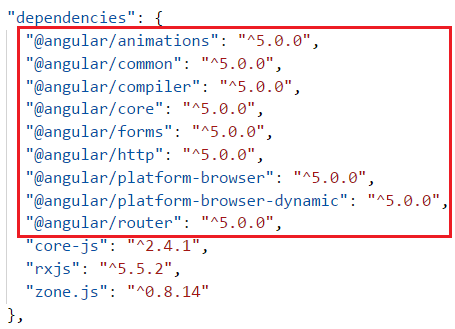
# 1. Angular project setup

This is Part 1 of **Angular CRUD tutorial**. In this video and in our upcoming videos in this series we will discuss performing CRUD operations in Angular i.e Creating, Reading, Updating and Deleting in Angular with simple examples.   
  
Setting up a new angular project from scratch is a tedious and time consuming process. However, Angular CLI makes it super-fast and easy. With angular development, we write same boiler plate code to create components, pipes, services, directives etc. Manually creating these consumes lot of time. Angular CLI can generate these with lightning speed while still following Angular's best practices and conventions. So basic knowledge of Angular CLI is very helpful.   
  
  
The prerequisites for this course are basic knowledge of **Bootstrap**, **Angular CLI** and **Angular 2**. If you are new to these please check out our courses using the links below.

1. [Angular CLI](http://csharp-video-tutorials.blogspot.com/2017/10/angular-cli-tutorial-for-beginners.html)
2. [Angular 2](http://csharp-video-tutorials.blogspot.com/2017/06/angular-2-tutorial-for-beginners_12.html)
3. [Bootstrap](http://csharp-video-tutorials.blogspot.com/2016/05/bootstrap-tutorial-for-beginners.html)

**Installing the tools required**   
  
**Node :** Install the latest version of node. Here is the link to download and install the latest version  
<https://nodejs.org/en/download/>  
  
As of this recording the latest version is 8.9.2 which is what I have installed. Run windows command prompt as an administrator and execute the following command to verify the version of node installed on your machine.  
node -v  
  
**Angular CLI :**Install the latest version of Angular CLI by executing the following command from the windows command prompt.   
npm install -g @angular/cli  
  
As of this recording, the latest version is 1.6.0 which is what I have installed on my machine. To verify the version of Angular CLI installed on your machine execute the following command.  
ng -v  
  
Another question that you might have is, which version of Angular are we using with this course. As of this recording, the latest version is Angular 5, which is what we will be using.   
  
After you have the latest version of Node and Angular CLI installed, launch windows command prompt as an administrator and execute the following command. This creates a new AngularProject with name AngularCrud. We do not want test files to be generated for the root component AppComponent, so we have set "skip-tests" option to true. We will discuss unit testing components in a later video.  
ng new AngularCrud --skip-tests true   
  
This command creates a brand new Angular Project with name AngularCrud. In windows command prompt change to the directory that contains your angular project using the following command.  
cd AngularCrud  
  
Once you are in the project directory, execute the following command to open the project with Visual Studio Code, by executing the following command from the windows command prompt   
code .  
  
Once you have the Angular project opened in Visual Studio Code, open “**package.json**”file and notice that we are using Angular 5.   
   
  
We will be using Bootstrap for styles in our application. So install Bootstrap by executing the following command from the command prompt.  
npm install bootstrap@3 --save  
  
Once Bootstrap is installed, open .angular-cli.json file and specify the path to the Bootstrap stylesheet (bootstrap.min.css) in the styles property as shown below. 

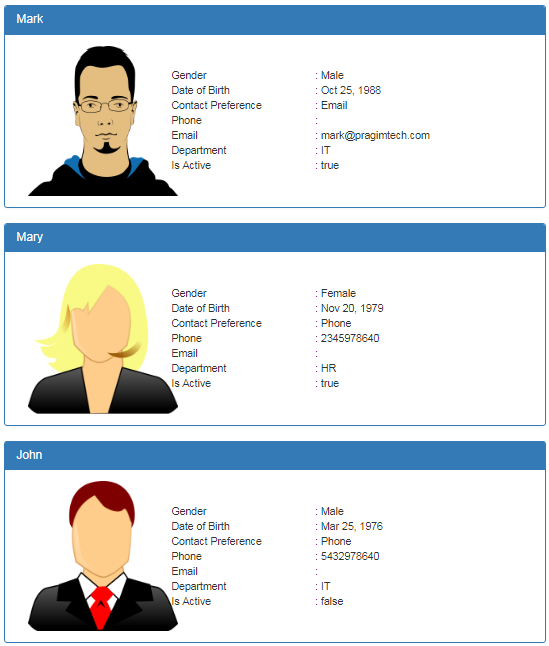
"styles": [

  "../node\_modules/bootstrap/dist/css/bootstrap.min.css",

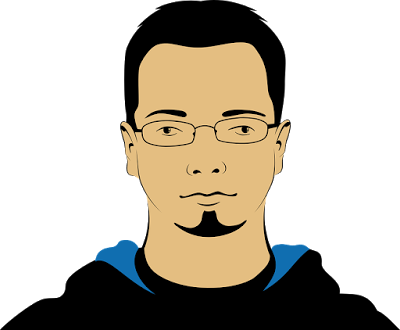
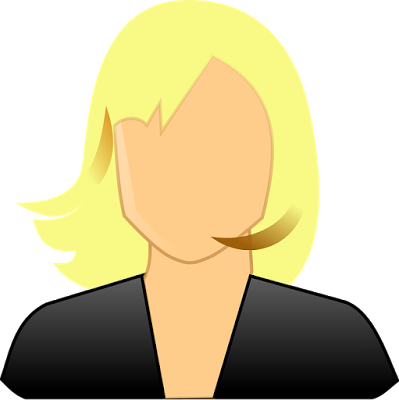
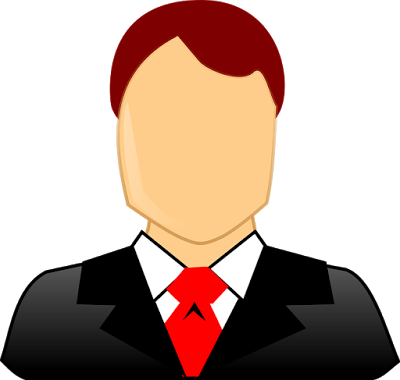
  "styles.css"

]

# 2. Reading data in angular

In this video we will discuss performing the **READ operation in Angular**. In our upcoming videos, we will discuss the rest of the CRUD operations i.e Creating, Updating and Deleting.   
  
  
Let us understand implementing the READ operation with an example. We want to display list of employees as shown below.   
   
  
  
At the moment, we do not have **Employee**model. First, let's create the Employee model.    
  
**Creating Employee model :** 

1. In Visual Studio Code, expand the "src" folder
2. Right click on the "App" folder, and select "New Folder" from the context menu
3. Name the folder "models". We will place all our models in this folder
4. Now add a new file in the "models" folder
5. Name it "employee.model.ts"
6. Copy and paste the following code in it

export class Employee {  
    id: number;  
    name: string;  
    gender: string;  
    email?: string;  
    phoneNumber?: number;  
    contactPreference: string;  
    dateOfBirth: Date;  
    department: string;  
    isActive: boolean;  
    photoPath?: string;  
}  
  
Next, create a component to display the list of employees. Name it **ListEmployeesComponent.**  
  
**Creating ListEmployeesComponent :** Use the following AngularCLI command to create ListEmployeesComponent. We will place all employee CRUD components in "employees" folder. This is the reason we prefixed the "employees" folder name in the command. Also, notice we have set --flat option to true as we do not want to place the ListEmployeesComponent files in it's own dedicated folder.   
  
ng g c employees/listEmployees --spec false --flat true  
  
The above command not only creates the **ListEmployeesComponent**, it also updates the **AppModule**. In the app.module.ts file it has imported ListEmployeesComponent and included it in the declarations array. So the Angular CLI has generated lot of boiler plate code, that we would have to write manually otherwise.  
  
**Creating images folder :**We will place all the images that we are going to use in "images" folder. We will have the images folder in the "assets" folder. So add a new folder in the "assets" folder and name it "images" and copy the following 3 images. Name the images mark.png, mary.png and john.png.   
  
[](https://2.bp.blogspot.com/-xG8wtGFhwd4/WjFWoJpjSPI/AAAAAAAAns0/-UCndX6XnlEsTYS-LeZyXa2_DsgQQlkQgCLcBGAs/s1600/mark.png)   
  
[](https://2.bp.blogspot.com/-VWh7J1pgrGo/WjFW-lYK-aI/AAAAAAAAns4/cjh5i2X7vhMhboHwirOlp-QLD3lDOFXXACLcBGAs/s1600/mary.png)   
  
[](https://2.bp.blogspot.com/-L8hccI1RetU/WjFXFp717LI/AAAAAAAAns8/zmoWhYWycR0SNT_u7Fxl5e73hEYsTMVIgCLcBGAs/s1600/john.png)   
  
**Changes in list-employees.component.ts :**The changes are commented and self-explanatory

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

// import Employee Model

import { Employee } from '../models/employee.model';

@Component({

  selector: 'app-list-employees',

  templateUrl: './list-employees.component.html',

  styleUrls: ['./list-employees.component.css']

})

export class ListEmployeesComponent implements OnInit {

  // Hard code the employee data. In a later video we will discuss

  // how to retrieve this employees data from a database table

  employees: Employee[] = [

    {

      id: 1,

      name: 'Mark',

      gender: 'Male',

      contactPreference: 'Email',

      email: 'mark@pragimtech.com',

      dateOfBirth: new Date('10/25/1988'),

      department: 'IT',

      isActive: true,

      photoPath: 'assets/images/mark.png'

    },

    {

      id: 2,

      name: 'Mary',

      gender: 'Female',

      contactPreference: 'Phone',

      phoneNumber: 2345978640,

      dateOfBirth: new Date('11/20/1979'),

      department: 'HR',

      isActive: true,

      photoPath: 'assets/images/mary.png'

    },

    {

      id: 3,

      name: 'John',

      gender: 'Male',

      contactPreference: 'Phone',

      phoneNumber: 5432978640,

      dateOfBirth: new Date('3/25/1976'),

      department: 'IT',

      isActive: false,

      photoPath: 'assets/images/john.png'

    },

  ];

  constructor() { }

  ngOnInit() {

  }

}

**Changes in list-employees.component.html :** Replace the existing HTML, with the following HTML. Notice we are using Bootstrap for styling.

<div class="panel panel-primary" \*ngFor="let employee of employees">

  <div class="panel-heading">

    <h3 class="panel-title">{{employee.name}}</h3>

  </div>

  <div class="panel-body">

    <div class="col-xs-10">

      <div class="row vertical-align">

        <div class="col-xs-4">

          <img class="imageClass" [src]="employee.photoPath" />

        </div>

        <div class="col-xs-8">

          <div class="row">

            <div class="col-xs-6">

              Gender

            </div>

            <div class="col-xs-6">

              : {{employee.gender}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Date of Birth

            </div>

            <div class="col-xs-6">

              : {{employee.dateOfBirth | date}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Contact Preference

            </div>

            <div class="col-xs-6">

              : {{employee.contactPreference}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Phone

            </div>

            <div class="col-xs-6">

              : {{employee.phoneNumber}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Email

            </div>

            <div class="col-xs-6">

              : {{employee.email}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Department

            </div>

            <div class="col-xs-6">

              : {{employee.department}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Is Active

            </div>

            <div class="col-xs-6">

              : {{employee.isActive}}

            </div>

          </div>

        </div>

      </div>

    </div>

  </div>

</div>

**Changes in list-employees.component.css :** Include the following CSS classes

.imageClass{

    width:200px;

    height:200px;

}

.vertical-align{

    display: flex;

    align-items: center;

}

**Changes in app.component.html :**Include the ListEmployeesComponent selector (app-list-employees) as a directive in the root component (app.component.html)

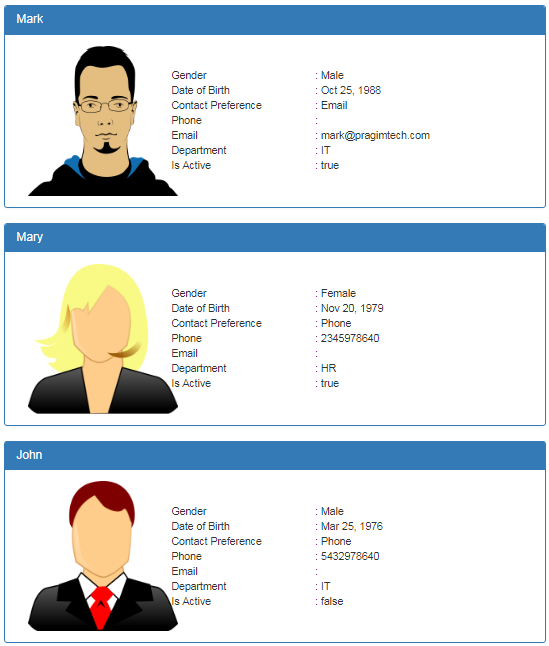
<div class="container">

    <app-list-employees></app-list-employees>

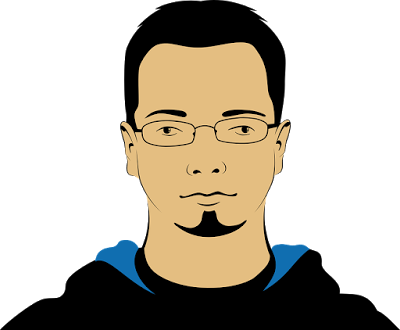
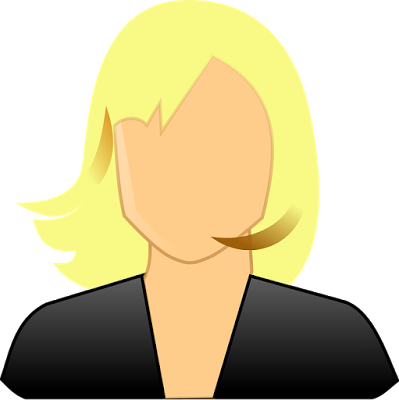
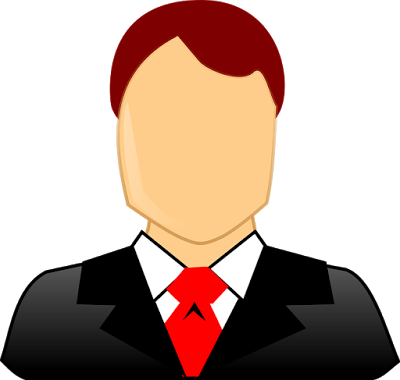
</div>

At this point, save all the changes and run the angular project using the following command. This command not only compiles the angular application, it also launches your default browser and displays the list of employees as expected.  
ng serve -o  
  
We have just seen how to perform the READ operation. In our next video, we will set up routing. 

# 3. Angular routing and navigation

In this video we will discuss performing the **READ operation in Angular**. In our upcoming videos, we will discuss the rest of the CRUD operations i.e Creating, Updating and Deleting.   
  
Let us understand implementing the READ operation with an example. We want to display list of employees as shown below.   
   
  
  
At the moment, we do not have **Employee**model. First, let's create the Employee model.    
  
**Creating Employee model :** 

1. In Visual Studio Code, expand the "src" folder
2. Right click on the "App" folder, and select "New Folder" from the context menu
3. Name the folder "models". We will place all our models in this folder
4. Now add a new file in the "models" folder
5. Name it "employee.model.ts"
6. Copy and paste the following code in it

export class Employee {  
    id: number;  
    name: string;  
    gender: string;  
    email?: string;  
    phoneNumber?: number;  
    contactPreference: string;  
    dateOfBirth: Date;  
    department: string;  
    isActive: boolean;  
    photoPath?: string;  
}  
  
Next, create a component to display the list of employees. Name it **ListEmployeesComponent.**  
  
**Creating ListEmployeesComponent :** Use the following AngularCLI command to create ListEmployeesComponent. We will place all employee CRUD components in "employees" folder. This is the reason we prefixed the "employees" folder name in the command. Also, notice we have set --flat option to true as we do not want to place the ListEmployeesComponent files in it's own dedicated folder.   
  
ng g c employees/listEmployees --spec false --flat true  
  
The above command not only creates the **ListEmployeesComponent**, it also updates the **AppModule**. In the app.module.ts file it has imported ListEmployeesComponent and included it in the declarations array. So the Angular CLI has generated lot of boiler plate code, that we would have to write manually otherwise.  
  
**Creating images folder :**We will place all the images that we are going to use in "images" folder. We will have the images folder in the "assets" folder. So add a new folder in the "assets" folder and name it "images" and copy the following 3 images. Name the images mark.png, mary.png and john.png.   
  
[](https://2.bp.blogspot.com/-xG8wtGFhwd4/WjFWoJpjSPI/AAAAAAAAns0/-UCndX6XnlEsTYS-LeZyXa2_DsgQQlkQgCLcBGAs/s1600/mark.png)   
  
[](https://2.bp.blogspot.com/-VWh7J1pgrGo/WjFW-lYK-aI/AAAAAAAAns4/cjh5i2X7vhMhboHwirOlp-QLD3lDOFXXACLcBGAs/s1600/mary.png)   
  
[](https://2.bp.blogspot.com/-L8hccI1RetU/WjFXFp717LI/AAAAAAAAns8/zmoWhYWycR0SNT_u7Fxl5e73hEYsTMVIgCLcBGAs/s1600/john.png)   
  
**Changes in list-employees.component.ts :**The changes are commented and self-explanatory

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

// import Employee Model

import { Employee } from '../models/employee.model';

@Component({

  selector: 'app-list-employees',

  templateUrl: './list-employees.component.html',

  styleUrls: ['./list-employees.component.css']

})

export class ListEmployeesComponent implements OnInit {

  // Hard code the employee data. In a later video we will discuss

  // how to retrieve this employees data from a database table

  employees: Employee[] = [

    {

      id: 1,

      name: 'Mark',

      gender: 'Male',

      contactPreference: 'Email',

      email: 'mark@pragimtech.com',

      dateOfBirth: new Date('10/25/1988'),

      department: 'IT',

      isActive: true,

      photoPath: 'assets/images/mark.png'

    },

    {

      id: 2,

      name: 'Mary',

      gender: 'Female',

      contactPreference: 'Phone',

      phoneNumber: 2345978640,

      dateOfBirth: new Date('11/20/1979'),

      department: 'HR',

      isActive: true,

      photoPath: 'assets/images/mary.png'

    },

    {

      id: 3,

      name: 'John',

      gender: 'Male',

      contactPreference: 'Phone',

      phoneNumber: 5432978640,

      dateOfBirth: new Date('3/25/1976'),

      department: 'IT',

      isActive: false,

      photoPath: 'assets/images/john.png'

    },

  ];

  constructor() { }

  ngOnInit() {

  }

}

**Changes in list-employees.component.html :** Replace the existing HTML, with the following HTML. Notice we are using Bootstrap for styling.

<div class="panel panel-primary" \*ngFor="let employee of employees">

  <div class="panel-heading">

    <h3 class="panel-title">{{employee.name}}</h3>

  </div>

  <div class="panel-body">

    <div class="col-xs-10">

      <div class="row vertical-align">

        <div class="col-xs-4">

          <img class="imageClass" [src]="employee.photoPath" />

        </div>

        <div class="col-xs-8">

          <div class="row">

            <div class="col-xs-6">

              Gender

            </div>

            <div class="col-xs-6">

              : {{employee.gender}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Date of Birth

            </div>

            <div class="col-xs-6">

              : {{employee.dateOfBirth | date}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Contact Preference

            </div>

            <div class="col-xs-6">

              : {{employee.contactPreference}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Phone

            </div>

            <div class="col-xs-6">

              : {{employee.phoneNumber}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Email

            </div>

            <div class="col-xs-6">

              : {{employee.email}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Department

            </div>

            <div class="col-xs-6">

              : {{employee.department}}

            </div>

          </div>

          <div class="row">

            <div class="col-xs-6">

              Is Active

            </div>

            <div class="col-xs-6">

              : {{employee.isActive}}

            </div>

          </div>

        </div>

      </div>

    </div>

  </div>

</div>

**Changes in list-employees.component.css :** Include the following CSS classes

.imageClass{

    width:200px;

    height:200px;

}

.vertical-align{

    display: flex;

    align-items: center;

}

**Changes in app.component.html :**Include the ListEmployeesComponent selector (app-list-employees) as a directive in the root component (app.component.html)

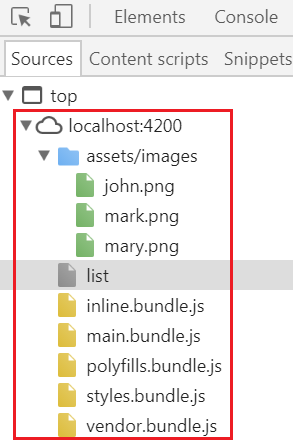
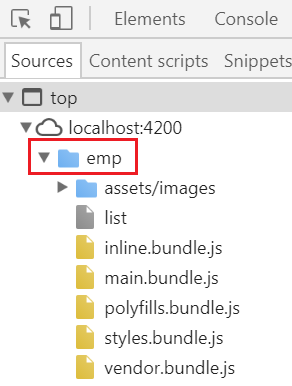
<div class="container">

    <app-list-employees></app-list-employees>

</div>

At this point, save all the changes and run the angular project using the following command. This command not only compiles the angular application, it also launches your default browser and displays the list of employees as expected.  
ng serve -o  
  
We have just seen how to perform the READ operation. In our next video, we will set up routing.

# 4. Angular base href

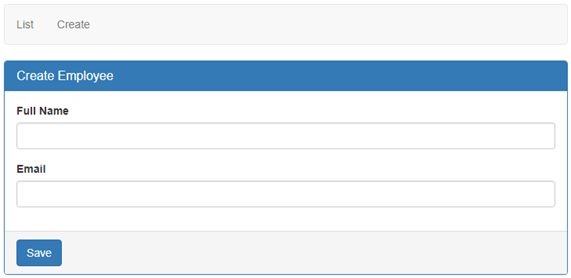
When setting up routing in an angular application, the first step is to set the base path using the base href element. The base path tells the angular router, how to compose the navigation URLs. The browser uses the <base href> value to prefix relative URLs when referencing CSS files, scripts, and images.   
  
  
During development we usually set this to a single forward slash as shown below.   
<base href="/">  
  
This means all the URLs now will be relative to the root of the application. So when we navaigate to "/list", the path "/list" will be appended to root UR and the complete URL will be as shown below. Notice "/list" is relative to the root URL.  
http://localhost:4200/list  
  
Along the same lines, when we navigate to "/create", the complete URL is http://localhost:4200/create  
  
When we deploy our application to a server, we typically deploy it to a sub folder on the server. For example, if we are deploying our application in a sub-folder called **"emp"**, then we set the base href element to /emp/ as shown below.  
<base href="/emp/">  
  
This means all the URLs now will be relative to the "emp" base path and will be as shown below.  
http://serverName/emp/list  
http://serverName/emp/create  
  
During development we usually set base href element to a single forward slash as shown below.   
<base href="/">  
  
At this point, if we execute the following command, all the URLs will be relative to the root URL "http://localhost:4200"  
ng serve -o   
  
Also, on the **"sources"** tab in the browser developer tools, you will find all the Script,  Images and Template files are relative to the root URL **"http://localhost:4200"** as shown in the image below.   
   
  
During development, if you want a different base path other than "/", simply execute the "ng serve" command with --base-href option set to your desired base path as shown below.  
ng serve -o --base-href /emp/  
  
At this point all the URLs will be relative to "http://localhost:4200/emp" as we have set the --base-href to /emp/. You can confirm this by looking at the URLs in the address bar and the "Sources" tab in the browser developer tools.   
   
  
On your local development machine, if you set the base href element in index.html to "/emp/" instead of a single "/" and if you run ng serve -o command without the "base-href" option  you will not see anything on the browser. When you open the browser developer tools, you will see the JavaScript bundle files failed to load. To fix this execute ng serve command along with the base href option as shown below.  
ng serve -o --base-href /emp/   
  
On your local development machine, if you set the base href element in index.html to a single forward slash and if you want to deploy your application on a server on sub-folder called "emp", then you will have to remember to update the base href element value in index.html to "/emp/". There are 2 ways we can do this. 

1. Manually update the index.html file OR
2. Use the --base-href option along with the ng build command as shown below. This will update the "base href" element value index.html.ng build --base-href /emp/

# 5. Angular forms tutorial

In [Part 2](https://www.youtube.com/watch?v=tPySwBVmGvg) of [Angular CRUD tutorial](https://www.youtube.com/watch?v=JYPyy-hvjYc&list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5), we discussed performing the READ operation. In this video and in the next few videos we will discuss performing the CREATE operation. To understand the CREATE operation, let us build a form that help us create a new employee. For this we will use the **createEmployee**component that we already created in one of our previous videos in this series. Along the way, we will also discuss performing validation and displaying meaningful error messages to the user.   
  
  
**There are 2 ways to create forms in Angular** 

1. Template Driven Forms
2. Model Driven Forms (Commonly called Reactive Forms)

**Both these approaches have their own pros and cons**. For example, Template Driven forms are generally used to create simple forms. On the other hand, Reactive forms are used to create complex forms. For example, if you want to add form controls dynamically or perform cross-field validation we use the Reactive forms approach. There are several other differences, between Template driven and Reactive forms. We will discuss those differences in detail, in a later video.   
  
In this video, we will use the Template driven approach to build the **"Create Employee"** form. As the name implies, template driven forms are heavy on the template. This means we do most of the work in the view template of the component.   
  
We want to design our "Create Employee" form as shown below. To keep this simple, at the moment we only have 2 fields (Full Name & Email). We will add the other fields like Gender, Department, Phone Number etc.. later. Also, at the moment, we only have textboxes on our form. In our upcoming videos we will discuss working with radio buttons, checkbox, dropdownlist etc   
   
  
Replace the HTML in "create-employee.component.html" file with the following HTML 

<form #employeeForm="ngForm" (ngSubmit)="saveEmployee(employeeForm)">

  <div class="panel panel-primary">

    <div class="panel-heading">

      <h3 class="panel-title">Create Employee</h3>

    </div>

    <div class="panel-body">

      <div class="form-group">

        <label for="fullName">Full Name</label>

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

               name="fullName" [(ngModel)]="fullName">

      </div>

      <div class="form-group">

        <label for="email">Email</label>

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

               name="email" [(ngModel)]="email">

      </div>

    </div>

    <div class="panel-footer">

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

    </div>

  </div>

</form>

Angular Generated Form Model : {{employeeForm.value | json}}   
  
**Code Explanation:**  
We are using Bootstrap CSS classes like panel, panel-primary, panel-heading, panel-title etc to style the form. There is no Angular here. If you are new to bootstrap, [please click here to check out our Bootstrap tutorial](http://csharp-video-tutorials.blogspot.com/2016/05/bootstrap-tutorial-for-beginners.html).  
  
**Consider the following line of code**  
<form #employeeForm="ngForm" (ngSubmit)="saveEmployee(employeeForm)">  
  
**#employeeForm**is called the template reference variable. Notice we have assigned "ngForm" as the value for the template reference variable employeeForm. So employeeForm variable holds a reference to the form. When Angular sees a form tag, it automatically attaches the ngForm directive to it. The ngForm directive supplements the form element with additional features. It holds all the form controls that we create with ngModel directive and name attribute, and monitors their properties like value, dirty, touched, valid etc. The form also has all these properties. We will discuss these properties at the individual control level and at the form level in detail in our upcoming videos.  
  
**The ngSubmit directive** submits the form when we hit the enter key or when we click the Submit button. When the form is submitted, saveEmployee() method is called and we are passing it the employeeForm. We do not have this method yet. We will create it in the component class in just a bit.  
  
**The ngForm directive** is provided by Angular FormsModule. So for us to be able to use it, we will have to import the FormsModule in our AppModule file (app.module.ts). So please make sure to include the following import statement. Also include "FormsModule" in the imports array of @NgModule decorator.  
import { FormsModule } from '@angular/forms';  
  
If "FormsModule" is not imported you will see the following error in the browser developer toolsthere is no directive with exportas set to ngform  
  
**Consider the following block of code** 

<div class="form-group">

  <label for="fullName">Full Name</label>

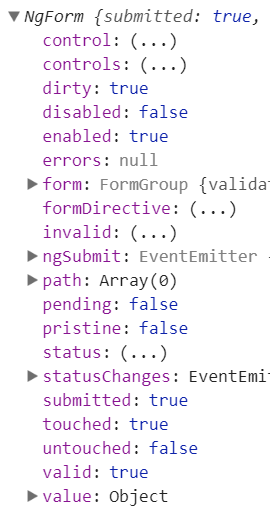
  <input id="fullName" type="text" class="form-control"

          name="fullName" [(ngModel)]="fullName">

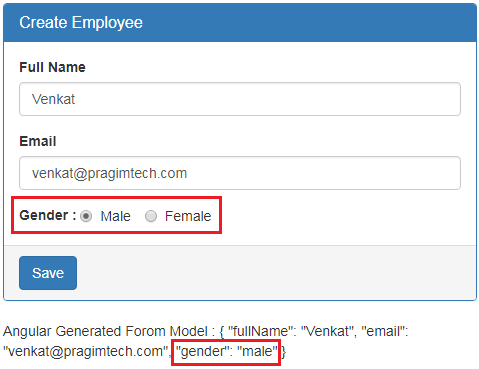
</div>

1. To style the "Full Name" field and it's associated label, we are using Bootstrap. So "form-group" and "form-control" are Bootstrap CSS classes used for styling. There is no Angular here.
2. The "for" attribute on the label, is used to link the label with it's associated "fullName" input control. With the "for" attribute in place, when we click on the label, it's associated input element automatically receives the focus. Again there is no Angular here. It's all standard HTML.
3. The **ngModel**directive is used for creating two-way data binding i.e to keep the HTML element value and it's corresponding component property in sync. We discussed two-way data binding in detail in our [Angular 2 course](https://www.youtube.com/watch?v=WWQZCDegWHg&list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6). [Click here to watch two-way data binding video.](https://www.youtube.com/watch?v=aBf1nLGuVz8&index=15&list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6)
4. Notice we have set **ngModel** directive to "fullName". We do not have "fullName" property in the component class. Angular automatically creates "fullName" property using the value of the "name" attribute of the HTML input element. This is why "name" attribute is also required when we use **ngModel**directive. If we remove the "name" attribute, we get the following error.  
   If ngModel is used within a form tag, either the name attribute must be set or the form control must be defined as 'standalone' in ngModelOptions
5. So the bottom line is, if you want an input element to be tracked by the form make sure to include both the **name attribute**and **ngModel**directive. Otherwise that input element will not be part of the Form model created by Angular.

**Consider the following piece of code :**We are using the value property of the employeeForm to display fullName and email property values of the Form Model that angular automatically generates for us. We are using the Angular "json" pipe to format the JSON data.

**Angular Generated Forom Model : {{employeeForm.value | json}}**  
  
Finally in the CreateEmployeeComponent class include the following saveEmployee() method. At the moment we are simply logging the value of the Angular generated Form model to the console. In our upcoming videos, we will discuss how to save the new employee to a database table.   
  
saveEmployee(employeeForm: NgForm): void {  
  console.log(employeeForm.value);  
}  
  
**Please note :**Make sure to import **NgForm**type from '@angular/forms'  
import { NgForm } from '@angular/forms';  
  
Remember we discussed, The **ngForm**directive supplements the form element with additional features and properties like value, dirty, touched, valid etc. To see all these properties, knock of the value property and log just the employeeForm as shown below.  
  
saveEmployee(employeeForm: NgForm): void {  
  console.log(employeeForm);  
}  
  
At this point, if you fill in the Full Name and Email text boxes and when you submit the form either by click the "Save" button or by pressing the "Enter" key you will see the form logged to the browser console and you can see all these properties.   
   
  
These properties are greatly useful for performing form validation. We will discuss them in detail in our upcoming videos.

# 6. Bootstrap radio buttons in Angular

In this video we will discuss **working with radio buttons in Angular Template Driven forms**.   
  
We want to include **"Gender"** radio buttons in the Create Employee form as shown below. When we select employee "Gender" using the radio buttons, the selected gender value should reflect in the Angular generated form model as shown in the image below. Also, we we click the **"Save"** button we want the selected gender value to be logged to the console.   
   
  
  
To achieve this all you have to do is include the following HTML in **create-employee.component.html** file 

<div class="form-group">

  <label>Gender</label>

  <div class="form-control">

    <label class="radio-inline">

      <input type="radio" name="gender" value="male" [(ngModel)]="gender">

      Male

    </label>

    <label class="radio-inline">

      <input type="radio" name="gender" value="female" [(ngModel)]="gender">

      Female

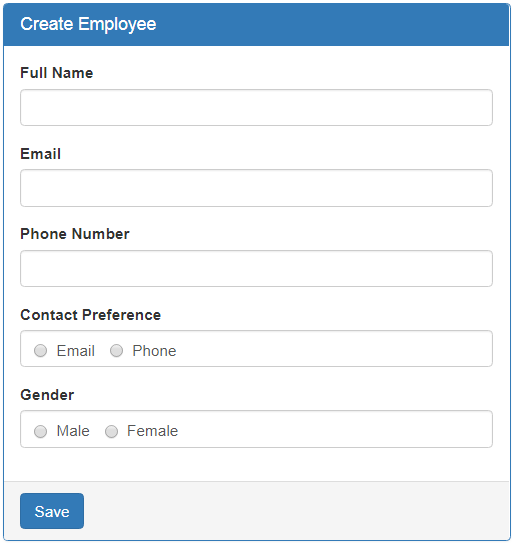
    </label>

  </div>

</div>

**Code Explanation** 

* The **name**attribute is required to group the radio buttons as one unit and make the selection mutually exclusive. Make sure both the radio buttons have the same value for the **"name"** attribute. Otherwise the radio button selection won't be mutually exclusive.
* It is also important that you set the **"value"** attribute for each radio button. This value is posted to the server when the form is submitted.

While we are here, let's also include a textbox to capture **"Phone Number"** and **"Contact Preference"** radio button. So the form should now look as as shown below.   
   
  
For your referece, here is the complete HTML in create-employee.component.html  

<form #employeeForm="ngForm" (ngSubmit)="saveEmployee(employeeForm)">

  <div class="panel panel-primary">

    <div class="panel-heading">

      <h3 class="panel-title">Create Employee</h3>

    </div>

    <div class="panel-body">

      <div class="form-group">

        <label for="fullName">Full Name</label>

        <input id="fullName" type="text" class="form-control" name="fullName"

        [(ngModel)]="fullName">

      </div>

      <div class="form-group">

        <label for="email">Email</label>

        <input id="email" type="text" class="form-control" name="email"

        [(ngModel)]="email">

      </div>

      <div class="form-group">

        <label for="phoneNumber">Phone Number</label>

        <input id="phoneNumber" type="text" class="form-control" name="phoneNumber"

        [(ngModel)]="phoneNumber">

      </div>

      <div class="form-group">

        <label>Contact Preference</label>

        <div class="form-control">

          <label class="radio-inline">

            <input type="radio" name="contactPreference" value="email"

            [(ngModel)]="contactPreference">

            Email

          </label>

          <label class="radio-inline">

            <input type="radio" name="contactPreference" value="phone"

            [(ngModel)]="contactPreference">

            Phone

          </label>

        </div>

      </div>

      <div class="form-group">

        <label>Gender</label>

        <div class="form-control">

          <label class="radio-inline">

            <input type="radio" name="gender" value="male" [(ngModel)]="gender">

            Male

          </label>

          <label class="radio-inline">

            <input type="radio" name="gender" value="female" [(ngModel)]="gender">

            Female

          </label>

        </div>

      </div>

    </div>

    <div class="panel-footer">

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

    </div>

  </div>

</form>

Angular Generated Forom Model : {{employeeForm.value | json}}

# 7. Angular radio button checked by default

**In this video we will discuss**

* How to have a radio button checked by default
* How to disable a radio button

**How to get a radio button checked by default in Angular :**Consider the following HTML, that displays "Gender" radio buttons 

<div class="form-group">

  <label>Gender</label>

  <div class="form-control">

    <label class="radio-inline">

      <input type="radio" name="gender" value="male" [(ngModel)]="gender" >

      Male

    </label>

    <label class="radio-inline">

      <input type="radio" name="gender" value="female" [(ngModel)]="gender">

      Female

    </label>

  </div>

</div>

If we include **checked**attribute on one of the radio buttons, we expect that radio button to be checked by default when the form initially loads. But you will discover that is not the case. In the following example, we have included "checked" attribute on "Male" radio button, but when the form is displayed it is not checked. 

<input type="radio" name="gender" value="male" [(ngModel)]="gender" checked>

However, if you remove the "ngModel" directive from the radio button, then it gets checked as expected. Notice the "ngModel" directive is removed from the radio button.   
  
<input type="radio" name="gender" value="male" checked>  
  
With Angular Template Driven forms, we use "ngModel" directive for two-way data binding. So the moment we put it back in place the "checked" attribute does not work. To make it work include **"gender"** property in the component class and initialise to the value of the radio button that you want to have checked by default. In our case, let us say, we want the "Male" radio button to be checked by default. To achieve this include "gender" property initialised to value of "male" in the component class as shown below.  
  
gender = 'male';  
  
At this point you will have "Male" radio button checked by default when the form loads. Now, even if we remove the "**checked**" attribute from the "Male" radio button it is still checked by default when the form loads. This is because of the two-way data binding that we get with "ngModel" directive. For our form we do not want any radio button to be checked by default, so remove the "checked" attribute and the "gender" property from the component class.  
  
How to disable a radio button : To disable a radio button, use the disabled attribute on that radio button. "Male" radio button in this case will be disabled when the form initially loads. 

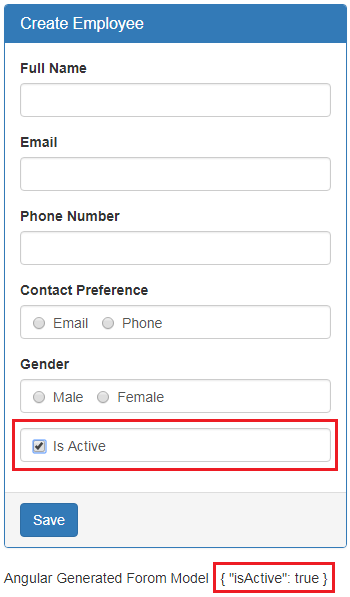
<input type="radio" name="gender" value="male" [(ngModel)]="gender" disabled>

Another important point to keep in mind. By default, disabled form controls are not included in the Angular auto generated form model. Since, the "Male" radio button is disabled, the gender property will not be included in the Angular generated form model.  
  
In our form, we do not want any radio button to be disabled, so please remove the disabled attribute.  
  
In our next video, we will discuss working with **CheckBox control in Angular Template Driven forms**.

# 8. Bootstrap checkbox in angular

**In this video we will discuss**

1. Working with a checkbox control in Angular Template Driven forms
2. How to have a checkbox checked by default
3. How to disable a checkbox

**Working with a checkbox in Angular is very similar to working with a radio button**. We want to include "Is Active" checkbox in the **Create Employee**form as shown below. When we check the checkbox, "isActive" property should reflect in the Angular generated for model as shown in the image below. Also, when we click the "Save" button we want the "isActive" property value to be logged to the console.   
   
  
  
To achieve this all you have to do is include the following HTML in **create-employee.component.html**file 

<div class="form-group">

  <div class="form-control">

    <label class="checkbox-inline">

      <input type="checkbox" name="isActive" [(ngModel)]="isActive">Is Active

    </label>

  </div>

</div>

If we include **"checked"** attribute on a checkbox, we expect checkbox to be checked by default when the form initially loads. But you will discover that is not the case.

<input type="checkbox" name="isActive" [(ngModel)]="isActive" checked>Is Active

However, if you remove the **"ngModel"** directive from the checbox, then it gets checked as expected. Notice the "ngModel" directive is removed from the checkbox.   
  
<input type="checkbox" name="isActive" checked>Is Active  
  
With Angular Template Driven forms, we use **"ngModel"** directive for two-way data binding. So the moment we put it back in place the **"checked"** attribute does not work. To make it work include **"isActive"** property in the component class and initialise it to true.  
  
isActive = true;   
  
At this point you will have **"Is Active"** checkbox checked by default when the form loads. Now, even if we remove the **"checked"** attribute from the checkbox it is still checked by default when the form loads. This is because of the two-way data binding that we get with "ngModel" directive. For our form we do not want the checkbox to be checked by default, so remove the "checked" attribute and the "isActive" property from the component class.  
  
How to disable a checkbox : To disable a checkbox, use the disabled attribute 

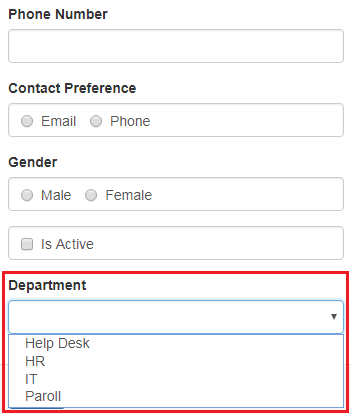
<input type="checkbox" name="isActive" [(ngModel)]="isActive" disabled>Is Active

Another important point to keep in mind. By default, disabled form controls are not included in the Angular auto generated form model. Since, the "Is Active" checkbox is disabled, it will not be included in the Angular generated form model.  
  
In our form, we do not want the checkbox to be disabled, so please remove the **disabled**attribute.

# 9. Angular Bootstrap select list

**In this video we will discuss** 

* Working with a select list in Angular Template Driven forms
* How to have one of the select list option selected by default
* How to disable select list

Let us **understand working with a select element in Angular with a simple example**. We want to include "Department" select list as shown in the image below.   
   
  
  
**Here is the HTML for the "Department" select list**

<div class="form-group">

  <label for="department">Department</label>

  <select id="department" name="department"

          [(ngModel)]="department" class="form-control">

    <option value="1">Help Desk</option>

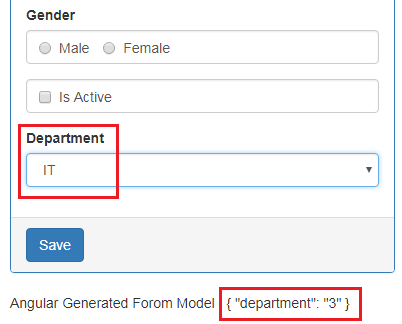
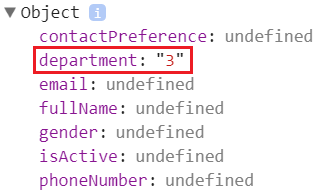
    <option value="2">HR</option>

    <option value="3">IT</option>

    <option value="4">Paroll</option>

  </select>

</div>

At the moment, we have hard coded the select list options in the HTML. In our next video we will discuss, how to get the select list options from the component class. Notice each option also has a corresponding value. The value is the department id which is what we want to save in the database table when the form is submitted. We will discuss, saving the data to a database table in a later video.  
  
At this point, when we select an option, notice the corresponding option value is included against the **"department"**property in the **Angular auto-generated form model**.   
   
  
Also notice, when we click the **"Save"** button, the **"department"**property along with the selected option value is logged to the console in browser developer tools.   
   
  
**How to have one of the select list option selected by default**  
  
If we include **"selected"** attribute on one of the options of the select list, we expect that option to be selected by default when the form initially loads. In the example below, we have included the "selected" attribute on the **"IT"** option, but when the form reloads, the **"IT"**option is not selected.  
  
<option value="3" selected>IT</option>  
  
If you remove the **"ngModel"** directive from the select list, then the the **"IT"** option gets selected as expected. Notice the **"ngModel"** directive is removed from the select list. 

<div class="form-group">

  <label for="department">Department</label>

  <select id="department" name="department" class="form-control">

    <option value="1">Help Desk</option>

    <option value="2">HR</option>

    <option value="3" selected>IT</option>

    <option value="4">Paroll</option>

  </select>

</div>

In Angular, we use **"ngModel"** directive for two-way data binding. So the moment we put it back in place the **"selected"** attribute does not work. To make it work include **"department"** property in the component class and initialise it with one of the option value which you want to have selected by default. In our case, we want the **"IT"** option to be selected by default. The **"IT"** option value is **"3"**. So, I have initialised **"department"** property with a value of **'3'**  
  
department = '3'  
  
At this point you will have **"IT"** option selected by default when the form loads. Now, even if we remove the **"selected"** attribute from the "IT" option, it is still selected by default when the form loads. This is because of the two-way data binding that we get with **"ngModel"** directive.   
  
**How to disable a select list :**To disable a select element, use the **disabled**attribute 

<select id="department" name="department" [(ngModel)]="department"

        class="form-control" disabled>

Another important point to keep in mind. By default, d**isabled form controls are not included in the Angular auto generated form model**. Since, the "department" select element is disabled, it will not be included in the Angular generated form model.   
  
In our form, we do not want the select element to be disabled, so please remove the **disabled**attribute. Also, we do not want any option to be selected by default, so remove the **"department"** property from the component class.  
  
In our next video, we will discuss, **how to get the select list options from the component class**, instead of having them **hard-coded in the HTML.**

# 10. Angular select options from array

In this video, we will discuss, how to get the select list options from an array in the component class, instead of having them hard-coded in the HTML. This is continuation to [Part 9](https://www.youtube.com/watch?v=HMK4P_jx0y8&index=9&list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5). So, please watch [Part 9](https://www.youtube.com/watch?v=HMK4P_jx0y8&index=9&list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5) from [Angular CRUD tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5) before proceeding.   
  
**Step 1 :** Create the **Department**class.    
Add a TypeScript file to the models folder. Name it department.model.ts. Copy and paste the following code. Notice the Department class has 2 properties - id and name of the department.   
  
export class Department {  
    id: number;  
    name: string;  
}   
  
**Step 2 :** Import the Department class    
  
Include the following import statement in create-employee.component.ts file  
import { Department } from '../models/department.model';  
  
**Step 3 :**Include the following array of departments in CreateEmployeeComponent class in create-employee.component.ts file  
  
departments: Department[] = [  
  { id: 1, name: 'Help Desk' },  
  { id: 2, name: 'HR' },  
  { id: 3, name: 'IT' },  
  { id: 4, name: 'Payroll' }  
];  
  
**Please note :**The "Department" type is not required for the application to work, but it adds great value during development. Using it provides us intellisense, error checking and type saftey.   
  
**Step 4 :**In create-employee.component.html file, modify the HTML that displays the "Department" dropdownlist as shown below.

<div class="form-group">

  <label for="department">Department</label>

  <select id="department" name="department" [(ngModel)]="department"

          class="form-control">

    <option \*ngFor="let dept of departments" [value]="dept.id">

      {{dept.name}}

    </option>

  </select>

</div>

**Code explanation:**

* On the "option" element we are using ngFor structural directive to loop over the array of departments we have in the "departments" property of the component class
* For each "Department" object in the "departments" array, we get an option.
* The option value is the **department id** and the display text is the **department name**
* Notice the square brackets around the [value] property. This is property binding in Angular. We discussed property binding in detail in [Part 9](https://www.youtube.com/watch?v=RGYfTx9AAQA&list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6&index=9) of [Angular 2 tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6). If you remove the square brackets the value for each option will be the literal text "dept.id" instead of the department id (1 or 2 or 3 etc.)
* To display the deprtment name we are using **interpolation**. We discussed interpolation in [Part 8](https://www.youtube.com/watch?v=2FkkAqFWRk8&list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6&index=8) of [Angular 2 tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhWqGD8BuKuX-VTKqlNBj-m6).
* Since ngFor is a structural directive there is an asterisk before it.
* Structural directives modify the DOM, i.e they add or remove the elements from DOM. Adding and removing elements from DOM is different from showing and hiding. We will discuss all these in detail in our upcoming videos.

At this point, when we select a department, the respective department id is included in the Angular generated form model. Along the same lines, when we click the "Save" button the respective department id is logged to the console.  
  
**Please note :**It is important that we include the ngFor directive on the element that we want to be repeated. In our case we want an option element for each department we have in the array. So we included the **ngFor**directive on the option element. If we instead include the **ngFor**directive on the "div" element that has the bootstrap "form-group" class as shown below.

<div class="form-group" \*ngFor="let dept of departments">

  <label for="department">Department</label>

  <select id="department" name="department" [(ngModel)]="department"

          class="form-control">

    <option [value]="dept.id">

      {{dept.name}}

    </option>

  </select>

</div>

We get 4 department dropdownlists. That is one for each department in the array. So it is important we include the ngFor directive on the right element.   


# 11. Angular datepicker tutorial

**In this video we will discuss**

* Why is not a good practice to use the browser built-in DatePicker control
* Installing ngx-bootstrap
* Using ngx-bootstrap datepicker in Angular

**Why is not a good practice to use the browser built-in DatePicker control :**This is because the implem entation of datepicker is different from browser vendor to vendor. This means our end users may have different experience depending on the browser they use. Let us understand this with an example.   
  
  
On our **"Create Employee"** form we want to capture **Date of Birth** of an employee. **Datepicker**control is very useful in capturing dates from users. When we use the HTML5 input type date, the browser automatically displays it's built-in datepicker control. Include the following piece of HTML on "create-employee.component.html" file just below the "Department" field HTML 

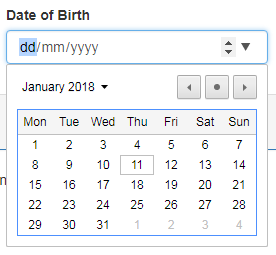
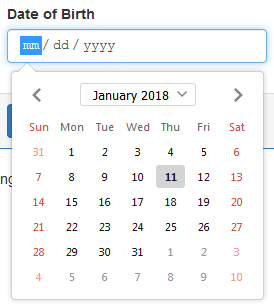
<div class="form-group">

  <label for="dateOfBirth">Date of Birth</label>

  <input id="dateOfBirth" name="dateOfBirth" [(ngModel)]="dateOfBirth"

          type="date" class="form-control" />

</div>

Notice we have set the input element type to date. At this point if we run the project and navigate to http://localhost:4200/create in Google chrome, we see the date-picker as shown below.   
   
  
Now, if we navigate to the same url in firefox, we see a date-picker control that is very different from the date-picker control that is on Google chrome browser.   
   
  
So, this means our end users have different experience depending on the browser they use. What we want here is consistency. There are many third party Date-picker controls that we can use, to provide consistent experience to our end users. ngx-bootstrap datepicker control is one of them.   
  
Please refer to the UI components section on the following page, to see the list of all third party UI components that we can use in Angular  
<https://angular.io/resources>  
  
**Installing ngx-bootstrap :**The following are the steps to install ngx-bootstrap   
  
**Step 1 :** Execute the following command to npm install ngx-bootstrap  
npm install ngx-bootstrap --save  
  
**Step 2 :**If you do not have Bootstrap installed, please install it using the following npm command. If you are following along we have already installed bootstrap in Part 1 of this Angular CRUD tutorial. So I am not going to execute this command again.  
npm install bootstrap@3 --save   
  
**Please note :**We are usng Bootstrap 3. We can also use Bootstrap 4 with ngx-bootstrap. Please refer to the documentation available at the following link on how to use Bootstrap 4 with ngx-bootstrap.  
<https://valor-software.com/ngx-bootstrap/#/getting-started>  
  
**Step 3 :**Once Bootstrap is installed, open .angular-cli.json file and specify the path to the Bootstrap stylesheet (bootstrap.min.css) in the styles property as shown below. Again, we have already done this in [Part 1](https://www.youtube.com/watch?v=JYPyy-hvjYc) of [Angular CRUD tutorial](https://www.youtube.com/watch?v=JYPyy-hvjYc&list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5).

"styles": [

  "../node\_modules/bootstrap/dist/css/bootstrap.min.css",

  "styles.css"

]

**Using ngx-bootstrap datepicker in Angular :** The following are the steps to use ngx-bootstrap datepicker in Angular  
  
**Step 1 :**In app.module.ts file, include the following import statement to import BsDatepickerModule   
import { BsDatepickerModule } from 'ngx-bootstrap/datepicker';  
  
Also, include BsDatepickerModule in the imports array of @NgModule decorator as shown below  
@NgModule({  
  imports: [BsDatepickerModule.forRoot(),...]  
})  
  
**Step 2 :**In "create-employee.component.html" file, make the following 2 changes to the HTML that displays the "Date of Birth" field 

* Change the "type" attribute value from "date" to "text"
* Include "bsDatepicker" directive on the input element

<div class="form-group">

  <label for="dateOfBirth">Date of Birth</label>

  <input id="dateOfBirth" name="dateOfBirth" [(ngModel)]="dateOfBirth"

          type="text" bsDatepicker class="form-control" />

</div>

**Step 3 :**Include a reference to the bs-datepicker.css file in .angular-cli.json file.

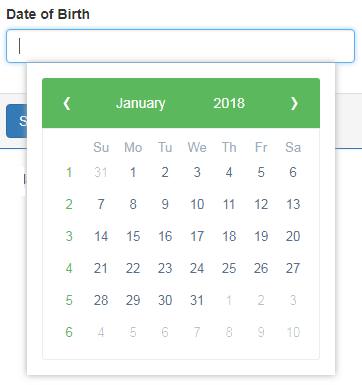
"styles": [

  "../node\_modules/bootstrap/dist/css/bootstrap.min.css",

  "../node\_modules/ngx-bootstrap/datepicker/bs-datepicker.css",

  "styles.css"

]

At this point when you view the page in Google chrome or Firefox, you get the same datepicker and hence the same experience.   
   
  
When we select a date from the date-picker control, the "Date of Birth" textbox is automatically populated with the selected date and it is also captured by the angular generated form model.  
  
**With this datepicker control, it is also very easy to capture a date range.** For example, you have an open job role, and you want to capture a date range for accepting CV's, we can very easily do this. All we have to do is use bsDaterangepicker directive instead of bsDatepicker directive on the input element as shown below. 

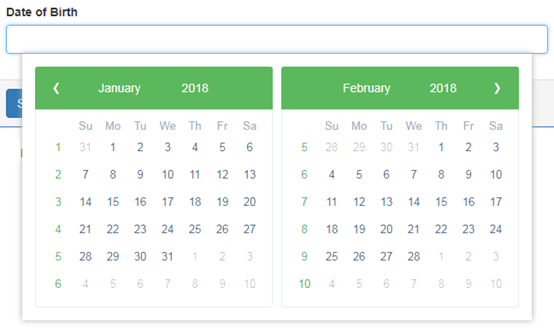
<div class="form-group">

  <label for="dateOfBirth">Date of Birth</label>

  <input id="dateOfBirth" name="dateOfBirth" [(ngModel)]="dateOfBirth"

          type="text" bsDaterangepicker class="form-control" />

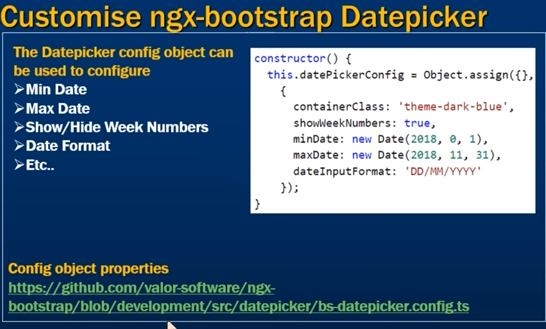
</div>

The above simple change, will display Daterange picker as shown below. When we select a date range, the corresponding input field is automatically populated with the selected date range and it is also captured by the angular generated form model.   
 

* At the moment, the Datepicker is using the **default green theme**. We want to change it to **dark-blue theme**, so it matches with the rest of the form.
* The date is captured in the textbox in mm/dd/yyyy format. We want to change it to dd/mm/yyyy format
* At the moment there is no default date. We want to set a default date
* The input element is spanning across the entire width of the form. We want to limit it's width

Datepicker is a highly configurable component. We will discuss how to do all of the above in our next video.

# 12. Customise ngx bootstrap datepicker

In this video we will discuss **customising the ngx-bootstrap datepicker**component with an example. This is continuation to [Part 11](https://www.youtube.com/watch?v=edaN6iCcqP4). Please watch [Part 11](https://www.youtube.com/watch?v=edaN6iCcqP4) from [Angular CRUD tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5) before proceeding.   


At the moment we have date range picker. Use bsDatepicker directive instead of bsDaterangepicker directive on the input element so we can capture single date i.e the Date of Birth of the employee.

<div class="form-group">

  <label for="dateOfBirth">Date of Birth</label>

  <input id="dateOfBirth" name="dateOfBirth" [(ngModel)]="dateOfBirth"

          class="form-control" type="text" bsDatepicker  />

</div>

**Changing ngx-bootstrap datepicker theme :**At the moment, the Datepicker is using the default green theme. We want to change it to dark-blue theme, so it matches with the rest of the form. As of this recording ngx-bootstrap datepicker component has the following 6 color schemes.

1. theme-default
2. theme-green
3. theme-blue
4. theme-dark-blue
5. theme-red
6. theme-orange

We can change the default colour-scheme, by manipulating containerClass property in bsConfig object. Here are the steps.

**Step 1 :**Make the following changes in CreateEmployeeComponent class (i.e create-employee.component.ts file)

// Import BsDatepickerConfig type. This is the Config object for datepicker. Using this

// config object we can set minDate, maxDate, whether to show/hide week numbers and

// change the color theme using the containerClass property.

import { BsDatepickerConfig } from 'ngx-bootstrap/datepicker';

// In the CreateEmployeeComponent class make the following changes

export class CreateEmployeeComponent implements OnInit {

  // create a property of type Partial<BsDatepickerConfig>

  datePickerConfig: Partial<BsDatepickerConfig>;

  // In the constructor set containerClass property to the preferred theme

  constructor() {

    this.datePickerConfig = Object.assign({}, { containerClass: 'theme-dark-blue' });

   }

   // Rest of the code...

}

**Please note :**  
We are using the TypeScript partial type here to set only the "containerClass" property of BsDatepickerConfig object. To learn more about the partial type please refer to the following article.  
<https://www.typescriptlang.org/docs/handbook/release-notes/typescript-2-1.html>  
  
**Object.assign()** copies the property values from one or more source objects to a target object. The target object is the first parameter and the rest are the sources. Object.assign() is useful for merging objects or cloning them shallowly.  
  
**Step 2 :**In the view template (i.e in create-employee.component.html file) bind the "datePickerConfig" property in the component class we created above in Step 1, to the bsConfig input property.

<div class="form-group">

  <label for="dateOfBirth">Date of Birth</label>

  <input id="dateOfBirth" name="dateOfBirth" [(ngModel)]="dateOfBirth"

          class="form-control" type="text" bsDatepicker [bsConfig]="datePickerConfig" />

</div>

At this point, you should see the Datepicker using the **dark-blue** theme colour as shown below.   
   
  
**Showing or hiding week numbers :**By default, the weeknumber are displayed. If you want to hide them, all you have to do is set "showWeekNumbers" boolean property to false in the config object as shown below.

constructor() {

  this.datePickerConfig = Object.assign({},

    {

      containerClass: 'theme-dark-blue',

      showWeekNumbers: false

    });

}

You can find all the properties of the config object at the following page.   
<https://github.com/valor-software/ngx-bootstrap/blob/development/src/datepicker/bs-datepicker.config.ts>  
  
Along the same lines we can also set the **min and max dates**. Please note that the month numbers start from 0 and not 1. So for January it is 0, February it is 1, so on and so forth.

constructor() {

  this.datePickerConfig = Object.assign({},

    {

      containerClass: 'theme-dark-blue',

      showWeekNumbers: true,

      minDate: new Date(2018, 0, 1),

      maxDate: new Date(2018, 11, 31),

    });

}

To change the **date format**, use dateInputFormat property of the config object.

constructor() {

  this.datePickerConfig = Object.assign({},

    {

      containerClass: 'theme-dark-blue',

      showWeekNumbers: true,

      minDate: new Date(2018, 0, 1),

      maxDate: new Date(2018, 11, 31),

      dateInputFormat: 'DD/MM/YYYY'

    });

}

To set a default date, create a property (dateOfBirth) in the component class and set it to the default value you want. Since we are using 2 way databinding, the defualt date is displayed in the corresponding input field when them form loads. In this case we have set default date to January 30, 2018.  
  
dateOfBirth: Date = new Date(2018, 0, 30);  
  
At the moment, the "Date of Birth" input element is spanning across the entire width of the form. There are sevral options to limit it's width. One option is to use the Bootstrap row and grid classes (Example: col-md-4, col-md-5, etc...)

<div class="row">

  <div class="form-group col-md-4">

    <label for="dateOfBirth">Date of Birth</label>

    <input id="dateOfBirth"  name="dateOfBirth" [(ngModel)]="dateOfBirth"

           class="form-control" type="text" bsDatepicker

           [bsConfig]="datePickerConfig" />

  </div>

</div>

To control the placement of the Datepicker use the placement property. The allowed values are "top" | "bottom" | "left" | "right". The default is "bottom".  
  
For our form we do not want a default date to be set. So please remove the dateOfBirth property from the component class. We also do not want minDate and maxDate, so delete these properties as well from the datePickerConfig object. Also delete, showWeekNumbers property as it is set to true by default. This means our datePickerConfig object in the constructor has just 2 properties (dateInputFormat and containerClass)

constructor() {

  this.datePickerConfig = Object.assign({},

    {

      containerClass: 'theme-dark-blue',

      dateInputFormat: 'DD/MM/YYYY'

    });

}

You can find all the properties of the config object at the following page:

<https://github.com/valor-software/ngx-bootstrap/blob/development/src/datepicker/bs-datepicker.config.ts>