ANGULAR

ChandraSekhar(CS) Baratam

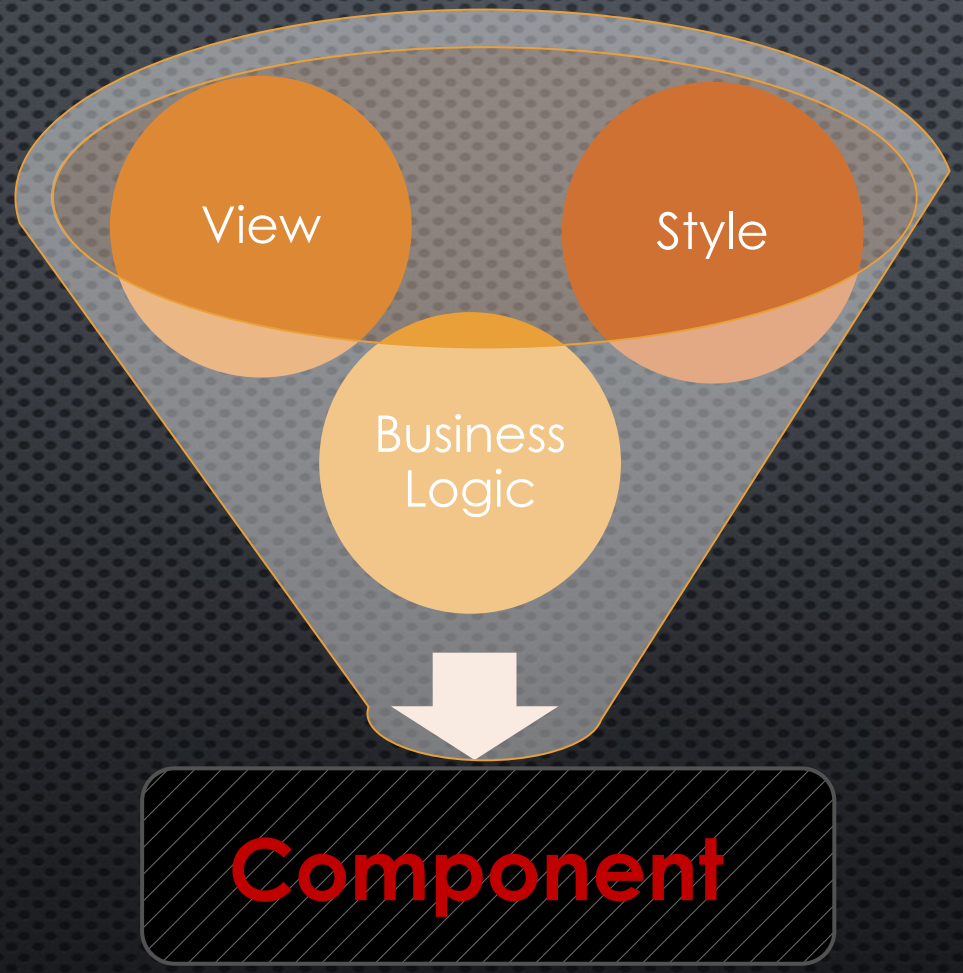
1. Component
2. Data Binding
3. Directives
4. Service
5. Router
6. Pipes
7. Forms & Validation
8. HTTP Service
9. Component Communication
10. CRUD

ChandraSekhar(CS) Baratam

COMPONENTS

ChandraSekhar(CS) Baratam

WHAT IS COMPONENT

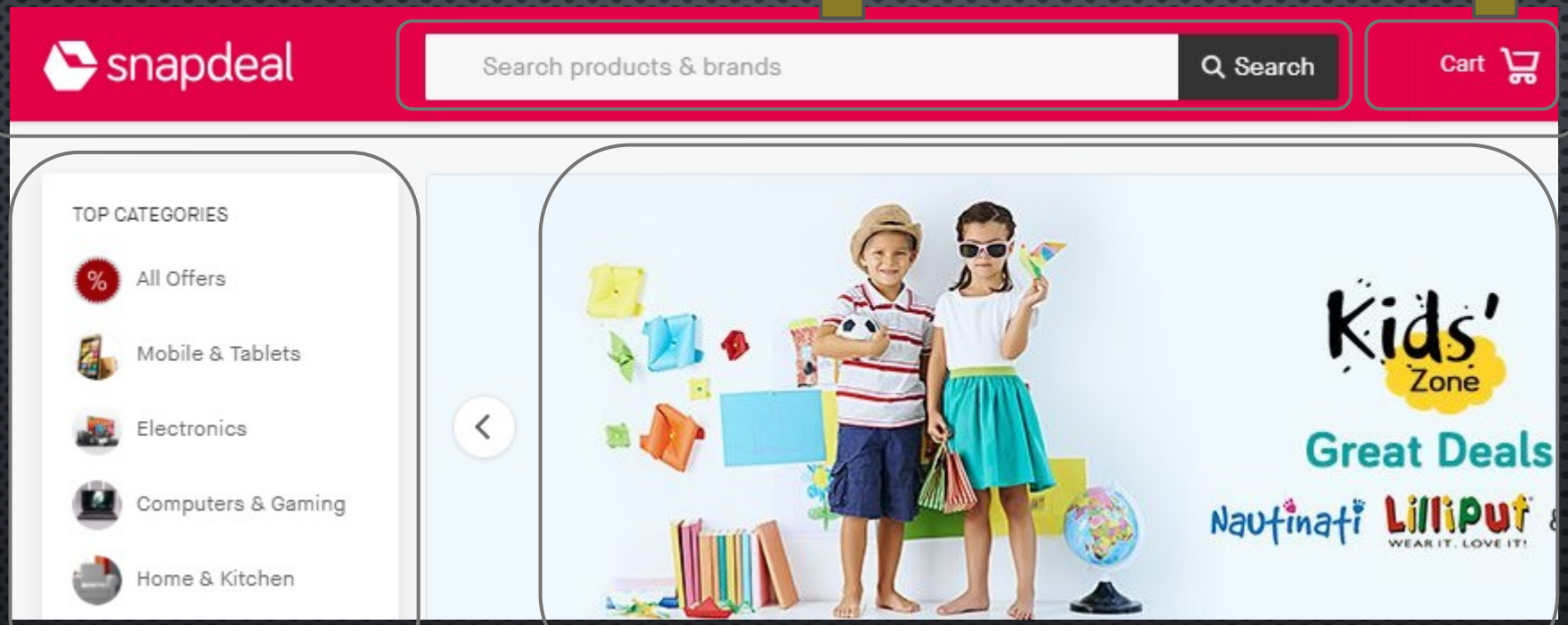


EXAMPLE

Header Component

Search Component

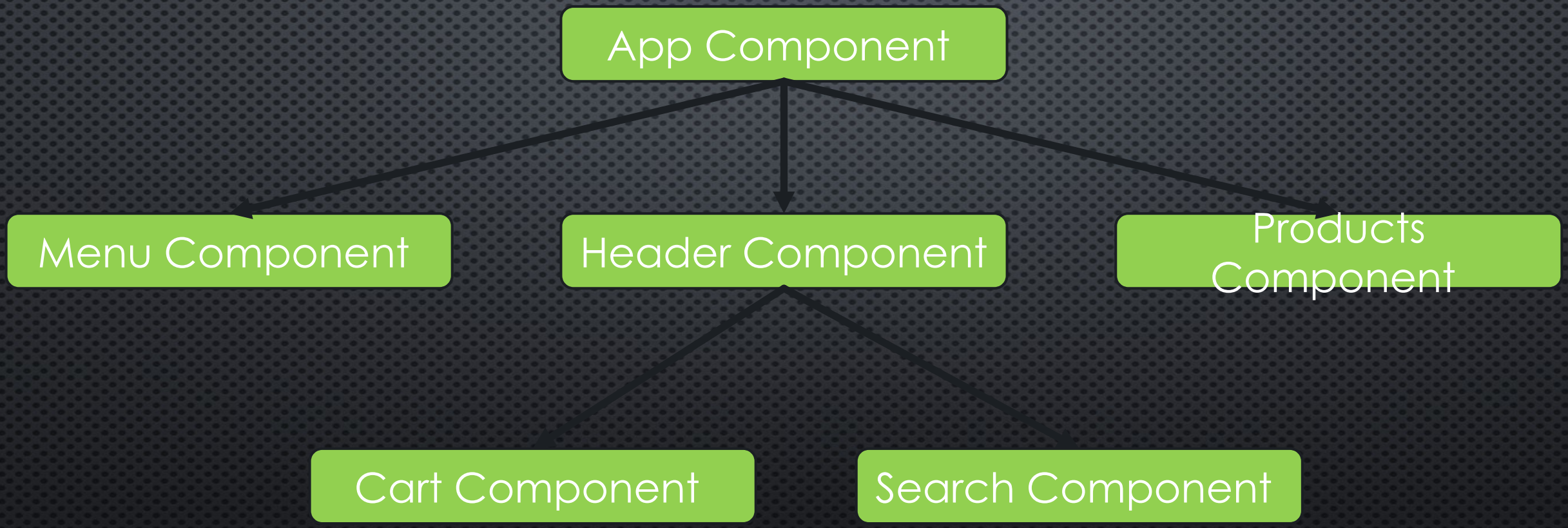
Cart Component



Menu Component

Products Component

COMPONENT HIERARCHY



COMPONENT CREATION

```
export class MyClass
  {

  }
```

+

```
@Component({

})
```

```
@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyClass    {   }
```

HELLO WORLD

ChandraSekhar(CS) Baratam

```html
<html>
<head>
  <meta charset="utf-8">
  <title>AngularWebProject</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Index.html

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

@Component({
  selector: 'app-root',
  templateUrl:
'./app.component.html',
  styleUrls:
['./app.component.css']
})
export class AppComponent {

}
```

App.component.ts

```
<app-my-app></app-my-app>
```

App.component.html

```
import { BrowserModule } from
'@angular/platform-browser';
import { NgModule, Component } from
'@angular/core';

import { AppComponent } from
'./app.component';

import { MyAppComponent } from './my-app/my-app.component';

@NgModule({
  declarations: [
    AppComponent,
    MyAppComponent
  ],
  imports: [
    BrowserModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.module.ts

```
<div>
{{message}}
</div>
```

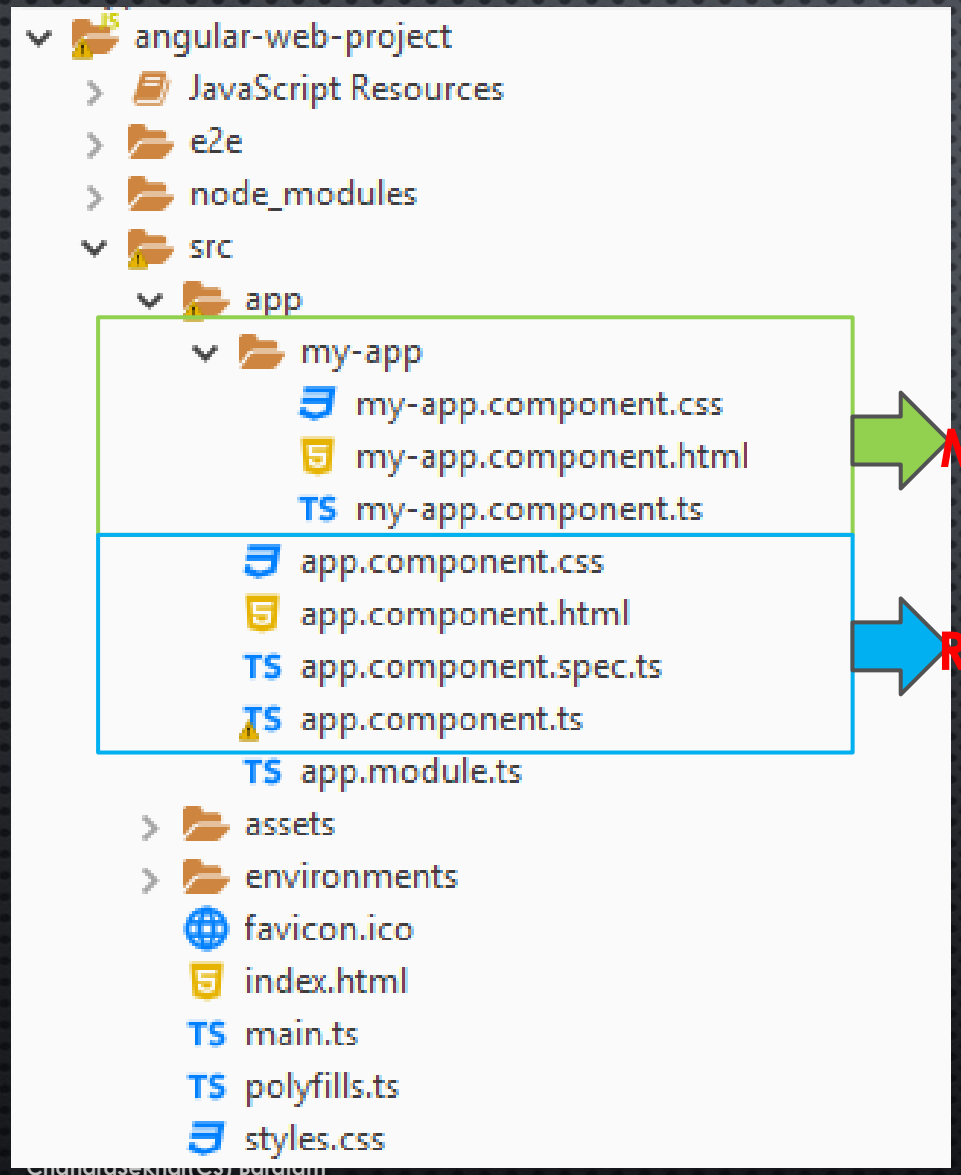
my-app.component.html

```
div {
margin: 0 auto;
border-width: 2px;
border-color: #FF3838;
position: absolute;
top: 141px;
left: 280px;
background: white;
width: 650px;
height: 458px;
box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.7);
font-family: lato;
font-size: 200px;
color: #808000;
border-radius: 10px;
text-align: center;
}
```

my-app.component.css

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

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  message = 'Hello World';
  constructor() {}

}
```

my-app.component.ts



My Component

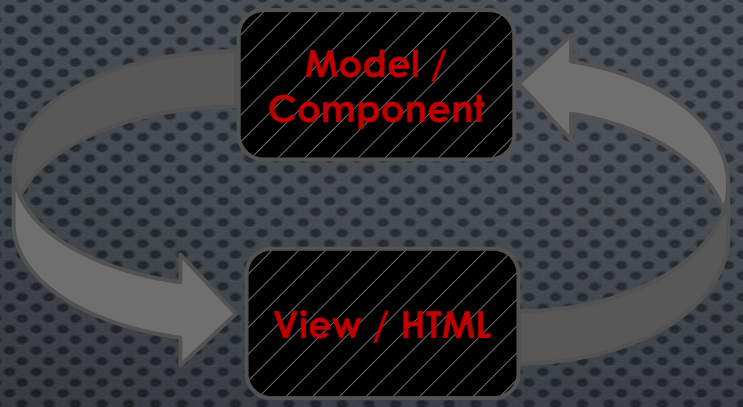
Root Component

Hello World

DATA BINDING

ChandraSekhar(CS) Baratam

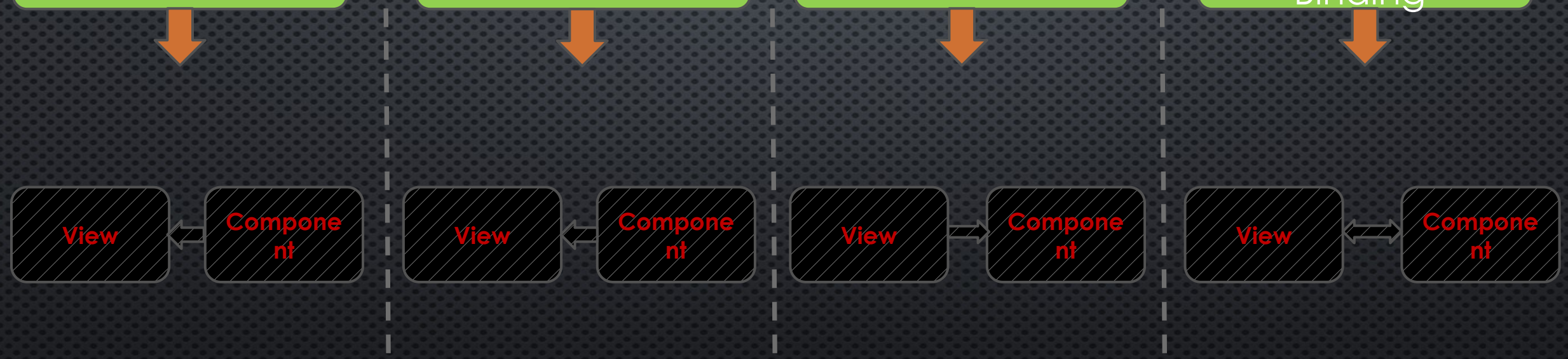
ANGULAR DATA BINDING



1. One Way Binding
2. Two Way Binding

ANGULAR DATA BINDING

1. Interpolation
2. Property Binding
3. Event Binding
4. 2 Way Data Binding

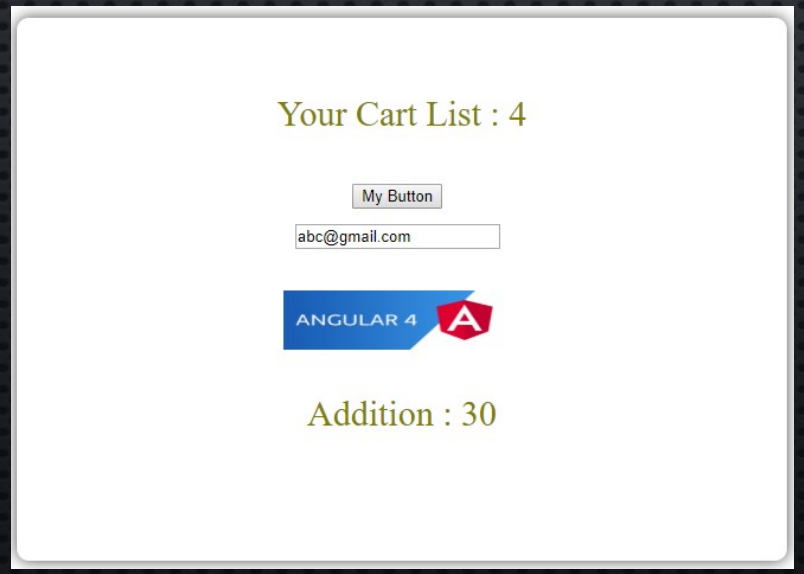


ChandraSekhar(CS) Baratam

1 INTERPOLATION - EXAMPLE

My-app.component.html

```
<div>
<br>
<p>Your Cart List : {{count}}</p>
<input type="submit" value="{{btnText}}"/>
<br>
<input type="text" value="{{text}}"/>
<br>
<br>
<img src="{{myImage}}" width="200"
height="50"/>
<p>Addition : {{getSum(10,20)}}</p>
</div>
```



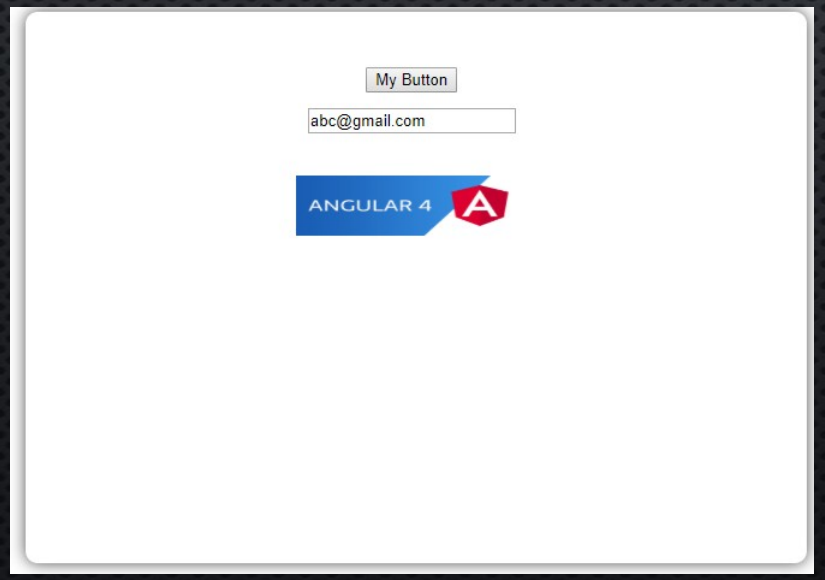
My-app.component.ts

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

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  count: number;
  btnText: string;
  text: string;
  myImage: string;
  constructor() {
    this.count = 4;
    this.btnText = 'My Button';
    this.text = 'abc@gmail.com';
    this.myImage = '../assets/logo.png';
  }
  getSum(a, b) {
    return a + b;
  }
}
```

2 PROPERTY BINDING - EXAMPLE

My-app.component.html

```
<div>
<br>
<input type="submit" [value]="btnText"/>
<br>
<input type="text" [value]="text"/>
<br>
<br>
<img [src]="myImage" width="200"
height="50"/>
</div>
```



My-app.component.ts

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

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  btnText: string;
  text: string;
  myImage: string;
  constructor() {
    this.btnText = 'My Button';
    this.text = 'abc@gmail.com';
    this.myImage = '../assets/logo.png';
  }
}
```

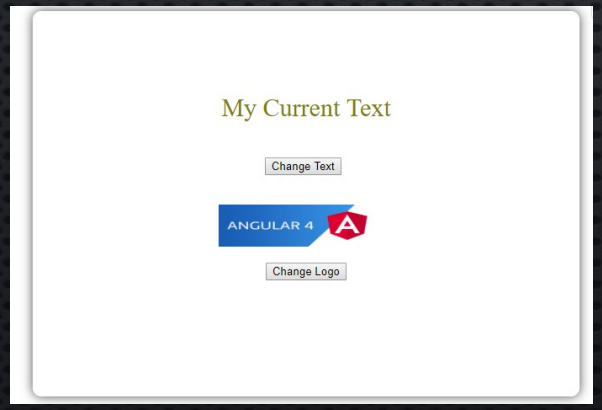
3 EVENT BINDING - EXAMPLE

My-app.component.html

```
<div>
<br>
<br>
<p>{{myText}}</p>
<input type="submit" (click)="changeText()" value="Change Text"/>
<br>
<br>
<br>
<img [src]="myLogo" width="200" height="50"/>
<br>
<input type="submit" (click)="changeLogo()" value="Change Logo"/>
</div>
```

My-app.component.ts

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

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  myText: string;
  myLogo: string;
  constructor() {
    this.myText = 'My Current Text';
    this.myLogo = '../assets/logo1.png';
  }
  changeText() {
    this.myText = 'This is my New Text';
  }
  changeLogo() {
    this.myLogo = '../assets/logo2.png';
  }
}
```



4 2 WAY DATA BINDING - EXAMPLE

My-app.component.html

```
<div>
<br>
<br>
<input type="text" [(ngModel)]="name" placeholder="Enter
Your name"/>
<p>Welcome {{name}}</p>
</div>
```

My-app.component.ts

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

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  name: string;
  constructor() {}
}
```

App.module.ts

```
import {FormsModule} from '@angular/forms';
.
.
.
.

imports: [
    BrowserModule,
    FormsModule
  ],
```

DIRECTIVES

ChandraSekhar(CS) Baratam

ANGULAR DIRECTIVES

1. Components

Directive with a template.
Ex : @Component

2. Structural Directives

Modify the DOM structure.
Add/Remove elements from DOM.

3. Attribute Directives

Change the appearance of elements.

1 STRUCTURAL DIRECTIVES

Structural Directives → **Modify the DOM structure. Add/Remove elements from DOM.**

1. ngIf
2. ngFor
3. ngSwitch

STRUCTURAL DIRECTIVES - EXAMPLE

My-app.component.html

```
<div class="myDiv">
<button (click)="myFunction1()">Click Me</button>
<div *ngIf="flag">
<ul>
<li *ngFor="let vehicle of vehicles "
(click)="myFunction2(vehicle)">{{vehicle}}</li>
</ul>
</div>
<div [ngSwitch]="selectedVehicle">
<font color="green" *ngSwitchCase="'Two Wheeler'">Bike</font>
<font color="green" *ngSwitchCase="'Three Wheeler'">Auto</font>
<font color="green" *ngSwitchCase="'Four Wheeler'">Car</font>
<font color="green" *ngSwitchDefault>Invalid</font>
</div>
</div>
```

My-app.component.ts

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

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  flag: boolean;
  vehicles: string[];
  selectedVehicle: string;
  constructor() {
    this.flag = false;
    this.vehicles = ['Two Wheeler', 'Three
Wheeler', 'Four Wheeler'];
  }
  myFunction1() {
    this.flag = !this.flag;
  }
  myFunction2(vehicle) {
    this.selectedVehicle = vehicle;
  }
}
```

2 ATTRIBUTE DIRECTIVES

Attribute Directives → Change the appearance of elements.

1. ngStyle
2. ngClass
3. ngModel

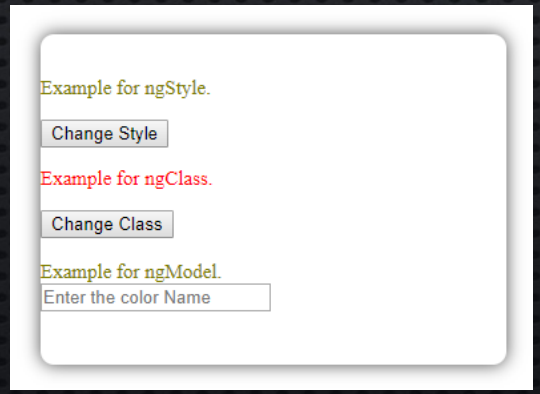
ATTRIBUTE DIRECTIVES - EXAMPLE

My-app.component.html

```
<div class="myDiv">
<br>
<p [ngStyle]="style">Example for ngStyle.</p>
<button (click)="myFunction1()">Change Style</button>
<br>
<p [ngClass]="currentCSSClass">Example for ngClass.</p>
<button (click)="myFunction2()">Change Class</button>
<br>
<div [style.color]="inputColor">
Example for ngModel.<br>
<input type="text" [(ngModel)]="inputColor" placeholder="Enter
the color Name"/>
</div>
</div>
```

My-app.component.ts

```
import {Component} from '@angular/core';
@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  style: {};
  currentCSSClass: string;
  inputColor: string
  constructor() {
    this.currentCSSClass = 'myClass1';
  }
  myFunction1() {
    this.style = {
      'font-size': '15px',  'font-style': 'bold',  'color': '#' +
this.generateColor()
    };
  }
  generateColor() {
    let temp: number = Math.floor(Math.random() * 1000000);
    return temp;
  }
  myFunction2() {
    if (this.currentCSSClass === 'myClass1') {
      this.currentCSSClass = 'myClass2';
    }
    else {
      this.currentCSSClass = 'myClass1';
    }
  }
}
```



My-app.component.css

```
.
.
.
.myClass1{
color: red;
}
.myClass2{
color: green;
}
```

CUSTOM DIRECTIVES

1. Create a Type Script file using @Directives.
2. Register in app.module.ts file.
3. Use the created directives in HTML elements.

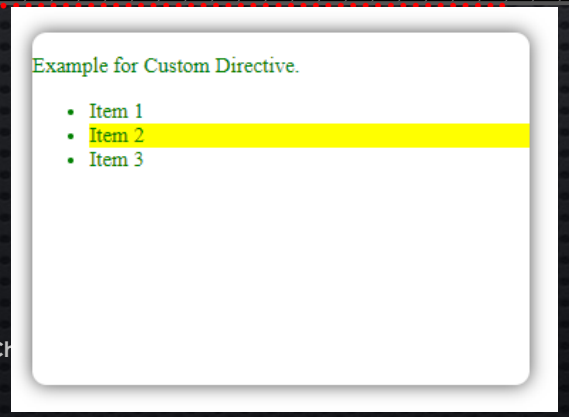
CUSTOM DIRECTIVES - EXAMPLE

My-app.component.html

```
<div class="myDiv">
<p myTextColor>Example for Custom Directive.</p>
<ul>
<li *ngFor="let i of items" myTextColor
bgColor="yellow">{{i}}</li>
</ul>
</div>
```

App.module.ts

```
import { MyCustomDirective } from './custom-directives/my-
bgcolor.directive';
@NgModule({
 declarations: [
   AppComponent,
   MyAppComponent,
   MyCustomDirective
 ],
```



My-bgcolor.directive.ts

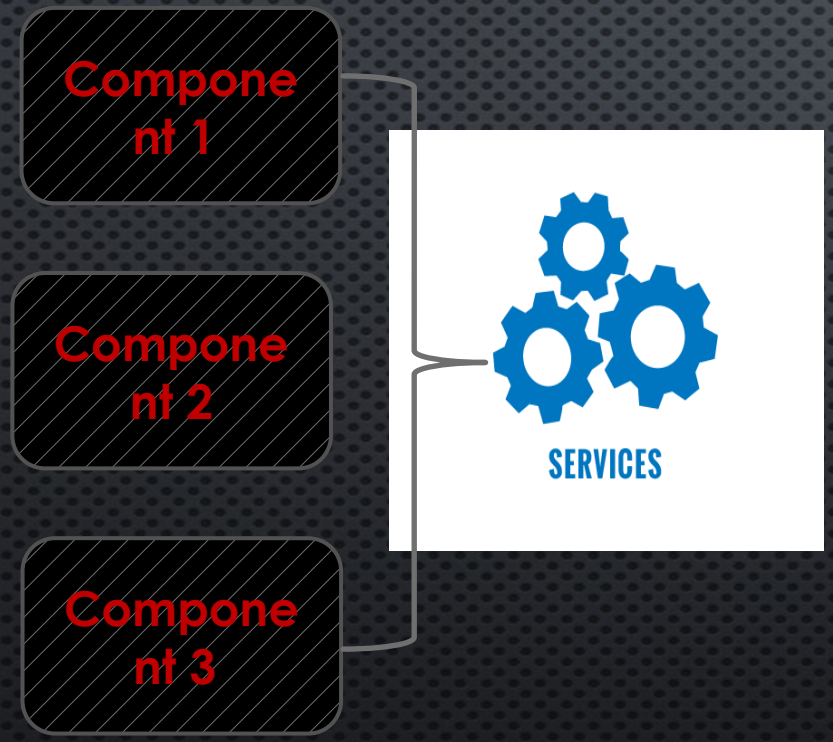
```
import {Directive, ElementRef, Input, HostListener} from
'@angular/core';
@Directive({
  selector: '[myTextColor]'
})
export class MyCustomDirective {
  @Input()
  bgColor: string;
  constructor(private e: ElementRef) {
    e.nativeElement.style.color = 'green';
  }

  @HostListener('mouseenter') onmouseenter() {
    this.e.nativeElement.style.backgroundColor =
this.bgColor;
  }

  @HostListener('mouseleave') onmouseleave() {
    this.e.nativeElement.style.backgroundColor = 'white';
  }
}
```

SERVICE

ChandraSekhar(CS) Baratam

WHAT IS SERVICE

Reusable code that can be accessed from multiple components



SERVICE – EXAMPLE 1

My-app.component.html

```
<div class="myDiv">
<br><br><br>
<div class="menu">
<p>Food Menu</p>
<ul>
<li *ngFor="let food of foodItems">{{food}}</li>
</ul>
</div>
</div>
```

Food.service.ts

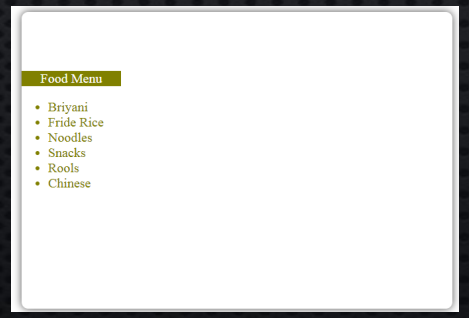
```
import {Injectable} from '@angular/core';
@Injectable()
export class FoodService {
  foods: string[];
  constructor() {
    this.foods = ['Briyani', 'Fride Rice', 'Noodles', 'Snacks',
'Rools', 'Chinese'];
  }
  getFoodItems() {
    return this.foods;
  }
}
```

My-app.component.ts

```
import {FoodService} from '../my-services/food.service';
import {Component} from '@angular/core';

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  foodItems: string[];
  constructor(private service: FoodService) {
    this.foodItems = service.getFoodItems();
  }
}
```

App.module.ts

```
import { FoodService } from
'./my-services/food.service';
providers: [FoodService],
```



SERVICE – EXAMPLE 2

My-app.component.html

```
<div class="myDiv">
<br><br><br>
<div class="menu">
<p>Food Menu</p>
<ul>
<li *ngFor="let food of foodItems" myTextColor
bgColor="Lavender"
(click)="myFunction(food)">{{food}}</li>
</ul>
</div>
<div class="items" *ngIf="flag">
<p align="center">{{selectedFood}} List</p>
<hr/>
<table width="100%"><tr>
<td *ngFor="let arr of selectedFoodList">
<div *ngFor="let f of arr">
<img src={{f}} alt="{{f}}"/>
</div>
</td>
</tr></table>
</div>
</div>
```

My-app.component.ts

```
import {FoodService} from
'../my-services/food.service';
import {Component} from '@angular/core';

@Component({
  selector: 'app-my-app',
  templateUrl: './my-app.component.html',
  styleUrls: ['./my-app.component.css']
})
export class MyAppComponent {
  foodItems: string[];
  flag: boolean;
  selectedFood: string;
  selectedFoodList: string[][];
  constructor(private service: FoodService) {
    this.flag = false;
    this.foodItems = service.getFoodItems();
  }
  myFunction(foodName) {
    this.flag = true;
    this.selectedFood = foodName;
    this.selectedFoodList =
this.service.getSelectedFoodItems(foodName);
  }
}
```

SERVICE – EXAMPLE 2 …

My-app.component.css

```
.myDiv {
margin: 0 auto;
border-width: 2px;
border-color: #FF3838;
position: absolute;
top: 141px; left: 280px; background: white;
width: 650px;
height: 448px;
box-shadow: 0px 0px 20px rgba(0, 0, 0, 0.7);
font-family: lato;
font-size: 20px;
color: #808000;
border-radius: 10px;
}

.menu {   width: 150px;  height: 100px;  }

.items {
position: absolute;
background-color: Lavender;
top: 132px;  left: 150px; width: 450px;
height: 300px;
}
```

costom-directive/My-bgcolor.directives.ts

```
import {Directive, ElementRef, Input, HostListener} from
'@angular/core';
@Directive({
  selector: '[myTextColor]'
})
export class MyCustomDirective {
  @Input()
  bgColor: string;
  constructor(private e: ElementRef) {
    }

  @HostListener('mouseenter') onmouseenter() {
    this.e.nativeElement.style.backgroundColor =
this.bgColor;
    this.e.nativeElement.style.cursor = 'pointer';
  }

  @HostListener('mouseleave') onmouseleave() {
    this.e.nativeElement.style.backgroundColor =
'white';
  }
}
```

SERVICE – EXAMPLE 2 …

food.services.ts

```
import {BriyaniService} from './briyani.service';
import { NoodleService } from './noodle.service';
import {Injectable} from '@angular/core';
@Injectable()
export class FoodService {
  foods: string[];
  selectedFoodList: string[][];
  constructor(private briyaniservice: BriyaniService,
private noodleservice: NoodleService) {
    this.foods = ['Briyani', 'Fride Rice', 'Noodles', 'Snacks',
'Rools', 'Chinese'];
  }
  getFoodItems() {
    return this.foods;
  }
  getSelectedFoodItems(foodName) {
    if (foodName === 'Briyani') {
      this.selectedFoodList =
this.briyaniservice.getBriyaniList();
    }
    if (foodName === 'Noodles') {
      this.selectedFoodList =
this.noodleservice.getNoodlesList();
    }
    return this.selectedFoodList;
  }  }
```

Briyani.services.ts

```
import {Injectable} from '@angular/core';
@Injectable()
export class BriyaniService {
  briyaniList: string[][];
  constructor() {
    this.briyaniList = [
      ['Chicken Briyani', '../assets/briyani.png', 'Rs : 120'],
      ['Mutton Briyani', '../assets/briyani.png', 'Rs : 140'],
      ['Fish Briyani', '../assets/briyani.png', 'Rs : 160']
    ];
  }
  getBriyaniList() {
    return this.briyaniList;
  }  }
```

SERVICE – EXAMPLE 2 ...

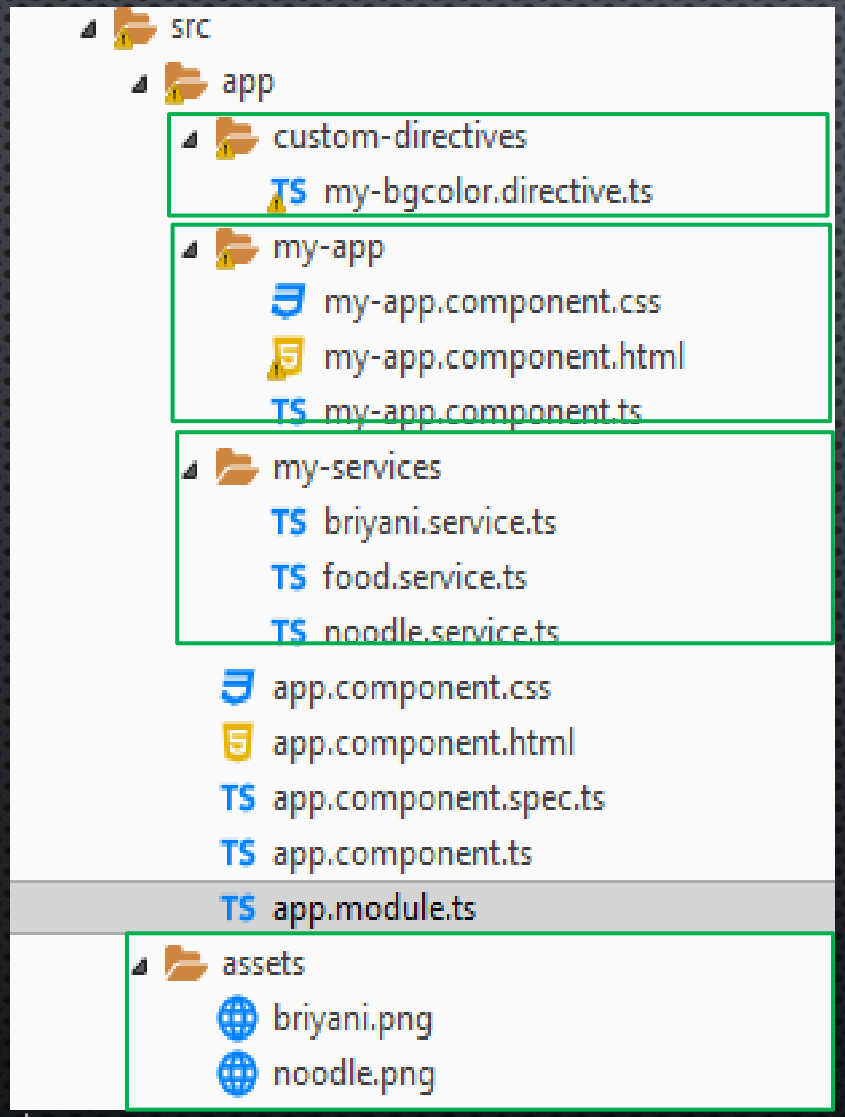
noodle.services.ts

```
import {Injectable} from '@angular/core';

@Injectable()
export class NoodleService {
  noodlesList: string[][];
  constructor() {
    this.noodlesList = [
      ['Chicken Noodle', '../assets/noodle.png', 'Rs : 110'],
      ['Mutton Noodle', '../assets/noodle.png', 'Rs : 130'],
      ['Fish Noodle', '../assets/noodle.png', 'Rs : 140']];
  }
  getNoodlesList() {
    return this.noodlesList;
  }
}
```

app.module.ts

```
import {BrowserModule} from '@angular/platform-
browser';
import {NgModule} from '@angular/core';
import {FormsModule} from '@angular/forms';
import {AppComponent} from './app.component';
import {MyCustomDirective} from './custom-
directives/my-bgcolor.directive';
import {MyAppComponent} from './my-app/my-
app.component';
import {BriyaniService} from
'./my-services/briyani.service';
import {FoodService} from
'./my-services/food.service';
import { NoodleService } from
'./my-services/noodle.service';
@NgModule({
  declarations: [
    AppComponent,
    MyAppComponent,
    MyCustomDirective
  ],
  imports: [
    BrowserModule,
    FormsModule
  ],
  providers: [FoodService, BriyaniService,
NoodleService],
  bootstrap: [AppComponent]
})
```

SERVICE – EXAMPLE 2 …



SERVICE – EXAMPLE 2 …

Food Menu

- Briyani
- Fride Rice
- Noodles
- Snacks
- Rools
- Chinese

Food Menu

- Briyani
- Fride Rice
- Noodles
- Snacks
- Rools
- Chinese

Briyani List

Chicken Briyani

Rs : 120

Mutton Briyani

Rs : 140

Fish Briyani

Rs : 160

Food Menu

- Briyani
- Fride Rice
- Noodles
- Snacks
- Rools
- Chinese

Noodles List

Chicken Noodle

Rs : 110

Mutton Noodle

Rs : 130

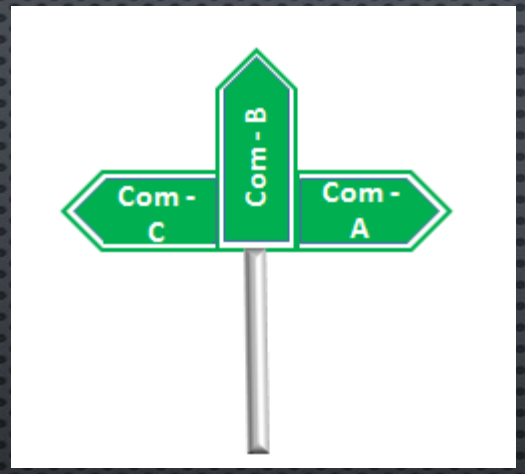
Fish Noodle

Rs : 140

ROUTER

ChandraSekhar(CS) Baratam

WHAT IS ROUTER



Routes between Components

ROUTING – EXAMPLE

Home.component.html

```
<p>
 I am Home Component.
</p>
```

Contact.component.html

```
<p>
 I am Contact Component.
</p>
```

about.component.html

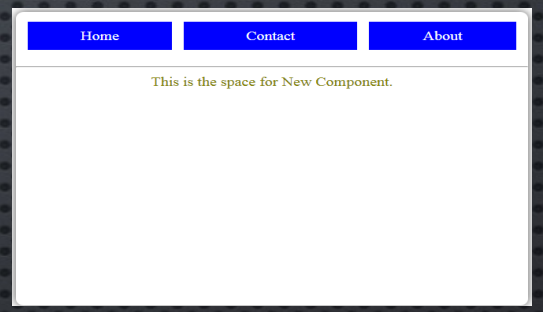
```
<p>
 I am About Component.
</p>
```

app.component.html

```
<div class="myDiv">
<table width="100%" [cellSpacing]="15"
[cellPadding]="10">
<tr>
<td><a routerLink="Home">Home</a></td>
<td><a routerLink="Contact">Contact</a></td>
<td><a routerLink="About">About</a></td>
</tr>
</table>
<hr/>
<div>
 This is the space for New Component.
 <router-outlet></router-outlet>
</div>
</div>
```

app.module.ts

```
import { Routes, RouterModule } from
'@angular/router';

const ROUTES: Routes = [
{ path : 'Home', component :
HomeComponent},
{ path : 'Contact', component :
ContactComponent},
{ path : 'About', component :
AboutComponent}
];
@NgModule({
.........
imports: [
   RouterModule.forRoot(ROUTES),
```



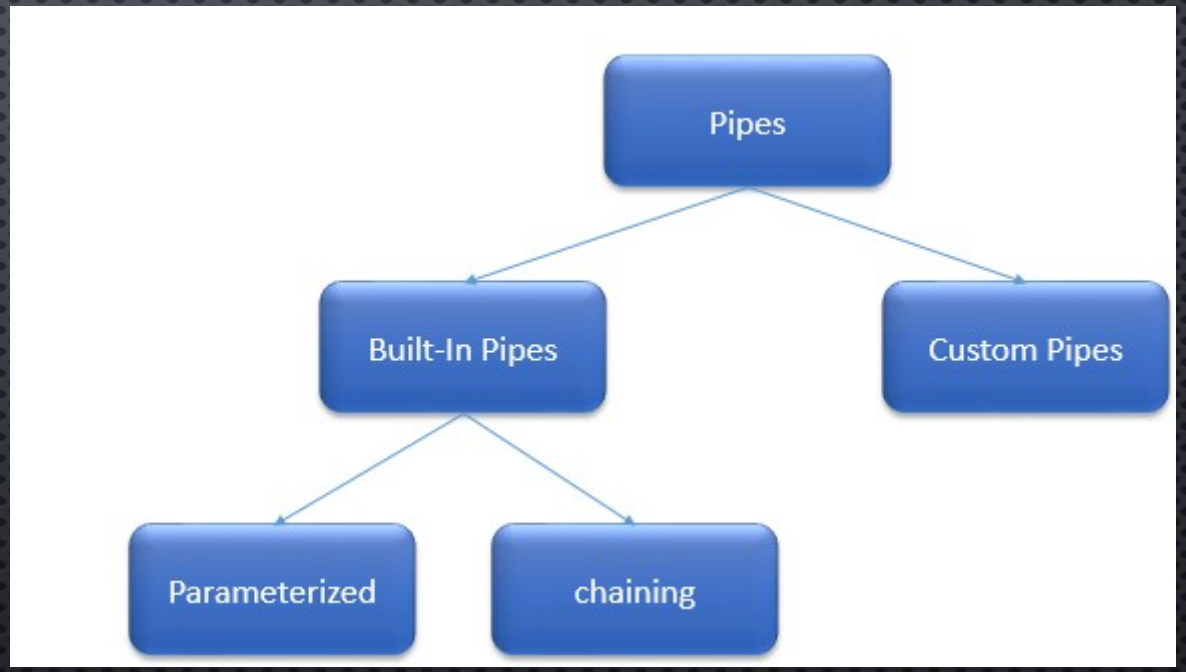
PIPES

ChandraSekhar(CS) Baratam

WHAT IS PIPE

valan

VALAN



1 BUILT-IN PIPES

app.component.html

```
<div>
<table width="80%">
<tr><td>Upper Case Pipe  </td><td>{{ name | uppercase }}</td></tr>
<tr><td>Lower Case Pipe  </td><td>{{ name | lowercase }}</td></tr>
<tr><td>Title Case Pipe  </td><td>{{ name | titlecase }}</td></tr>
</table>
<hr/><br>
<table width="100%">
<tr><td>Date Pipe  </td><td>{{ date }}</td></tr>
<tr><td>Date Pipe  </td><td>{{ date | date: 'dd/MMM/yyyy' }}</td></tr>
<tr><td>Date Pipe  </td><td>{{ date | date: 'dd-MM-yy' }}</td></tr>
<tr><td>Date Pipe  </td><td>{{ date | date: 'fullDate' }}</td></tr>
<tr><td>Date Pipe  </td><td>{{ date | date: 'fullDate' |
uppercase}}</td></tr>
<tr><td>Date Pipe  </td><td>{{ date | date: 'shortTime' }}</td></tr>
</table>
<hr/><br>
<table width="80%">
<tr><td>Percent Pipe  </td><td>{{00.54565 | percent}}</td></tr>
<tr><td>Currency Pipe  </td><td>{{6589.23 | currency:"USD"}}</td></tr>
</table>
</div>
```

app.component.ts

```
import {Component} from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
 name: string = 'valan arasu';
  date: Date = new Date();}
```

Upper Case Pipe	VALAN ARASU
Lower Case Pipe	valan arasu
Title Case Pipe	Valan Arasu
Date Pipe	Sun Jul 15 2018 11:11:59 GMT+0530 (India Standard Time)
Date Pipe	15/Jul/2018
Date Pipe	15-07-18
Date Pipe	Sunday, July 15, 2018
Date Pipe	SUNDAY, JULY 15, 2018
Date Pipe	11:11 AM
Percent Pipe	55%
Currency Pipe	$6,589.23

2 CUSTOM PIPES

app.component.html

```
<div>
<br>{{ arr1 }}
<br><br>{{ arr1 | sort: 'reverse' }}
<br><br>{{ arr1 | sort: 'ascending' }}
<br><br>{{ arr1 | sort: 'descending' }}
<hr/>
<br>{{ arr2 }}
<br><br>{{ arr2 | sort: 'reverse' }}
<br><br>{{ arr2 | sort: 'ascending' }}
<br><br>{{ arr2 | sort: 'descending' }}
</div>
```

App.module.ts

```
import { SortPipe } from
'./sort.pipe';
@NgModule({
  declarations: [
    AppComponent,
    SortPipe
  ],
```

Sort.pipe.ts

```
import {Pipe, PipeTransform} from
'@angular/core';
@Pipe({
  name: 'sort'
})
export class SortPipe implements
PipeTransform {
  transform(value: any[], args: string):
any[] {
    if (args === 'ascending') {
      return value.sort();
    }
    if (args === 'descending') {
      return value.sort().reverse();
    }
    if (args === 'reverse') {
      return value.reverse();
    }
  }
}
```

app.component.ts

```
import {Component} from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  arr1: number[] = [3, 7, 1, 9, 2];
  arr2: string[] = ['valan', 'arasu', 'hi', 'hello',
'wipro'];
}
```

3,7,1,9,2

2,9,1,7,3

1,2,3,7,9

9,7,3,2,1

valan,arasu,hi,hello,wipro

wipro,hello,hi,arasu,valan

arasu,hello,hi,valan,wipro

wipro,valan,hi,hello,arasu

Forms & Validation

ChandraSekhar(CS) Baratam

1 TEMPLATE DRIVEN FORMS

Template-driven-form-container.component.html

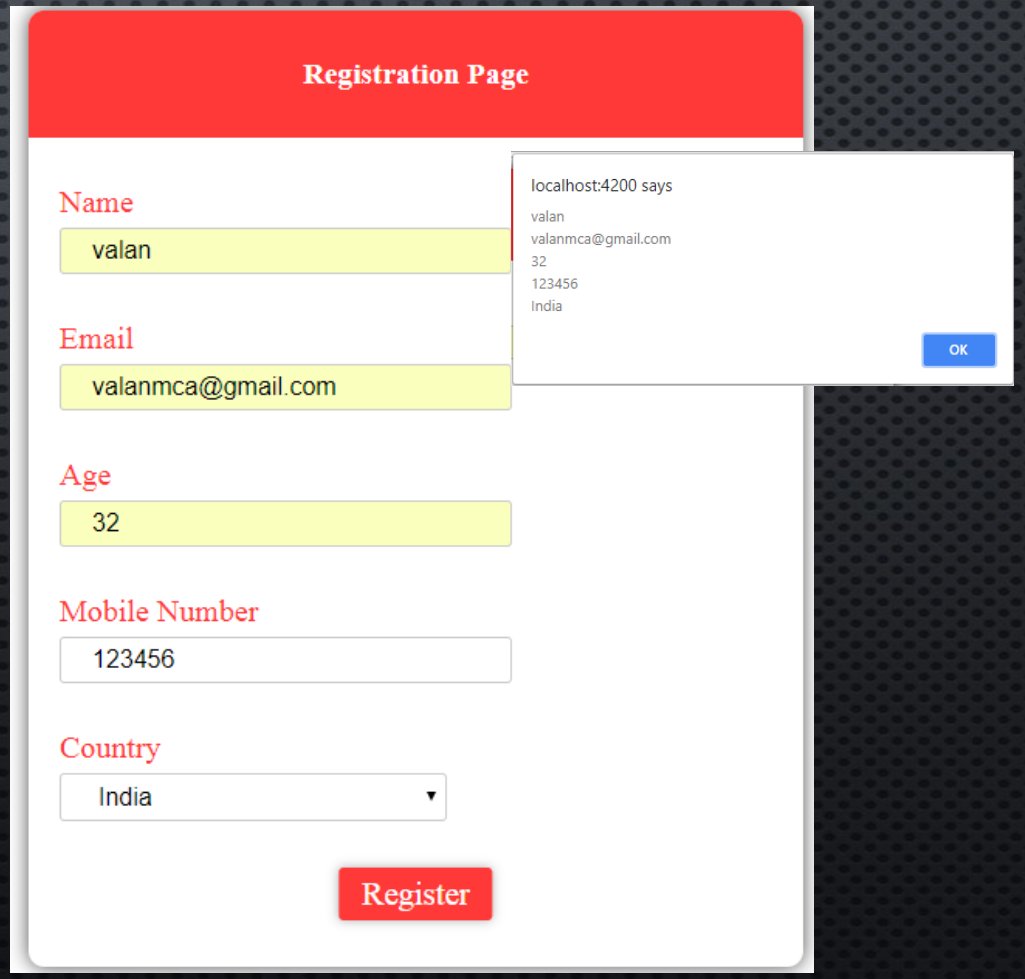
```
<div class="form_container">
 <form #myForm="ngForm" (ngSubmit)="myFunction(myForm.value)">
 <header>Registration Page</header>
 <label>Name</label><input type="text" name="name" ngModel/>
 <label>Email</label><input type="text" name="email" ngModel/>
 <label>Age</label><input type="text" name="age" ngModel/>
 <label>Mobile Number</label><input type="text" name="mobileNo"
ngModel/>
 <label>Country</label>   <select name="country" ngModel>
 <option value="India">India</option>  <option
value="China">China</option>
 </select>
 <button>Register</button>
 </form>
</div>
```

Template-driven-form.component.ts

```
import {Component, OnInit} from '@angular/core';
@Component({
  selector: 'template-driven-form',
  templateUrl: './template-driven-form.component.html',
  styleUrls: ['./template-driven-form.component.css']
})
export class TemplateDrivenFormComponent {
  myFunction(data) {
    alert(data.name + "\n" + data.email + "\n" + data.age + "\n" +
data.mobileNo + "\n" + data.country);
  }    }
```

App.module.ts

```
imports: [
   BrowserModule,
   FormsModule
 ],
```



TDF Validation Table

State	Class if true	Class if false
Control has been visited	ng-touched	ng-untouched
Control's value has changed	ng-dirty	ng-pristine
Control's value is valid	ng-valid	ng-invalid

```
<label>Name</label><input type="text"
name="name" #nameRef  required
ngModel/>
<br>
<b>{{nameRef.className}}</b>
```

Name

ng-untouched ng-pristine ng-invalid

Name valan

ng-dirty ng-valid ng-touched

TDF Validation

Template-driven-
form.component.html

```
<div class="form_container">
 <form #myForm="ngForm" (ngSubmit)="myFunction(myForm.value)">
 <header>Registration Page</header>
 <label>Name</label><input type="text" name="name" required pattern="[a-zA-z][a-zA-Z ]+" #name="ngModel"
ngModel/>
 <div *ngIf="name.touched">
 <div *ngIf="name.errors?.required">
 <font color="red"><b>Name Required.</b></font>
 </div>
 <div *ngIf="name.errors?.pattern">
 <font color="red"><b>Invalid Name.</b></font>
 </div>
 </div>
 <label>Email</label><input type="text" name="email" required pattern="^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\
w{2,3})+$" #email="ngModel" ngModel/>
 <div *ngIf="email.touched">
 <div *ngIf="email.errors?.required">
 <font color="red"><b>Email Required.</b></font>
 </div>
 <div *ngIf="email.errors?.pattern">
 <font color="red"><b>Invalid Email.</b></font>
 </div>
 </div>
```

TDF Validation....

Template-driven-form.component.html

```
<label>Age</label><input type="text" name="age" required pattern="[0-9]+" #age="ngModel" maxlength=2
ngModel/>
 <div *ngIf="age.touched">
 <div *ngIf="age.errors?.required">
 <font color="red"><b>Age Required.</b></font>
 </div>
 <div *ngIf="age.errors?.pattern">
 <font color="red"><b>Invalid Age.</b></font>
 </div>
 </div>
 <label>Mobile Number</label><input type="text" name="mobileNo" required pattern="^[1-9]{1}[0-9]{9}$"
#mobileNo="ngModel" ngModel/>
 <div *ngIf="mobileNo.touched">
 <div *ngIf="mobileNo.errors?.required">
 <font color="red"><b>Mobile Number Required.</b></font>
 </div>
 <div *ngIf="mobileNo.errors?.pattern">
 <font color="red"><b>Invalid Mobile Number.</b></font>
 </div>
 </div>
 <label>Country</label>   <select name="country" required ngModel>
 <option value="India">India</option>  <option value="China">China</option>
 </select>
 <button [disabled]="myForm.invalid">Register</button>
 </form>
</div>
```

TDF VALIDATION....

Registration Page

Name

Name Required.

Email

valanmca@gmail.com

Age

32

Mobile Number

1

Invalid Mobile Number.

Country

India

Register

2 REACTIVE / MODEL DRIVEN FORMS

reactive-form.component.html

```
<div class="form_container">
 <form [formGroup]="myForm"
(ngSubmit)="myFunction(myForm.value)">
 <header>Registration Page</header>
 <label>Name</label><input type="text"
formControlName="name"/>
 <label>Email</label><input type="text"
formControlName="email"/>
 <label>Age</label><input type="text"
formControlName="age"/>
 <label>Mobile Number</label><input type="text"
formControlName="mobileNo"/>
 <button>Register</button>
 </form>
<ul>
<li *ngFor="let user of userList">{{user.name}}  {{user.email}}
{{user.age}}  {{user.mobileNo}}</li>
</ul>
</div>
```

App.module.ts

```
imports: [
  BrowserModule,
  ReactiveFormsModule,
  FormsModule
],
```

recative-form.component.ts

```
import {Component, OnInit} from '@angular/core';
import {FormGroup, FormControl} from
'@angular/forms';
export class ReactiveFormComponent {
  myForm: FormGroup;
  userList: User[] = [];
  constructor() {
    this.myForm = new FormGroup({
      name: new FormControl(''),
      email: new FormControl(''),
      age: new FormControl(''),
      mobileNo: new FormControl(''),
    });
  }
  myFunction(data) {
    this.userList.push(data);
  }
}
export interface User {
  name: string;
  email: string;
  age: number;
  mobileNo: number;
}
```

REACTIVE / MODEL DRIVEN FORMS...

Registration Page

Name

valan

Email

valanmca@gmail.com

Age

32

Mobile Number

123454

Register

Registration Page

Name

valan2

Email

valanmca@gmail.com

Age

32

Mobile Number

123454

Register

- valan valanmca@gmail.com 32 123454
- valan1 valanmca@gmail.com 32 123454
- valan2 valanmca@gmail.com 32 123454

Reactive Form Validation

reactive-form.component.html

```
<div class="form_container">
<form [formGroup]="myForm" (ngSubmit)="myFunction(myForm.value)">
<header>Registration Page</header>
<label>Name</label><input type="text" formControlName="name"/>
<div *ngIf="myForm.get('name').touched && myForm.get('name').invalid">
<div *ngIf="myForm.get('name').hasError('required')">
<font color=red><b>Name is Required.</b></font>
</div>
<div *ngIf="myForm.get('name').hasError('pattern')">
<font color=red><b>Invalid Name.</b></font>
</div>
</div>
<label>Email</label><input type="text" formControlName="email"/>
<div *ngIf="myForm.get('email').touched && myForm.get('email').invalid">
<div *ngIf="myForm.get('email').hasError('required')">
<font color=red><b>Email is Required.</b></font>
</div>
<div *ngIf="myForm.get('email').hasError('pattern')">
<font color=red><b>Invalid Email.</b></font>
</div>
</div>
```

Reactive Form Validation...

reactive-form.component.html

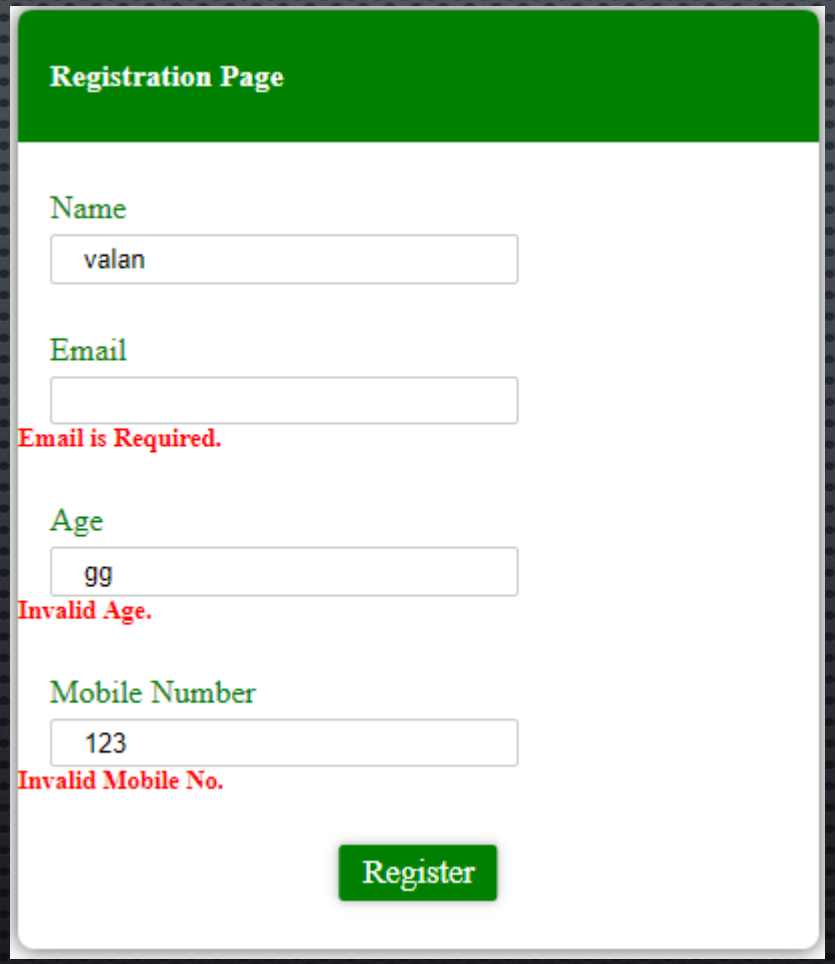
```
<label>Age</label><input type="text" formControlName="age"/>
<div *ngIf="myForm.get('age').touched && myForm.get('age').invalid">
<div *ngIf="myForm.get('age').hasError('required')">
<font color=red><b>Age is Required.</b></font>
</div>
<div *ngIf="myForm.get('age').hasError('pattern')">
<font color=red><b>Invalid Age.</b></font>
</div>
</div>
<label>Mobile Number</label><input type="text" formControlName="mobileNo"/>
<div *ngIf="myForm.get('mobileNo').touched && myForm.get('mobileNo').invalid">
<div *ngIf="myForm.get('mobileNo').hasError('required')">
<font color=red><b>Mobile No is Required.</b></font>
</div>
<div *ngIf="myForm.get('mobileNo').hasError('pattern')">
<font color=red><b>Invalid Mobile No.</b></font>
</div>
</div>
<button [disabled]="myForm.invalid">Register</button>
</form>
<ul>
<li *ngFor="let user of userList">{{user.name}}  {{user.email}}  {{user.age}}  {{user.mobileNo}}</li>
</ul>
</div>
```

Reactive Form Validation...

reactive-form.component.ts

```
export class ReactiveFormComponent {
  myForm: FormGroup;
  userList: User[] = [];
  constructor() {
    this.myForm = new FormGroup({
      name: new FormControl('', [Validators.required, Validators.pattern('[a-zA-z][a-zA-Z]+')]),
      email: new FormControl('', [Validators.required, Validators.pattern('^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\
w{2,3})+$')]),
      age: new FormControl('', [Validators.required, Validators.pattern('[0-9]+'), Validators.maxLength(2)]),
      mobileNo: new FormControl('', [Validators.required, Validators.pattern('^[1-9]{1}[0-9]{9}$')]),
    });
  }
  myFunction(data) {
    this.userList.push(data);
  }
}
export interface User {
  name: string;
  email: string;
  age: number;
  mobileNo: number;
}
```

REACTIVE FORM VALIDATION...



HTTP SERVICE

ChandraSekhar(CS) Baratam

SETTING UP A FAKE REST API

IN THE OPENED TERMINAL, TYPE THE FOLLOWING COMMAND.

NPM INSTALL -G JSON-SERVER

NOW, ADD THE FOLLOWING COMMAND TO THE SCRIPTS SECTION IN THE PACKAGE.JSON FI
SERVER:

"JSON-SERVER": "JSON-SERVER --WATCH DB.JSON --PORT 3004"

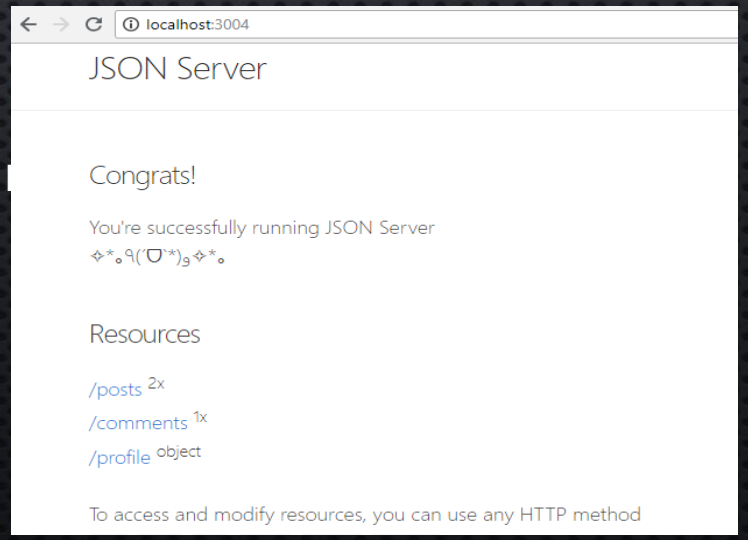
IN THE TERMINAL+ VIEW (ENSURE YOU HAVE THE CORRECT PROJECT SELECTED) RUN THE

NPM RUN JSON-SERVER

IT WILL CREATE A FILE CALLED "DB.JSON"

Package.json

```
{
  "name": "angular-dem",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build --prod",
    "test": "ng test",
    "lint": "ng lint",
    "json-server": "json-server --
watch db.json --port 3004",
    "e2e": "ng e2e"
  },
```



1 GET

App.module.ts

```
import {NgModule} from '@angular/core';
import {BrowserModule} from '@angular/platform-
browser';
import { HttpModule } from '@angular/http';

import {AppComponent} from './app.component';
import { DataService } from './data.service';
import { HttpServiceComponent } from './http-
service/http-service.component';

@NgModule({
  declarations: [
    AppComponent,
    HttpServiceComponent
  ],
  imports: [
    BrowserModule,
    HttpModule
  ],
  providers: [DataService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Data.service.ts

```
import {Injectable} from '@angular/core';
import {Http} from '@angular/http';

@Injectable()
export class DataService {

  private url: string =
'http://localhost:3004/posts';

  constructor(private http: Http) {}

  getAllEmployeeDetails() {
    return this.http.get(this.url);
  }

}
```

GET...

http-service.component.ts

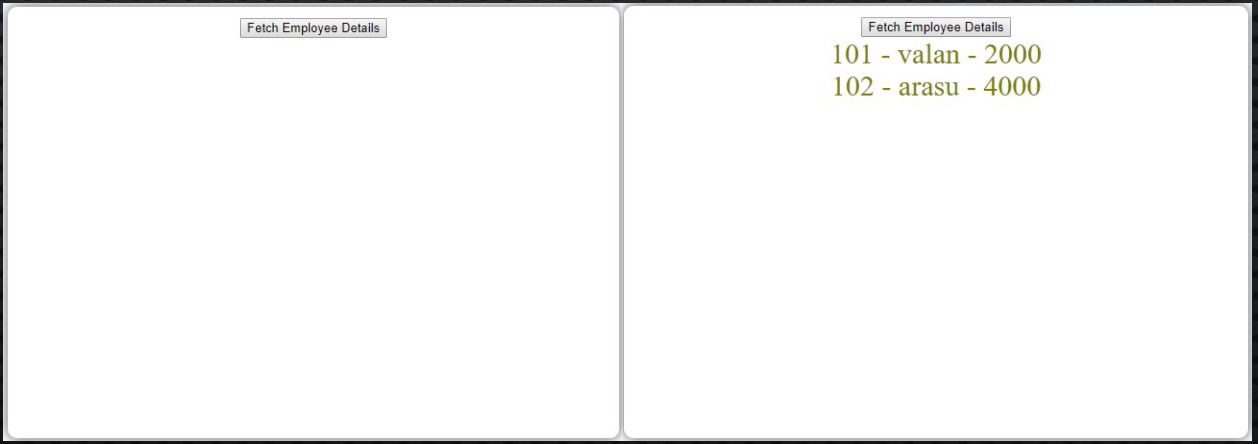
```
import {DataService} from '../data.service';
import {Component, OnInit} from '@angular/core';
@Component({
  selector: 'http-service',
  templateUrl: './http-service.component.html',
  styleUrls: ['./http-service.component.css']
})
export class HttpServiceComponent {
  employees: string;

  constructor(private service: DataService) {}

  getAllEmployeeDetails() {
    this.service.getAllEmployeeDetails().subscribe(res
=> this.employees = res.json());
  }
}
```

db.json

```
"posts": [
    {
    "id" : 101,
    "name" : "valan",
    "salary" : 2000
  },
   {
    "id" : 102,
    "name" : "arasu",
    "salary" : 4000
  }
  ],
```

http-service.component.html

```
<div class="MyDiv">
<button (click)="getAllEmployeeDetails()">Fetch
Employee Details</button>
<ul *ngFor="let emp of employees">
<li>{{emp.id}} - {{emp.name}} - {{emp.salary}}</li>
</ul>
</div>
```



2 POST

App.module.ts

```
import {NgModule} from '@angular/core';
import {BrowserModule} from '@angular/platform-
browser';
import { HttpModule } from '@angular/http';

import {AppComponent} from './app.component';
import { DataService } from './data.service';
import { HttpServiceComponent } from './http-
service/http-service.component';

@NgModule({
  declarations: [
    AppComponent,
    HttpServiceComponent
  ],
  imports: [
    BrowserModule,
    HttpModule
  ],
  providers: [DataService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Data.service.ts

```
import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class DataService {
  url: string = 'http://localhost:3004/posts';
  constructor(private http: Http) { }

  getAllEmployeesDetails() {
    return this.http.get(this.url);
  }

  saveEmployeeDetails() {
    return this.http.post(this.url, {id : 103, name : 'Raj',
salary : 5000}).map((res: Response) => res.json());
  }
}
```

POST…

http-service.component.ts

```
import { DataService } from '../data.service';
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'http-service',
  templateUrl: './http-service.component.html',
  styleUrls: ['./http-service.component.css']
})
export class HttpServiceComponent {
  employees: string;
  constructor(private service: DataService) { }

  getAllEmployeeDetails() {
    this.service.getAllEmployeesDetails().subscribe(res =>
this.employees = res.json());
  }
  saveEmployeeDetails() {
    this.service.saveEmployeeDetails().subscribe();
  }
}
```

http-service.component.html

```
<div>
<br>
<button (click)="getAllEmployeeDetails()">Fetch Employee
Details</button>
<ul *ngFor="let emp of employees">
<li>{{emp.id}} - {{emp.name}} - {{emp.salary}}</li>
</ul>
<button (click)="saveEmployeeDetails()">Save
Employee</button>
</div>
```

db.json

```
"posts": [
{
    "id": 101,
    "name": "valan",
    "salary": 2000
  },
  {
    "id": 102,
    "name": "arasu",
    "salary": 4000
  },
  {
    "id": 103,
    "name": "Raj",
    "salary": 5000
  } ],
```

3 DELETE

App.module.ts

```
import {NgModule} from '@angular/core';
import {BrowserModule} from '@angular/platform-
browser';
import { HttpModule } from '@angular/http';

import {AppComponent} from './app.component';
import { DataService } from './data.service';
import { HttpServiceComponent } from './http-
service/http-service.component';

@NgModule({
  declarations: [
    AppComponent,
    HttpServiceComponent
  ],
  imports: [
    BrowserModule,
    HttpModule
  ],
  providers: [DataService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Data.service.ts

```
import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import 'rxjs/add/operator/map';

@Injectable()
export class DataService {
  url: string = 'http://localhost:3004/posts';
  constructor(private http: Http) { }

  getAllEmployeesDetails() {
    return this.http.get(this.url);
  }

  saveEmployeeDetails() {
    return this.http.post(this.url, {id : 103, name : 'Raj',
salary : 5000}).map((res: Response) => res.json());
  }

deleteEmployeeDetails() {
    return this.http.delete(this.url + '/103');
  }
}
```

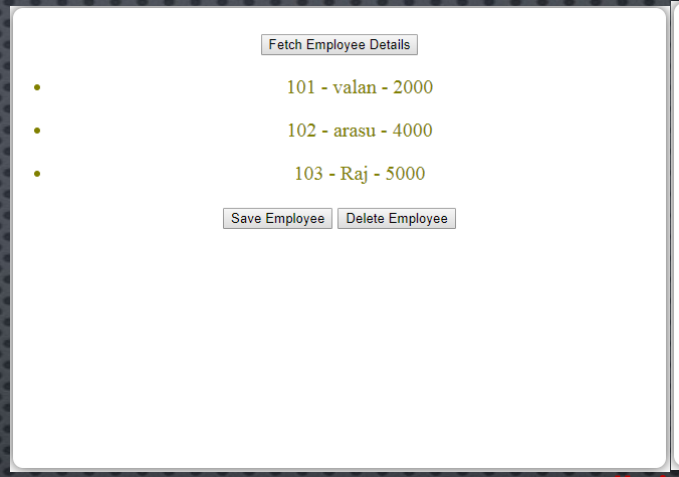
DELETE...

http-service.component.ts

```
import { DataService } from '../data.service';
import { Component, OnInit } from '@angular/core';
@Component({
  selector: 'http-service',
  templateUrl: './http-service.component.html',
  styleUrls: ['./http-service.component.css']
})
export class HttpServiceComponent {
  employees: string;
  constructor(private service: DataService) { }
  getAllEmployeeDetails() {
    this.service.getAllEmployeesDetails().subscribe(res =>
this.employees = res.json());
  }
  saveEmployeeDetails() {
    this.service.saveEmployeeDetails().subscribe();
  }
deleteEmployeeDetails() {
    this.service.deleteEmployeeDetails().subscribe();
  }
}
```

http-service.component.html

```
<div>
<br>
<button (click)="getAllEmployeeDetails()">Fetch Employee
Details</button>
<ul *ngFor="let emp of employees">
<li>{{emp.id}} - {{emp.name}} - {{emp.salary}}</li>
</ul>
<button (click)="saveEmployeeDetails()">Save
Employee</button>
<button (click)="deleteEmployeeDetails()">Delete
Employee</button>
</div>
```



db.json

```
"posts": [
{
    "id": 101,
    "name": "valan",
    "salary": 2000
  },
  {
    "id": 102,
    "name": "arasu",
    "salary": 4000
  },
],
```

4 PUT

App.module.ts

```
import {NgModule} from '@angular/core';
import {BrowserModule} from '@angular/platform-
browser';
import { HttpModule } from '@angular/http';

import {AppComponent} from './app.component';
import { DataService } from './data.service';
import { HttpServiceComponent } from './http-
service/http-service.component';

@NgModule({
  declarations: [
    AppComponent,
    HttpServiceComponent
  ],
  imports: [
    BrowserModule,
    HttpModule
  ],
  providers: [DataService],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

Data.service.ts

```
import { Injectable } from '@angular/core';
import { Http, Response } from '@angular/http';
import 'rxjs/add/operator/map';
@Injectable()
export class DataService {
  url: string = 'http://localhost:3004/posts';
  constructor(private http: Http) { }
  getAllEmployeesDetails() {
    return this.http.get(this.url);
  }
  saveEmployeeDetails() {
    return this.http.post(this.url, {id : 103, name : 'Raj',
salary : 5000}).map((res: Response) => res.json());
  }
deleteEmployeeDetails() {
    return this.http.delete(this.url + '/103');
  }
updateEmployeeDetails() {
    return this.http.put(this.url + '/103', {name : 'Kumar' ,
salary : '7000'});
  }
}
```

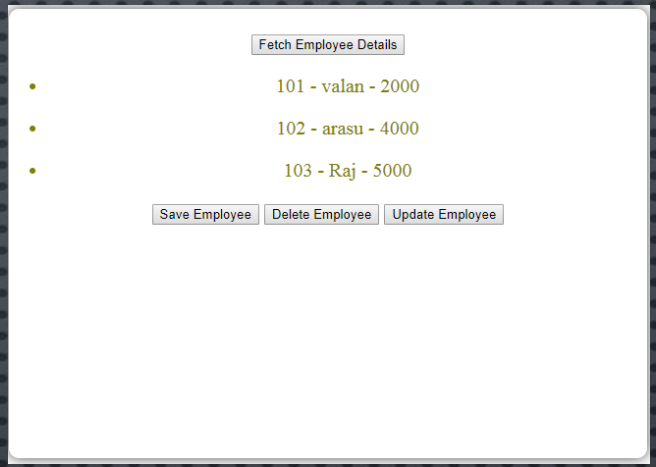
PUT…

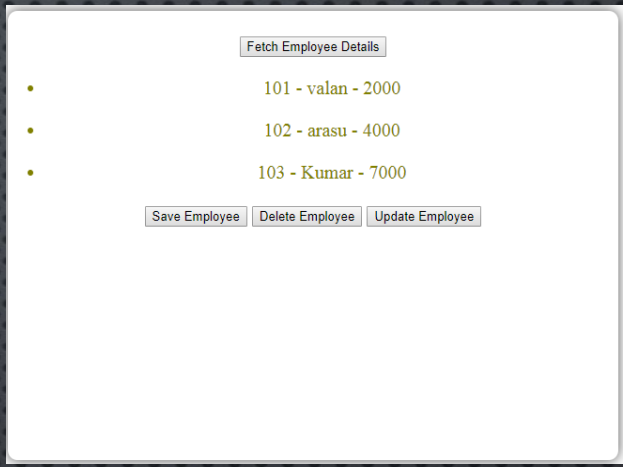
http-service.component.ts

```
……................................
export class HttpServiceComponent {
  employees: string;
  constructor(private service: DataService) { }
  getAllEmployeeDetails() {
    this.service.getAllEmployeesDetails().subscribe(res =>
this.employees = res.json());
  }
  saveEmployeeDetails() {
    this.service.saveEmployeeDetails().subscribe();
  }
deleteEmployeeDetails() {
    this.service.deleteEmployeeDetails().subscribe();
  }
updateEmployeeDetails() {
    this.service.updateEmployeeDetails().subscribe();
  }
}
```

http-service.component.html

```
<div> <br>
<button (click)="getAllEmployeeDetails()">Fetch Employee
Details</button>
<ul *ngFor="let emp of employees">
<li>{{emp.id}} - {{emp.name}} - {{emp.salary}}</li>
</ul>
<button (click)="saveEmployeeDetails()">Save
Employee</button>
<button (click)="deleteEmployeeDetails()">Delete
Employee</button>
<button (click)="updateEmployeeDetails()">Update
Employee</button>
```





db.json

```
"posts": [
{
    "id": 101,
    "name": "valan",
    "salary": 2000
  },
  {
    "id": 102,
    "name": "arasu",
    "salary": 4000
  },
{
    "name": "Kumar",
    "salary": "7000",
    "id": 103
  }    ],
```

COMPONENT COMMUNICATIONS

ChandraSekhar(CS) Baratam

COMMUNICATION BETWEEN COMPONENTS

Sender.component.html

```
<div>
  Sender Component
  <hr/>   <br><br>
  <form [formGroup]="myForm"
(ngSubmit)="myFunction(myForm.value)">
  <input type="text" placeholder="Enter the data"
formControlName="text"/><br>
  <button>Send</button>
  </form>
</div>
```

Sender.component.ts

```
import { MyService } from '../my.service';
import {Component, OnInit} from '@angular/core';
import {FormGroup, FormControl} from
'@angular/forms';

@Component({
  selector: 'sender',
  templateUrl: './sender.component.html',
  styleUrls: ['./sender.component.css']
})
export class SenderComponent {
  myForm: FormGroup;
  constructor(private service: MyService) {
    this.myForm = new FormGroup({
      text: new FormControl('')
    });
  }

  myFunction(data) {
    this.service.dataToService(data.text);
  }

}
```

My.service.ts

```
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';
@Injectable()
export class MyService {
  private messageSource = new BehaviorSubject('Default
Message');
  currentMessage = this.messageSource.asObservable();
  constructor() { }

  dataToService(text) {
    this.messageSource.next(text);
  }
}
```

COMMUNICATION BETWEEN COMPONENTS…

Receiver.component.html

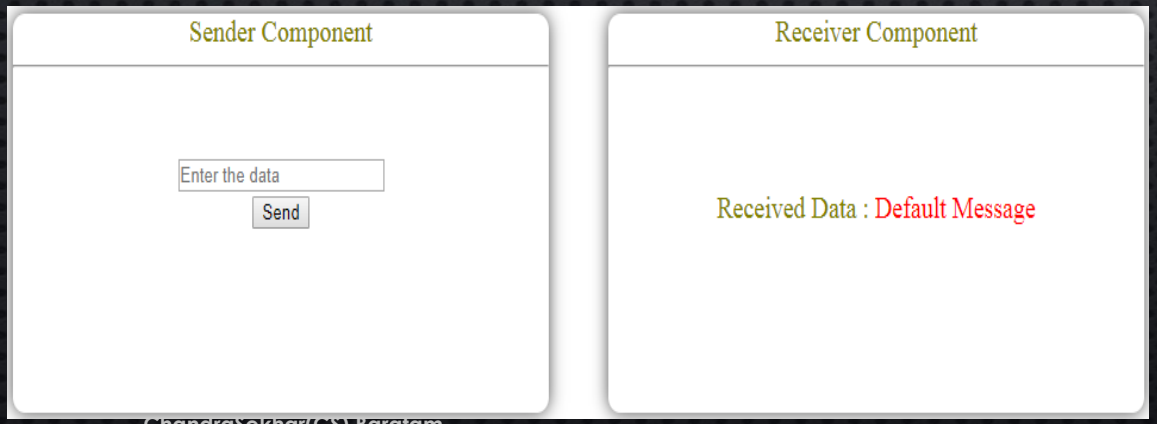
```
<div>
  Receiver Component
  <hr/><br><br><br>
  Received Data : <font color=red> {{message}} </font>
</div>
```

App.component.html

```
<sender></sender>
<receiver></receiver>
```

receiver.component.ts

```
import {MyService} from '../my.service';
import {Component} from '@angular/core';
@Component({
  selector: 'receiver',
  templateUrl: './receiver.component.html',
  styleUrls: ['./receiver.component.css']
})
export class ReceiverComponent {
  message: string;
  constructor(private service: MyService) {
    this.service.currentMessage.subscribe(msg => this.message
= msg);
  }
}
```



CRUD

ChandraSekhar(CS) Baratam

CRUD USING HTTP

1. CREATE OPERATION USING HTTP.POST METHOD.

2. READ OPERATION USING HTTP.GET METHOD.

3. UPDATE OPERATION USING HTTP.PUT METHOD.

4. DELETE OPERATION USING HTTP.DELETE METHOD.

CRUD EXAMPLE

center.component.ts

center.component.html

```
<form [formGroup]="myForm">
<div class="MyDiv">
<label>Id</label><input type="text" formControlName="id"/>
<label>Name</label><input type="text"
formControlName="name"/>
<label>Salary</label><input type="text"
formControlName="salary"/>
</div>
<button
(click)="saveEmployee(myForm.value)">Save</button>
<button
(click)="updateEmployee(myForm.value)">Update</button>
<button
(click)="deleteEmployee(myForm.value)">Delete</button>
<br><font color=green>{{result}}</font>
</form>
```

```
export class CenterComponent {
  myForm: FormGroup;
  result: string = '';
  employee = {id: '', name: '', salary: ''};
  constructor(private service: EmsService) {
    this.myForm = new FormGroup({
      id: new FormControl(''),name: new FormControl(''),salary:
new FormControl('')
    });
  }
  saveEmployee(data) {
    this.employee.id = data.id;
    this.employee.name = data.name;
    this.employee.salary = data.salary;
    this.service.saveEmployee(this.employee);
     this.result = this.service.result;
  }
  updateEmployee(data) {
    this.employee.id = data.id;
    this.employee.name = data.name;
    this.employee.salary = data.salary;
    this.service.updateEmployee(this.employee);
    this.result = this.service.result;
  }
  deleteEmployee(data) {
    this.service.deleteEmployee(data.id);
    this.result = this.service.result;
  }
}
```

CRUD EXAMPLE...

right.component.html

```
<br>
 Employee List
<hr/>
<ul *ngFor="let emp of allEmployeesList">
  <li>{{emp.id}} - {{emp.name}} - {{emp.salary}}</li>
 </ul>
```

right.component.ts

```
import {EmsService} from '../ems.service';
import {Component} from '@angular/core';

@Component({
  selector: 'right',
  templateUrl: './right.component.html',
  styleUrls: ['./right.component.css']
})
export class RightComponent {

  allEmployeesList: string;

  constructor(private service: EmsService) {
    this.getAllEmployeeDetails();
  }

  getAllEmployeeDetails() {
    this.service.getAllEmployeeDetails().subscribe(res =>
this.allEmployeesList = res);
  }
}
```

CRUD EXAMPLE...

top.component.html

```
<p>
 Employee Management System
</p>
```

app.component.ts

```
<crud-main></crud-main>
```

app.module.ts

```
@NgModule({
  declarations: [
    AppComponent,
    CrudMainComponent,
    TopComponent,
    RightComponent,
    CenterComponent
  ],
  imports: [
    BrowserModule,
    ReactiveFormsModule,
    FormsModule,
    HttpModule,
    ],
  providers: [EmsService],
  bootstrap: [AppComponent]
})
```

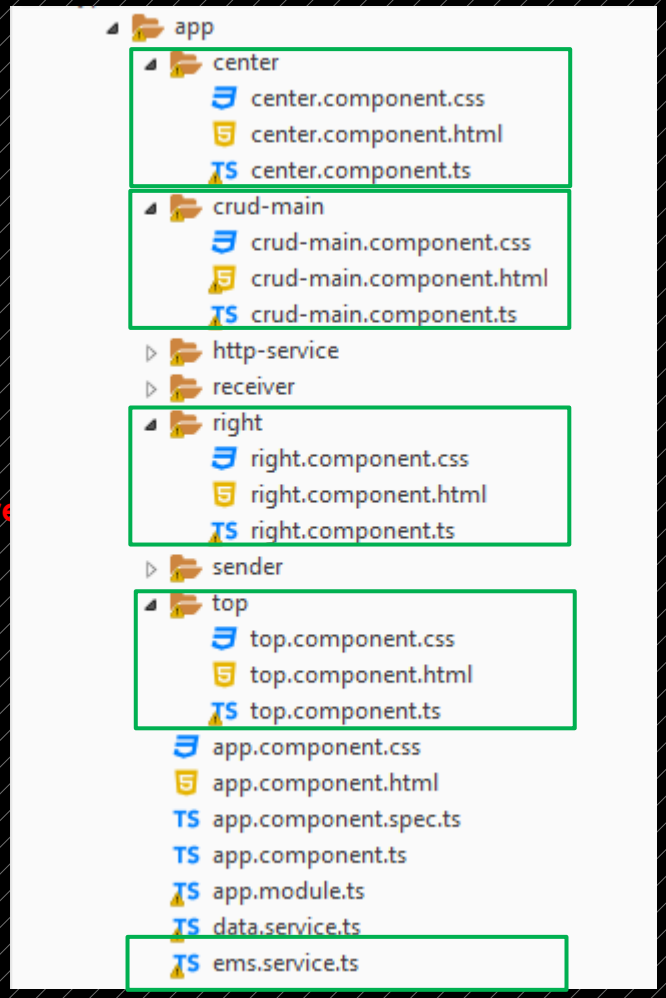
Crud-main.component.html

```
<div class="top">
<top></top>
</div>
<div class="center">
<center></center>
</div>
<div class="right">
<right></right>
</div>
```

CRUD EXAMPLE…

Ems.service.ts

```
import {HttpErrorResponse} from '@angular/common/http';  import {Injectable} from '@angular/core';  import {Http, Response} from '@angular/http';
import {Observable, BehaviorSubject} from 'rxjs';  import 'rxjs/add/operator/map';  import 'rxjs/add/operator/catch';
@Injectable()
export class EmsService {
  private messageSource = new BehaviorSubject('');
  url: string = 'http://localhost:3004/posts';
  result: string = '';
  constructor(private http: Http) {
    this.getAllEmployeeDetails();
  }
  getAllEmployeeDetails() {
    this.http.get(this.url).subscribe(res => this.messageSource.next(res.json()));
    let allEmployeeList = this.messageSource.asObservable();
    return allEmployeeList;
  }
  saveEmployee(employee) {
    this.http.post(this.url, employee).map((res: Response) => res.json()).subscribe(res => this.result = re
    this.getAllEmployeeDetails();
    this.result = 'Record ' + employee.id + ' Saved';
  }
  updateEmployee(employee) {
    this.http.put(this.url + '/' + employee.id, employee).subscribe();
    this.getAllEmployeeDetails();
    this.result = 'Record ' + employee.id + ' Updated';
  }
  deleteEmployee(id) {
    this.http.delete(this.url + '/' + id).subscribe();
    this.getAllEmployeeDetails();
    this.result = 'Record ' + id + ' Deleted';
  }  }
```



Employee Management System

Id

Name

Salary

Save Update Delete

Employee List

101 - valan - 2000

102 - arasu - 3000

103 - valan - 222

Employee Management System

Id
104

Name
Raj

Salary
7000

Save Update Delete

Record 104 Saved

Employee List

101 - valan - 2000

102 - arasu - 3000

103 - valan - 222

104 - Raj - 7000

Employee Management System

Id
104

Name
Kumar

Salary
5000

Save Update Delete

Record 104 Updated

Employee List

101 - valan - 2000

102 - arasu - 3000

103 - valan - 222

104 - Kumar - 5000

Employee Management System

Id
104

Name
Kumar

Salary
5000

Save Update Delete

Record 104 Deleted

Employee List

101 - valan - 2000

102 - arasu - 3000

103 - valan - 222