FORMS

 FORMS

Handling user input with forms is the cornerstone of many common applications. Applications use forms to enable users to log in, to update a profile, to enter sensitive information, and to perform many other data-entry tasks.

TBC

📖 ANGULAR FORMS

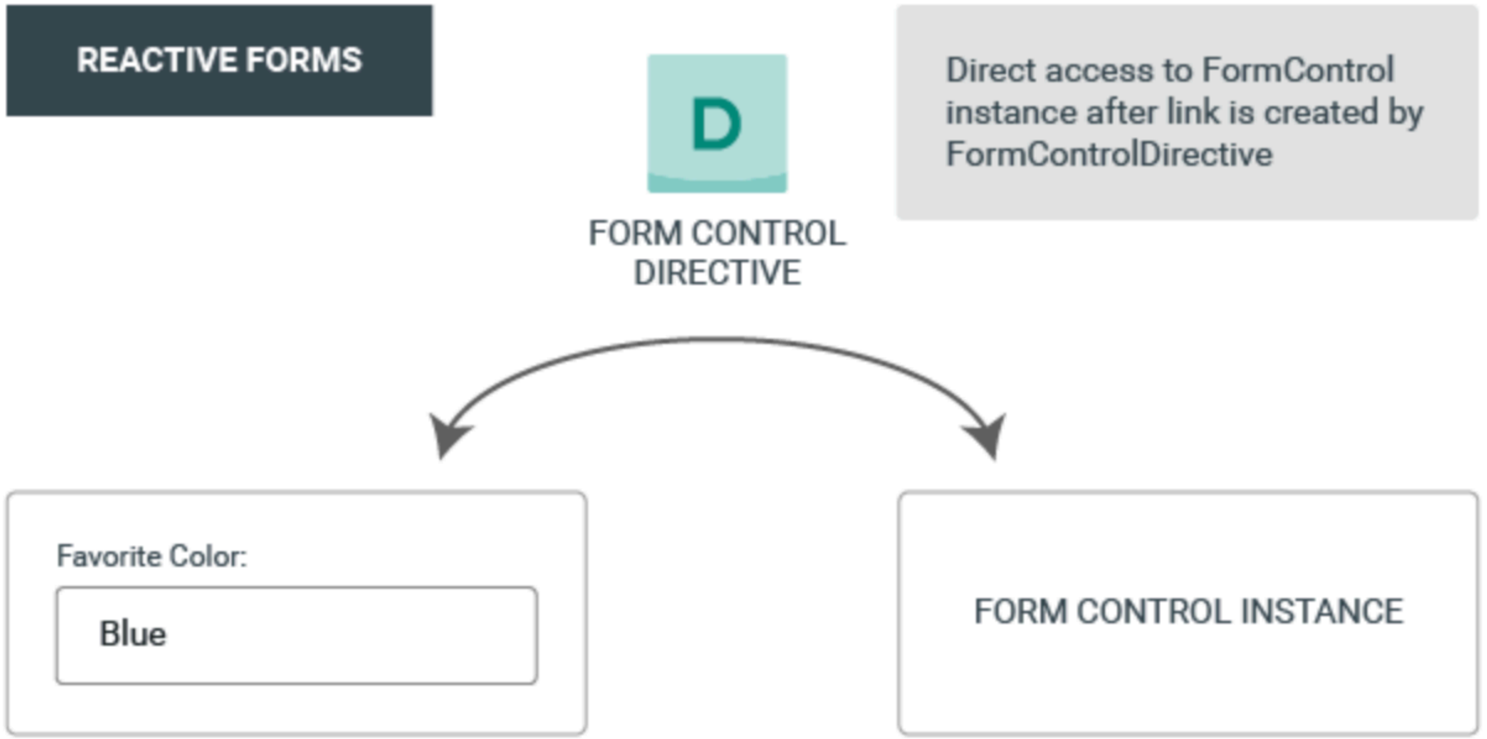
- Reactive forms are more robust: they're more scalable, reusable, and testable. If forms are a key part of your application, or you're already using reactive patterns for building your application, use reactive forms.
- Template-driven forms are useful for adding a simple form to an app, such as an 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.

TBC

📖 FORM BLOCKS

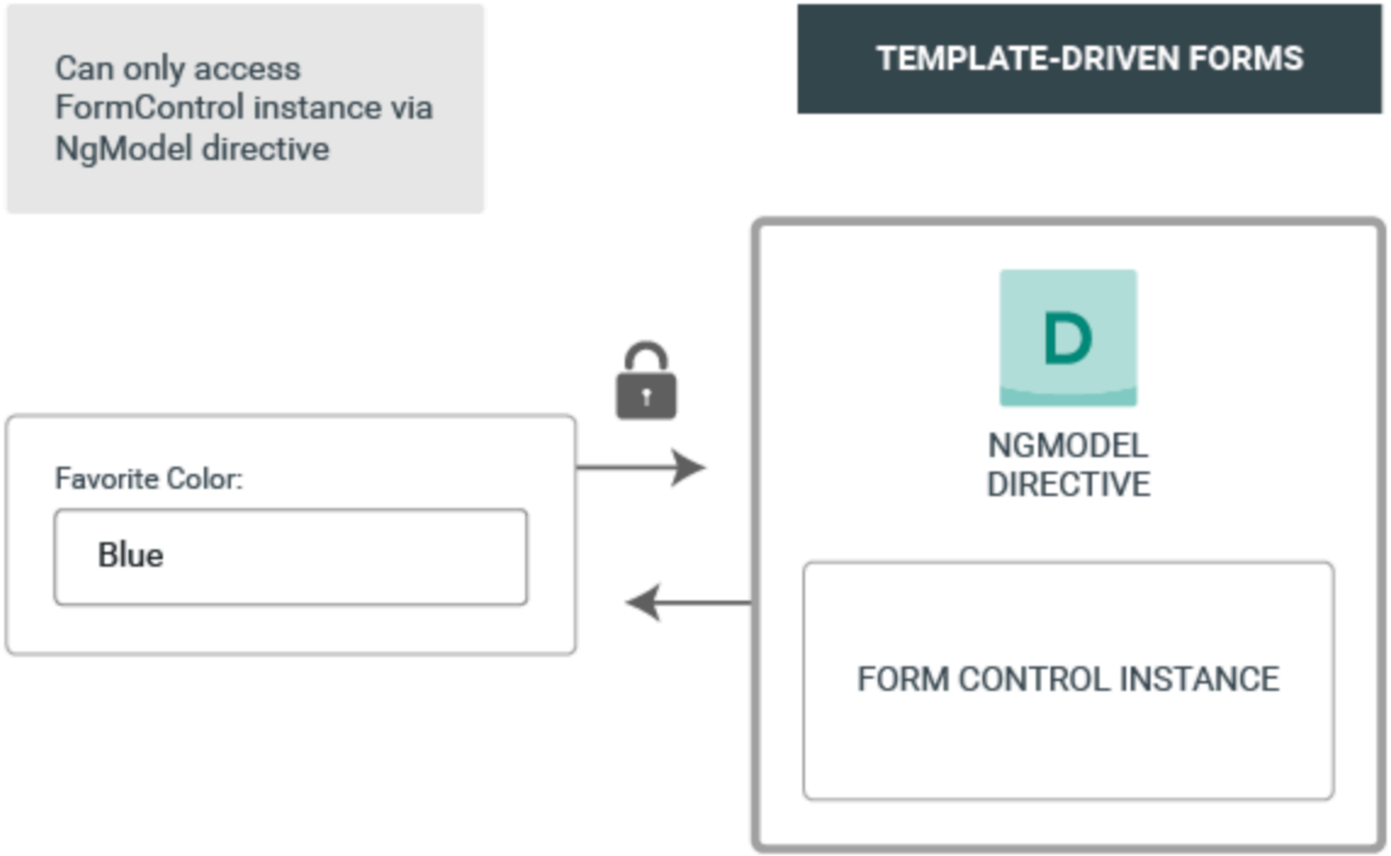
- FormControl
- FormGroup
- FormArray
- ControlValueAccessor

📖 REACTIVE FORMS



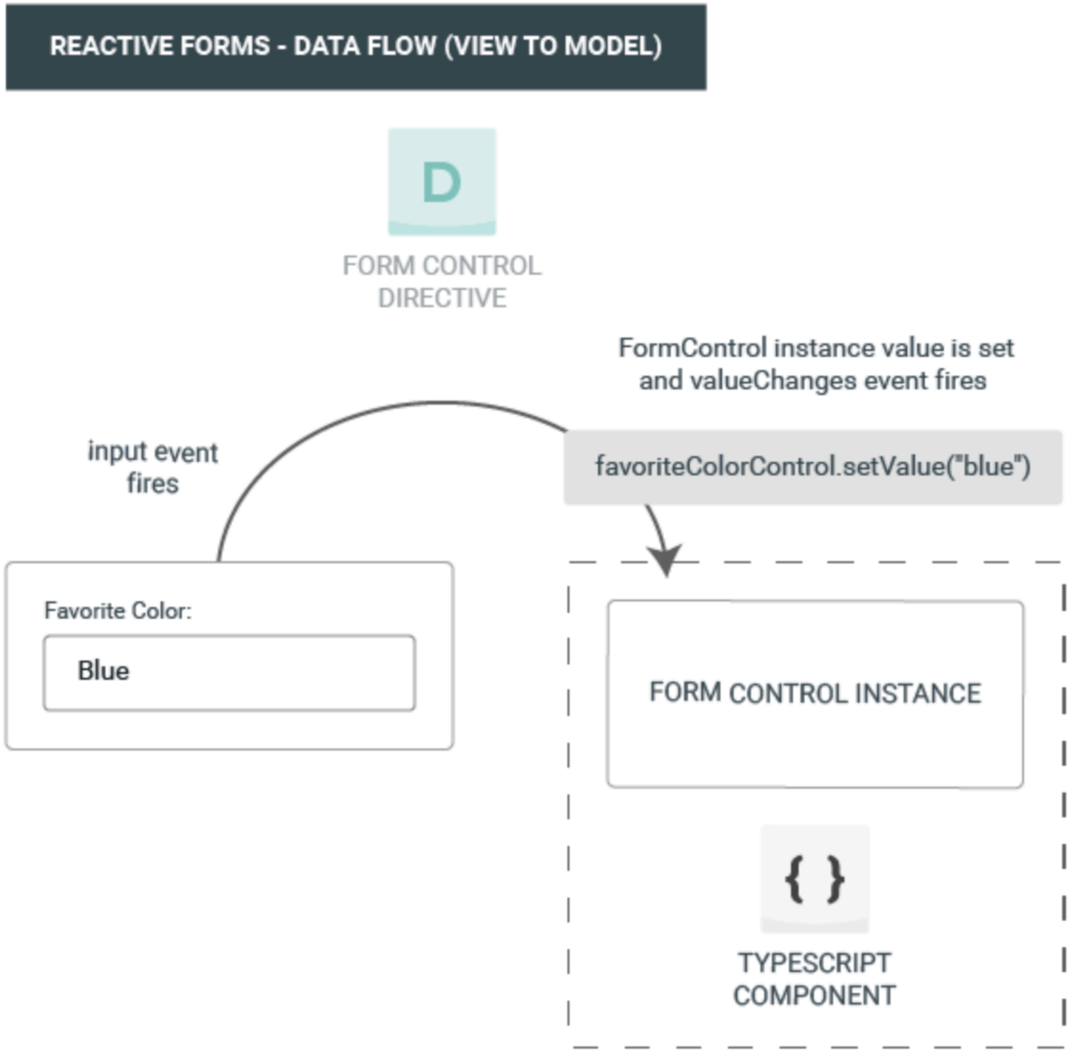
TBC

📖 TEMPLATE-DRIVEN FORMS



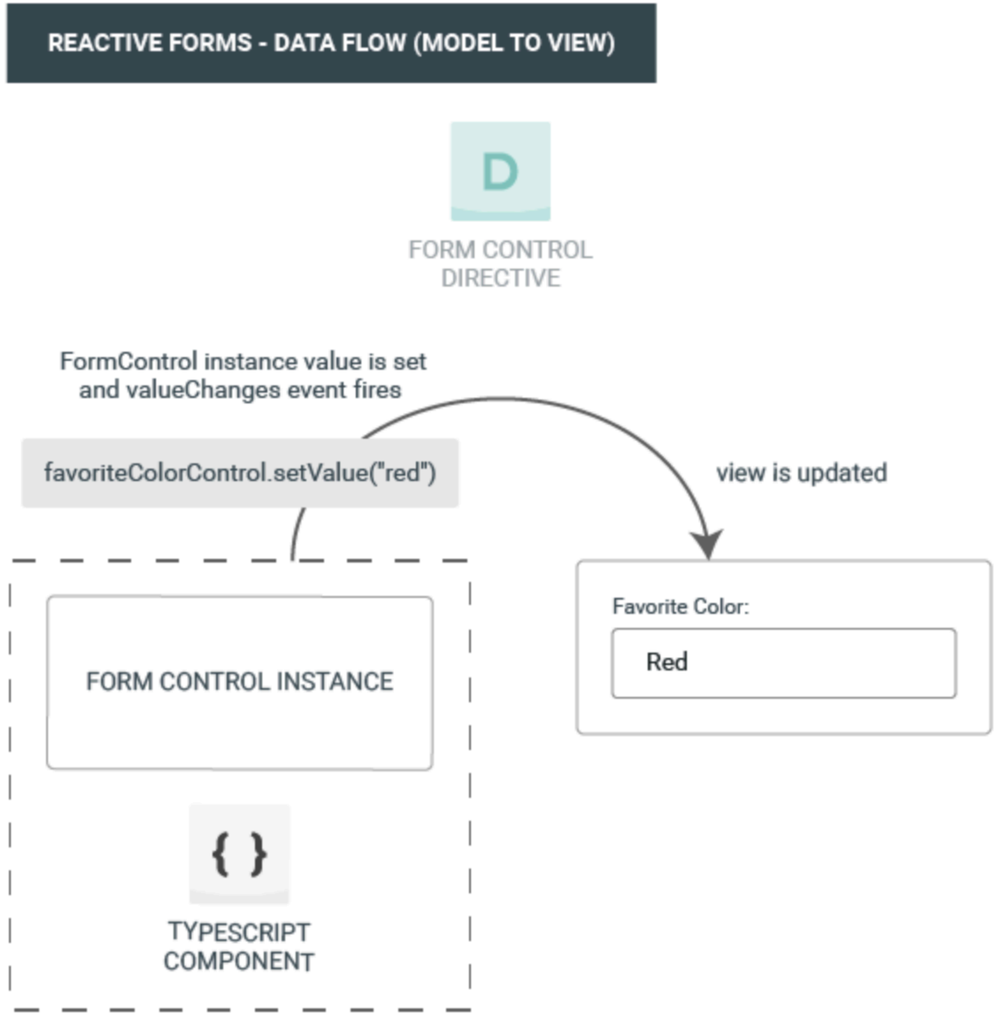
TBC

📖 DATA FLOW



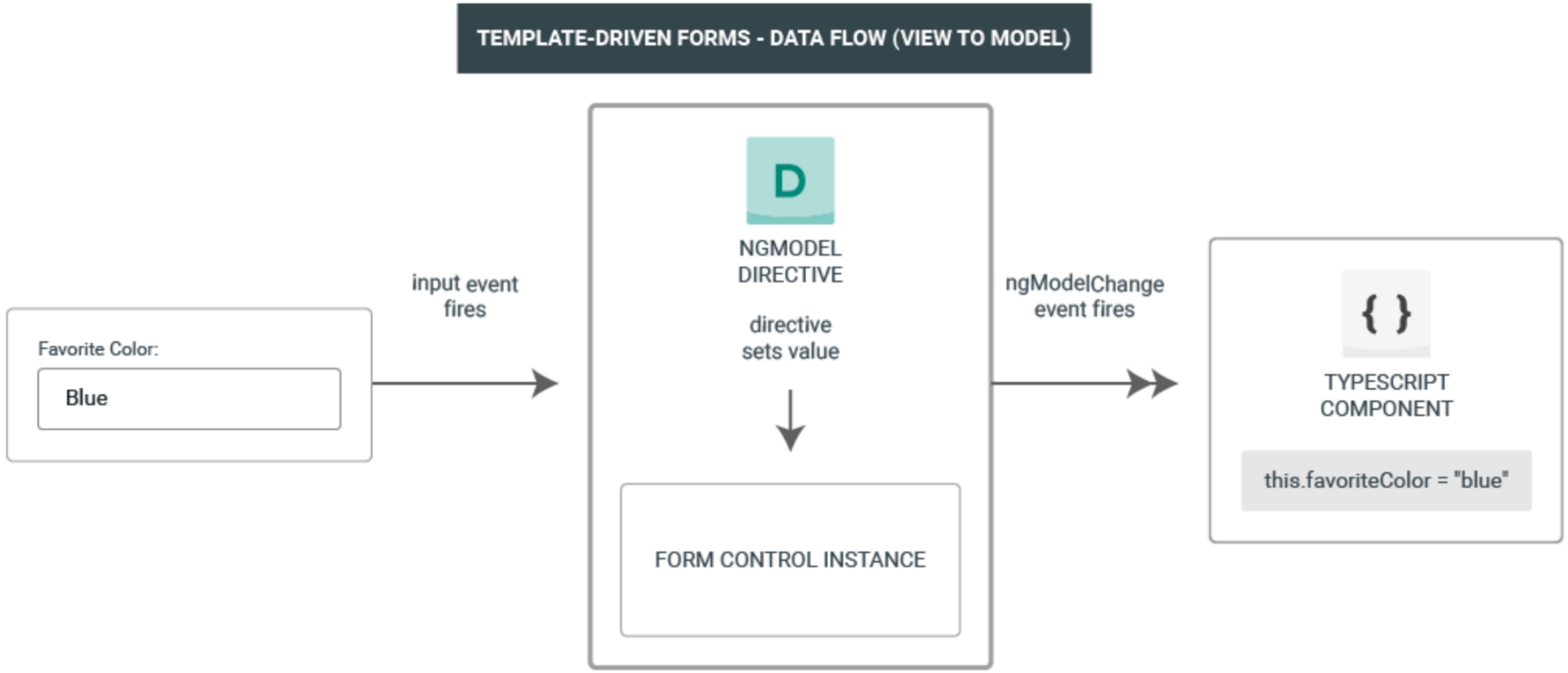
TBC

📖 DATA FLOW



TBC

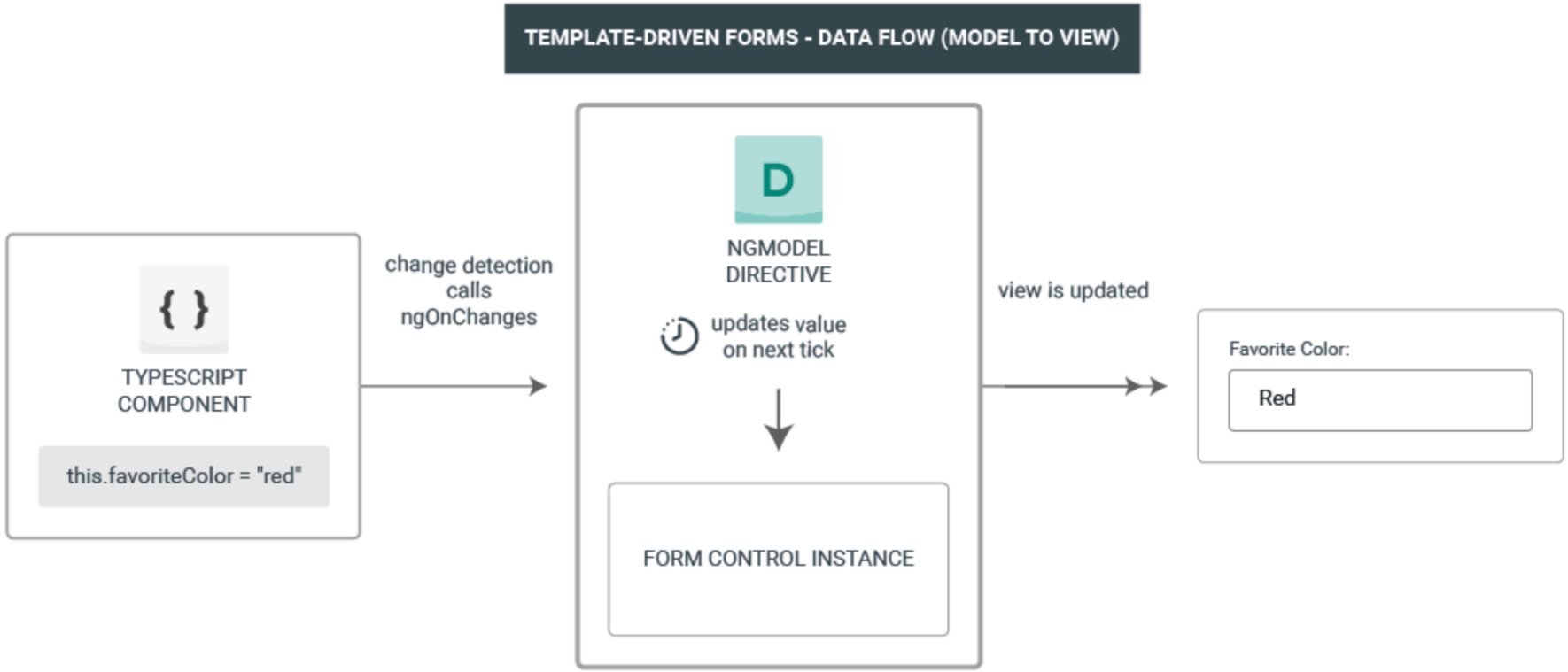
📖 DATA FLOW



START		DIRECTIVE SETS VALUE		END RESULT	
this.favoriteColor (ngModel)	RED	this.favoriteColor (ngModel)	RED	this.favoriteColor (ngModel)	BLUE
FormControl instance value	RED	FormControl instance value	BLUE	FormControl instance value	BLUE
view	BLUE	view	BLUE	view	BLUE

TBC

📖 DATA FLOW



START	
this.favoriteColor (ngModel)	RED
FormControl instance value	BLUE
view	BLUE

DIRECTIVE UPDATES VALUE	
this.favoriteColor (ngModel)	RED
FormControl instance value	RED
view	BLUE

END RESULT	
this.favoriteColor (ngModel)	RED
FormControl instance value	RED
view	RED

TBC

📖 MUTABILITY

- Reactive forms keep the data model pure by providing it as an immutable data structure. Each time a change is triggered on the data model, the FormControl instance returns a new data model rather than updating the existing data model.
- Template-driven forms rely on mutability with two-way data binding to update the data model in the component as changes are made in the template.

TBC

📖 MUTABILITY

- With reactive forms, the FormControl instance always returns a new value when the control's value is updated.
- With template-driven forms, the favorite color property is always modified to its new value.

📖 TEMPLATE DRIVEN VALIDATION

To add validation to a template-driven form, you add the same validation attributes as you would with native HTML form validation. Angular uses directives to match these attributes with validator functions in the framework.

📖 REACTIVE VALIDATION

In a reactive form, the source of truth is the component class. Instead of adding validators through attributes in the template, you add validator functions directly to the form control model in the component class. Angular then calls these functions whenever the value of the control changes.

TBC

HERE GOES!

TBC