**Page:**

**F1 (input)**

**F2 (input)**

**..**

**F2 (input)**

**[ngModel]**

**ModelDriven**

**Frm = FormBuilder(){**

**}**

**Reactive forms are more robust: they're more scalable, reusable, and testable. If forms are a key part of your application, or you're already using reactive patterns for building your application, use reactive forms.**

**Template-driven forms are useful for adding a simple form to an app, such as an email list signup form. They're easy to add to an app, but they don't scale as well as reactive forms. If you have very basic form requirements and logic that can be managed solely in the template, use template-driven forms.**

Key differences

The table below summarizes the key differences between reactive and template-driven forms.

|  | REACTIVE | TEMPLATE-DRIVEN |
| --- | --- | --- |
| Setup (form model) | More explicit, created in component class | Less explicit, created by directives |
| Data model | Structured | Unstructured |
| Predictability | Synchronous | Asynchronous |
| Form validation | Functions | Directives |
| Mutability | Immutable | Mutable |
| Scalability | Low-level API access | Abstraction on top of APIs |

## 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.

<input type="text" [formControl]="name">

import { FormControl } from '@angular/forms';

name = new FormControl('');

Value: {{ name.value }}

this.name.setValue(new'); //replace

import { FormGroup, FormControl } from '@angular/forms';

profileForm = new FormGroup({ firstName: new FormControl(''), lastName: new FormControl(''), });

<form [formGroup]="profileForm"> <label> First Name: <input type="text" formControlName="firstName"> </label> <label> Last Name: <input type="text" formControlName="lastName"> </label> </form>

### **Setup in reactive forms**

import { Component } from '@angular/core';

import { FormControl } from '@angular/forms';

@Component({

selector: 'app-reactive-favorite-color',

template: `

Favorite Color: <input type="text" [formControl]="favoriteColorControl">

`

})

export class FavoriteColorComponent {

favoriteColorControl = new FormControl('');

}

### **Setup in template-driven forms**

import { Component } from '@angular/core';

@Component({

selector: 'app-template-favorite-color',

template: `

Favorite Color: <input type="text" [(ngModel)]="favoriteColor">

`

})

export class FavoriteColorComponent {

favoriteColor = '';

}

## Data flow in forms

When building forms in Angular, it's important to understand how the framework handles data flowing from the user or from programmatic changes. Reactive and template-driven forms follow two different strategies when handling form input. The data flow examples below begin with the favorite color input field example from above, and then show how changes to favorite color are handled in reactive forms compared to template-driven forms.

### **Data flow in reactive forms**

As described above, in reactive forms each form element in the view is directly linked to a form model (FormControlinstance). Updates from the view to the model and from the model to the view are synchronous and aren't dependent on the UI rendered. The diagrams below use the same favorite color example to demonstrate how data flows when an input field's value is changed from the view and then from the model.

Reactive nesting

profileForm = new FormGroup({ firstName: new FormControl(''), lastName: new FormControl(''), address: new FormGroup({ street: new FormControl(''), city: new FormControl(''), state: new FormControl(''), zip: new FormControl('') }) });

<div formGroupName="address"> <h3>Address</h3> <label> Street: <input type="text" formControlName="street"> </label> <label> City: <input type="text" formControlName="city"> </label> <label> State: <input type="text" formControlName="state"> </label> <label> Zip Code: <input type="text" formControlName="zip"> </label> </div>

Template Driven

constructor( public id: number, public name: string, public power: string, public alterEgo?: string ) { }

in app.module.ts

import { NgModule } from '@angular/core'; import { BrowserModule } from '@angular/platform-browser'; import { FormsModule } from '@angular/forms';

imports: [ BrowserModule, FormsModule ],