Angular my notes

Hooks of component

Ngonchanges

Ngoninit

Ngdocheck

Ngaftercontentinit

Ngaftercontentchecked

Ngafterviewinit

Ngafterviewchecked

Ngondestroy

Angular dinesh notes

Component: basic building block of any app

3 parts:class,component decorator

Interpolation: to bind data in template from property

{{name}}

Property binding: [pname] like [‘id’]=’idvar’

Denotion[] for [disabled]->property binding

Class binding [class] <h2 [class]=’success’>

Ngclass for multiple classes

Style binding [style.color]: to apply inline style to tag

Event binding: template to class (click)=”onclick()”

Template reference variable:access an element from template html and all its dom properties we can do by template reference variable #myinput

<button click=”showvalue(myinput)”

Two way binding: combination of data binding and event binding (view to class) denoted by [(ngmodel)]

Directive

Import forms module to use ngmodel

Structural directives:ngif,ngswitch,ngofr ( to render list of elements)

Pipes: used to transform data in views

|lowercase

Slice and json pipe: slice : to get substring of a string

Json gives json of an object

Number,percent,currency

|number’1.2-3

Min int min dec and max dec

Date pipe: to transform date

Service:it is class to share data among component reusable code

Service.ts

Dependency injection: to inject service into component at run time

Create a service class

@injectable decorator used to tell service is injectable

d.i. pattern: to declare the dependency inside class and use them as arguments like constructor(engine, tyres)

disadv: dependency with in dependency create issue

imports: to include other modules into a module

providers: to register di service class into the module

@injectable is needed when service class has own dependency

Observable: returns sequesnce of items

By using httpclient we can fetch data

Typecast could be done of observable using interface

<iemployee[]>

Handle error in observable using catch

Difference in promise and observable

Single value streams of value

Not lazy lazy

Subscribe not require subscribe require

Promise invoke even cancel using unsubscribe(map, filter,foreach,reduce,retryand retry when)

operators used

then used or not

can not cancelled

this.subscription.unsubscribe();

rxjs/subscription used for unsubscribe

unsubscribe used in ngdestroy error

error handling using catch

error handling in forms using catch eroor and throw errors

forms in angular

template driven and reactiveforms

template driven: when more logic is written in template. Ngform,ngmodel, ngmodelgroup directives are used.

Ngmodel: it take care particular form controls

Ngmodelgroup: it take care group together multiple form controls.

Ngform: used to check validation status of form controls and to pass data.

Model: it is typescript class

Classes that binds angular form controls

Ng-untouched, ng-touched, ng-dirty, ng-valid, ng-invalid we can fetch using template reference variable

{{nam.untouched}} #name=”ngmodel”

Regular expression could be used. Pattern “^\d”

Errors could be shown using ref variable

Submit data: using ngsubmit=”onsubmit()”

Create a service and subscribe it in component

Reactive forms:

1) much logic in component class

2 we need to react to user input to update value

3 for complex scenario like dynamic form fields like additional phone no option give.

Custom validation: pwd and r.pwd match

4 dynamic validation : on checkbox we need to validate reg expression

5 unit test: unit test could be done easily

Limitation: more code in comparision template driven

Formgroup and formcontrols are imp classes of reactive forms

[formgroup] directive and property bind to form .

Nesting of formgroup using address (city,state) in registration formgroup

Setvalue to addvalue

Patch value for few control assign value.

Formbuilder service: used to create form with lesser code.

Angular interview questions

Two way binding: when data changes in model it reflects in view and if data changes in view it reflects in model

Spy:insight into dom which can not be changed directly

<div myspy>

Life cycle hooks

Ngoninit:to do complex initialization like api call and to set up component

Ngdocheck: to do changes which angular cannot do own like oldhero name and new heroname assign

Afterviewinit and afterviewchecked

@viewchild is used to query into child view.

Afterviewinit to do action after view is initialized.

Afterviewchecked : to do action after view is checked

Aftercontentinit:calls after angular projects external content

Content projection: to import html content from outside component and put into a spot. This is transclusion technique

Ng-content: this tag is placeholder for external content

Browser module: it is used when you want to run your application into the browser.

Common module: when use directive like ngif, ngfor

Router module: router link, forroot and forchild

Httpclientmodule: to initiate http request

Best practices of angular

1 ) use of angular cli

Ng new prjname

2 develop in modular fashion:core, shared and feature for complex application each module( own service, component and pipes)

Core- header, footer, navigation , side bar

Shared: component and pipes used by other modules

Feature: consists specific functionality collaborate with root module

3 lazy loading to feature module: like dashboard module

Loadchildren

4 use smart vs dummy: dummy for presentational logic

5 proper use of dependency injection: old way to insert service into providers. New way to assign service

Injectable( { providedin:’root’})

Root could be replaced by any module

6 create aliases for imports: like define in tsconfig.json

Performance in angular

1 trackby: use trackby function will return unique identifier. When array change then angular renders all but by trackby it will change dom for particular element.

2 const vs let: when declaring variable use const as much possible

3 pipeable operators: these are tree shakeable. It means only the code that need to be executed will be included on imports

.pipe(map(value=>value.item) earlier only map used.

4 isolate api hacks: isolate api calls in service so could be fixed easily on error.

5 subscribe in template: try subscribe in template instead of component file using async pipe lint rule could be used to unsubscribe.

6 clean up subscription: make sure unsubscribe done by using take and takeuntill operators. Memory leak could be there.

7 use appropriate operators:

Switch map: ignore previous emissions

Merge map: concurrently handle all emissions

Concat map: handle emission one after another

Exhaust map: when want to cancel all new emissions while processing previous emissions.

8 lazy load: load something when it is used. Application boot time will increase.

9 avoid subscription inside subscription: use chaining operators(with latestfrom, combinelatest)