**Angular Forms**

**Why we need Forms?**

Forms are the main building blocks of any type of application. When we use forms for login, registration, submission. Help request, etc., it is necessary that whatever forms we are developing, they should be user friendly. And it should have the indication of what went wrong by display user friendly message, etc.

Forms are really very very important to collect the data from the users. Often, each website contains forms to collect the user data.

You can use forms to login, submit a help request, place an order, book a flight, schedule a meeting, and perform other countless data entry tasks.

**What are Angular Forms?**

Developing forms requires design skill as well as framework support for two-way data binding, change tracking, validation, error handling, etc. The Angular Framework, provides two different ways to collect and validate the data from a user. They are as follows:

1. Template-Driven Forms
2. Model-Driven Forms (Reactive Forms)

**Template Driven Forms in Angular:**

Template Driven Forms are simple forms which can be used to develop forms. These are called Template Driven as everything that we are going to use in an application is defined into the template that we are defining along with the component.

In order to use Template Driven Forms, we need to import **FormsModule**into the application root module i.e. **app.module.ts** file.

**Features of Template Driven Forms:**

1. Easy to use.
2. Suitable for simple scenarios and fail for complex scenarios.
3. Similar to Angular 1.0 (Angular JS)
4. Two way data binding using NgModule syntax.
5. Minimal Component code
6. Automatic track of the form and its data.
7. Unit testing is another challenge

**Model-Driven Forms (Reactive Forms) in Angular:**

In a model driven approach, the model which is created in the .ts file is responsible for handling all the user interactions and validations. For this, first, we need to create the model using Angular’s inbuilt classes like formGroup and formControl and then we need to bind that model to the HTML form.

This approach uses the Reactive forms for developing the forms which favor the explicit management of data between the UI (User Interface) and the Model. With this approach, we create the tree of Angular Form Controls and bind them in the Native Form Controls. As we create the form controls directly in the component, it makes it a bit easier to push the data between the data models and the UI elements.

In order to use Reactive Forms, you need to import ReactiveFormsModule into the applications root module i.e. app.module.ts file.

**Features of Reactive Forms:**

1. More flexible, but need a lot of practice
2. Handles any complex scenarios.
3. No data binding is done (Immutable data model preferred by most developers).
4. More component code and less HTML Markup.
5. Easier unit testing.
6. Reactive transformations can be made possible such as
7. Handling a event based on a denounce time.
8. Handling events when the components are distinct until changed.
9. Adding elements dynamically.

**Which one is better – Template Driven or Reactive Forms?**

Neither reactive nor template driven are better over each other. 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, then you need to use the Reactive forms approach. There are several other differences, between Template driven and Reactive forms that we will discuss in our upcoming articles.

**Note:** They both are two different approaches, so you can use whichever suits your needs the most. You can use both in the same application.

## ****Angular Template Driven Forms****

**What are Angular Template Driven Forms?**

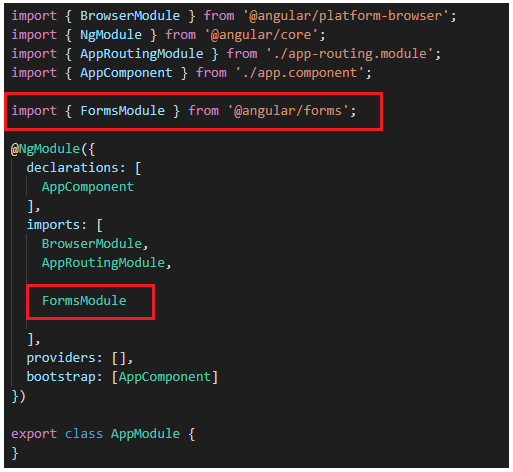
The Angular Template Driven Forms are simple forms which can be used to develop forms. These are called template driven as everything we are going to use in a application is defined into the template that we are defining along with the component.

**How we can develop and use these forms in Angular Application?**

Let see the step by step procedure to develop and use these forms in angular application.

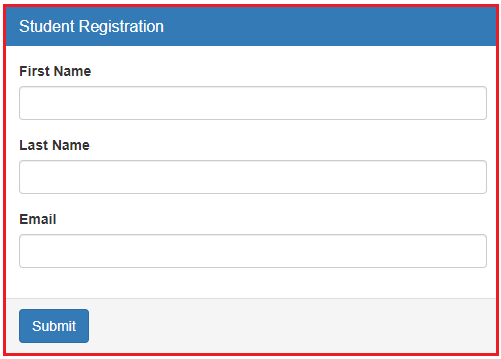
**Step1: Importing Forms module**

The **ngForm**directive is provided by Angular **FormsModule**. So, first we need ti import the Forms module in the applications root module i.e. in the **app.module.ts** file as shown in the below image. As you can see in the below image, we are doing two things. First the import the **FormsModule**from the angular library and then declare the **FormsModule** in the **import**array of the **@NgModule** decorator.



##### **Step2: Create a Registration Form**

We want to create a registration form as shown in the below image to register a student. To keep this example simple, at the moment we have only 3 fields (First Name, Last Name and Email). As we progress in this course, we will add the other fields like Gender, Branch, Phone Number etc. Also, at the moment, our form contains only textboxes. In our upcoming articles, we are going to discuss radio buttons, dropdownlist, checkbox, etc.



**Modifying app.component.html file:**

In order to create the above registration form, please modify the **app.component.html** file as shown below and then we will discuss the code in detail. Here, we are using Bootstrap CSS classes like panel, panel-primary, panel-heading, panel-title etc to style the form. If you are new how to use bootstrap in angular, then please read our **how to use bootstrap in angula**r article article.

<br/>

<div class="container">

  <div class="row">

      <div class="form-bg">

          <form #studentForm="ngForm" (ngSubmit)="RegisterStudent(studentForm)">

            <div class="panel panel-primary">

              <div class="panel-heading">

                <h3 class="panel-title">Student Registration</h3>

              </div>

              <div class="panel-body">

                <div class="form-group">

                  <label for="firstName">First Name</label>

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

                        name="firstName" ngModel>

                </div>

                <div class="form-group">

                  <label for="lastName">Last Name</label>

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

                        name="lastName" ngModel>

                </div>

                <div class="form-group">

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

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

                        name="email" ngModel>

                </div>

              </div>

              <div class="panel-footer">

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

              </div>

            </div>

          </form>

      </div>

  </div>

</div>

**Code Explanation:**

As you can see in the above code, we have created three text boxes and one button. But along with we are using something called NgForm, NgMoel, and many more things. Let us discuss these things in detail.

**NgForm:**

It is the directive which helps to create the control groups inside form directive. It is attached to the <form> element in HTML and supplements from tag with some additional features.

**NgModel:**

When we add ngModel directive to the control, all the input elements are registered with the NgForm. It created the instance of the FormControl class from Domain model and assign it to the form control elements. The control keeps track of the user information and the state and the validation status of the form control.

Next important thing is to consider is that when we use ngModel with form tag, then we should have to use the name property of the HTML control.

Two main functionalities are provided by NgForm and NgModel are the permission to retrieving the values of the control associated with the form and then retrieving the overall state of the controls in the form.

**Consider the following line of code:**

Understanding ngForm Directive in Angular

#studentForm is called the template reference variable and if you notice we have assigned “ngForm” as the value for the template reference variable studentForm. So the studentForm reference variable holds a reference to the form.

Now the questions arises, whether or not we need to use this local variable. Well the answer is no. We are exporting ngForm in the local variable just to use some of the properties of the form and these properties are as follows:

1. **studentForm.value** : It gives the object containing all the values of the field contain in the form.
2. **studentForm.valid**: This gives the value indicating if the form is valid or not. If it is valid then the value is true else the value is false.
3. **studentForm.touched** : It returns true or false when one of the field in the form is touched or entered.

As you can see, The form tag is not associated with any action method, then the question is how we post the form data to the component. The answer is using ngSubmit directive.

**Understanding ngSubmit directive:**

Please have a look at the following ngSubmit directive. Here, we are using the Event Binding concept and we binding to the RegisterStudent method of the component. Instead of the submit event of the form, we are using ngSubmit which will send the actual HTTP request instead of just submitting the form.

Understanding Angular ngSubmit directive

The ngSubmit directive will submits the form when we either hit the enter key or when we click the Submit button. When the form is submitted, RegisterStudent() method of the AppComponent class is called and we are passing it the studentForm. We do not have this method at the moment in the AppComponent class. We will create this method in just a bit.

**Step3: Creating RegisterStudent() method**

Now, open app.component.ts file and then copy and paste the below code in it. As you can see here, we created the RegisterStudent method. At the moment, we are simply logging the value of the Angular generated Form model to the console. In order to use the NgForm in our component class, first we need to import **NgForm**type from ‘**@angular/forms**‘.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

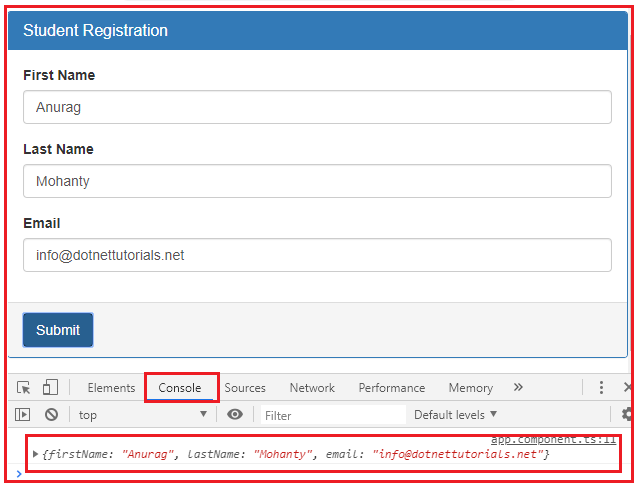
console.log**(**studentForm.value**)**;

**}**

**}**

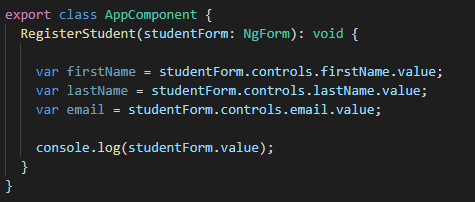
As we already discussed, The ngForm directive supplements the form element with additional features and properties like value, dirty, touched, valid etc. In the above example, we use the value property, similarly you can also use the other properties. These properties are very useful for form validations and we will discuss this in our upcoming articles.

At the moment save all changes and browse the application and you should the output as expected. Now, launch browser developer tools by pressing F12 key and then click on the console tab. Fill the form and click on the submit button, it should log details in the console window as shown in the below image.



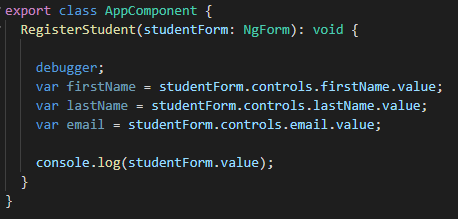
**How to fetch the form values and store in local variable:**

If you want to fetch the form control values and store in local variables then you need to modify the RegisterStudent method as shown below.



**How to debug the code in Angular?**

If you want to debug to make sure the values getting properly you can also add debugger as shown in the below image.



## ****Radio Buttons in Angular Template Driven Forms****

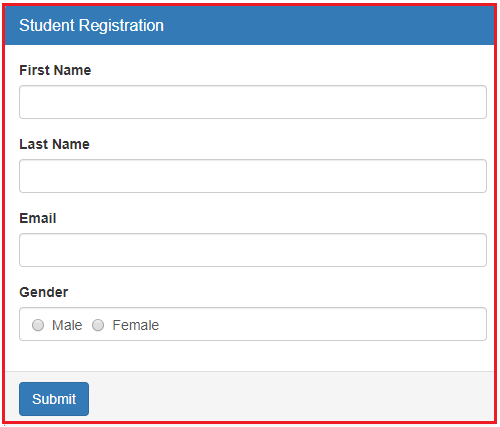
**What is a Radio Button?**

A Radio Button is an HTML element which is basically allows the user to select a single option from a predefined list of options. For example, you can create radio buttons for gender (Male and Female) and the user can only select either male radio button option or female radio button option but not the both.

**Example to understand Radio Buttons in Angular Template Driven Forms:**

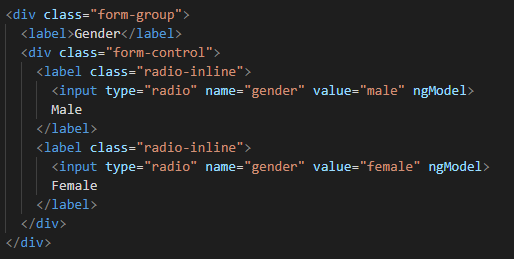
Let us understand how to create and use Radio Buttons in Angular Template Driven Forms. We are going to work with the same example that we started in our previous article.

Now, we want to include the “Gender” radio buttons in the student registration form as shown in the below image. When we select student “Gender” using the radio buttons and when we click the “Submit” button, we want the selected gender value to be logged to the console.



**How to create radio button in angular template driven forms?**

Please have a look at the below code which will create gender radio buttons with male and female options.



**Code Explanation**

In the above code, the name attribute of the input element radio is used to group the radio buttons as one unit which makes the selection mutually exclusive. The most important point that you need to keep in mind is that both the radio buttons should have the same value for the “**name**” attribute. Otherwise the radio button selection won’t be mutually exclusive.

Again if you notice we have set the value attribute of each radio button to make and female and this is the value which is going to be posted to the server when the form is submitted.

**The complete code of app.component.html:**

Following is the complete code of app.component.html file.

**<br/>**

**<div** class="container"**>**

**<div** class="row"**>**

**<div** class="form-bg"**>**

**<form** #studentForm="ngForm" (ngSubmit)="RegisterStudent(studentForm)" **>**

**<div** class="panel panel-primary"**>**

**<div** class="panel-heading"**>**

**<h3** class="panel-title"**>**Student Registration**</h3>**

**</div>**

**<div** class="panel-body"**>**

**<div** class="form-group"**>**

**<label** for="firstName"**>**First Name**</label>**

**<input** id="firstName" type="text" class="form-control"

name="firstName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label** for="lastName"**>**Last Name**</label>**

**<input** id="lastName" type="text" class="form-control"

name="lastName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

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

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

name="email" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label>**Gender**</label>**

**<div** class="form-control"**>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="male" ngModel**>**

Male

**</label>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="female" ngModel**>**

Female

**</label>**

**</div>**

**</div>**

**</div>**

**<div** class="panel-footer"**>**

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

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**

**</div>**

**Modifying the app.component.ts file:**

We want to log the posted form values into the console. So, modify the **app.component.ts** file as shown below.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

// debugger;

// var firstName = studentForm.controls.firstName.value;

// var lastName = studentForm.controls.lastName.value;

// var email = studentForm.controls.email.value;

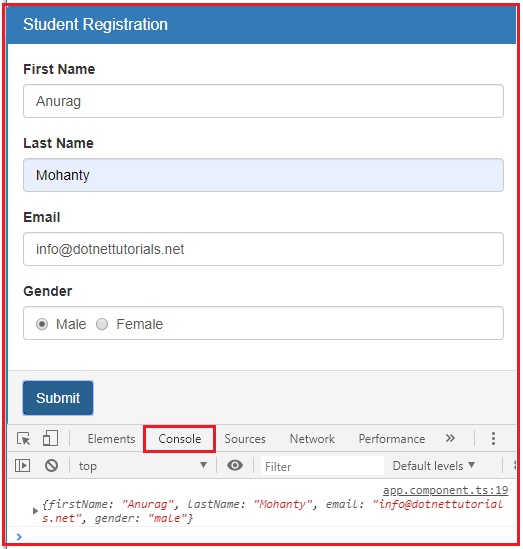
// var gender = studentForm.controls.gender.value;

console.log**(**studentForm.value**)**;

**}**

**}**

With the above changes in place, now browse the application,open browser developers tool by pressing F12 key and open console tab. Then fill the form and click on the submit button and you should see the posted form values in the console tab as shown in the below image.



**How to select a radio button checked by default in Angular?**

As we know when working with real-time applications, sometimes we need to provide the one radio button to be checked by default when the form load initially and normally we can do this by adding the checked attribute of the radio button.

If you include the checked attribute to one of the radio buttons, then you may expect that the radio button to be checked by default. But in our example, you will not get that default checked. In our example, lets include the “checked” attribute on the “Male” radio button. So. Modify the gender HTML code as shown below.

**<input type=”radio” name=”gender” value=”male” checked ngModel>**

With the above changes now browse the application and you will see the Male radio button is not checked.

However, if you remove the “ngModel” directive from the radio button as shown below, then you will see that the Male radio button is checked when the form is load.

**<input type=”radio” name=”gender” value=”male” checked>**

In Angular Template Driven forms, we generally use the “ngModel” directive for two-way data binding. So when we put the ngModel directive back into the control then the “checked” attribute will not work as expected.

**How to make it works?**

In order to make it work include the “gender” property in the component class and initialize its value to the value of the radio button that you want to have checked by default. In our case, we want the “Male” radio button to be checked by default. So, we need to add “gender” property initialized to value of “male” in the component class as shown below.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

gender = 'male';

RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

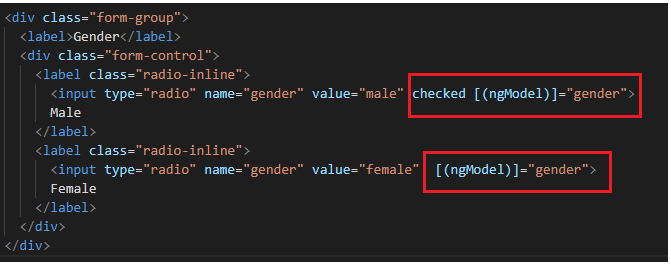
console.log**(**studentForm.value**)**;

**}**

**}**

**Modifying the app.component.html**

Now modify the **app.component.html** file as shown below where we included the **ngModel**with **gender**property of the component class.



At this point when you browse the application, you will see that the “Male” radio button is checked by default.

Now, if you remove the “checked” attribute from the “Male” radio button, then it is still checked by default when the form loads. This is possible because of the two-way data binding in angular. In our example, we do not want any radio button to be checked by default, so we remove the “checked” attribute and the “gender” property from the component class and ngModel directive.

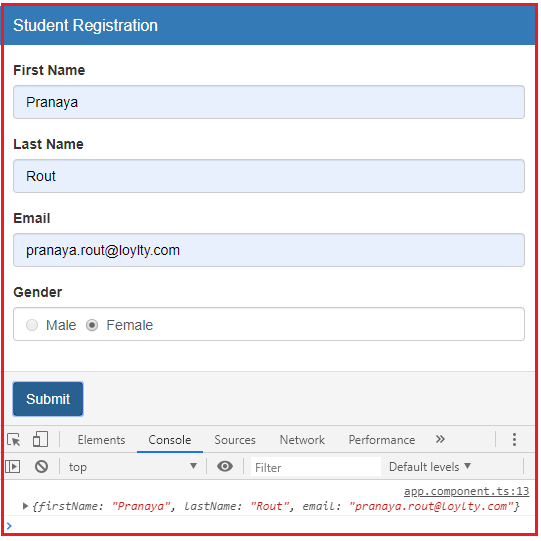
**How to disable a radio button in Angular Template Driven Forms?**

In order to disable a radio button in Angular Template Driven Form, we need use the disabled attribute on that radio button. For example, if you want to make the “Male” radio button disabled when the form initially loads, then you need to modify the Male radio button as shown below.

**<input type=”radio” name=”gender” value=”male” ngModel disabled>**

Note: The most important point that you need to remember is, by default, the disabled form controls are not included in the Angular auto generated form model.

In our example, the “Male” radio button is disabled, so, the gender property will not be included in the Angular generated form model. At this point, even if you select the Female radio button and submit the form, then also you will not see the gender property as shown in the below image.



In our example, we do not want any radio button to be disabled, so please remove the disabled attribute from the radio button.

## ****Checkbox in Angular Template Driven Forms****

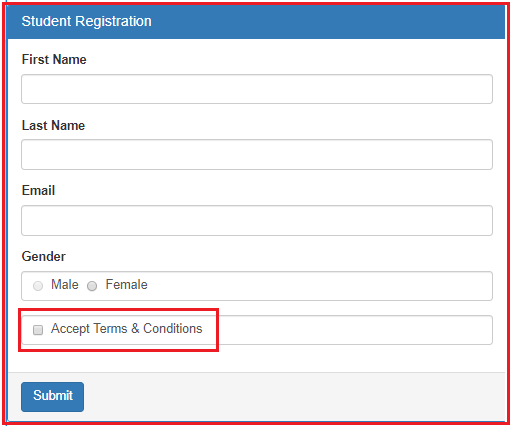
**What is a Checkbox?**

A **Checkbox** is an HTML element which allows the users to select multiple options from the available options. For example, in most of the website when you are filling a form you may find a checkbox for terms and conditions which needs to be accept in order to submit the form.

**Example to understand Checkbox in Angular Template Driven Forms:**

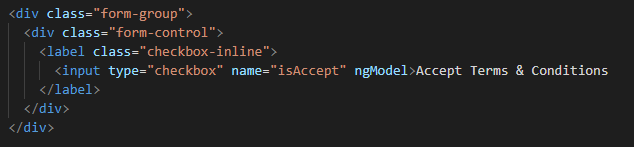
Let us understand how to create and use **Checkbox** in Angular Template Driven Forms. We are going to work with the same example that we worked in our previous article.

Now, we want to include “Accept Terms & Conditions” checkbox in the student registration form as shown in the below image. When we select the Accept Terms & Conditions checkbox and when we click the “Submit” button, we want to display the selected value of the checkbox in the console. Here, if the use select the checkbox then true will be logged into the console else false will be logged into the console.



**How to create check box in angular template driven forms?**

Please have a look at the below code which will create Accept Terms and Conditions checkbox.



**Code Explanation**

In the above code, we set the name attribute of the input element checkbox to isAccept. We have not set the value property here. This is because its value can be true of false. If the checkbox is checked or selected then the value is true else the value is false.

**The complete code of app.component.html:**

Following is the complete code of **app.component.htm**l file.

**<br/>**

**<div** class="container"**>**

**<div** class="row"**>**

**<div** class="form-bg"**>**

**<form** #studentForm="ngForm" (ngSubmit)="RegisterStudent(studentForm)" **>**

**<div** class="panel panel-primary"**>**

**<div** class="panel-heading"**>**

**<h3** class="panel-title"**>**Student Registration**</h3>**

**</div>**

**<div** class="panel-body"**>**

**<div** class="form-group"**>**

**<label** for="firstName"**>**First Name**</label>**

**<input** id="firstName" type="text" class="form-control"

name="firstName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label** for="lastName"**>**Last Name**</label>**

**<input** id="lastName" type="text" class="form-control"

name="lastName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

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

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

name="email" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label>**Gender**</label>**

**<div** class="form-control"**>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="male" ngModel**>**

Male

**</label>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="female" ngModel**>**

Female

**</label>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<div** class="form-control"**>**

**<label** class="checkbox-inline"**>**

**<input** type="checkbox" name="isAccept" ngModel**>**Accept Terms & Conditions

**</label>**

**</div>**

**</div>**

**</div>**

**<div** class="panel-footer"**>**

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

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**

**</div>**

**Modifying app.component.ts file:**

We want to log the posted form values into the console tab. So, modify the **app.component.ts**file as shown below.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

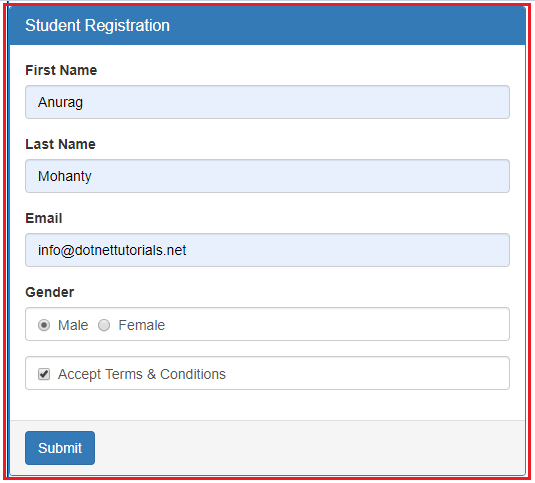
RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

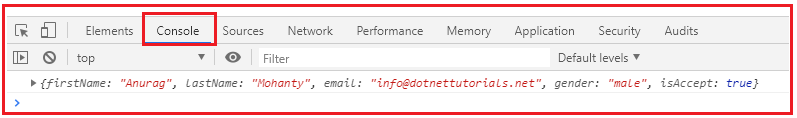
console.log**(**studentForm.value**)**;

**}**

**}**

Save the changes and browse the application, then open browser developers tool by pressing F12 key and click on the console tab. Fill the form and click on the submit button and you should see the posted form values in the console tab as shown in the below image.





**How to get a radio checkbox checked by default in Angular?**

As we know when working with real-time applications, many a times we need to provide the checkbox to be checked by default when the form load initially and we normally do this by adding the checked attribute of the checkbox.

If we include the checked attribute on the checkbox, then we may expect that the checkbox to be checked by default when the page load. But in angular template driven forms, you will not get that default checked when the page loads initially.So, lets include the “checked” attribute on the checkbox and verify this. So. Modify the Checkbox HTML code as shown below.

**<input type=”checkbox” name=”isAccept” ngModel checked>Accept Terms & Conditions**

With the above changes in place, now browse the application and you will see the the checkbox is not checked by default when the page load.

However, if we remove the “ngModel” directive from the checkbox as shown below, then you will see that the checkbox is checked when the form is load.

**<input type=”checkbox” name=”isAccept” checked>Accept Terms & Conditions**

As we already discussed, in Angular, we generally use the “ngModel” directive for two-way data binding. So when we put the ngModel directive back into the control then the “checked” attribute will not work on the checkbox. If we remove the ngModel then checked attributes work but two way data binding will not work.

**How to make it works?**

In order to make it work, we need to include a property lets say “isAccept” in the component class and initialize its value to true. So.modify the app.component.ts file as shown below.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

isAccept = **true**;

RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

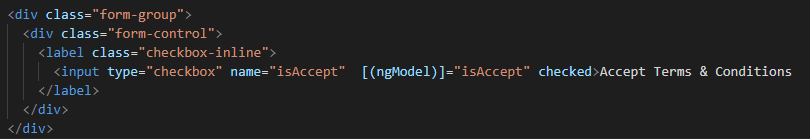
console.log**(**studentForm.value**)**;

**}**

**}**

**Modifying the app.component.html**

Now we include to include the ngModel directive and bind it with the component property isAccept. To do so, modify the checkbox HTML code in the **app.component.html** file as shown below.



With the above changes in place, now browse the application, you should see the Accept Terms & Conditions checkbox is checked by default when the form loads initially.

Now, even if you remove the “checked” attribute from the checkbox, then it still checked by default when the form loads. This is possible because of the Angular two-way data binding. In our example, we do not want the checkbox to be checked by default, so we remove the “checked” attribute and the “isAccept” property from the component class and ngModel directive.

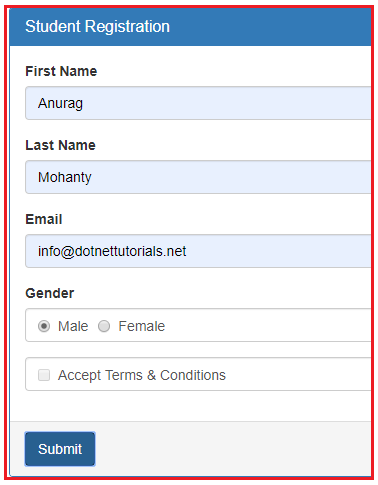
**How to disable a Checkbox in Angular Template Driven Forms?**

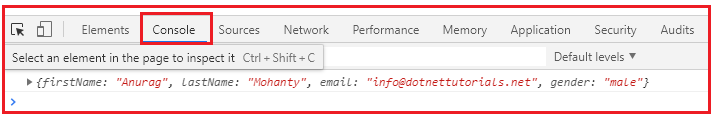
In order to disable a checkbox in Angular Template Driven Form, we need use the disabled attribute on the checkbox as shown below.

**<input type=”checkbox” name=”isAccept” ngModel disabled>Accept Terms & Conditions**

With the above changes, now it is not possible to select the checkbox. The most important point that you need to remember is, by default, the disabled form controls are not included in the Angular auto generated form model.

Since, the isAccept checkbox is disabled, it will not be included in the Angular generated form model. So, after the filling when you click on the button, in the console tab, you will not find the isAceept value as shown in the below image.





In our example, we do not want the checkbox to be disabled, so please remove the disabled attribute from the checkbox.

## ****DropDownList in Angular Template Driven Forms****

**What is a DropDownList?**

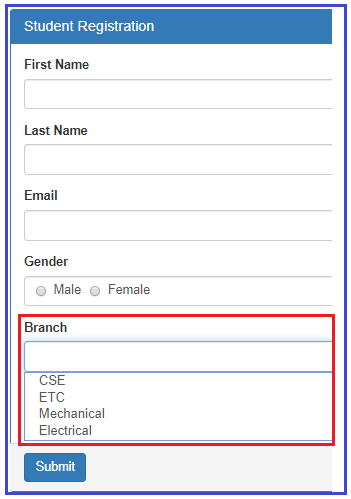
A DropDownList is an HTML Element which is nothing but a collection of list items from which it will allows the user to select a single list item. Depending on your business requirement you may either hard code the values or you may retrieve the values from a database table.

In this article, I am going to discuss both the approaches. First, we will discuss creating the DropDownList using the hard-coded value then we will see how to create the DropDownList with the values coming from a component.

**Example to understand DropDownList in Angular Template Driven Forms:**

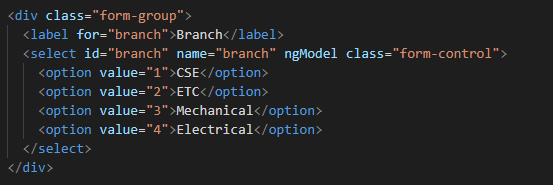
Let us understand how to create and use **DropDownList**in Angular Template Driven Forms with an example. We are going to work with the same example that we worked in our previous article.

Now, we want to include Branches dropdownlist in the student registration form as shown in the below image. When the user select a particular branch from the dropdownlist and click on the “Submit” button, then we want to display the selected drop down list value on the console.



**How to create the dropdownlist in angular using template driven forms?**

Please have a look at the following code which will create a Drop Down List with the required items.



**Code Explanation**

As shown in the code, we have hard coded the drop down list options in the HTML. Notice each option also has a corresponding value attribute and its value is the branch 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 our upcoming article.

**The complete code of app.component.html:**

Following is the complete code of **app.component.htm**l file.

**<br/>**

**<div** class="container"**>**

**<div** class="row"**>**

**<div** class="form-bg"**>**

**<form** #studentForm="ngForm" (ngSubmit)="RegisterStudent(studentForm)" **>**

**<div** class="panel panel-primary"**>**

**<div** class="panel-heading"**>**

**<h3** class="panel-title"**>**Student Registration**</h3>**

**</div>**

**<div** class="panel-body"**>**

**<div** class="form-group"**>**

**<label** for="firstName"**>**First Name**</label>**

**<input** id="firstName" type="text" class="form-control"

name="firstName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label** for="lastName"**>**Last Name**</label>**

**<input** id="lastName" type="text" class="form-control"

name="lastName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

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

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

name="email" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label>**Gender**</label>**

**<div** class="form-control"**>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="male" ngModel**>**

Male

**</label>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="female" ngModel**>**

Female

**</label>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<label** for="branch"**>**Branch**</label>**

**<select** id="branch" name="branch" ngModel class="form-control"**>**

**<option** value="1"**>**CSE**</option>**

**<option** value="2"**>**ETC**</option>**

**<option** value="3"**>**Mechanical**</option>**

**<option** value="4"**>**Electrical**</option>**

**</select>**

**</div>**

**<div** class="form-group"**>**

**<div** class="form-control"**>**

**<label** class="checkbox-inline"**>**

**<input** type="checkbox" name="isAccept" ngModel**>**Accept Terms & Conditions

**</label>**

**</div>**

**</div>**

**</div>**

**<div** class="panel-footer"**>**

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

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**

**</div>**

**Modifying app.component.ts file:**

We want to log the posted form values into the console tab. So, modify the **app.component.ts**file as shown below.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

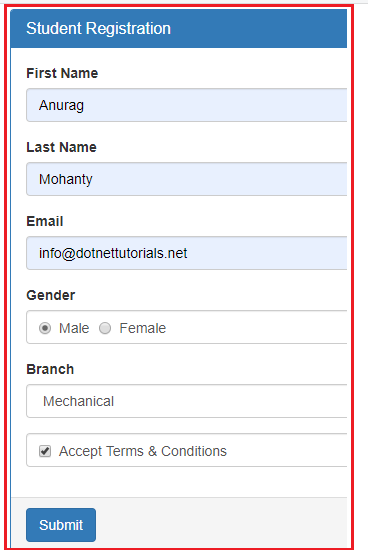
RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

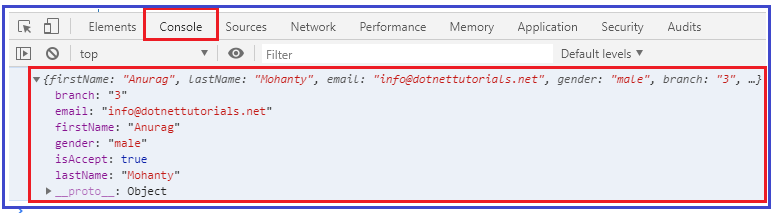
console.log**(**studentForm.value**)**;

**}**

**}**

Save the changes and browse the application, then open browser developers tool by pressing F12 key and click on the console tab. Fill the form, select one value from the drop down list and click on the submit button and you should see the posted form values in the console tab as shown in the below image.

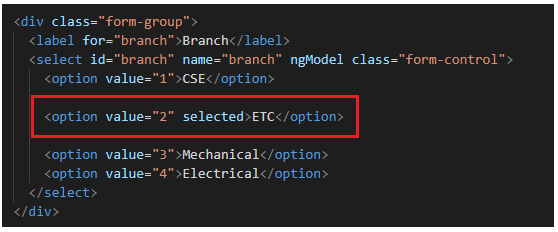




**How to have one of the dropdownlist item selected by default in Angular?**

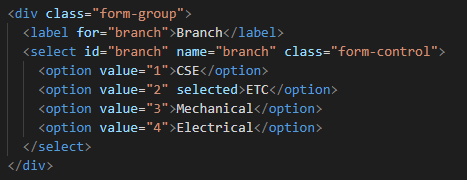
As we know when working with real-time applications, many a times we need to provide one option to be selected in the drop down list by default when the form load. And we normally do this by adding the **selected**attribute on one of the option of drop down list.

If we include the **selected**attribute on the **dropdownlist**, then we may expect that option or item to be selected by default. But in angular template driven forms, that will not work. Lets include the “**selected**” attribute on the ETC branch option to verify this. So. Modify the Drop Down List HTML code as shown below.



With the above changes in place, now browse the application and you will see that the ETC department is not selected by default when the page load.

However, if you remove the “ngModel” directive from the select list as shown below, then you will see that the ETC branch is selected when the form is load.



As we already discussed, we use the “ngModel” directive in angular for two-way data binding. So when we put the ngModel directive back into the control then the “**selected**” attribute will not work on the drop down list or select list. If we remove the ngModel directive from the control then **selected**attribute work but two way data binding will not work.

**How to make it works?**

In order to make it work, what we need to do is, we need to add a property lets say “BranchId” in the component class and initialize its value with the branch value that you want to be selected when the page load. As we want ETC branch to be selected by default and as its value is 1, so, we need to BranchId property and initialize its value to 2 in the AppComponent class. So, modify the **app.component.ts** file as shown below.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

BranchId = "2";

RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

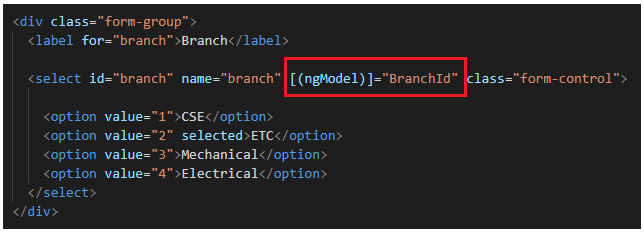
console.log**(**studentForm.value**)**;

**}**

**}**

**Modifying the app.component.html file**

Next, we include to include the ngModel directive and bind it with the component property BranchId. To do so, modify the Select List code in the **app.component.html** file as shown below.



With the above changes in place, now if you browse the application, then you should see the ETC branch is selected by default in the Branch Drop Down List when the form loads .

Now, even if you remove the “selected” attribute from the Option, then also it will work i.e. it will select the ETC by default. This is possible because of the two-way data binding which is provided by angular. In our example, we do not want the ETC to be selected by default, so we remove the “selected” attribute and the “BranchId” property from the component class and ngModel directive.

**How to disable a Drop Down List in Angular Template Driven Forms?**

In order to disable a drop down list in Angular Template Driven Form, we need use the disabled attribute on the select element as shown below.

**<select id=”branch” name=”branch” class=”form-control” ngModel disabled>**

With the above changes, now it is not possible to select any item from the drop down list. As we already discussed, by default, the disabled form controls are not included in the Angular auto generated form model.

In our example, we do not want the dropdownlist to be disabled, so please remove the disabled attribute from the select element.

**How to get the select list options from the component class?**

As of now we have hard-coded the select list options in HTML itself. In most of the real-time applications, you will get this data from a database. So, modify the **app.component.ts** file as shown below. Here, we created one property called **Branches**which will return the list of items that we want to show in the drop down list. It has two properties id and name.

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

**import** **{** NgForm **}** from '@angular/forms'

@Component**({**

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: **[**'./app.component.css'**]**

**})**

**export** **class** AppComponent **{**

RegisterStudent**(**studentForm: NgForm**)**: **void** **{**

console.log**(**studentForm.value**)**;

**}**

Branches: **any[]** = **[**

**{** id: 1, name: 'CSE' **}**,

**{** id: 2, name: 'ETC' **}**,

**{** id: 3, name: 'Mechanical' **}**,

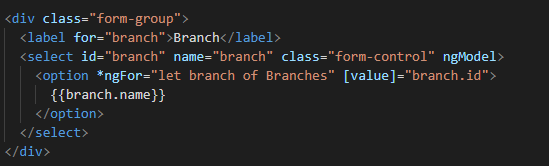
**{** id: 4, name: 'Electrical' **}**

**]**;

**}**

**Modifying the app.component.html file:**

Next we need to modify the drop down list code in the app.component.html file as shown below.



**Code explanation:**

As you can see on the “option” element we are using ngFor structural directive to loop through all the branches that we have in the “Branches” property of the component class.

For each “Branch” store in the “Branches” array, we get an option. The option value is the Branch id and the display text is the Branch name.

Please have a look at the square brackets around the [value] property. This is property binding in Angular. If you remove the square brackets the value for each option will be the literal text “branch.id” instead of the branch id (1 or 2 or 3, etc.). To display the Branch name we are using angular interpolation.

**The complete code of app.component.html:**

**<br/>**

**<div** class="container"**>**

**<div** class="row"**>**

**<div** class="form-bg"**>**

**<form** #studentForm="ngForm" (ngSubmit)="RegisterStudent(studentForm)" **>**

**<div** class="panel panel-primary"**>**

**<div** class="panel-heading"**>**

**<h3** class="panel-title"**>**Student Registration**</h3>**

**</div>**

**<div** class="panel-body"**>**

**<div** class="form-group"**>**

**<label** for="firstName"**>**First Name**</label>**

**<input** id="firstName" type="text" class="form-control"

name="firstName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label** for="lastName"**>**Last Name**</label>**

**<input** id="lastName" type="text" class="form-control"

name="lastName" ngModel**>**

**</div>**

**<div** class="form-group"**>**

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

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

name="email" ngModel**>**

**</div>**

**<div** class="form-group"**>**

**<label>**Gender**</label>**

**<div** class="form-control"**>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="male" ngModel**>**

Male

**</label>**

**<label** class="radio-inline"**>**

**<input** type="radio" name="gender" value="female" ngModel**>**

Female

**</label>**

**</div>**

**</div>**

**<div** class="form-group"**>**

**<label** for="branch"**>**Branch**</label>**

**<select** id="branch" name="branch" class="form-control" ngModel**>**

**<option** \*ngFor="let branch of Branches" [value]="branch.id"**>**

{{branch.name}}

**</option>**

**</select>**

**</div>**

**<div** class="form-group"**>**

**<div** class="form-control"**>**

**<label** class="checkbox-inline"**>**

**<input** type="checkbox" name="isAccept" ngModel**>**Accept Terms & Conditions

**</label>**

**</div>**

**</div>**

**</div>**

**<div** class="panel-footer"**>**

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

**</div>**

**</div>**

**</form>**

**</div>**

**</div>**