Angular 2 Notes

Contents

[Angular CLI 2](#_Toc512719943)

[Installing node\_module after git pull. 2](#_Toc512719944)

[Angular 2 Forms by Mark Zamoyta from Pluralsight 2](#_Toc512719945)

[Module 4 Data binding in Augular Forms 2](#_Toc512719946)

[This module will go over how to bind different form controls to the componet file. 2](#_Toc512719947)

[Data binding and ngModel 2](#_Toc512719948)

[Two way Data binding (bananna in a box) 3](#_Toc512719949)

[Examples used in Forms 3](#_Toc512719950)

[Add Model folder 4](#_Toc512719951)

[Add ngModel to a textbox 4](#_Toc512719952)

[Debug info for Model and Angular in the HTML. 4](#_Toc512719953)

[Adding ngModel to Checkbox 5](#_Toc512719954)

[Adding ngModel to radio button 6](#_Toc512719955)

[Module 5: Form validation 7](#_Toc512719956)

[HTML 5 Attributes for form Validation 7](#_Toc512719957)

[Styling a select tag for form validation. 8](#_Toc512719958)

[Form Level validation Intro. 9](#_Toc512719959)

[Angular and CSS Classes for validation 9](#_Toc512719960)

April 28, 2018

# Angular CLI

### Installing node\_module after git pull.

If you download an angular module from git there will not be the large and important node\_module folder that has all the angular dependencies. To install those

npm install

From the command line in vscode.

# Angular 2 Forms by Mark Zamoyta from Pluralsight

## Module 4 Data binding in Augular Forms

### This module will go over how to bind different form controls to the componet file.

### Data binding and ngModel

ngModel is a directive placed on input fields in a Html form.

In using ngModel you need to have a name specified on the input field.

<input name=”firstName” ngModel>

The use of ngModel above does not have any databinding but Angular is made aware of it.

<input name=”firstName” [ngModel]=”firstName”>

The use above will one way data bind from the compoent to the form. You need to have a property in the home.componet.ts named firstName for this to work.

<input name=”firstName” [ngModel] = “firstName”

(ngModelChange)=”firstName=$event”>

Above will two-way bind to the typescript componet. The square brackets [ngModel] will bind the changes to the property in the typscript file to the input field. The parentheses (ngModelChange) is event binding that will set firstName from the componet to the change of the input box.

<input name = “firstName” (ngModelChange) = “firstName = doStuff($event)”>

Above will call doStuff method in the compoent typescript when the input is changed and pass it the event handler.

<input #firstName = “ngModel” (ngModelChange) = “firstName = doStuff(firstName.value)”>

Above binds firstName with # so you can get properties directly out of the event rather then sending the whole $event which is better.

### Two way Data binding (bananna in a box)

<input name = “firstName” [(ngModel)] = “firstName”>

Above is simple two way binding for a to the componet class. Note you can’t call an event with the banana in a box syntax.

### Examples used in Forms

Add ngModel to the form to make Angular aware of the form and keep a model for it.

Below, you want to add #form=”ngForm” to the form tag

<form #form=”ngForm” novalidate>

Above, the novalidate property is set be we don’t want the browser to use its own validation, we want to set it ourselves.

Also #form=”ngForm” does both set a name of componet to form and adds a template reference varable called form to the angular model.

Below is another way of doing mostly the same thing.

<form name = “form” ngModel novalidate>

Above, the only problem with this way is you can’t interpolate info from the form directly with a template reference varable.

### Add Model folder

Add an employee.model.ts to hold the information about the form.

-add a folder to src file called models.

-add file called employee.models.ts

Add code

export class Employee {

constructor(

public firstName: string,

public lastName: string

){}

}

This will create a class with two properties (firstName and lastName). One benefit of typescript is you can create properties in the constructor like above.

Now just add employee to the componet.

Imports ~

@Componet( ~)

export class HomeComponet{

languages = ["English", "Spanish", "French", "Other"];

model = new Employee(‘John’, ‘Smith’);

}

This will allow you to bind to model property of the componet and bind it to the forms.

### Add ngModel to a textbox

<div class = ‘form-group>

<label> First Name </label>

<input #firstName = ‘ngModel’ type = ‘text’ class = ‘form-control’

[(ngModel)] = ‘firstName’>

</div>

### Debug info for Model and Angular in the HTML.

Use interpolation to see what the model is and what angular is thinking.

Model: {{ model | json }}

<br>

Angular: {{ form.value | json }}

Above uses the form template reference varable to get the angular tracked inputs in the form class. The pipe | is use to filter the output as a json format. Pipes are part of angular filters found in the Angular 2 Getting started video.

Add a event binding to the input box to change the input to uppercase

<div class = ‘form-group>

<label> First Name </label>

<input #firstName = ‘ngModel’ type = ‘text’ class = ‘form-control’

[ngModel] = ‘model.firstName’

(ngModelChange) = ‘textToUpperCase($event)>

</div>

In the componet class have…

export class HomeComponent {

languages = ["English", "Spanish", "Foo", "Other"];

model = new Employee('', 'Smith', true, "w2", "Spanish");

doStuff(value: string){

this.model.firstName = value.toUpperCase();

}

}

Above changes the model to uppercase and the from is bound to the model so then the form is update with the only uppercase.

Note: I tried to use the template varable, $fileName.value rather then just passint the event with $event to the function but it updated the input box only after the second letter was entered. A one letter delay. Sometimes passing the raw event is better it looks like.

Note: you can’t use bananna in a box syntax to call the function.

### Adding ngModel to Checkbox

First add a property to the model to handel the checkbox

export class Employee {

constructor(

public firstName: string,

public lastName: string,

public isFullTime: boolean

){}

}

Then add it to the model in the componet class.

model = new Employee(‘John’, 'Smith', true);

Then add the checkbox to the html

<div class="checkbox">

<label>

<input type="checkbox" name="is-full-time"

[(ngModel)] = "model.isFullTime">

Full Time Employee

</label>

</div>

Above will add a checkbox to the form with a name is-full-time and two way bind it to the model. The initail value in the for is whatever is passed in with model.

### Adding ngModel to radio button

Add radio button to the model class

export class Employee {

constructor(

public firstName: string, //syntax used to create a local property of this type

public lastName: string,

public isFullTime: boolean,

public paymentType: string,

public primaryLanguage: string

){}

}

Note: I just added the rest of the properties used in this module. The paymentType is added for this section.

Add the payment type to the model in the componet. Note I just added primary language too for the next secion.

model = new Employee('John', 'Smith', true, "w2", "Spanish");

Above sets the default paymentType to w2.

Then add to radio button to the form.

<div class='form-group'>

<label>Payment Type</label>

<div class="radio">

<label>

<input type="radio" name="paymentType" value="w2" checked

[(ngModel)] = "model.paymentType">

W2

</label>

</div>

<div class="radio">

<label>

<input type="radio" name="paymentType" value="1099"

[(ngModel)] = "model.paymentType">

1099

</label>

</div>

</div>

Above will create a check box with name of paymentType and values w2 and 1099

## Module 5: Form validation

### HTML 5 Attributes for form Validation

Required = if put onto an input tag it will require the there to be input for the form to be valid.

Look at caniuse.com to check browser compatablity.

Maxlength = put onto an input which will make the max number of characters allowed for that form. (highly supported)

Minlength = put on t an input tag to specify the min length. (not well supported)

Pattern = ‘^Q.\*’ will match with any word that begins with a Q

Pattern = ‘…+’ will match any word three letters or greater.

Search for patterns for email address, area codes, phone numbers, etc.

### Styling a select tag for form validation.

Adding an additional option for “Select a language…”

Lots of stuff. You need to add a varable to the component to flag the hasPrimaryLanguageError. Angular does not have a good way of assessing the fields of a select so you have to create a custom function.

Also add a class control-label so bootstrap will turn the lable read.

Add a template reference varable to the select because you will need to use it to get the current value of the select. Just sending $event to the function will not work easy because you will have to query the select for what the current value is. It is just easier to use template reference varable.

Add two way binding with banana in a box for the model.primaryLanguage.

Add blur event binding to the validatePrimaryLanguage function and pass it the value of the temple reference varable primaryLanguage.

Add the function to the component validatePrimaryLanguage that simply sets the hasPrimaryLanguageError flag in the componet to value = = = ‘default’ (identically equals to with three equals signs)

Add a change event binding too with the same settings as the blur.

Add the default option that will be the label for “select a language”.

<div class="form-group" [class.has-error] = 'hasPrimaryLanguageError'>

<label class='control-label'>Primary Language</label>

<select #primaryLanguage class="form-control" name="primaryLanguage"

(blur) = 'validPrimaryLanguage(primaryLanguage.value)'

(change) = 'validPrimaryLanguage(primaryLanguage.value)'

[(ngModel)]='model.primaryLanguage'>

<option value='default'>Select a language...</option>

<option \*ngFor="let lang of languages">

{{lang}}

</option>

</select>

<div \*ngIf='hasPrimaryLanguageError'class="alert alert-danger" role="alert">

Please select a Language.

</div>

</div>

Note: I also added an \*ngIf div to the bottom of the form group to give the user a message.

If the user click on the select without seting a value an error will occur. If the user click on the select a lauguage option an error will occur. If a valid selection is made then the error is taken away.

This is a good example of using multiple event bindings on one field.

### Form Level validation Intro.

If you want to have a text field verified that it has input then required on the input tag. Then when you put the template reference varable for the form to ngForm with…

#form=’ngForm’

You can simply disable the submit button with a property binding…

<button type=’submit’ class = ‘btn btn-primary’ [disabled] = ‘form.invalid’>OK</button>

This will turn off the button if the form is invalid.

You can also put in at the bottom

{{ form.invalid | json }}

If you want to see the current status of the form validation for debuging.

### Angular and CSS Classes for validation

CSS classes (note you will have to manage adding and subtracting these classes)

ng-touched // if a field has been touched.

ng-untouched

ng-dirty // if the user has changed anything with the field

ng-pristine

ng-invalid // if the form or control is invalid.

ng-valid

Angular model property have the same set of classes and purposes but just remove the ng- like valid instead of ng-valid.

To use the angular directives thou you will need to set the control or form with a template refence varable to ngForm for the form and ngModel for controls.

## Module 6: Posting data to the server

### Creating a Node test server

Create a new folder for the server > mkdir node\_server

(I created this in the Sandbox/Node/node\_server)

Then run > npm init

This will run you through a setup of a node server.

Use the default for everything for the test server. Keep hitting enter until you are back at the command prompt.

Then create a server.js file in the project and copy the server info into it. I don’t want to go over all the server stuff here. I will watch another pularsight video for that.

I also copied the Angular 2 forms at the

# Forms: wrap the form output into a object

Wrap the inputs for the inputs to be wrapped in an object

<div ngModelGroup=”location”>

<input (ngModel)=’address’ ~

<input (ngModel)=’city’ ~

<input (ngModel)=’country’ ~

</div>

This will wrap the values in a location array

location [adress: ‘foo’, city: ‘bar’, country: ‘usa’]