**Angular**

* Version
  + AngularJs
    - Deprecated
  + Angular 2
    - Angular 3
    - Angular 4
    - Angular 5
    - Angular 6
* Framework for developing Single Page Application
  + SPA contains only one page (index.html)
  + The page changes the elements (contents) dynamically
* Pre-requisite
  + Can use JS or TS (TS is proffered)
  + ng-cli
    - cli: Command Line Interface
    - Tool(s) to create/manage the angular application
    - Install using npm

sudo npm install –g @angular/cli

* + - Commands
      * **new**: used to create a new application
        + ng new <app name>
      * **serve**: used to run the angular application
        + ng serve
      * **generate**:
        + used to generate component/service/pipe/directive
        + ng generate <type> <name>
        + ng generate component my
        + ng generate service my
        + ng generate pipe my
        + ng generate directive my

**Angular application structure**

* **e2e**: end to end testing (test cases)
* **node\_modules**:
  + contains the dependency modules
  + @angular/core:
  + @angular/common:
  + @angular/forms:
  + @angular/http:
  + @angular/animations
  + @angular/router:
* **src**:
  + contains application’s pages/components/assets/configurations
  + **app**:
    - contains application source code
    - app.module.ts:
      * contains the configuration about the module(s)
      * declarations:
      * imports:
      * providers:
      * bootstrap:
  + **assets**:
    - contains application assets
    - e.g. images/audio/video etc
  + **environments**:
    - contains the configurations about the environments (machine)
    - e.g.
      * environment.ts: development
      * environment.prod.ts: production
      * environment.test.ts: testing
  + **favion.ico:** image which is displayed on the browser’s tab
  + **index.html:** the page which is loaded at the beginning
  + **main.ts**: loads the angular application
  + **polyfills.ts**: contains the implementation of the features which are introduced in ES5/ES6 but are missing in the standard JS (browser does not understand these features)
  + **styles.css:** contains the global styles
  + **test.ts**: contains the test cases
  + **tsconfig.app.json**: used to specify application-specific ts configuration
  + **tsconfig.spec.json**: used to specify application-specific ts configuration related to test cases
* configuration files
  + **angular.json**: contains the ng-cli + application configuration
  + **.editorconfig**: contains configurations related to editor
  + **.gitignore**: used to ignore files at the time checking – in the files in git repository
  + **karma.conf.js**: used to configure the karma testing tool to run the test cases
  + **package.json**:
    - used to configure the application
    - contains dependencies
      * used to run the application
    - contains dev-dependencies
      * used to run the application on development environment
  + **README.md**
    - Read me file which contains
      * steps to install/configure application
      * steps to deploy the application
  + **tsconfig.json**: contains the configuration related to typescript
  + **tslint.json**: contains the ts-linter configuration

**Angular Fundamentals**

* Module
  + Collection of classes/enums/functions
  + Angular uses **NgModule** to create a module
  + Every application must contain at least one module (AppModule)
  + E.g.

import { NgModule } from ‘@agular/core’;

@NgModule({

declarations: [],

imports: [],

providers: [],

bootstrap: []

})

export class AppModule {

}

* + where:
    - declarations:
      * contains the list of classes which part of this module
      * it may contain
        + Component class
        + Pipe class
        + Directive class
    - imports:
      * Contains list of dependency module
    - providers:
      * contains list of classes which provide services
      * contains list of services (classes)
    - bootstrap:
      * list of classes which will be loaded by default
* Component
  + Re-usable entity
  + Encapsulates
    - Data
      * To render the data, class uses data members
    - View:
      * Uses the template and CSS to render a view
    - Logic
  + Angular uses @Component to create a new component
  + E.g.

import {Component} from ‘@angular/core’;

@Component({

selector: ‘<tag name>’,

template: ‘<html code>’

})

export class MyComponent {

}

* + Where
    - selector: tag which is used as a container
    - template: used to define the html code (view)
    - styles:
    - templateUrls:
    - styleUrls:
  + To add a custom component:
    - Create a component class inside src/app (my.component.ts)
    - Add meta-data related to Component

import { Component } from '@angular/core';

@Component({

selector: 'my-component',

template: '<h1>My Component</h1>'

})

export class MyComponent {

}

* + - Declare the component class in AppModule’s declarations (src/app/app.module.ts)
* Metadata
  + Extra information added to the class

**Binding**

* One way binding
  + **Interpolation**
    - Use {{}} to insert the value of a variable
    - E.g.

// display the value of personName inside the div

<div>{{personName}}</div>

* + **Attribute binding**
    - Use [] to get the value of a variable and attach it to one or more attributes
    - E.g.

**// color = ‘red’;**

<div [style.color]=”color”>sample</div>

* + **Class Binding**
    - Use [] to achieve the class binding
    - used to attach/add a class based on a condition
    - the class gets attached/added to the html element only when the condition evaluates to true
    - E.g.

// css: .selected {…}

// ts: isSelected = true;

<div [class.selected]=”isSelected == true”>text</div>

* + **Event Binding**
    - Use () to achieve the even binding
    - Used to bind the event to a class method
    - E.g.

// ts: myMethod() { alert(‘my method’); }

<button (click)=”myMethod()”>click here</button>

* Two way binding
  + used to get the input from user and set it to the class property
  + import FormsModule in app.module.ts
    - import { FormsModule } from '@angular/forms';
    - inside NgModule’s imports section add the name of the module
  + e.g.

// ts: name

<input type=”text” [(ngModel)]=”name”>

**Add bootstrap to angular project**

* download the bootstrap
* load bootstrap css
  + use @import to load bootstrap css
  + open styles.css and add
    - @import ‘~bootstrap/dist/css/bootstrap.css’
    - where ~ represents node\_modules
* Steps
  + Copy the bootstrap.min.css to src/assets/
  + Copy the fonts directory to src/
  + Open src/styles.css and add
    - @import ‘./assets/bootstrap.min.css’;

**Creating Re-usable Components**

* every component can receive input from its consumer
  + to receive an input from consumer
    - import Input from @angular/core
    - call @Input function on the class members
    - user the class members
* every component may emit an event as its output
  + to emit an event
    - import Output and EventEmitter from @angular/core
    - call @Output() on the event emitter object
    - call emit() on event emitter when the event is getting occurred

export class FavoriteComponent {

**@Input()** isFavorite:boolean = false;

**@Output** favoriteToggled = new EventEmitter();

.

.

toggle() {

// occur an event

this.favoriteToggled.emit();

}

}

**Directive**

* Instruction(s) given to angular to perform some action(s)
* Types
  + \*ngFor
    - used to iterate over a collection
    - exposes built-in variables
      * index: position
      * first: true if the element is at the first position
      * last: true if the element is at the last position
      * even: true if the position is even position
      * odd: true if the position is odd position
    - syntax:

<div \*ngFor=”let <tmp> of <collection>”></div>

* + - E.g.

<div \*ngFor=”let contact of contacts”></div>

* + \*ngIf
    - used to add a conditional logic
    - Syntax:

<div \*ngIf=”<condition>”></div>

* + - E.g.

<div \*ngIf=”contacts.length == 0”>no contacts</div>

* + - * the div will be added to the page when the contacts array is empty
      * otherwise the div element will NOT be added
  + \*ngSwitchCase
    - used to select an element based on condition
    - Syntax:

<div [ngSwitch]=”<variable>”>

<div \*ngSwitchCase=”<value>”> contents</div>

</div>

* + - E.g.

selectedValue = 1;

<div [ngSwitch]=”selectedValue”>

<div \*ngSwitchCase=”1”> value 1</div>

<div \*ngSwitchCase=”2”> > value 2</div>

<div \*ngSwitchCase=”3”> > value 3</div>

</div>

**Add Http module**

* Import HttpModule in app.module.ts

import { HttpModule } from '@angular/http';

* Add HttpModule in imports list

Imports: [

BrowserModule,

HttpModule

]

* import the Http class from http module

import { Http } from ‘@angular/http’;

**Angular Routing**

* to launch a component by using route [url parameter]
* Steps
  + in app.module.ts

import { RouterModule } from '@angular/router';

* + configure the routes

RouterModule.forRoot([

{ path: ‘’, component: <component name>}

])

where

path: the identifier used in the url to launch the component

component: the component to be launched

* + add the router-outlet in the app.component.html

<router-outlet></router-outlet>

Visual Studio Code plugins

* <https://marketplace.visualstudio.com/items?itemName=steoates.autoimport>
* <https://marketplace.visualstudio.com/items?itemName=NuclleaR.vscode-extension-auto-import>
* <https://marketplace.visualstudio.com/items?itemName=johnpapa.Angular2>
* <https://marketplace.visualstudio.com/items?itemName=danwahlin.angular2-snippets>
* <https://marketplace.visualstudio.com/items?itemName=formulahendry.auto-close-tag>
* <https://marketplace.visualstudio.com/items?itemName=sidthesloth.html5-boilerplate>
* <https://marketplace.visualstudio.com/items?itemName=abusaidm.html-snippets>
* <https://marketplace.visualstudio.com/items?itemName=geyao.html-snippets>