**Forms**

**Large category of application are form sensitive and many of the application are basically huge forms and with validation business logic.**

**Every form intensive application has to provide a solution for**

1. **How to keep a track of the global form state**
2. **Know which part is valid and invalid**
3. **Displaying proper error messages**

**Angular 2 works with the Javascript representation of a form.**

**{controls:controlname,value:{controlName}}**

* **Angular Create a FormGroup Represent the form**
* **FormGroup is in sync with HTML form**
* **It handles validation and submission etc**

**Angular 2 provides 2 ways for Form handling**

1. **Data Driven approach**
2. **Template Driven Forms**

**Template Driven**

* **It tackles forms using ng-Model directive**
* **Forms built with this can only be tested in an end to end test because it requires presence of DOM.**
* **This mechanism is very useful and simple to understand.**

**Model driven**

* **Its somewhat same as template driven form**
* **We need to add FormGroup , FormControl**

1. **FormGroup**

**- FormGroup encapsulates the state of all of its inner controls**

**So Formgroup is valid only if all of tis inner controls are valid.**

1. **FormControl**

* **FormControl is created for each template form control**
* **Each form control in the template is represented by an instance of FormControl**
* **This encapsulates the state of the control whether its valid or invalid and event its current value**

**Advantage of Model Driven forms**

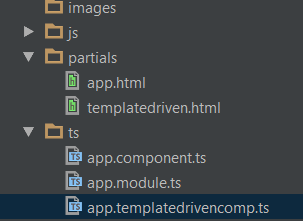
1. **Unit Testable**
2. **Listen To Form and Control Changes**

**Different between Template Driven and Model Driven approach**

1. **Form is set up and configured in HTML code while in Data driven its setup and configured programmatically in class body.**
2. **Angular 2 infers Formgroup from HTML code while in model driven Formgroup is created .**
3. **Form data is pass through ngSubmit while . Form(data) can be used throughout class body without passing it via ngsubmit.**

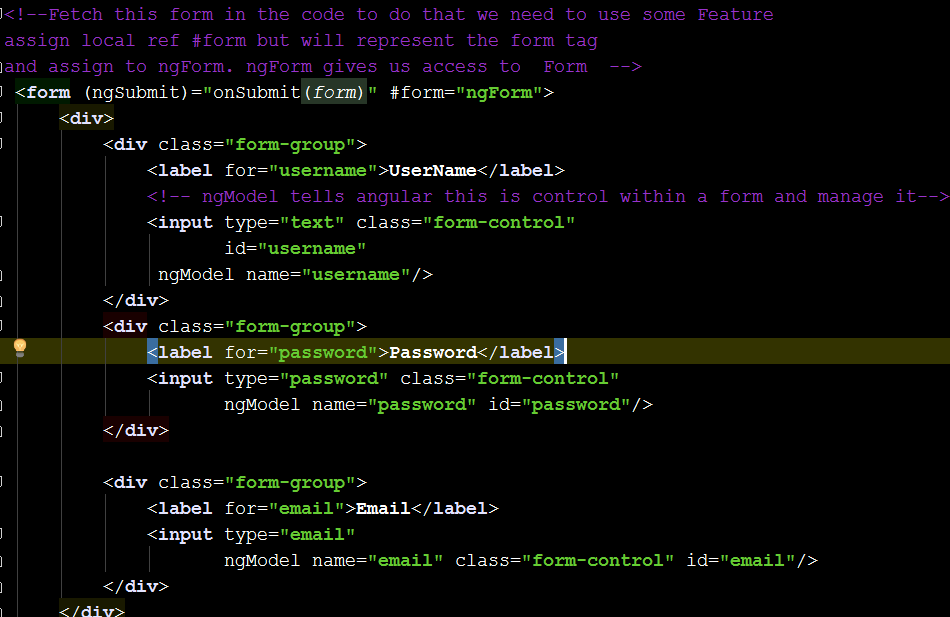
**Demo 1: Template Driven**

1. **Add new TemplateDriven Component and its HTML file in the partial folder**

****

1. **Add Templatedriven.html along with with ngModel ngSubmit and onsubmit.**

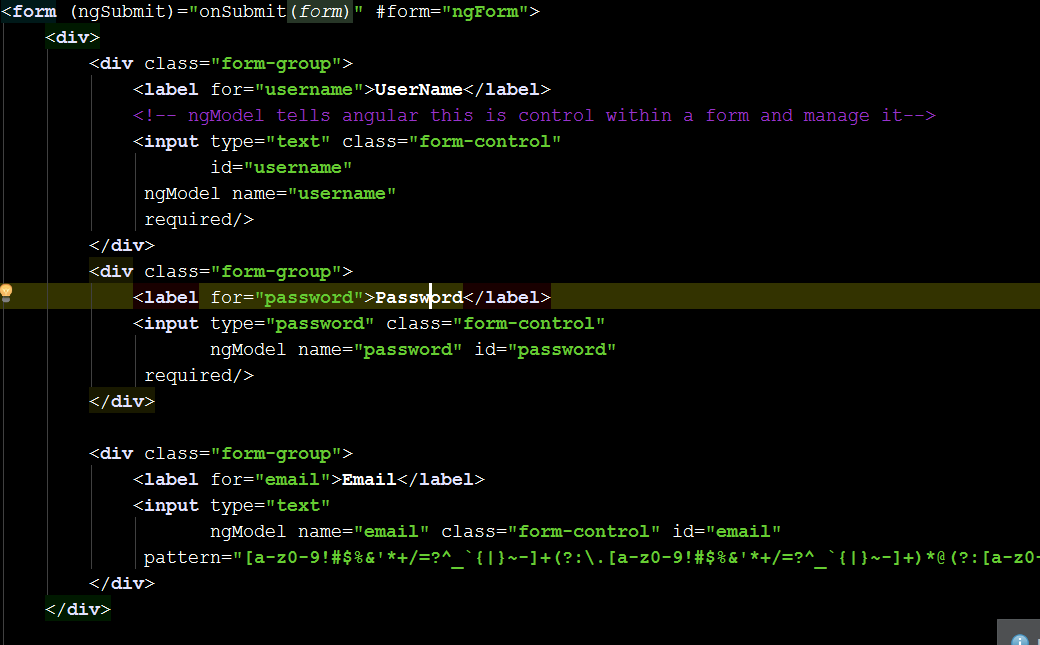
**NOTE : DON’T use input type email .. rather use text**

****

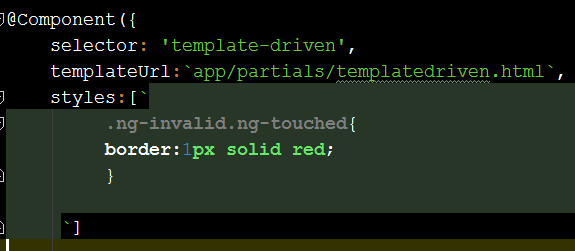
1. **Add a method in Template Component to Submit the data**

****

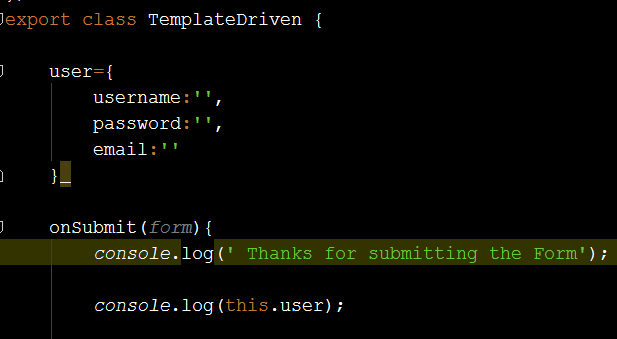
1. **Add Required attribute and Pattern in Email**

****

1. **Add Form State for ng-invalid . it will show red border if input type is in invalid state.**

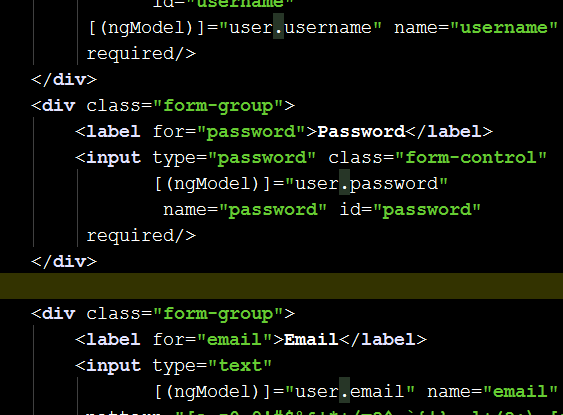
****

1. **How to add Two Way Data Binding**
2. **Add a user object that holds username,password and email.**

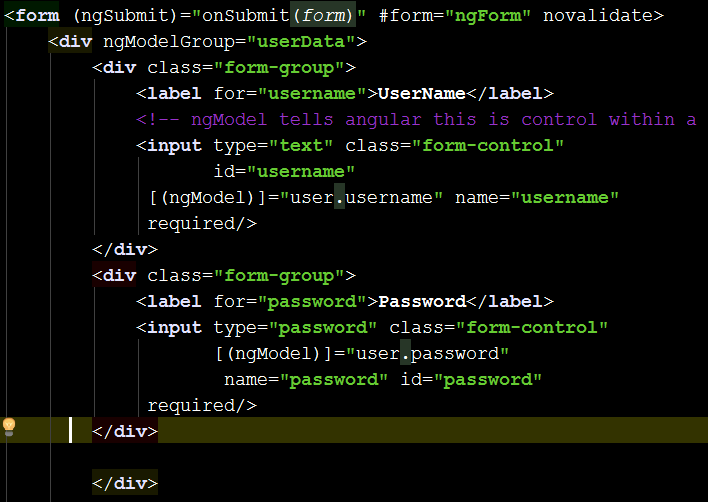
****

1. **Change the username,password and email binding (ngModel) in the template code.**

**Notice ngModel**

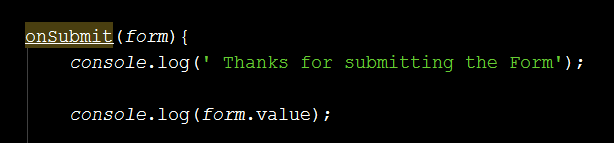
****

1. **Suppose we want to group the form control. We need to use ngModelGroup**

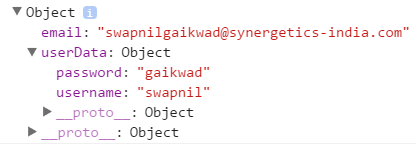
****

**In above example, I have grouped bothe username and password so it will appear as a single object.**

1. **Print the result in console.**

****

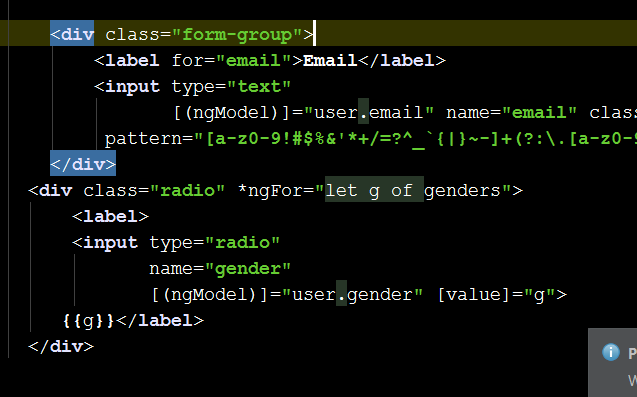
1. **Check the output in console both the username and password is wrapped as a group.**

****

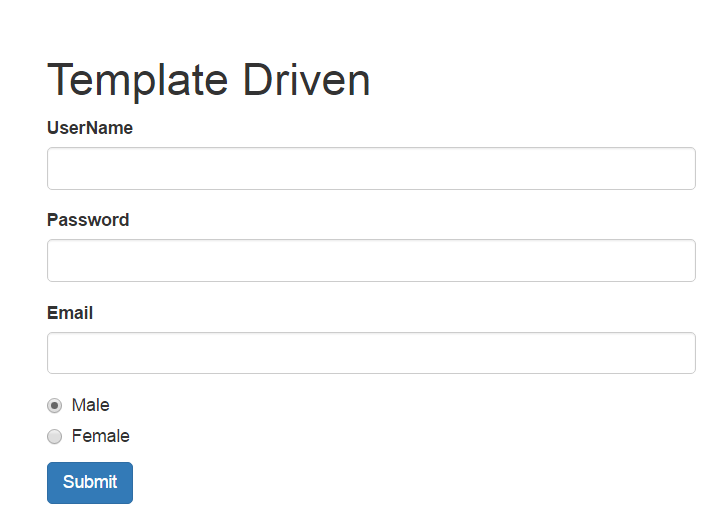
1. **Working with Radio button**
2. **Add Genders array and keep some values there also add entry of it in user object.**

****

1. **Change the HTML add new radio buttons using ngFor and add a value property binding to it. Add radio button code from below.**

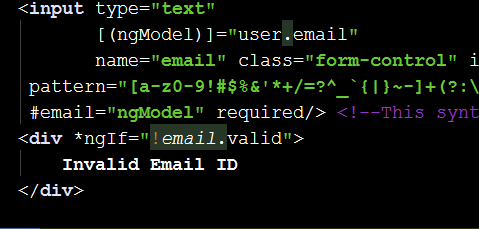
****

**Output**

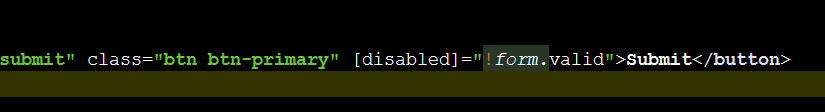
****

1. **Add some validation messages to this form .**

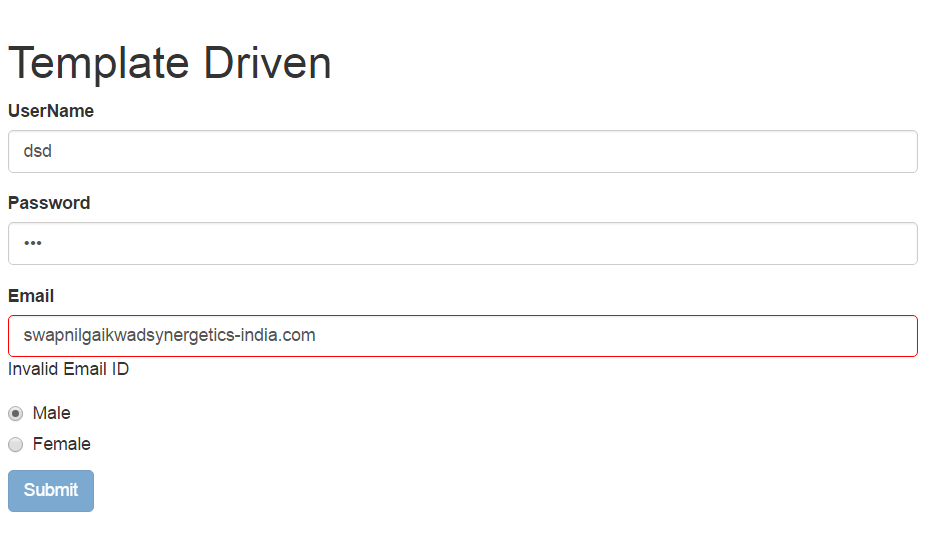
**Go to form and add #email=ngModel . it gives access the email control.**

****

1. **Form should be submitted only if its valid . so use the form state management syntax to enable the button only if the form is valid.**

****

**Output**

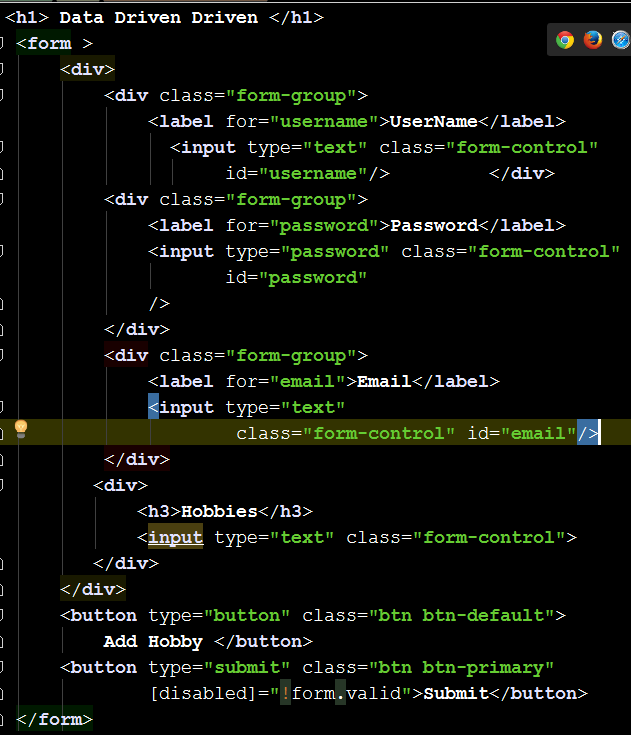
****

**Model Driven /Data Driven Form Validation**

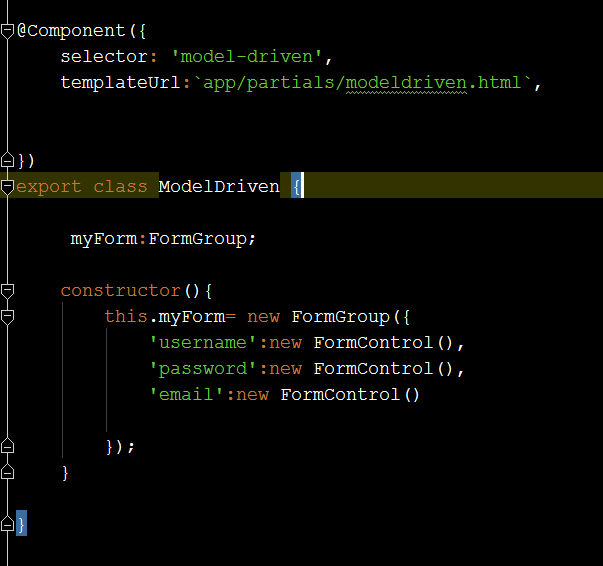
* **Validators are rules which an input control has to follow. If the input doesn’t match the rule then control is said to be invalid.**
* **We can apply validator either by adding attributes to the template or by defining it to the form model**
* **Angular comes with prebuilt validators which we can access via Validators module**
* **Required**
* **Minlength**
* **Maxlength**
* **Pattern**
* **FormControl constructor takes the initial value of the control . The second parameter is single or list of validator.**
* **You need to add Validators from angular/forms**

**Demo 1:**

1. **Add a HTML Code**

****

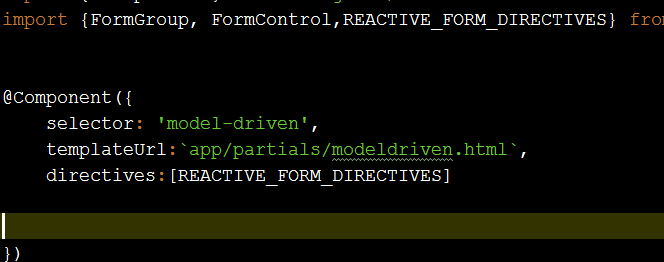
1. **Add a modeldriven component . What we are trying to do here is , rather than creating formgroup and ngForm automatically . we are taking control over it. We are creating our own form.**

****

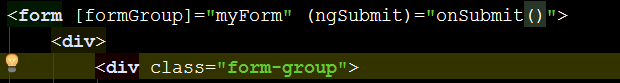
1. **Now associate this model to the HTML. To do this use formGroup and formControl.**

****

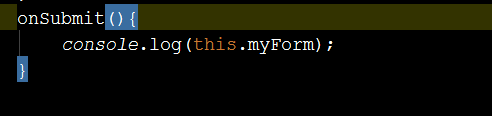
1. **This wont work unless you import REACTIVE\_FORMS\_DIRECTIVES which allows us to access this formGroup and formControlName**

****

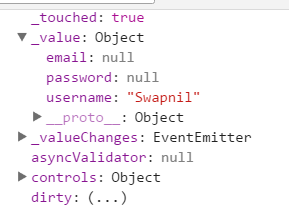
1. **Add ngSubmit in form**

****

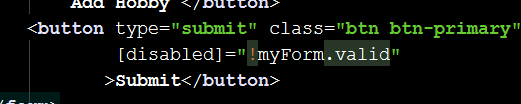
1. Add Onsubmit Code



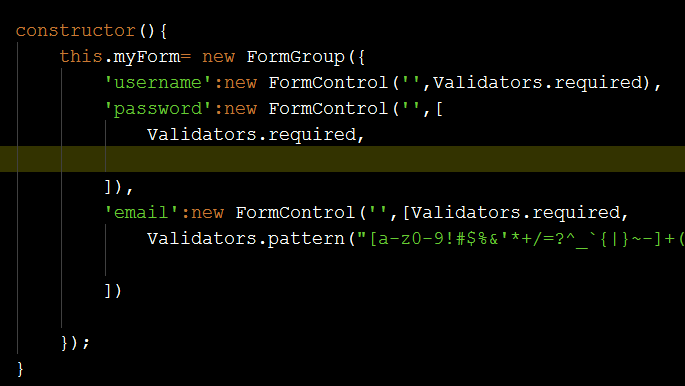
1. **Check the output in console.**

****

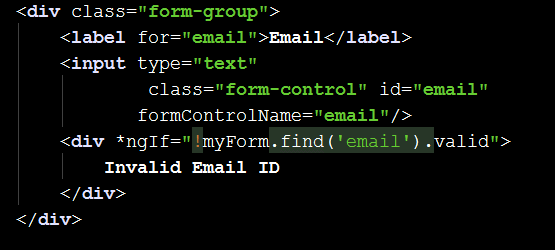
1. **Disable the submit button until entire form is valid.**

****

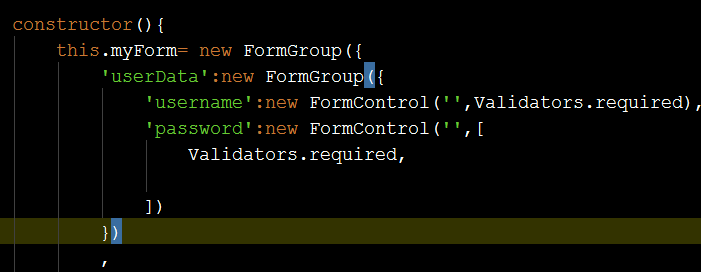
1. **Add some Validation in the model driven**

****

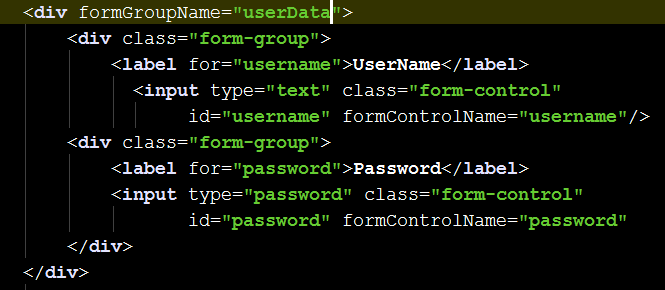
1. **Suppose you want to add conditional message to show the error.**

****

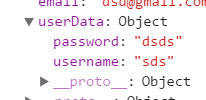
1. **How to add model Group the way we added in Template driven**
2. **Add new formGroup in model driven**

****

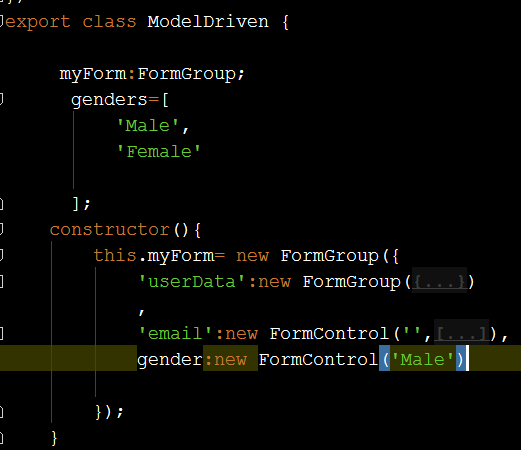
1. **We need to arrange HTML to tell myForm where in HTML you will get this controls.**

****

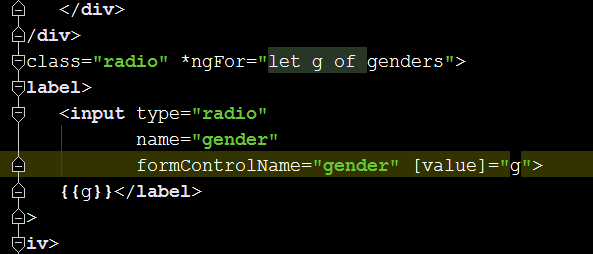
1. **See the output the username and password is grouped inside modelgroup**

****

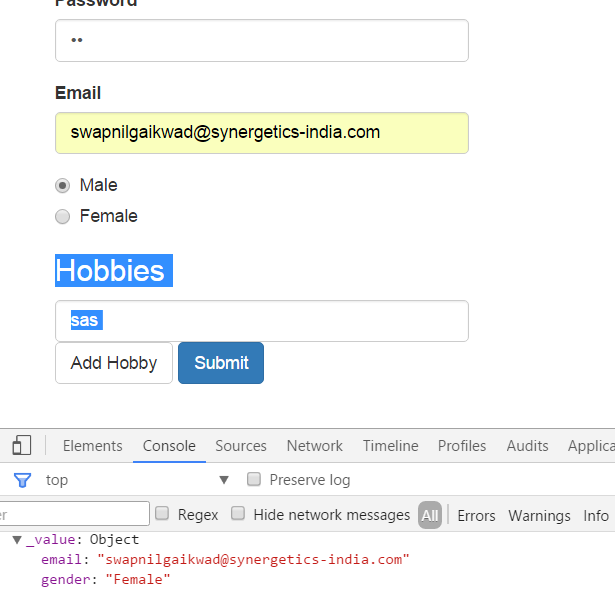
1. **Now add Radio Button**
2. **Create genders array and register as a controlname in the constructor**

****

1. **Iterate over it in the HTML code and display it.**

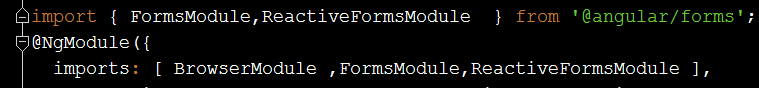
****

1. **See the output**

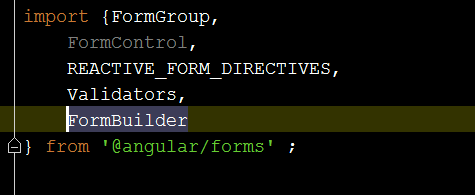
****

**Using FormBuilder**

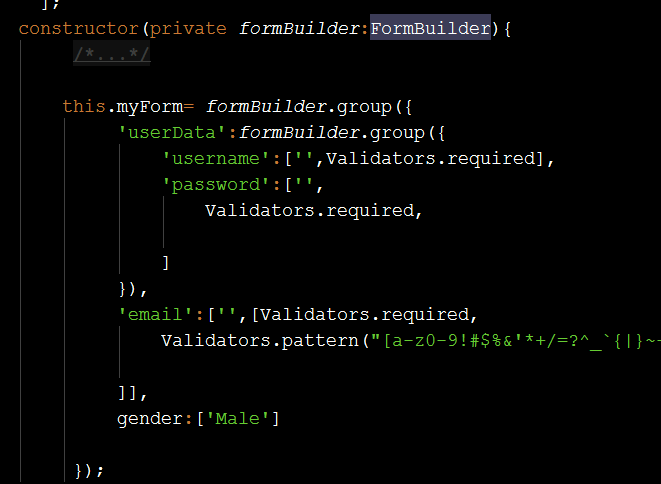
1. **Change the FormGroup to use formbuilder , it’s a different way of achieving same result**
2. **Add ReactiveModule in app.module and register with import**

****

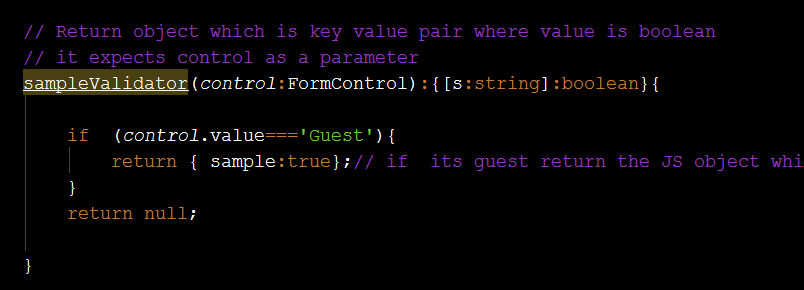
1. **Import formBuilder in import**

****

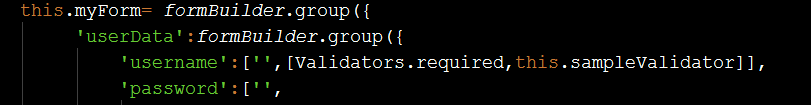
1. **.Instantiate formbuilder in constructor. And use array syntax to create formgroup and formcontrol**

****

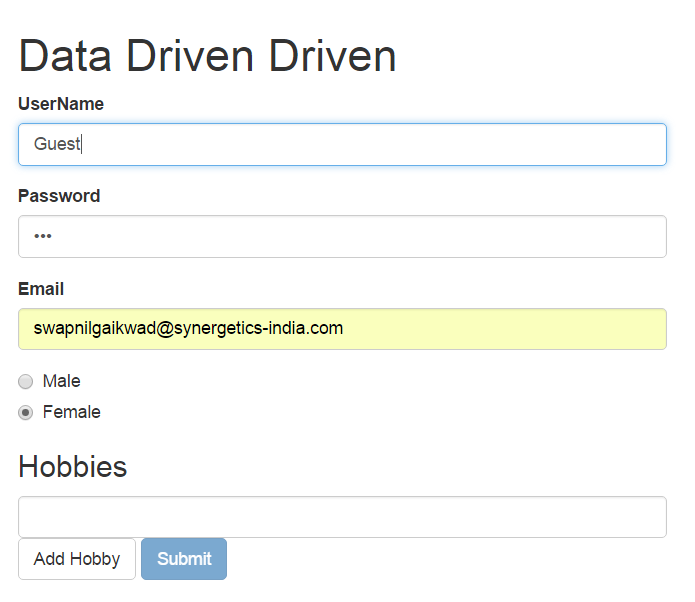
1. **How to add custom validator**
2. **Add a function which will validate the formcontrol.**

****

1. **Add this validator in usermail validator collection.**

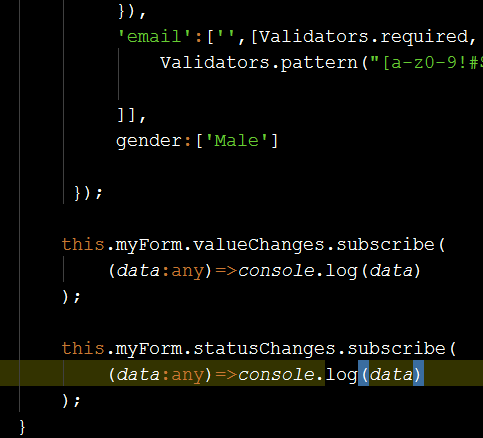
****

1. **See the output : it will not allows Guest as a username**

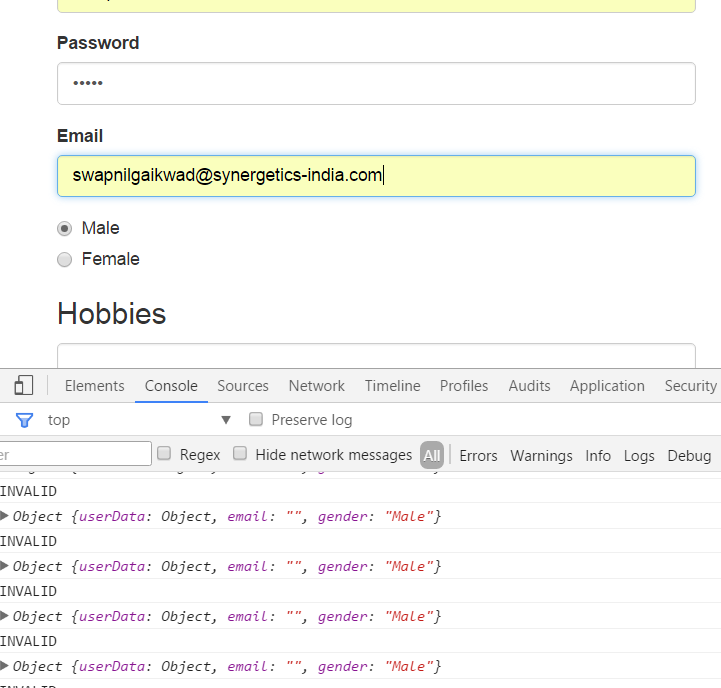
****

**Value Changes and Status Changes**

1. **You can write some responsive code for your application as soon as the value changes or status of the form changes . it returns observable hence we can use subscribe to find the changes**

****

1. **See the output.**

****

**Form Control State**

1. **Dirty and Pristine**

* **If the dirty is true if the user has changed the value of control.**
* **Pristine is opposite of dirty that is , its true of the user hasnot changed the value and false if the user has.**

1. **Touched and UnTouched**

* **A control is touched if the user has a focus on the control and then focused on something else**
* **Difference between touched and dirty is that touched the user doesn’t need to change the value of the input control.**
* **If the touched is true then the field is touched by user else false.**
* **Untouched is opposite of that**

1. **Valid and Invalid**

* **Valid is true if all the validators are passing**
* **Otherwise its invalid**

**Validation Styling**

* **Bootstrap has classes for showing visual feedback for controls when they are invalid.**

1. **Has-danger class : it adds red border**
2. **Has-success class adds green border**