# Angular2 Startup

# Angular2 Architecture

* Consists of components
* Components are piece of code block where we may/maynot have templates and/or css
* Components consists of views and templates
* Components can access data using services, and these services are injected into components using injectable keyword
* Ideally Angular2 supports **Uni-Directional** Data Binding i.e. from View to Component or from Component to view
* In some cases we may implement **Two-Way Directional** Data Binding

# Setting Up Developing Environment

* Developed using typescripts, which include class, Decorators and other ECMA6 standards
* Make you have **nodejs/npm** installed and updated to latest version
* Download and install typescript using **npm install –g typescript**
* Download/Clone the boilerplate code with following git repo [**https://github.com/mschwarzmueller/angular-2-beta-boilerplate**](https://github.com/mschwarzmueller/angular-2-beta-boilerplate)
* Then install all the packages from package.json via **npm install**
* Start the server using **npm start**
* In order to compile typescript file to plain old js file run **gulp** in cmd

# Building First Component

* Component are nothing but the advanced Directives/used to extend the functionality
* Components consists of several imp parts i.e. import, export, decorators and so on…
* **Import** : used to import other classes default/user developed functionality
* **Export**: used to export the class to other files/modules
* **Decorators**: attach configurations to class
* Also in order to launch the application we need to add entry of AppComponent into boot.ts file
* ///<reference path="../node\_modules/angular2/typings/browser.d.ts"/>
* import {bootstrap} from 'angular2/platform/browser';
* import {AppComponent} from "./app.component";
* bootstrap(AppComponent);
* And also in index.html file where we need to replace default text (Loading in this case ) with view by using selectors from Component Defn
* <my-app>Loading...</my-app>
* Inside component Decorator it is defined as …
* selector: 'my-app',

# Click Handler

* (click)="onContactClicked()" // inside template
* onContactClicked() { // inside Class
* this.showConatct = true;
* }

# Conditional CSS Class

* If condition satisfies as true then only apply css class on element and this can be achieved by
* [class.clicked]="showConatct"
* And by importing css as
* styleUrls: ['../src/css/app.css']

# ngIf

* <div \*ngIf="showConatct">
* The above will be shown only if showContact == true
* And written as \*ngIf

# Data Binding

* Ideally Angular2 provides uni-directional data bindings
* **() round brackets** signifies data binding from **view to component**
* **[] square brackets** signifies data binding from **Component to view**
* Angular2 also provides Two-Way Data binding in a way as follows
* <input type="text" [(ngModel)]="contact.firstName" />
* **[(ngModel)]=”contact.firstName”** is used to implement Two-Way

# ngFor

* ngFor is written as **\*ngFor=”#key of array”** and then key is used to iterate though array of objects like key.firstName, key.phone and so on…
* <li
* \*ngFor="#contact of contactList"
* (click)="onContactClicked(contact)"
* class="contact-name"
* [class.clicked]="showConatct"
* >
* {{contact.firstName}} {{contact.lastName}}
* </li>

# Creating custom Component

* New .ts file is created such as contact-list.component.ts
* Need to import Component from angular2/core
* Need to add @Component decorator and need to add Class and export the same class as
* export class ContactListComponent {}
* need to add selector as an html selector so that this component can be loaded into another or parent component and that I sto be added in @Component say selector : “contact-list”
* need to add this <contact-list> inside parent or another component and need to load the directive in the another/parent component

# Sending selected data/object from One Component to Another

* need to add configuration parameter as inputs : [‘contact’], inside newly created component
* where as this contact must be same as public contact = {} created inside class body
* in order to send data for selected item from one component to another need to add the selected value selectedContact in this case to [contact] where this contact should be same as that of defined at inputs: [‘contact’]
* <contact [contact]="selectedContact"></contact>
* And inside contact.component.ts we have
* inputs: ['contact']

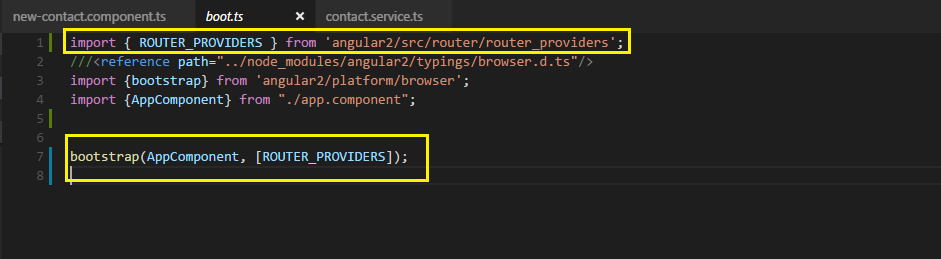
# Services

* Services are used to fetch data from Server and inject in different components
* Created using @Injectable
* For now lets create a mock source of data and then it would be replaced by actual http call
* Create interface as contact.ts where in we create type of object that is contact in this case
* export interface Contact {
* firstName: string,
* lastName: string,
* phone: string,
* email: string
* }
* Created a mock-contact.ts file where in we loaded hardcoded data of contacts that we need to display
* import { Contact } from './contact';
* export const CONTACTS: Contact[] =  [
* { firstName: 'Andy',lastName: 'Stan',phone: '4122139550',email: 'andy@gmail.com'},
* { firstName: 'Bran',lastName: 'Clove',phone: '5122136981',email: 'bran@gmail.com'},
* {firstName: 'Chris', lastName: 'Nolan', phone: '2233556688', email: 'chris@gmail.com' },
* {firstName: 'Dan',  lastName: 'Brown', phone: '7788994455', email: 'dan@gmail.com' },
* ];
* Created a Service now with contact.service.ts
* Inside this we added @Injectable which can be imported from angular2/core lib.
* Also created custom class as ContactService and added method which fetches data/contacts from CONTACTS ir from mock-contact.ts and return using promise object as Promise.resolve()
* import { CONTACTS } from './mock-contact';
* import { Injectable } from 'angular2/core';
* @Injectable()
* export class ContactService {
* getContacts() {
* return Promise.resolve(CONTACTS);
* }
* }
* Now in order to get this service/rather call functions inside service from Component we need to inject in component, and this can be done via adding **provides : [ContactService]** and after that we may need to import the service at top if IDE does not support this
* providers: [ContactService]
* Constructor is added to instantiate the service and private variable of type Service is created
* constructor(private \_contactService: ContactService) {}
* With this \_contactService we call getContacts() from service and this is then assigned to the conatctList[] inside fat arrow functions
* getContactsList() {
* this.\_contactService.getContacts().then(
* (data: Contact) => this.contactList = data
* )
* }
* ngOnInit() : any is added as onload function to load the data on component load
* ngOnInit() {
* this.getContactsList();
* }

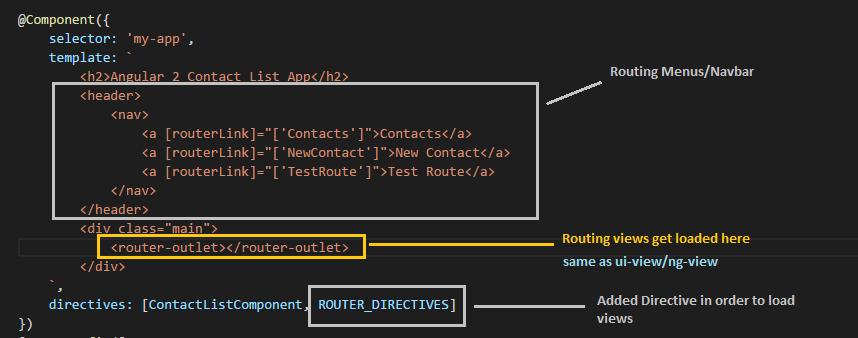
# Routes

* To implement routes need to add <base href=”/”> inside <head> of index.html
* It should be just after opening <head>
* Need to add ROUTER\_PROVIDERS on boot.ts in bootstrap(AppComponent, [ROUTER\_PROVIDERS])
* bootstrap(AppComponent, [ROUTER\_PROVIDERS]);
* Once we define that we see that we import ROUTER\_PROVIDERS from angular2 router lib on top as follows

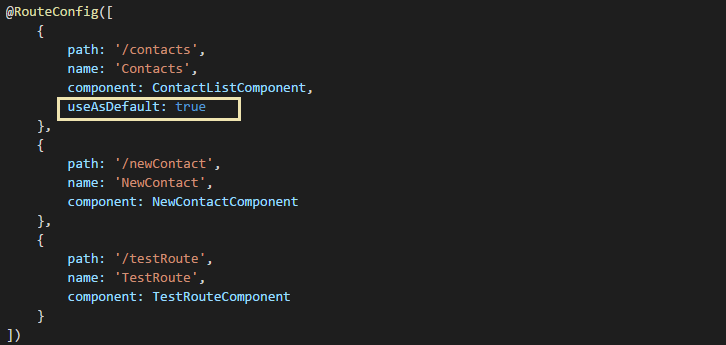
import { ROUTER\_PROVIDERS } from 'angular2/src/router/router\_providers';



* Inside AppComponent we need to add ROUTER\_DIRECTIVES in directives : [] and corresponding import has to done on top from angular2/core lib
* We need to display different menus in order to implement routing and that can be done by adding <header><nav> … </nav></header> section inside AppComponent



* Need to configure the different routes and need to set the Default routes so that when the UI loads we would see first/default route and then we would navigate through different routes and this can be achieved by **RouteConfig**  and its corresponding import is added @top



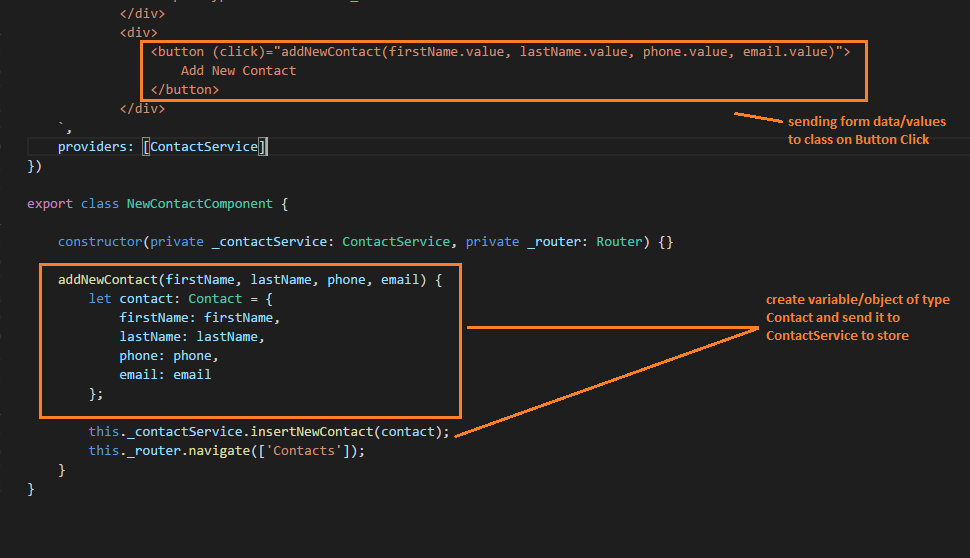
* Also need to make sure all the Components listed above should be in place in order to implement routing successfully

# Inserting Data with Service and More on Routes

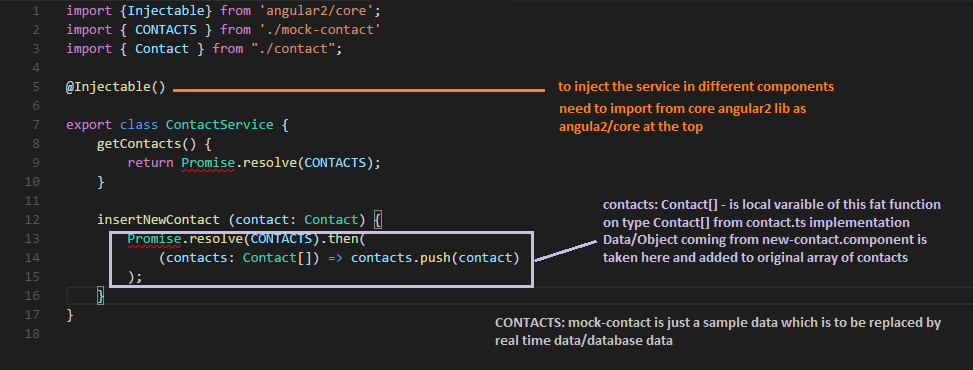
* Create a form/ input placehoders in NewContactComponent Template in the following manner
* For now we have created local variables in each of the fields with #value such as **#firstName** and we can get its value in Component Class as **firstName.value** by passing it though button click event to the function addNewContact() in class defn



* We need to send this data to Service in order to update the original Contact List with this new entry
* To achieve this, we first need to add/import/inject Service to our Component
* Adding **providers: [ContactService]** into Component Definition and then importing same at the top would make ContactService/Service available in Component



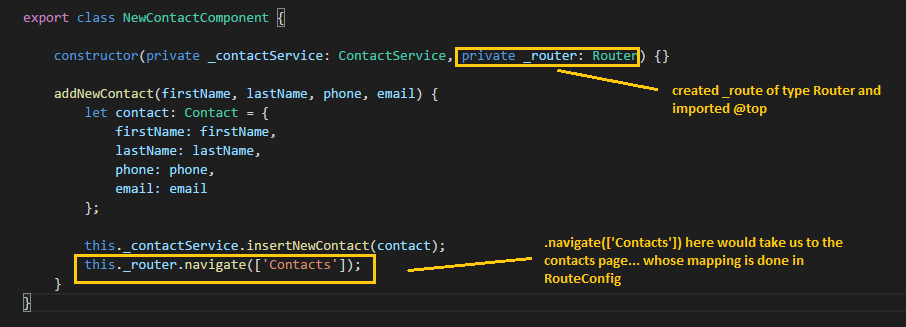
* When we have created let contact: Contact we need to import contact interface
* import { Contact } from './contact';
* then we send this to ContactService/(any other Service) for further processing
* we need to inject Service in order to send data and this can be achieved by creating Constructor and calling private service variable of type Service and we can then access method inside service with that private variable of type Service
* constructor(private \_contactService: ContactService, private \_router: Router) {}
* Consider the Service Implementation as follows



* Promise object used here to resolve/reject the data and send back this updated data to invoking component using Promise.resolve()

# Internal Navigation

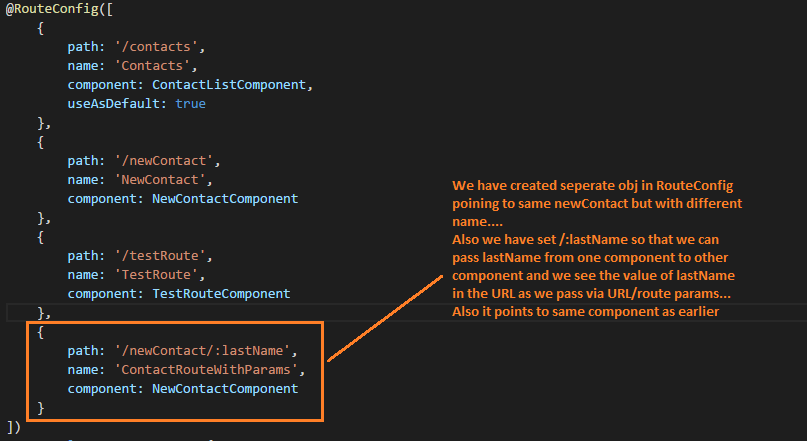
* When we get the updated data we may need to display this data into some other component and update the data there… for this purpose we may need to implement internal routing/navigation
* This can be achieved by creating private variable of type Router and then calling .navigate([‘name of the route where to route from routeConfig’])
* Explained as



# Passing Parameters with routes

* We may need to pass the parameters from one component to other in order to send data to and fro between components and this can be achieved in following way

**1st Method : Creating one more route in RouteConfig**



* We have then added button in Contact.component on-click of which we route to different component
* Need to add constructor to add/instantiate the \_router of Router object by importing from angular2 router
* Using this \_router and calling .navigate method we may implement routing
* In this case we have added additional parameter/object in .navigate([‘name’, **{}**]) where in pass parameter that is to be passed into routes





* In the new-contact.component where we need to fetch this parameter coming via routeparams we add RouteParams to constructor
* Created \_routeParams of type RouteParams and then calling .get(‘lastName’) we can fetch the parameter and then we may set into form
* For this purpose we create some public string lastNameFromParams = “” to this var we would assign the value that we received form routeParams
* public lastNameFromParams = "";
* This can be stated as
* ngOnInit() {
* this.lastNameFromParams = this.\_routeParams.get("lastName");
* }

# Forms (Two Way Data-binding)

* When Angular2 finds <form> tag in DOM, it automatically adds ngForm directive/class to the form and all the fields of form are part of ngControl group.
* Need to create a form as follows
* <form #newContactForm="ngForm" (ngSubmit)="onSubmitNewConatctForm()">
* <div class="form-grp">
* <label for="txt\_firstName">First Name</label>
* <input type="text" id="txt\_firstName"
* ngControl="firstName"
* [(ngModel)]="newContact.firstName"
* required
* />
* </div>
* <div class="form-grp">
* <label for="txt\_lastName">Last Name</label>
* <input type="text" id="txt\_lastName" value="{{lastNameFromParams}}"
* ngControl="lastName"
* [(ngModel)]="newContact.lastName"
* required
* />
* </div>
* <div class="form-grp">
* <label for="txt\_phone">Phone</label>
* <input type="text" id="txt\_phone"
* ngControl="phone"
* [(ngModel)]="newContact.phone"
* required/>
* </div>
* <div class="form-grp">
* <label for="txt\_email">Email</label>
* <input type="text" id="txt\_email"
* ngControl="email"
* [(ngModel)]="newContact.email"
* required/>
* </div>
* <div class="form-grp">
* <!--<button (click)="onCreateNewContactClicked(firstName.value, lastName.value,
* phone.value, email.value)">Create Contact</button>-->
* <button type="submit">Submit Contact</button>
* </div>
* </form>
* Created newContact of type Contact in class and values from form are accessed with Two Way Data-bindings using [(ngModel)] attribute and these values are then sent to the Service to modify the data
* Router is implemented to navigate back to the Contact-list component where we see the new contact added to the previous list of contacts
* export class NewContactComponent implements OnInit {
* public lastNameFromParams = "";
* newContact: Contact;
* constructor(private \_contactService: ContactService, private \_router: Router,
* private \_routeParams: RouteParams) {}
* onCreateNewContactClicked(firstName, lastName, phone, email) {
* let contact: Contact = {
* firstName: firstName, lastName: lastName, phone: phone, email: email
* };
* this.\_contactService.insertContact(contact);
* this.\_router.navigate(['Contacts']);
* }
* onSubmitNewConatctForm() {
* this.\_contactService.insertContact(this.newContact);
* this.\_router.navigate(["Contacts"]);
* }
* ngOnInit() {
* this.lastNameFromParams = this.\_routeParams.get("lastName");
* this.newContact = {
* firstName: '',
* lastName: this.lastNameFromParams || '',
* phone: '',
* email: ''
* };
* }
* }

# Forms with FormBuilder

* <https://www.youtube.com/watch?v=LyRgKQjsGa4&list=PL55RiY5tL51qIb5VW2ywbT12UZeqmzBAu&index=16>

# HTTP GET/POST