view -> input -> validate -> form and data model

Reactive Forms

Robust

Scalable

Reusable

Testable

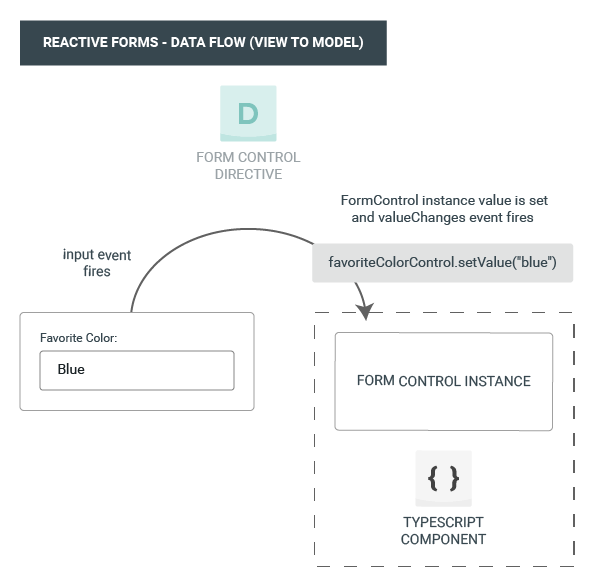
You are already using reactive pattern

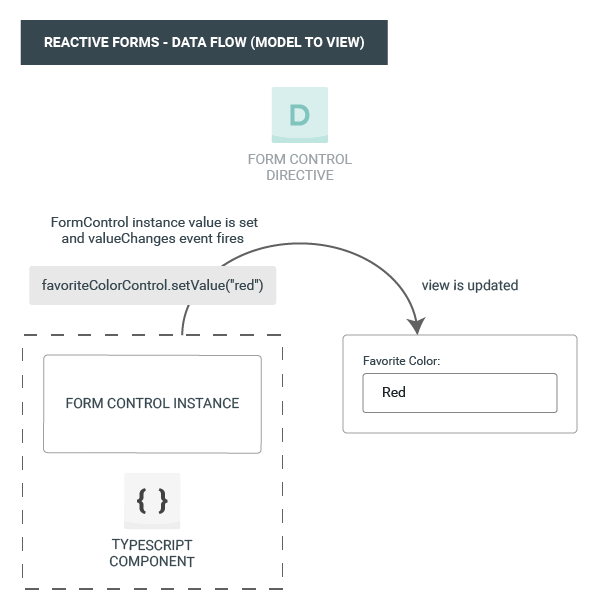
Form model is source of truth

With reactive forms, the form model is explicitly defined in the component class. The reactive form directive (in this case, FormControlDirective) then links the existing FormControl instance to a specific form element in the view using a value accessor (ControlValueAccessor instance).

The reactive form directive (in this case, FormControlDirective)

ControlValueAccessor instance

Updates from the view to the model and from the model to the view are synchronous and aren't dependent on the UI rendered



**Note :** To use reactive form add ReactiveFormsModule to imports array

Template Driven Forms

Simple forms

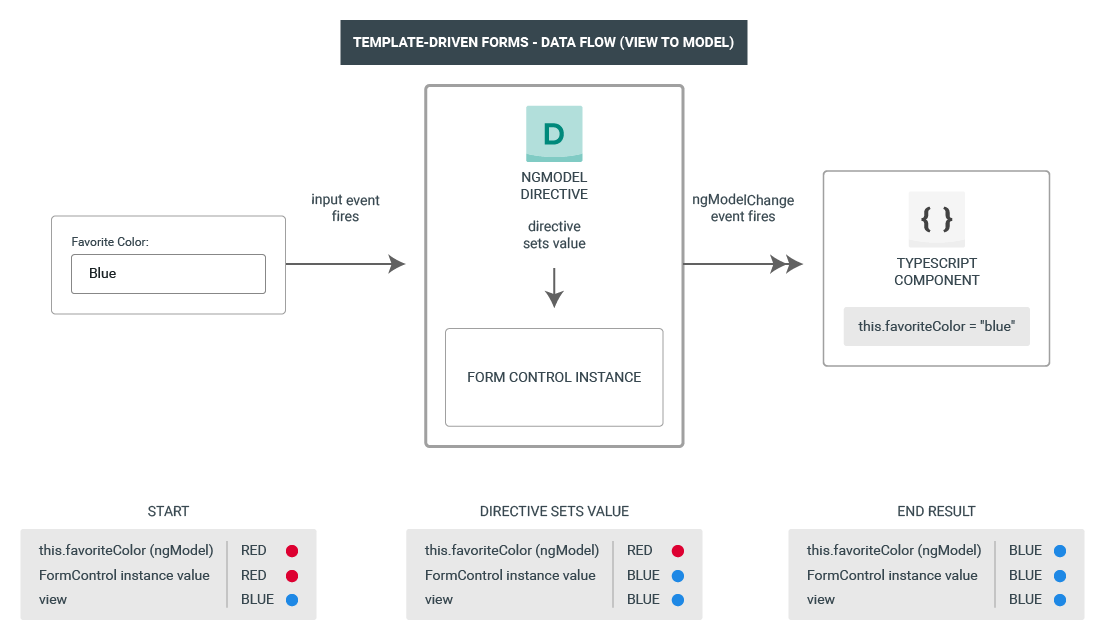
basic form requirements and logic that can be managed solely in the template, use template-driven forms.

template-driven form directive NgModel

you no longer have direct control over the form model.

You can display the value in these ways:

* Through the valueChanges observable where you can listen for changes in the form's value in the template using AsyncPipe or in the component class using the subscribe() method.
* With the value property. which gives you a snapshot of the current value.



Common foundation

Both reactive and template-driven forms share underlying building blocks.

**FormControl** tracks the value and validation status of an individual form control.

**FormGroup** tracks the same values and status for a collection of form controls.

**FormArray** tracks the same values and status for an array of form controls.

**ControlValueAccessor** creates a bridge between Angular FormControl instances and native DOM elements.

Form validation

* Reactive forms define custom validators as **functions** that receive a control to validate.
* Template-driven forms are tied to template directives, and must provide custom validator directives that wrap validation functions.

Partial model update

Patching the model value:

There are two ways to update the model value:

* Use the setValue() method to set a new value for an individual control. The setValue() method strictly adheres to the structure of the form group and replaces the entire value for the control.
* Use the patchValue() method to replace any properties defined in the object that have changed in the form model.

Conclusion :

I Practiced

* reactive form vs template driven form.
* Reactive Form
  + AsyncPipe
  + jsonPipe
  + testing
  + FormControl
  + Nested formControl
  + setValue
  + Updating value partially by patchValue