Angular Course

=====

HTML5 Cheat sheet: https://www.wpkube.com/html5-cheat-sheet/

Check html5, css3 - browser support: https://caniuse.com/

ES2015 E-Book: https://leanpub.com/understandinges6/read

ES2015- Ref: http://es6-features.org/

Github Link - Project 1 : https://github.com/arunprabu/lnt-website-aug-2020

Github Link - Project 2: https://github.com/arunprabu/lnt-ph-book-sep2020

Authentication -- Refer this project

https://github.com/arunprabu/phonebook-ng8-lntfeb

Final Feedback and Project Submission Link: https://drive.google.com/file/d/1DWcmG9TTAdj\_YHjLHNTg5hnijEr60eil/view

Angular E-Book:

====

Web

---

Client Server Arch

------

Client - Browser

Server - Tomcat, Apache, JBoss, NodeJS Server

Web App Develop

1. Server Side Rendering Apps / Traditional Webapps

Server: 1 (front end + backend)

Executes HTML + CSS+JS + Backend + DB queries

Response would be HTML, CSS, JS, Data

Disadvantages

---

1. Page will refresh

2. Heavy lifting,

3. delayed response

4. Higher Bandwidth consumption

5. Expensive

6. Not better UX

2. Client Side Rendering / Single Page Apps (SPA) / Modern Webapps

Servers : 2 (1 Server for the front end + 1 server for the backend )

Front End Server ( responding with HTML, CSS, JS, may be running on 4200)

Back end server ( respond with data - JSON, may be running on 8080)

Backend will be known as REST API

Browser ( will render it with the data in right places)

Advantages

---

1. Page will not refresh

2. Faster websites

3. Lesser Bandwidth consumption

4. Less Expensive

5. Better UX

Examples: gmail, angular website, reactjs website, our own app.

-----

Web Front End layers

---

1. Structural Layer (HTML)

2. Presentational Layer (CSS)

3. Interaction Layer (JS)

HTML

---

Semantic elements / tags

1. Block level elements

div, p, header, footer, nav, main, section, form, aside

2. Inline Elements

a, img, span, input, textarea, button, video

HTML5

------

Learn about more html5 elements

Tasks:

====

Learn about HTML5 form elements and attributes

CSS

===

3 Ways to use CSS

---

1. External CSS -- app-wide css

2. Internal CSS -- page-wide css

3. Inline CSS -- css will be limited to one element

Syntax:

---

selector {

// style definition

color: red; // property: value of the property;

text-align: center;

}

Types of selectors

----

1. element/tag selector h1, h2, div, input

2. id selector #para1, para2, #passwordInput

3. class selector .main-text, .container

4. attribute selector [type='text']

5. pseudo selector -- todo

6. universal selector \*

Todo:

HTML5 form attributes

color picker, date picker

pseudo selector,

understand units in css - px, em, vh, %

experiment with float left and float right

learn more about css precedence

learn about embedding fonts using css3 @font-face

-------------------

Refer Bootstrap Starter template

---

https://getbootstrap.com/docs/4.5/examples/starter-template/

todos:

====

Work with Bootstrap to develop other pages

//todos -- 31st Aug 2020

====================

// disable the button on click

// work with mouseover

// try drag and drop -- refer w3schools

// experiment with innerHTML

bootstrap-demo.html

===

<div class="row justify-content-md-center">

<div class="col-md-6">

<h2>Contact Form</h2>

<br>

<form>

<div class="form-group">

<label for="exampleInputName">Name</label>

<input type="text" class="form-control" id="exampleInputName"

placeholder="Name">

</div>

<div class="form-group">

<label for="exampleInputEmail1">Email address</label>

<input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" placeholder="Enter email">

<small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>

</div>

<div class="form-group">

<label for="exampleInputPhoneNo">Phone No</label>

<input type="tel" class="form-control" id="exampleInputPhoneNo"

placeholder="Phone Number">

</div>

<button type="submit" class="btn btn-primary">Submit</button>

</form>

</div>

</div>

==========

Rest client

---

https://chrome.google.com/webstore/detail/advanced-rest-client/hgmloofddffdnphfgcellkdfbfbjeloo

or try postman

REST API URL

---

https://jsonplaceholder.typicode.com/users

TypeScript

====

TypeScript

----

Microsoft,

TypeScript = JS + Datatyping + OOP

TypeScript => Superset of JS

-----

1. Loosely typed

var x = 10;

x = true; //works

x = []; //works

(js, php, python)

2. Strongly typed

String myName = 'Arun';

myName = true; //error -- compilation error

(C, C++, C#, java)

3. Optionally-typed language

----

TypeScript Code Example

----

var myName: string = 'John';

myName = 100; //error

var x: number = 10;

x = true; // error

var isLoggedIn: boolean = true;

isLoggedIn = false; //valid

isLoggedIn = 120; //error

var u: undefined;

var n: null = null;

var skillList1: string[] = ['html', 'css', 'js'];

var skillList2: Array<string> = ['html', 'css', 'js'];

-----

var everything: any = 10; //valid

everything = true; //valid

everything = 'Steve'; //valid

var randomList: any[] = [ true, 10, 'Wow', {}, [] ]; //valid

------------------

Types of Datatyping

1. Explicit Datatyping

var myName: string = 'John';

myName = 100; //error

2. Implitic Datatyping

var z = 10; // valid -- auto accommodates to the data type

z = 100; // valid

z = true; // error

TS => (TSC) => JS

===

Class in TS

===

class Car {

// public var and private vars go here

maker: string;

model: string;

private fuel: string = 'Petrol';

isLoggedin = false;

constructor( \_maker: string, \_model: string ){

this.maker = \_maker;

this.model = \_model;

// private vars -- var, let, const

let x = 10;

var z = 10;

}

drive(){ // public method

console.log(`I am Driving ${this.maker} ${this.model}`);

// if user loggedin

var getData = function(){

}

// if user not loggedin

var accessDenied = function(){

}

if(this.isLoggedin){

getData();

}

else{

accessDenied()

}

}

// private method

private getFuel(){

}

}

const myCar: Car = new Car('Honda', 'City');

myCar.drive(); //I am Driving Honda City

==========

NodeJS Ecosystem

====

NPM (Node Package Manager)

===

https://www.npmjs.com/

Dependencies

---

1. Project Dependencies

Packages that are required for the project to run in deployment

(examples: bootstrap, jquery, dropzone, datatables )

npm i bootstrap === npm install bootstrap === npm i --save bootstrap ===

npm i bootstrap -S

2. Development Dependencies

Packages that are required for us to do the development

(examples: Typescript, webpack )

npm i typescript --save-dev === npm i typescript -D

Commands

---

npm uninstall jquery

npm update bootstrap

npm uninstall typescript --save-dev

------

npm i @angular/cli -g == npm i @angular/cli --global

// todos- Sep 1, 2020

1. npm prune

2. Learn about various Package version specifications

3. Where will the npm global packages be in Windows?

4. learn yarn package manager

========

Dev machine setup

=====

npm uninstall @angular/cli -g (if the cli installed, already)

npm i @angular/cli -g

=======

Todo: Sep 2, 2020

====

1. earn about starting the server in different port

2. conditional content projection in angular 4,5,6,7,8,9

3. integrate bootstrap with angular.json (refer: https://www.smashingmagazine.com/2019/02/angular-application-bootstrap/)

ng g c home -is

ng g c concepts -is

ng g c contacts -is

ng g c about -is

Routing

-----

Characteristics of SPA

-----

1. Page should not refresh

2. URL should change

3. History should be maintained

4. Keep header and footer intact

5. In between header and footer, show contextual updates about the page

6. Active menu should be set

7. Page title should be changed

Step 0:

----

Identify the URL's

localhost:4200/ - Home

localhost:4200/concepts - Concepts

localhost:4200/contacts - Contacts

localhost:4200/about - About

Step 1:

----

ng g c home -is

ng g c concepts -is

ng g c contacts -is

ng g c about -is

Step 2

---

Have the href in menu.comp.html

Step 3:

-----

Setup routing configuration in app-routing.module.ts

Step 4:

---

Check the app. URL will be stable. Now, load the component in b/w header and footer using

<router-outlet></router-outlet>

Step 5:

----

Check the app. Page will refresh. Let's fix it.

in menu.comp.html replace href with routerLink

Step 6:

----

Let's work on active menu

in menu.comp.html,

all li's should have routerLinkActive='active'

Step 7:

----

Home menu is always active. Let's fix it.

in first li add the following

[routerLinkActiveOptions]="{exact: true}"

=========

Data Binding

-----

Types of Data Binding

1. String Interpolation

TS => HTML

{{ }}

2. PRoperty Binding

TS => HTML

[]

3. Event Binding

TS => HTML

()

4. Two Way Binding

TS <=> HTML

[(ngModel)]

Cross Component Communication

---------------

1. Parent Comp to Child Comp Communication

Custom Property Binding

2. Child Comp to Parent Comp Communication

1. Custom Event Binding

2. @ViewChild() -- todo

also explore @ViewChildren() -- todo

3. Any to Any Comp Communication

Prereq: Services, RxJS/Observables, Subscription

===============

Directives

===

<pre>

Directives

-----

Spl instructions to the DOM

1. Attribute Directives

routerLink, routerLinkActive, ngClass, ngStyle,

2. Structural Directives

impact will be on html

1. \*ngIf

2. \*ngFor

3. ngSwitch

</pre>

CRUD App

----

localhost:4200/contacts/add --- add a contact - AddContact

localhost:4200/contacts --- list all contacts - Contacts

localhost:4200/contacts/1 --- contact details - ContactDetails

localhost:4200/contacts/1 --- update contact details

localhost:4200/contacts/1 --- delete contact

=======

Forms in Angular

----------------

1. Template Driven Forms

html

easiest version of ng form

learning curve is not a complicated one

adv

---

easiest

Disadv

---

1. not suitable for complex validation

2. not suitable for Unit Testing

2. Reactive Forms (Recommended)

html + TS

easy

learning curve is a little bigger one

Adv

---

1. suitable for complex validation

2. suitable for Unit Testing

Disadv

----

1. will take sometime

===========

Service

---

Intermediate layer -- connects comp with the rest api

Building block in Angular

ng g s contacts/services/contact

====

URL

---

https://www.example.com/

https://www.example.com?ref=amazon&campaign=sep2020 - query params

https://www.example.com/contacts - no url params, no query params

https://www.example.com/contacts?status=active - no url params, query param

https://www.example.com/contacts/1 - URL Param (id)

https://www.example.com/contacts/1000?status=active - URL Param (id) and query param

https://www.example.com/contacts/1000/edit - URL Param (id)

https://www.example.com/contacts/1000/delete - URL Param (id)

https://www.amazon.in

https://www.amazon.in/electronics/ - listpage URL param(category)

https://www.amazon.in/electronics/mobiles - listpage URL params(category, subcat)

https://www.amazon.in/electronics/mobiles/samsung - listpage URL params(category, subcat, maker)

https://www.amazon.in/electronics/mobiles/samsung/galaxy - listpage URL param(category, subcat, maker, model)

https://www.amazon.in/electronics/mobiles/samsung/galaxy/45q34523453 --- pdt details URL param(category, subcat, maker, model, pdtId)

https://www.amazon.in/furnitures/sofa/fdbdsfgdf/dsfggf/34646436545 --- pdt details - URL param(category, subcat, maker, model, pdtId)

==========

Possible Project Structure

----

src/

app/

shared/

components/

services/

guards/

pipes/

contacts/

components/

services/

directives/

....

auth/

components/

login/

signup/

services/

==============

Building blocks of Angular

===

Components

Directives

Services

Interfaces

Classes

Models

Guards

Pipes

Modules

======

=====

Pipes

Utilities

Transform info one format to another

lowercase to uppercase

timestamp to human readable date

========

For running any angular project in our machine

step 1. download zipped folder from GitHub

step 2. npm i // install npm

============

for building project

----------

ng build --prod --baseHref=lnt-ph-book-sep2020

deployment

-------------

after building the project, a new folder is created automatically Under the DIST folder

from there open index.html and open with GoLive extension