# Title : Angular Journey 3 - Angular JS Form Building Block (Template vs Reactive Forms)

# Description

Angular Form aka Html form is generally used it as a communication endpoint to submit data to the server for processing. In Angular 2, there are two ways to design forms namely Template vs Reactive driven form. Both approaches have its own benefits and difference implementation. In this section, I would like to share some of the knowledge and experience which I have learned from Deborach Kurata’s Angular 2: Reactive Forms [Course].

There is a quote saying “the best way to learn is by doing”. Learning is easy these days especially with practical-demo and videos provided by lecturer. All we need to do is follow the instructions and work on it. Things start to become uncluttered, when we tried to build the things with our idea and logic. After I finished learning the Deborach Video, I started building my own movie review web site with Angular 2 web site in reactive way and found the a few issues…..

If you like to view some of the key notes given by Deborach, please click on the Notes Button. You can also down my movie review demo, if you like to play around. So far, it is fun to work with Angular 2. Technology amaze people constantly….

# Notes

The following are the key notes which I have learned from Deborach Kurata’s Angular 2: Reactive Forms [Course]

## Form Building Blocks

In Angular 2, there are two ways to create Forms.

* Template Form Type
* Reactive Form Type

Both approaches have its own advantages. Depending on the business requirements, we can tailor our form design using one of those approach

|  |  |
| --- | --- |
| Template Driven | Reactive |
| Easy to Use | More Flexible and suitable for complex scenarios (eg changing validation based user’s selection or form state) |
| Similar to Angular 1 | Immutable data Model (reactive form do not use data binding, so the form cannot mutate our data model) |
| Two-way data Binding & Minimal Component Code | Easier to perform an action on a value change |
| Automatically tracks form and input element state | Reactive transformations -> DebounceTime (to delay reacting to user input), and DistinctUntilChanged to Ignore values that are same as previously processed values. |
|  | Easily add input elements dynamically |
|  | Easier Unit Testing |

## Form State

Angular has a set of predefined form and input element states. These state defined whether user has already updated or not. There are three types of states existed in Angular namely;

* Value Changed
  + It has two states
    - **Pristine**
    - **Dirty**
  + Form state is Pristine, if all input elements on a form is pristine.
* Validity
  + It has two states
    - **Valid** - input value is value is valid if it is passed defined validation rules
    - **Error** - input value failed defined validation rules
* Visited
  + It has two states
    - **Touched** – use has set focus into input element and left the input element
    - **Untouched** – opposite to touched

These form states are used by both template and reactive driven forms to track state and values

## Form Control and Form Group

* FormControl – tracks the value and state of individual input element (eg, input box)
* FormGroup – tracks the value and state of a group of form controls (eg, address inputs group). FormGroup themselves can be nested within other FormGroups.

## Form Model

Form Model is data structure that represents the HTML form. Below is the screenshot provided in Deborach’s Video. Form Model retains a few values including form state, form value, child controls, formcontrols, nested formgroups. This form model is the same for both template-driven and Reactive Forms, but how it is created is different.



In **Template-Driven approach**, Form Model Creation process are passed it template file and write HTML in our template for the form element, input elements, data binding, validation rules using attributes and validation error messages. In component class, it define properties aka data model for data binding and implement methods for form operations.

In **Reactive** approach, it shift the responsibility for creating form model to component class. We define the form model by creating instance of FormGroup and FormControl building blocks in our component class. It defines validation rules in the class.

## Form Directives

Template-driven and reactive forms use difference sets of directives for binding FormControl and FormGroup.

|  |  |
| --- | --- |
| Template-driven (FormsModule) | Reactive (ReactiveFormsModule) |
| It first import form module to bring set of directives, which includes   * **ngForm** – to access form model * **ngModel** – for two-way binding * **ngModelGroup** – for grouping input elements within the form   Angular creates form model for us | It first import reactive form module to bring set of appropriate directive, which includes   * formGroup * formControl * formControlName * formGroupName * formArrayName   Angular does not create form model for us |

## Template Driven Form Syntax

Below screenshot is illustrates template-driven and reactive form samples from Deborach’s video.

|  |  |
| --- | --- |
|  |  |
| It used “#firstNameVar” template reference variables to access model data | It access form model directly wihout using reference. |