Forms :

**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.

##### ****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.

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 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