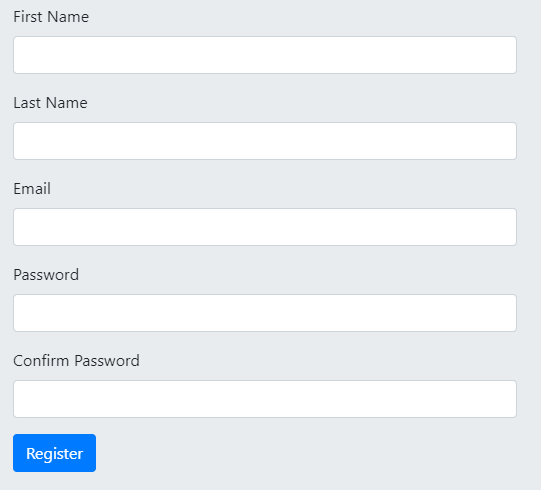
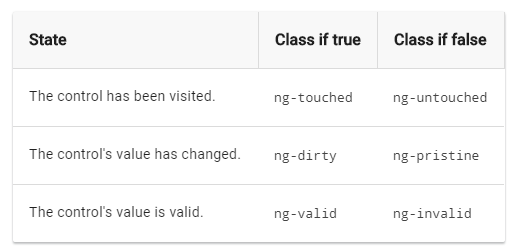
# Template driven forms

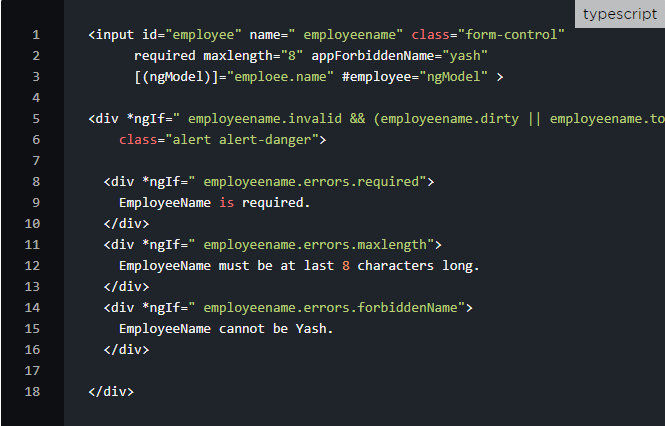
Template driven forms are forms where we write logic, validations, controls etc, in the template part of the code (html code). The template is responsible for setting up the form, the validation, control, group etc. Template driven forms are suitable for simple scenarios, uses two way data binding using the [(NgModel)] syntax, easier to use though unit testing might be a challenge.

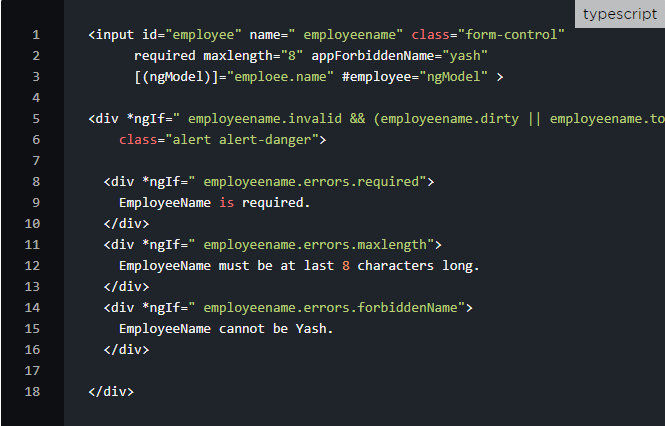




# Template Driven Forms Validation

In template-driven form, we will be adding the same attributes as we would with native HTML form validation. Angular uses directives to match these attributes with validator functions in the framework. Whenever the value of a form control changes, Angular runs validation and generates either a list of validation errors, which results in an INVALID status or null, which results in a VALID status. We can check the control’s state by exporting ngModel to a local template variable. Let’s consider the below example, which will explain how the exported NgModel to a variable called name.

****

****

In the above example, the input tag contains the validation attributes like required and minlength. Note that it also carries a custom validator directive forbiddenEmployeeName. #employeename="ngModel" exports NgModel into a local variable called employeename. NgModel contains many properties of underlying FormControl instance. By using this, we will be able to check the control states such as valid and dirty. The \*ngIf on the div tag validates if the employeename is invalid and if the control is either dirty or touched.

**Template reference variables (#var)**

A template reference variable is often a reference to a DOM element within a template. It can also refer to a directive (which contains a component), an element, TemplateRef, or a web component.

# @Viewchild decorator

A ViewChild is a component, directive, or element as a part of a template. If we want to access a child component, directive, DOM element inside the parent component, we use the decorator @ViewChild() in Angular.

# Form State and Input State

Input State:

untouched The field has not been touched yet

touched The field has been touched

pristine The field has not been modified yet

dirty The field has been modified

invalid The field content is not valid

valid The field content is valid

Form State:

pristine No fields have been modified yet

dirty One or more have been modified

invalid The form content is not valid

valid The form content is valid

submitted The form is submitted

