Angular

1.Changing DateOfBirth into Blue theme.

2.To make week number off then need to showWeekNumbers:false”­.

example:

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

3.To maintained min or max date .

Example:

constructor() {

this.datePickerConfig = Object.assign({}, {

containerClass: "theme-dark-blue" ,

showWeekNumbers:false,

minDate:new Date(2018, 0, 1),

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

}

4.to make date format need to update in ts

constructor() {

this.datePickerConfig = Object.assign({}, {

containerClass: "theme-dark-blue" ,

showWeekNumbers:false,

minDate:new Date(2018, 0, 1),

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

dateInputFormat:'DD/MM/YYYY'

});

}

5.To get by default date need to update in create.ts file.

dateOfBirth: Date= new Date(2018,11,30);

6.to get DateOfBirth normal size not so big as like above all then need to update in create.html file.

Example:

<div class="row">

<div class="form-group col-md-5">

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

<input type="text" [(ngModel)]="dateOfBirth" [bsConfig]="datePickerConfig" name="dateOfBirth" class="form-control" bsDatepicker>

</div>

</div>

7.to make DateOfBirth place either left,right,Middel have to change.

<div class="row">

<div class="form-group col-md-5">

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

<input type="text" [(ngModel)]="dateOfBirth" [bsConfig]="datePickerConfig"

name="dateOfBirth" class="form-control" bsDatepicker placement="right">

</div>

</div>

Note:

The DatePickerConfig object can be used to configure

Min-date

Max-date

Show/hide Week Numbers

Date Format

Etc...

Lecture:13

**Angular ngif directive**

**Export class create-employee.component{**

previewPhoto= false;

togglePhotoPreview(){

this.previewPhoto= !this.previewPhoto;

}

**}**

<div class="form-group">

<label for="photoPath">Photo Path</label>

<input type="text" class="form-control" id="photoPath" [(ngModel)]="photoPath" name="photoPath">

</div>

<div class="form-group">

<button class="btn btm-primary" type="button" (click)="togglePhotoPreview()">

{{previewPhoto?"Hide":"Show"}} Preview</button>

</div>

<div class="form-group">

<img [src]="photoPath" height="200" width="200" \*ngIf="previewPhoto" />

</div>

1. **Every time clicking show preview button then submit alys executing to fix this issue need to write type=”button” in create.html.**

**Example->**

</div>

<div class="form-group">

<button class="btn btm-primary" type="button" (click)="togglePhotoPreview()" >Show Preview</button>

</div>

2.now some enhacement if photo has been showed then need one button which will behave as a Hide Preview or Show Vise-Versa...

Example->

<div class="form-group">

<button class="btn btm-primary" type="button" (click)="togglePhotoPreview()">

{{previewPhoto?"Hide":"Show"}} Preview</button>

</div>

Lecture 14.

Angular disable browser validation

1. By default Angular 4and later version disable brower native Validation.

How to Enable Browser validation.

2.to make enable

<form #employeeForm="ngForm" ngNativeValidate (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" required [(ngModel)]="fullName" name="Fullname" type="text" class="form-control" placeholder="Enter your Fullname"

/>

</div>

**Lecture #14.** **Angular form validation**

**Angular** form validation properties->

**1.Touched**

**2.Un-Touched**

**3.Pristine**

**4.Dirty**

**5.Valid**

**6.Invalid**

**HTML5 validation Attributes**

**1.required**

**2.maxLength**

**3.pattern**

**4.min**

**5.max**

**To check validation properties we will do some changes**

<div class="panel-body">

<div class="form-group">

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

<input id="fullName" required [(ngModel)]="fullName" name="Fullname" type="text" class="form-control" placeholder="Enter your Fullname"

#fullNameControl="ngModel" />

</div>

<div>

touched ->{{fullNameControl.touched}}<br>

un-touched ->{{fullNameControl.untouched}}

</div>

Note->

1.To include the html5 validation attribute such as requied for example:

<input id="fullName" type=”text”/>

2.Export ngModel directive to a local templte variable

<input id=”fullName” required #fullNameControler=”ngModel” type=”text”/>

3.use the local template variable to access the validation properties (touched,untouched,dirty,valid,etc).

<div>touch{{ fullNameControler .touched}}</div>

<div>untouch{{ fullNameControler .untouched}}</div>

<div>dirty{{ fullNameControler .dirty}}</div>

4.to check entire for validation properties

Export ngForm directive to a local template reference variable.

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

5.use the template reference variable to access the validation properties at form lavel (touched,valid,dirty).

<div>touch{{ employeeForm.touched}}</div>

<div>untouch{{ employeeForm.untouched}}</div>

<div>dirty{{ employeeForm.dirty}}</div>

**Lecture #16 .** **Displaying angular form validation error messages**

**BootStarp classes for styling validation error messages.**

**->has-error**

**->control-label**

**->help-block**

<div class="form-group" [class.has-error]="fullNameControl.invalid && fullNameControl.touched" [class.has-success]="fullNameControl.valid">

<label for="fullName" class="control-label">Full Name</label>

<input id="fullName" required [(ngModel)]="fullName" name="Fullname" type="text" class="form-control" placeholder="Enter your Fullname"

#fullNameControl="ngModel" />

<span class="help-block" \*ngIf="fullNameControl.invalid && fullNameControl.touched" >FullName Required</span>

</div>

**Note->**

**To make from submit button disable.**

<button type="submit" class="btn btn-primary" [disabled]="employeeForm.invalid">Save</button>

**Lecture #17.** **Model binding in angular template driven forms**

1.Eailer discussed Binding Angular form to our own Model.

Error-> can not assign a variable or reference.

At the moment, in **CreateEmployeeComponent**we are using the Angular Auto-generated form model. Instead of using the Angular generated form model, we can use our model class.

In employee.model.ts file in the models folder, we have Employee class. We want to use this class as the model when creating a new employee. Here are the steps.

**Step 1 :** In create-employee.component.ts file, import the Employee model  
import { Employee } from '../models/employee.model';

**Step 2 :** In CreateEmployeeComponent class, include **employee**property. Notice we have set the type to Employee and initialised all properties with NULL value.   
  
export class CreateEmployeeComponent implements OnInit {  
  employee: Employee = {  
    id: null,  
    name: null,  
    gender: null,  
    contactPreference: null,  
    phoneNumber: null,  
    email: null,  
    dateOfBirth: null,  
    department: null,  
    isActive: null,  
    photoPath: null  
  };

**Step 3 :** In the view template, bind the ngModel directive of an input field to it's corresponding property on the employee object. The employee property we created in Step 2 returns an employee object, which is the model for our form.   
  
For example, bind ngModel directive on the email input field to the email property on the employee object.  
[(ngModel)]="employee.email"

Except fullName, bind the ngModel directive of the rest of the input fields with the corresponding properties on the employee object.   
  
In the employee class we do not have **fullName**property. we have **name**instead. On the view template, the corresponding input field name is fullName. To keep things consistent let's change fullName to name on the label and the input field as shown below. 

<div class="form-group" [class.has-error]="name.invalid && name.touched">

  <label for="name" class="control-label">Name</label>

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

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

  <span class="help-block" \*ngIf="name.invalid && name.touched">

    Name is required

  </span>

</div>

At this point, if you view the page in the browser, you will see the following error.  
**Cannot assign to a reference or variable**   
  
We get this error because, Angular generated form model creates **name** property and we are also creating a local template variable with the same name by exporting **ngModel**to **#name**. Hence we get the error - Cannot assign to a reference or variable.   
  
One way to fix this error is, by giving our local template reference variable a different name other than ~~name~~. So if we change #name="ngModel" to #nameControl="noModel" the error goes away. We discussed this in detail in [Part 15](https://www.youtube.com/watch?v=d8XONHXTv_4) of [Angular CRUD tutorial](https://www.youtube.com/playlist?list=PL6n9fhu94yhXwcl3a6rIfAI7QmGYIkfK5).   
  
The other way to fix this error is by using our own model. Using the **ngModel**directive, bind the **name**property of the employee object to the **name**input field 

[(ngModel)]="employee.name"

At this point, if you view the page in the browser and notice the error is gone and all the properties in the Angular generated form model are NULL as expected.   
  
To see our own employee model, include the following code in the view template file (create-employee.component.html) 

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

<br/>

<br/>

Our Employee Model : {{ employee | json}}

At this point, on the browser we should see both - Angular generated form model and our own employee model. Notice as we change the values in the input fields, the respective properties in both the models are updated as expected.   
  
At the moment, when we click the "Save" button, we are logging the **employeeForm.value**to the console. We instead want to log our employee model object. To do this   
In the view template, pass the employee object to the saveEmployee() method.

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

Modify saveEmployee() method in create-employee.component.ts file as shown below.   
  
saveEmployee(newEmployee: Employee): void {  
  console.log(newEmployee);  
}   
  
At this point, when we click the Save button, the employee object is logged to the console as expected.

**Lecture #18.** **Angular email validation example**

**In this video we will discuss** 

* Validating Email form field in Angular
* Using multiple validators on a single input field
* Angular safe navigation operator

**Email validation in Angular :**There are 2 different ways to validate email form field in Angular. We can either use **pattern validator**or **email validator**. Email validator is introduced in Angular 4. So if you are using Angular 4 or later version you may use email validator or pattern validator to validate email. If you are using Angular 2, then your only choice is to use Pattern validator.   
  
  
In this video we will discuss using the Angular built-in **Email validator**and in our next video we will discuss using the **Pattern validator**.    
  
**Consider the following HTML. Notice we are using Bootstrap classes for styling.**

<div class="form-group">

  <label for="email">Email</label>

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

          [(ngModel)]="employee.email">

</div>

The above HTML would produce the following Email input field   
   
  
We want to validate this email input field for 2 things 

* Email is required and
* Valid email must be provided

To make email, a required field modify the HTML as shown below

<div class="form-group" [class.has-error]="email.invalid && email.touched">

  <label for="email" class="control-label">Email</label>

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

          [(ngModel)]="employee.email" #email="ngModel">

  <span class="help-block" \*ngIf="email.invalid && email.touched">

    Email is required

  </span>

</div>

**Code Explanation :** 

* [class.has-error]="email.invalid && email.touched". This is class binding in angular. If the email field is touched and invalid, then the Bootstrap class has-error is added to the div element, else the class is removed.
* On the label that displays "Email" text, we applied control-label Bootstrap class. This class turns the label text to red if there is a validation error.
* \*ngIf="email.invalid && email.touched". Notice the \*ngIf structural directive on the span element. If the email field is touched and invalid the span element is added to the DOM, else it is removed. The Bootstrap help-block class on the span element is for styling.

At this point, if you touch the email field and leave it without typing in anything, you will see the validation error message "Email is required"

   
  
We also want to make sure the user enters a valid email. If someone types ABC, that is not a valid email. Angular 4 has built-in email validator, that we can use to validate if the user has entered a valid email. Here are the steps.  
  
**Step 1 :**On the email input field, place the **email** directive 

<input id="email" required email

**Step 2 :**Use the following HTML, to display the validation error message. If the email is invalid, angular attaches email key to the errors collection. On the other hand, if the email field is valid, the key email will not be in the errors collection. The question mark here is called the safe navigation operator. It protects against null and undefined values in property paths. It is generally used when we are not sure if a property exists or not. It safely handles null and undefined values, and very useful to prevent null-reference exceptions.

<span class="help-block" \*ngIf="email.errors?.email && email.touched">

  Email is Invalid

</span>

Here is the complete HTML that makes the email filed required and also checks if the email has a valid format

<div class="form-group" [class.has-error]="email.invalid && email.touched">

  <label for="email" class="control-label">Email</label>

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

          [(ngModel)]="employee.email" #email="ngModel">

  <span class="help-block" \*ngIf="email.errors?.required && email.touched">

    Email is required

  </span>

  <span class="help-block" \*ngIf="email.errors?.email && email.touched">

    Email is Invalid

  </span>

</div>

As of this recording, email validator provided by Angular does not allow null or empty values. When we leave the email field empty, the email validator is still fired. This is wrong. Checking NULL and empty values should be the job of the required validator. The following is the work around.  
  
Bind email directive to a boolean expression. The email validator is only added when the email field value is not an empty string. This ensures that, when we type something in the email field, the email validator is attached to the input field and it validates if the email format is valid or not.   
  
<input id="email" required [email]="employee.email!==''"  
  
**Please note :**Do not forget to initialise the email property in the employee object to an empty string.s

### #Lecture19. Angular regular expression validation

In this video we will discuss using **pattern validator in angular** to meet most of your application complex validation requirements.    
  
  
**With the pattern validator we use a regular expression**. Regular expressions are extremely useful when you want to validate if a given string conforms to a specified pattern.   
  
For example, you can use regular expressions to check if a given email conforms to a a valid email format. Similarly you can also check if provided postcode conforms to a specific country postcode format.   
  
Apart from checking conformity with a pattern, they can also be used to extract sub-strings from a given input string.  
  
  
**To validate** if the provided email has a valid email pattern we can use the pattern validator in angular. To use the pattern validator use the pattern attribute along with the regular expression on the input field you want to validate.

<input pattern="^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$" type="text"

       name="email" [(ngModel)]="employee.email" #email="ngModel">

It is easy to learn regular expressions. Initially they may appear complicated, but if you get the basics right it is very easy to understand them. However, you can also find the commonly used regular expressions on the internet. For example, if you want to find a regular expression to validate email address, simply search the internet with the following string  
Regular expression for email validation  
  
**Use the following HTML**, to display the validation error message. If the email is invalid, angular attaches pattern key to the errors collection. On the other hand, if the email field is valid, the key pattern will not be in the errors collection. The question mark here is called the **safe navigation operator**. We discussed this operator in detail in our previous video. If you are new to this operator, please check out our previous video.

<span class="help-block" \*ngIf="email.errors?.pattern && email.touched">

  Email is Invalid

</span>

The following example, shows both **required**and **pattern**validators on the Email input field.

<div class="form-group" [class.has-error]="email.invalid && email.touched">

  <label for="email" class="control-label">Email</label>

  <input required pattern="^[a-zA-Z0-9\_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+$"

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

         [(ngModel)]="employee.email" #email="ngModel">

  <span class="help-block" \*ngIf="email.errors?.required && email.touched">

    Email is required

  </span>

  <span class="help-block" \*ngIf="email.errors?.pattern && email.touched">

    Email is Invalid

  </span>

</div>

### Let's take this pattern validation to the next level. I want to validate emails against a specific domain. For example pragimtech.com is the only valid domain that I want to allow. Any other domain should be considered invalid. This can be very easily achieved with the following regular expression. ^[a-zA-Z0-9\_.+-]+@(?:(?:[a-zA-Z0-9-]+\.)?[a-zA-Z]+\.)?(pragimtech)\.com$