# Angular Forms

Forms are the mainstay of business applications. You use forms to log in, submit a help request, place an order, book a flight, schedule a meeting, and perform countless other data-entry tasks.

In developing a form, it's important to create a data-entry experience that guides the user efficiently and effectively through the workflow.

Developing forms requires design skills (which are out of scope for this page), as well as framework support for two-way data binding, change tracking, validation, and error handling, which you'll learn about on this page.

# Types of Forms:

1. Template-driven Forms
2. Reactive Forms

# Template-driven Forms

Gives more control on HTML.

The variable #formName is reference to the NgForm directive that governs the form as a whole.

Angular automatically creates and attaches an NgForm directive to the <form> tag.

The NgForm directive supplements the form element with additional features. It holds the controls you created for the elements with an ngModel directive and name attribute, and monitors their properties, including their validity. It also has its own valid property which is true only if every contained control is valid.

|  |  |  |
| --- | --- | --- |
| **State** | **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 |

# Conclusion

The Angular form discussed in this page takes advantage of the following framework features to provide support for data modification, validation, and more:

* An Angular HTML form template.
* A form component class with a @Component decorator.
* Handling form submission by binding to the NgForm.ngSubmit event property.
* Template-reference variables such as #heroForm and #name.
* [(ngModel)] syntax for two-way data binding.
* The use of name attributes for validation and form-element change tracking.
* The reference variable’s valid property on input controls to check if a control is valid and show/hide error messages.
* Controlling the *Submit* button's enabled state by binding to [NgForm](https://angular.io/api/forms/NgForm) validity.
* Custom CSS classes that provide visual feedback to users about invalid controls.