Building a template-driven form

Template-driven forms use [two-way data binding](https://angular.dev/guide/templates/two-way-binding) to update the data model in the component as changes are made in the template and vice versa.

Template vs Reactive forms

Angular supports two design approaches for interactive forms. Template-driven forms allow you to use form-specific directives in your Angular template. Reactive forms provide a model-driven approach to building forms.

Template-driven forms are a great choice for small or simple forms, while reactive forms are more scalable and suitable for complex forms. For a comparison of the two approaches,

Template-driven forms rely on directives defined in the FormsModule.

| **Directives** | **Details** |
| --- | --- |
| NgModel | Reconciles value changes in the attached form element with changes in the data model, allowing you to respond to user input with input validation and error handling. |
| NgForm | Creates a top-level FormGroup instance and binds it to a <form> element to track aggregated form value and validation status. As soon as you import FormsModule, this directive becomes active by default on all <form> tags. You don't need to add a special selector. |
| NgModelGroup | Creates and binds a FormGroup instance to a DOM element. |
|  |  |

Validations in [Template-driven forms](https://www.tektutorialshub.com/angular/angular-template-driven-forms/) are provided by the Validation directives. The [Angular Forms Module](https://www.tektutorialshub.com/angular/angular-forms-fundamentals/) comes with several built-in validators. You can also create your own custom Validator.

The Built-in validators use the HTML5 validation attributes like required, minlength, maxlength & pattern. Angular interprets these validation attributes and add the validator functions to [FormControl](https://www.tektutorialshub.com/angular/formcontrol-in-angular/) instance.

Angular forms module keep track of the state of our form and each of its form elements. These states are exposed to the user through FormGroup, FormArray & FormControl objects.

We get the reference to the top-level FormGroup instance by creating a template variable and bind it to ngForm. We have already done it when we had added the #contactForm="ngForm" in our form tag.

The [FormGroup](https://www.tektutorialshub.com/angular/formgroup-in-angular/) has a valid property, which is set to true if all of its child controls are valid. We use it to set the disabled attribute of the submit button.

[Track control states](https://angular.dev/guide/forms/template-driven-forms#track-control-states)

Adding the NgModel directive to a control adds class names to the control that describe its state. These classes can be used to change a control's style based on its state.

The following table describes the class names that Angular applies based on the control's state.

| **States** | **Class if true** | **Class if false** |
| --- | --- | --- |
| The control has been visited. | ng-touched | ng-untouched |
| The control's value has changed. | ng-dirty | ng-pristine |
| The control's value is valid. | ng-valid | ng-invalid |

When you use [(ngModel)] on an element, you must define a name attribute for that element. Angular uses the assigned name to register the element with the NgForm directive attached to the parent <form> element.

<p>

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

      <input    type="text"

                id="firstname"

                name="firstname"

                required minlength="10"

                #firstname="ngModel"

                [(ngModel)]="contact.firstname">

    </p>

    <div \*ngIf="!firstname?.valid && (firstname?.dirty || firstname?.touched)" class="error">

      <div \*ngIf="firstname.hasError('required')">

        First Name is required

      </div>

      <div \*ngIf="firstname.hasError('minlength')">

        First Name Minimum Length is 10

      </div>

    </div>

Extend the input tag with a template reference variable that you can use to access the input box's Angular control from within the template. In the example, the variable is #firstname="ngModel"..

The template reference variable (#firstname) is set to "ngModel" because that is the value of the [NgModel.exportAs](https://angular.dev/api/core/Directive" \l "exportAs) property. This property tells Angular how to link a reference variable to a directive.

References

<https://www.tektutorialshub.com/angular/template-driven-form-validation-in-angular/>

<https://angular.dev/guide/forms/template-driven-forms>