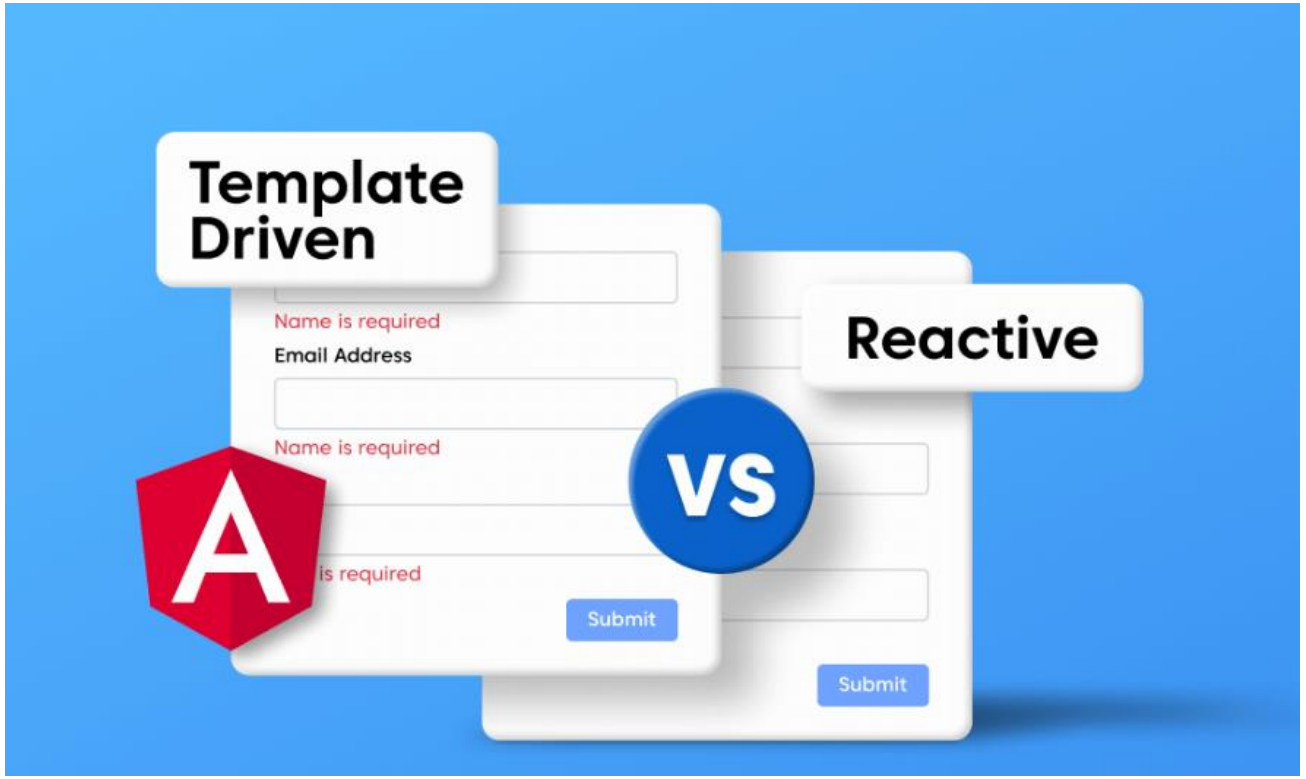
Los formularios casi siempre están presentes en cualquier sitio web o aplicación, ya que la entrada de datos del usuario es una de las funciones requeridas más comunes. Permiten realizar innumerables tareas de entrada de datos, como crear cuentas, iniciar sesión, realizar pedidos, buscar registros, etc.



Understanding Angular Forms 1.1 Types of Forms in Angular

1. Template-Driven Forms
2. Reactive Forms

Template-Driven Forms (Formularios basados en plantillas)

```
imports: [FormsModule],
```

Los formularios basados en plantillas **son ideales para formularios más sencillos** donde se desea que **Angular se encargue de gran parte del trabajo pesado**. Son declarativos y **se basan en directivas integradas en la plantilla HTML** para gestionar la vinculación y validación de datos del formulario.

Reactive Forms (Formularios reactivos)

```
imports: [ReactiveFormsModule]
```

Los formularios reactivos **ofrecen mayor control y flexibilidad**. Se basan **en programación typescript** y son adecuados para **formularios complejos con requisitos de validación dinámicos**. Con los formularios reactivos, se define la estructura y el comportamiento del formulario en la clase del componente.

Template-Driven Forms (en el hml)

```
<form #myForm="ngForm" (ngSubmit)="save()">
        <input id="name" type="text"
               [ (ngModel) ]="user.Name"
               name="Name"
               #Name="ngModel"/>
</form>
```

#myForm="ngForm" → Crea una **variable de plantilla local** llamada myForm que **hace referencia al objeto** **NgForm** asociado a ese <form>.

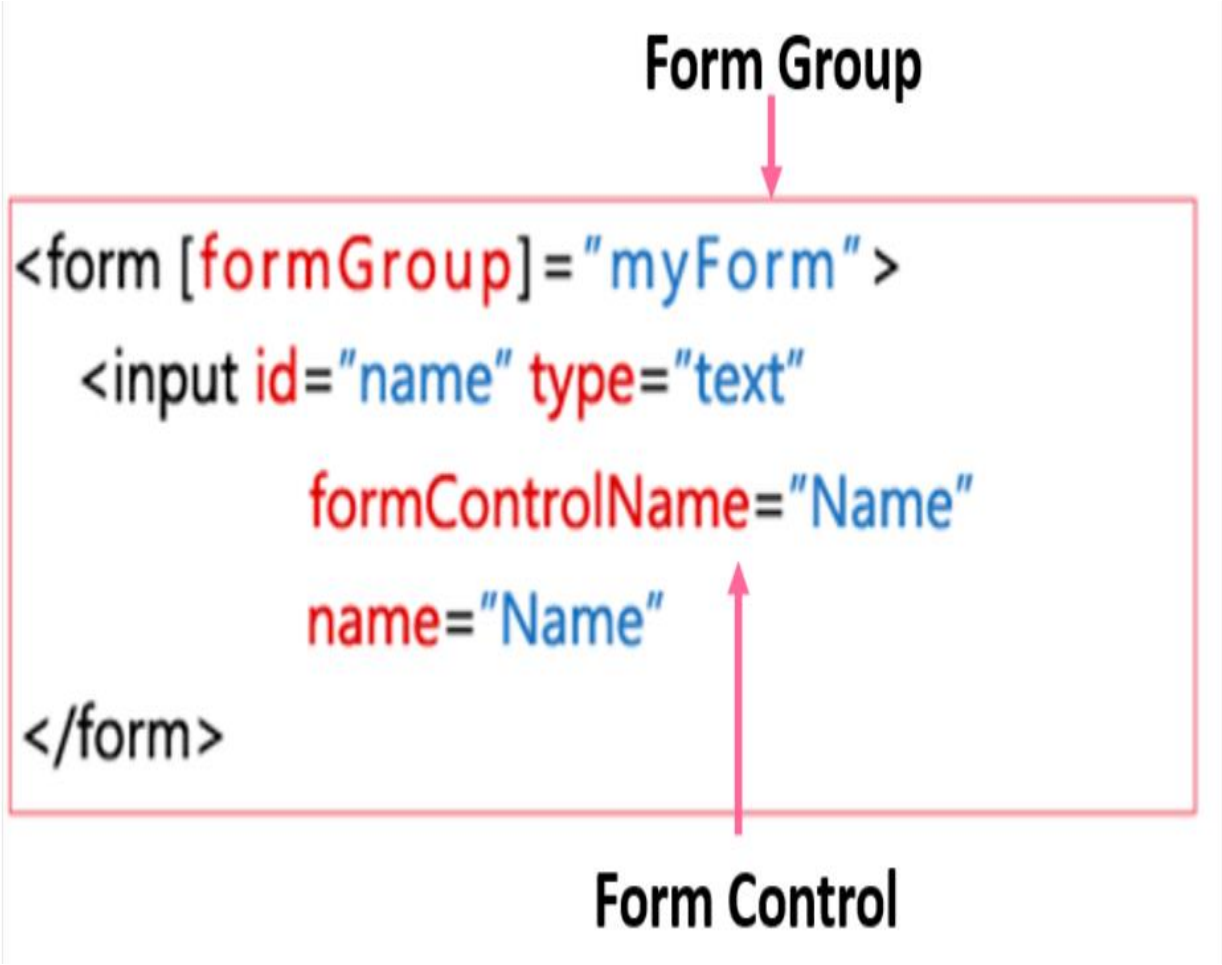
- Luego puedes acceder a:
  - `formTemp.value` → los valores del formulario
  - `formTemp.valid` → si el formulario es válido

En los **formularios basados en plantillas**, no creamos objetos de control de formulario de Angular, sino que las directivas de Angular los crean automáticamente utilizando la información de nuestra configuración de enlace de datos.

No tenemos que insertar ni extraer valores de datos, ya que Angular se encarga de ello mediante la directiva ngModel y actualiza el modelo de datos según los cambios del usuario al usar el formulario.

Los formularios basados en plantillas se configuran en código HTML. Son fáciles de usar y se adaptan a formularios sencillos. Utilizan directivas (ngForm, ngModel) y nombres de referencia **(#refName**) para crear formularios.

Reactive Forms (en el hml)



Los formularios reactivos se configuran en la clase del componente. **Se basa en el uso de clases como FormControl y FormGroup**, que rastrean el valor y el estado de validación. **Los formularios reactivos facilitan las pruebas y la validación.**

Template-Driven Forms (en el ts)

```ts
import { Component } from '@angular/core';
import { FormsModule } from '@angular/forms';

@Component({
  selector: 'app-formulario-template',
  imports: [FormsModule],
  templateUrl: './formulario-template.component.html',
  styleUrl: './formulario-template.component.css'
})
export class FormularioTemplateComponent {
  nombre = '';
  email = '';
  edad: number | null = null;

  enviarTemplate(form: any) {
    if (form.valid) {
      alert('Formulario por template: datos enviados');
      this.nombre="";
      this.edad=null;
      this.email="";
    }
  }
}
```

Template-Driven Forms (en el html)

Go to component

```
1  <h2>Template-Driven Form</h2>
2
3  <form #formTemp="ngForm" (ngSubmit)="enviarTemplate(formTemp)">
4
5    <div>
6      <label>Nombre:</label>
7      <input name="nombre" [(ngModel)]="nombre" required>
8    </div>
9
10   <div>
11     <label>Email:</label>
12     <input name="email" [(ngModel)]="email" type="email" required>
13   </div>
14
15   <div>
16     <label>Edad:</label>
17     <input name="edad" [(ngModel)]="edad" type="number" required min="1">
18   </div>
19
20   <div>
21   <button type="submit" [disabled]="formTemp.invalid">Enviar</button>
22   </div>
23
24 </form>
25
26 <br>
27 <div>
28 Formulario valido: {{formTemp.valid}}
29
30 </div>
```

Reactive Forms (en el ts)

```typescript
import { Component } from '@angular/core';
import { FormBuilder, FormGroup, ReactiveFormsModule, Validators } from '@angular/forms
import { CommonModule } from '@angular/common';

@Component({
  selector: 'app-formulario-reactive',
  imports: [ReactiveFormsModule, CommonModule],
  templateUrl: './formulario-reactive.component.html',
  styleUrl: './formulario-reactive.component.css'
})
export class FormularioReactiveComponent {

  miFormulario: FormGroup;

  constructor(private fb: FormBuilder) {
    this.miFormulario = this.fb.group({
      nombre: ['', Validators.required],
      email: ['', [Validators.required, Validators.email]],
      edad: [null, [Validators.required, Validators.min(1)]]
    });
  }

  enviar(): void {
    if (this.miFormulario.valid) {
      alert('Formulario reactivo: datos enviados');
      this.miFormulario.reset();
    }
  }

}
```

Reactive Forms
(en el html)

Go to component

```
<h2>Formulario Reactivo con FormBuilder</h2>

<form [formGroup]="miFormulario" (ngSubmit)="enviar()">

  <div>
    <label for="nombre">Nombre:</label>
    <input id="nombre" type="text" formControlName="nombre">
  </div>

  <div>
    <label for="email">Email:</label>
    <input id="email" type="email" formControlName="email">
  </div>

  <div>
    <label for="edad">Edad:</label>
    <input id="edad" type="number" formControlName="edad">
  </div>

  <div>
    <button type="submit" [disabled]="miFormulario.invalid">Enviar</button>
  </div>
</form>

<div>
Formulario valido: {{miFormulario.valid}}

</div>

@if(miFormulario.valid){
<div>
  <h4>Datos a enviar:</h4>
  <pre>{{ miFormulario.value | json }}</pre>
</div>
}
```

Template-Driven Forms (en el html)

¿Qué es #nombreCtrl="ngModel"?

Es una **referencia local** a la **directiva NgModel** que Angular aplica a ese input.

Te permite acceder al **estado del input** directamente en el HTML.

```
<form #formTemp="ngForm" (ngSubmit)="enviarTemplate(formTemp)">

  <div>
    <label>Nombre:</label>
    <input name="nombre" [(ngModel)]="nombre" required #nombreCtrl="ngModel">
    @if (nombreCtrl.invalid && nombreCtrl.touched) {
      <p style="color: red;">El nombre es obligatorio</p>
    }
  </div>

  <div>
    <label>Email:</label>
    <input name="email" [(ngModel)]="email" type="email" required #emailCtrl="ngModel">
    @if (emailCtrl.errors?.['required'] && emailCtrl.touched) {
      <p style="color: red;">El email es obligatorio</p>
    }
    @if (emailCtrl.errors?.['email'] && emailCtrl.touched) {
      <p style="color: red;">Formato de email no válido</p>
    }
  </div>

  <div>
    <label>Edad:</label>
    <input name="edad" [(ngModel)]="edad" type="number" required min="1" #edadCtrl="ngModel">
    @if (edadCtrl.errors?.['required'] && edadCtrl.touched) {
      <p style="color: red;">La edad es obligatoria</p>
    }
    @if (edadCtrl.errors?.['min'] && edadCtrl.touched) {
      <p style="color: red;">La edad debe ser mayor a 0</p>
    }
  </div>

  <div>
    <button type="submit" [disabled]="formTemp.invalid">Enviar</button>
  </div>

</form>
```

Reactive Forms (en el html)

- Angular marca un campo como **inválido** cuando su valor rompe alguna regla que tú le definiste con **validadores**

```
<h2>Reactive Form</h2>

<form [formGroup]="miFormulario" (ngSubmit)="enviarReactivo()">
  <label>Nombre:</label>
  <input formControlName="nombre">
  @if (miFormulario.get('nombre')?.invalid && miFormulario.get('nombre')?.touched) {
    <div>Nombre es obligatorio.</div>
  }

  <label>Email:</label>
  <input formControlName="email" type="email">
  @if (miFormulario.get('email')?.invalid && miFormulario.get('email')?.touched) {
    <div>Email no válido.</div>
  }

  <label>Edad:</label>
  <input formControlName="edad" type="number">
  @if (miFormulario.get('edad')?.invalid && miFormulario.get('edad')?.touched) {
    <div>Edad debe ser mayor a 0.</div>
  }

  <button type="submit" [disabled]="miFormulario.invalid">Enviar</button>
</form>
```