Angular Course Topics

1. Module
2. Component
3. AppModule/Component
4. Custom Component
5. CLI and Nesting
6. Component Styles
7. Component Selector
8. Databinding
9. String Interpolation
10. Property Binding
11. Event Binding
12. Bindable property and events
13. 2-way binding
14. All forms of databinding
15. ngIf, ngStyle, ngClass, ngFor
16. Event and property binding
17. Binding custom properties
18. View Encapsulation
19. Local References
20. @ViewChild()
21. Ng-content
22. Component lifecycle – Hooks
23. @ContentChild()
24. Directives
25. Renderer
26. @HostListener and @HostBinding - Binds a property with listener. Works best for working with internal element
27. Set properties for directives
28. Structural directives -vcRef.CreateEmbeddedView()
29. Services
    a. Need
    b. Injecting Service
    c. Hierarchical injector - AppModule, AppComponent, Any other component
    d. @Injectable
    e. Using Services for Cross-Component Communication
30. Routing
    a. Keywords: router-outlet, router, router module, routerlink, routeractivelink, routerLinkActiveOptions, router.navigate - relativeTo, ActivatedRoute, route paramters – snapshot

b. Route.params can also be subscribed so that data in component gets updated on reload
c. Query Params and fragments - Subscribe it so that we always receive updated value
d. Child Routing - children[]
e. queryParamsHandling: preserve
f. Wildcard
g. AuthGaurds - AuthService, CanActivateChild, canDeactivate, static data routing, **Dynamic data with Resolve Guard**
h. Usehash: true

31. Observables
    a. Various data sources = Events, Http Request
    b. Custom observable
    c. RxJS – Operators
    d. Subjects
32. Forms
    a. Template-Driven (TD) vs Reactive Approach
    b. Angular infers the form object from DOM / Form is created programmatically and synced with the DOM
    c. Using FormArrayControl

```
Example1: getControls() {
        return (<FormArray>this.signupForm.get('hobbies')).controls;
        }
        In the template, you can then use:
        *ngFor="let hobbyControl of getControls(); let i = index"

Example2: get controls() {
    return (this.signupForm.get('hobbies') as FormArray).controls;
    }
    and then in the template:
    *ngFor="let hobbyControl of controls; let i = index"
```

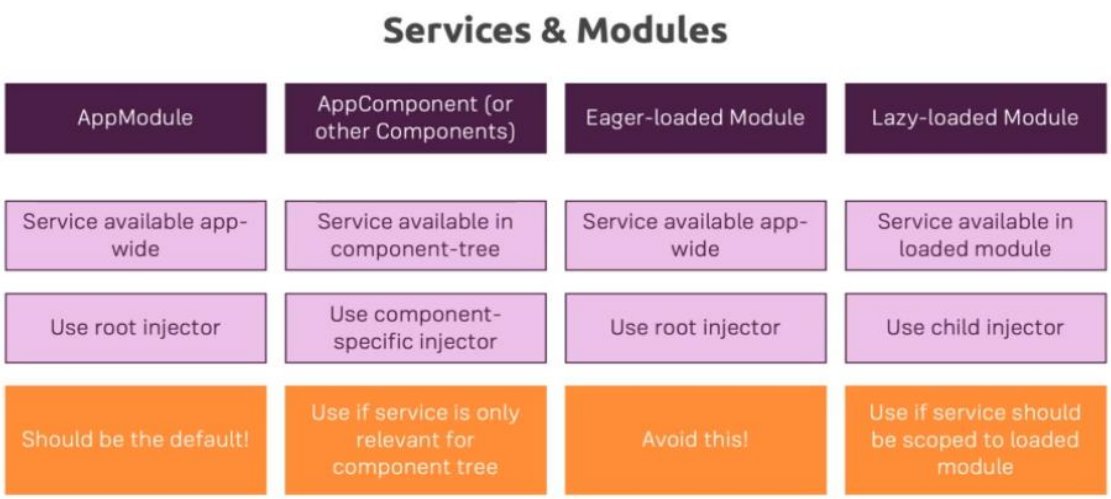
    d. Custom Validators
    e. Using reactive error codes
    f. Async validator
    g. As of Angular 8+, there's a new way of clearing all items in a FormArray.

```
(<FormArray>this.recipeForm.get('ingredients')).clear();
```

    The clear() method automatically loops through all registered FormControls (or FormGroups) in the FormArray and removes them.
    It's like manually creating a loop and calling removeAt() for every item.
33. Pipes
    a. Inbuilt pipes

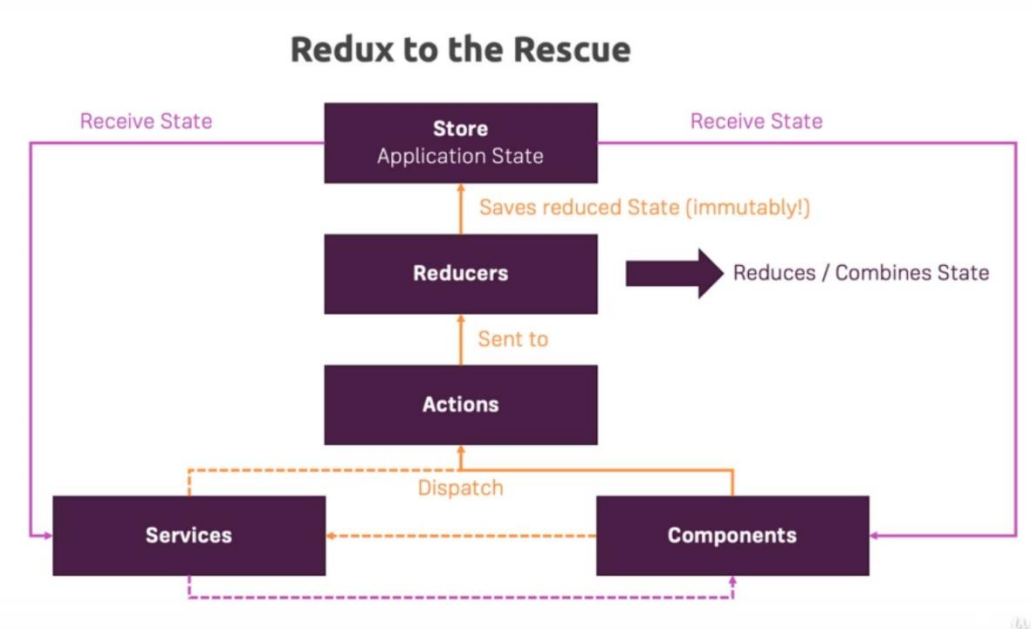
b. Custom pipes
c. Pure/Impure
d. Async

34. HTTP requests
    a. Anatomy of a HTTP request
    b. Using RxJS transform – pipe(), map()
    c. Using types with HTTP client
    d. Use subject for error handling
    e. catchError()
    f. observing different types of responses
    g. Interceptors – intercept(), manipulating request objects, response interceptors, multiple interceptors
35. Authentication
    a. BehaviorSubject
    b. take(), exhaustMap()
    c. Auth Interceptor - HTTP_INTERCEPTORS
    d. Auth Guard
36. Dynamic Components
    a. ngIf vs Programmatic approach
    b. Programmatic approach - ComponentFactoryResolver
    c. entryComponents()
37. Angular Modules & Optimizing Angular Apps
    a. Splitting modules
    b. Lazy Loading
    c. Preloading lazy-loaded code – preloadingStrategy
    d. Aot vs JiT

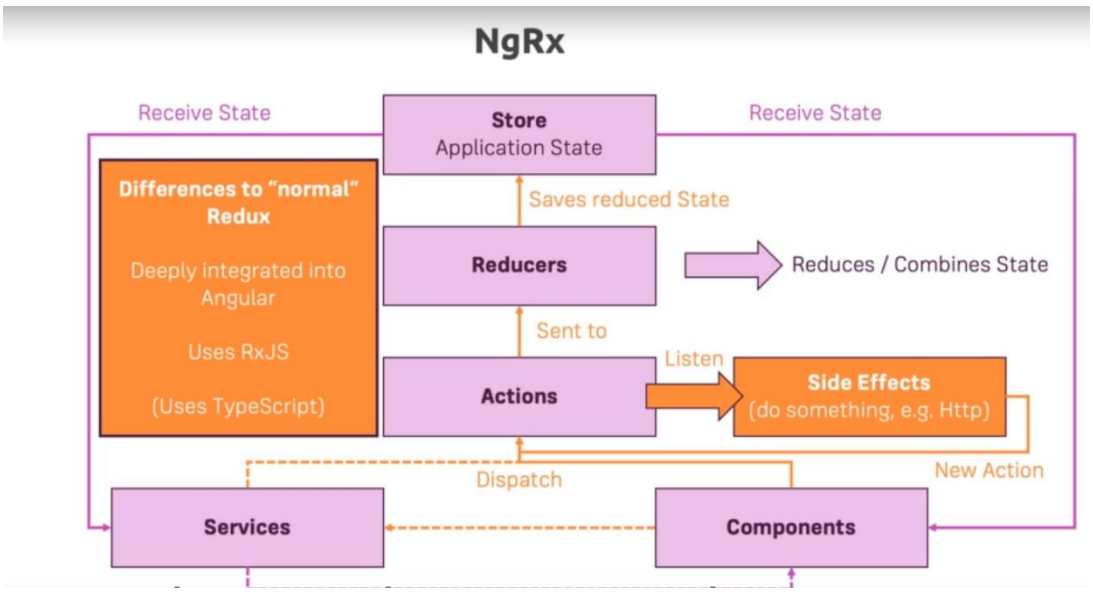


38. NgRx – Manage application state
    a. RxJs has few issues so prefer NgRx inspired by Redux

b.



c.



d. All about reducers
e. Action, State
f. select(), dispatch()
g. One root state
h. Effects, Error handling
i. Store Dev tools – Redux extension
j. The router store – ngrx/router-store
k. Refer ngrx-github for example projects

39. Angular Universal
    a. ModuleMapLoader
    b. NestJS

40. Angular Animations
    a. @angular/animations
    b. Animation triggers and state
    c. Transitions – can use <=>
    d. Transition phases
    e. The void state
41. Adding Offline Capabilities with Service Workers
    a. Official Angular Service Worker Docs: https://angular.io/guide/service-worker-intro
    b. Academind Resources on PWAs: https://academind.com/learn/progressive-web-apps
42. Unit Testing
    a. Why? Analyse. Setup. Run.
    b. Add test cases, test dependencies: components and service
    c. Async Tasks, Using fakeAsync and tick
    d. Isolated vs Non-Isolated Tests

    Official Docs: https://angular.io/docs/ts/latest/guide/testing.html

    I can also recommend the following article: https://semaphoreci.com/community/tutorials/testing-components-in-angular-2-with-jasmine

    For more Information on how to run Tests with the CLI have a look at their official Docs:

    => Unit Tests: https://github.com/angular/angular-cli/wiki/test

    => E2E Tests: https://github.com/angular/angular-cli/wiki/e2e
43. Closer loot to CLI and Angular as a platform
    a. ng new, ng add, ng generate, ng update, ng build, ng test, ng lint, ng deploy
    b. IDE and setup, CLI Commands
    c. Understanding angular.json, differential loading
44. Angular Elements – Create custom element from component
45. Typescript
    a. Classes
    b. Interfaces
    c. Generics
    d. Dive deep