentation
    5. StackOverflow
15. What function do you call on a Promise to register the callbacks that will process it?
    1. success()
    2. broken()
    3. new()
    4. error()
    5. then()
16. What Angular construct should only be used to control and manage the user interface?
    1. component
    2. pipe
    3. service
    4. route
    5. Module
17. What is the name of the decorator conventionally added to all Angular services?
    1. NgModule
    2. Component
    3. Service
    4. Injectable
    5. Core
18. What TypeScript language construct is used to create a new Angular service?
    1. function
    2. class
    3. import
    4. enum
    5. Interface
19. Which of the following best describe an Angular service?
    1. Orchestrator of view logic
    2. Collection of methods that don't fit anywhere else
    3. Discrete piece of reusable functionality
    4. Coordinator of routing
    5. Encapsulated data presented in UI
20. Which type of module should provide should be used to declare components that can be shared with other modules?
    1. A Shared Module
    2. The Core Module
    3. A Feature Module
    4. The App Module
21. Where is the recommended place to put data retrieval logic (such as http calls to an api)?
    1. In a custom data retrieval service
    2. Anywhere in your application where you need the data
    3. In a pipe
    4. In the component where the data is needed
22. What is the recommended way to identify input and output properties on components?
    1. There is no need to identify input and output properties
    2. Declare them in the module where the components are declared
    3. Use the @Input() and @Output() decorators on component properties
    4. Declare them in the `inputs` and `outputs` component metadata properties
23. What is the recommended folder structure for Angular apps?
    1. Group files of the same type together (/components/, /services/, etc)
    2. Use feature-based folders (/users/, /catalog/, etc)
    3. Group by layer (/templates/, /business-logic/, etc)
    4. Don't use any folder structure until it's absolutely necessary
24. What coding principle suggests that a service, component, or other class shouldn't take on too many behaviors?
    1. The Law of Yuri
    2. The Single Responsibility Principle
    3. The One-to-One Rule
    4. The Simplicity Principle