Angular - Basic Terminologies

Friday, September 1, 2017 1:59 PM

N Murugan Nagarajan at 10/09/2017 12:11 AM

ANGULAR

- **WHAT:** One of the Frontend UI Framework
  LIKE - REACT, VUE

- **Purpose**: Code using vanilla JS becomes difficult to maintain for big applications.

- **SIMPLE ARCH:** UI (ANGULAR/HTML/CSS) <==========> BACK END (DATA/API/LOGIC)
  HTTP ENDPT or PORT / HTTP REQUEST or SHIP / DATA or LOADS

- **TOOLS:** NODEJS Installation,
  will be installed automatically
  Then, npm install -g @angular/cli
  Verify the version : ng -v

- **New App Creation** : ng new <projectname> , npm install, ng serve

- **Version:** Angular 1.x to Angular 4. No version number please.

TYPESCRIPT - Superset of JavaScript

Install: npm install -g typescript
Verify the version : tsc -v

- **LET -** keyword can be used instead of var in js - `let n; //By default the type is any..,`
- **DATATYPES -** `let n: number;/let b: boolean;/let s: string;/let a: any;/let na: number[];/let aa: any[];/enum Color {blk=0,wht=1}`
- **TYPECAST -** `(<string>mynameAny) or (mynameAny as string)`
- **ARROW -** `(msg1, msg2) => { console.log(msg1); console.log(msg2); };`
- **INTERFACES -** `can act as object type to pass parameter inputs inside a function`
- **CLASSES -** `useful to group the parameters and all the related functions`
- **CONSTRUCTORS -** `ass parameter input inside a function`
- **ACCESS MODIFIERS -** `Private / Public .. Just declare the variables in constructors whey will be accessible at class automatically`
- **PROPERTIES -** `Also think about using either " age / get AGE() " or " _age / get age() ".`
- **MODULES -** `export class LikeComponent {….} / import { LikeComponent } from './like.component'`

ANGULAR BUILDING BLOCKS

- **COMPONENT** - Encapsulates Data & Logic & HTML
  - Create a component
  - Register it in a module
  - Use it /Refer it in a HTML

  Easiest way to do the above steps is by,,
  ng g c <component name> ------- Validate **declarations** whether it is injected properly in the app.module.ts / manually add it in the required module
  ng g s <servicename> ------- Validate **providers** whether it is injected properly in the app.module.ts / manually add it in the required module

- **MODULES** - Collection of Related Components. (Think of grouping/collection of items in a Super Market like Groceries/Meat/Veggies)

- **INTERPOLATION -** {{ }}
- **DISPLAY DATA** -
  - Property Binding , `<img [src] = 'imageUrl' /> Alternative of String Interpolation {{}}`
  - Attribute Binding , `<td [attr.colspan] = 'colspanVal'>`
- **APPLY CLASSES/STYLES -**
  - Class Binding , `[class.active]="isactive"`
  - Style Binding , `[style.backgroundColor] = "isactive ? '#87ceeb' : 'WHITE'"`
- **EVENT HANDLING -**

- Avoid Event Bubbling effect when coding event handler.
- `<button (click) = onSaveEvent()>`
- `<input (keyup.enter) = "onKeyEnter()"/>`

- **VARIABLE BINDING -**
  - [(Banana in the box)] - The present better way to bind the variables between the view & component
- **PIPES -** Are useful for formatting the different data types in the view to display in a presentable way.

```
- NAME:{{user.name | uppercase}}
- AGE :{{user.age | number }}
- SAL :{{user.salary | currency : 'INR' : true}}
- DOB :{{user.dob | date: 'shortDate'}}
- TAX :{{user.taxrate | number : '3.1-2'}}
- PROFILE: {{user.profiledesc | summary : 100}}
```

ANGULAR COMPONENTS API

- Component becomes reusable when they are able to accept Inputs & provide Outputs based on User action
- The flow would be STATE --> INPUT --> {COMPONENT} --> OUTPUT --> EVENT
- `@Input('favoriteflag') isFavorite = true;` -- Use Alias names to reduce the dependency between the view & component
- `@Output('favoriteclick') change = new EventEmitter();` -- EventEmitter used to trigger the event on a specific user action
- Now the above two attributes can be used to pass the input and trigger the corresponding event as a Ouput.
- `<favoritestar [favoriteflag]="favFlag" (marked)="printChange($event)" (favoriteclick)="favoriteChange($event)"></favoritestar>`
- Thus, this component can be reused through-out the application by providing different inputs and doing different output actions based on demand.
- **SHADOW DOM :** Allows to scope the css style only inside a specific component.

ANGULAR DIRECTIVES

DIRECTIVES TYPES

- ngFor, ngIf, ngSwitchCase, ngModel, hidden etc..,
- **STRUCTURAL DIRECTIVES ,** Modify the DOM Structure - ngIf, ngFor
- **ATTRIBUTE DIRECTIVES ,** Modify the attributes of DOM elements - ngModel
- Decide whether to use **ngIf** or hidden based on need & performance impact.
- CHANGE DETECTION -- DOM Events, AJAX , TIMERS etc triggers a change detection event in angular to restructure the DOM accordingly.
- SAFE TRAVERSAL -- Useful to handle null objects. :: **object?.value** .., Avoid exception in view console
- TRACKBY - Remember to use trackby while using For Loops to avoid multiple memory allocation for the same list of objects.

ANGULAR FORMS

FORM TYPES

- **TEMPLATE DRIVEN -** For simple forms and common validation purpose
- **ngForm - FormGroup -** value, touched, untouched, dirty, pristine, valid, errors - Collection of FormControls
- **ngModel - FormControl -** value, touched, untouched, dirty, pristine, valid, errors
- **ngModelGroup**
- ngModelGroup doesn't have ngSubmit like ngForm, it's only purpose is use to group similar form elements and check their validity/status as a whole.

- **REACTIVE FORMS -** For complex forms which requires specific user defined validations. Unit Testable..,
- Creating controls and add Validation
- Implement custom user defined validation
- Implement async validation which goes to server side for validation purpose
- Try to build forms that include an array of objects for practice purpose
- **ASYNC VALIDATIONS** makes use of Promise/Observable
- FormControl, FormGroup, FormControlName, FormGroupName(sub-groups/forms), FormArrays(array of FormControls), FormBuilder,

ANGULAR HTTP SERVICES

- **CRUD Operations**
- **Creating Reusable Data Service**
- **How to Handle Errors -** Error Code from Response, Creating Application Error classes.
- Define Proper **Separation of Concerns**
  - Create a service layer to handle the HTTP requests instead of overloading the component.
  - Create Application error classes and throw them instead of handling different error codes inside the components.
  - Create a common generalized data service which can be reused by other service classes.

- **Setup local JSON-Server** in your machine using **npm install -g json-server** or add required dependency in package.json
  - Configure it `"json": "json-server --watch db.json"` in package.json
  - ! Execute it by **npm run json** before triggering the **ng serve** to access the local json-server which acts as a backend

- Configuring **proxy.conf**.json to map a URL with the required server.
- Perform CRUD operations using ADD, PUT, PATCH, GET, DELETE
- Lifecycle Hooks: OnInit, OnChanges, DoCheck, AfterContentInit etc….,
- **Error Handling: Unexpected** (Server down, Network issue etc..,) & **Expected** (Client Input Validation failure, Business logic failure etc..,)
- Throwing Application Specific Errors and creating instance of different error classes based on error codes.

- **OBSERVABLES vs PROMISES**
  - Observable - lazy call, the request will not be sent until there is a subscribe part.
    - Collection of asynchronous data that arrives over time
  - Promises - eager call, the request will be forwarded even if the response is not handled/subscribed.

ANGULAR ROUTING & NAVIGATIONS

- **Routing Config**uration
- **Dynamic**/Programmatic **Navigation**
- Developing **SPA** - Single Page Application
- Step1: Configure Route , Step2: Add Router outlet, Step3: Add Links
- **ROUTERLINK**
  - **Simple Routes** `<a routerLink="followers">GIT Followers</a></li>`
    - `- Attribute Binding is sufficient`
  - **Simple Routes** `<a [routerLink]="['/followers', follower.login]">{{follower.login}}</a></li>`
    - `- Property Binding has to be done`

ANGULAR AUTHENTICATION

- JWT - Json WebTokens [Header, Payload, Signature/Secret Key] - All three in encrypted format
- Use angular2-jwt library to access/decode these tokens.
- Based on Authentication/login details & Authorization/role information - we can
  - Show/Hide Elements
  - Protect Routes using Guards
  - Protect API endpoints in the server using WebToken informations. We can send the token in the Request Header

ANGULAR DEPLOYMENT - #TODO Have a lot of pending activity in this topic

- **Old Deployment Method** - Copy Paste the Source code to Prod/Deployment environment
  - Lots of files, Big Bundle size
- Minification, Uglification, Bundling, Dead Code removal, Ahead of Time
- Solution : User **ng build -prod**
- **JIT (Just In Time) & AOT (Ahead of Time)**
  - Faster Startup
  - Small Bundle
  - Template error is noticed earlier
  - Security as the template is not sent to client as it is and it will be in a compiled JS format. (Like a class file)
- Either install the TSLINT plugin in VSCODE or use ng lint --fix command to resolve Typescript minor refactoring issues.

Deploy in GITHUB PAGES /GITHUB.IO

```
#Useful Commands to Deploy
    STEP1: Create a Github Repo in the Git website (Ex:- RepoName: angular-app)
    STEP2: Navigate to your project folder and follow the execute the below commands.
    STEP3: #Add the Origin to Remote Repo.git remote add origin https://github.com/murugan425/angular-app
    STEP4: #Push the Remote to Master Branch git push origin master
    #Enable the Github Pages in Settings of Github.com.
    STEP5: #Install the angular-cli-pages Globally npm i -g angular-cli-ghpages
    STEP6: #Execute a build  ng build --prod --base-href='https://murugan425.github.io/angular-app/'
    STEP7: #Trigger the angular-cli-gh command 'ngh'
    #We can merge Step6 & 7 in a single command by adding a script in Package.json
    "ghdeploy": "ng build --prod --base-href='https://murugan425.github.io/angular-app/' && ngh"
    #Then Execute npm run ghdeploy
```

Deploy in FIREBASE

```
#Useful Commands to Deploy
    STEP1: Install Firebase : npm i -g firebase-tools
    STEP2: Navigate to the application folder and run 'firebase login'
    STEP3: Use the Googel Oauth verification to authenticate.
    STEP2: Build the application : ng build --prod
    STEP3: Configure the Firebase JSON file added in the application.
```

```
        STEP4: Deploy in Firebase
```

- **Deploy in HEROKU**

```
#Useful Commands to Deploy
     STEP1: Install Heroku: Better to download and install Heroku based on your OS instead of npm
```

- **ANGULAR WITH REDUX**
  - Used to store or manage the state of the application
  - Too much useful to avoid data sharing between multiple components that are independent of each other.
  - STORE - ACTIONS - REDUCERS
    - STORE: Single JS object which contains the data.
    - ACTION: In general terms **events** - tells some change has happened in the state.
    - REDUCER: Event Handler based on the change happened
  - Advantages:
    - Testability
    - Go back in Time by debugging and check the state/events for analysis
    - Thereby, we can perform undo/redo
  - **Action ----> Store ---> Reducer ----> New State to Store**
    - Component dispatches the action
    - Based on action type, the Reducer triggers the required action method and updates the state.
    - Then the new state is returned back to Component
    - Using the @select & async pipe the new change reflects in the UI
  - Try to refactor the reducers, store and action by grouping them and use combineReducers to merge them together to create a si ngle store.
  - Make use of redux-devtools extension in devmode which enables to save states and reload them, very helpful to reproduce events

- **ANGULAR ANIMATIONS**
  - **CSS** (animate.css) **& JAVASCRIPT** (Web Animations API)
  - Function available: trigger, transition, state, animate etc etc..,
  - **Basic Principle:** State1 ----transits/changesto-----> State2
  - Different States: **Void** (ex.., add/remove) ===> **Default**(current state) ===> **Custom**(ex.., collapse/expand)
  - **void => * or * => void** can be combined as **void <=> *** or we can use alias **:enter , :leave**
  - **Easing**: linear, ease-in, ease-out, ease-in-out and cubic-bezier curve..,

- **ANGULAR MATERIAL**
  - Reusable UI component built with Typescript & Angular
  - Internationalized, Clean, Simple, Customizable, No much performance impact, etc…