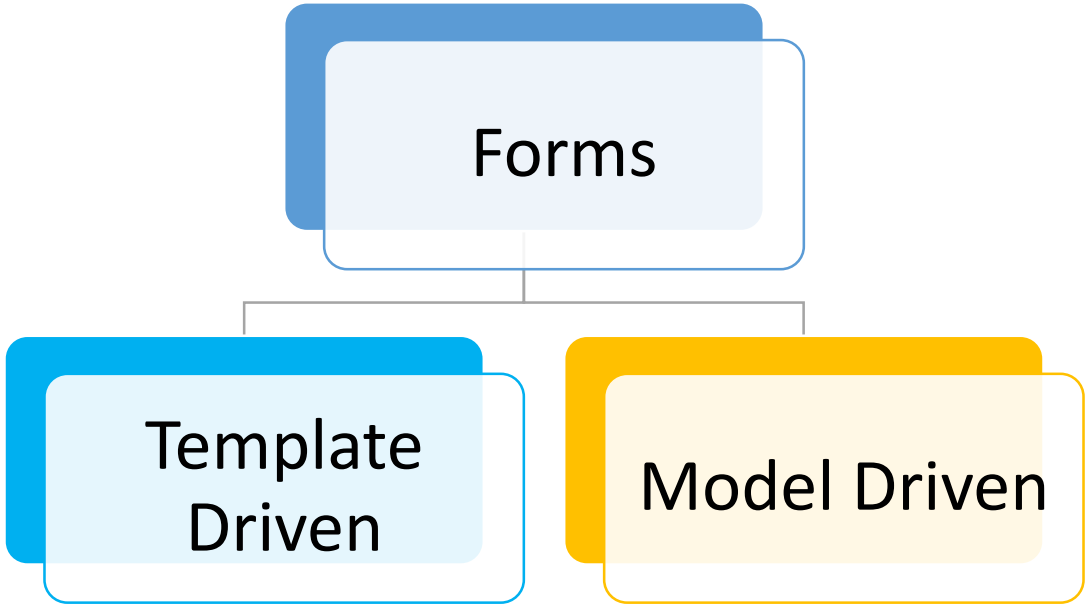
Angular Forms and Validations

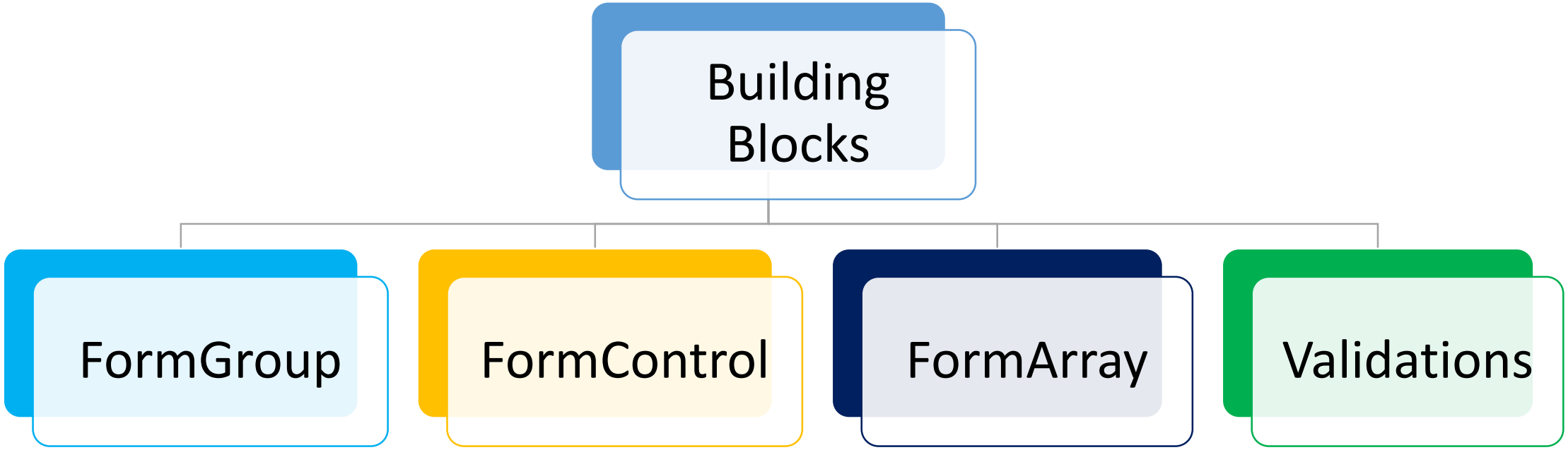
DotNetTricks

Angular Forms

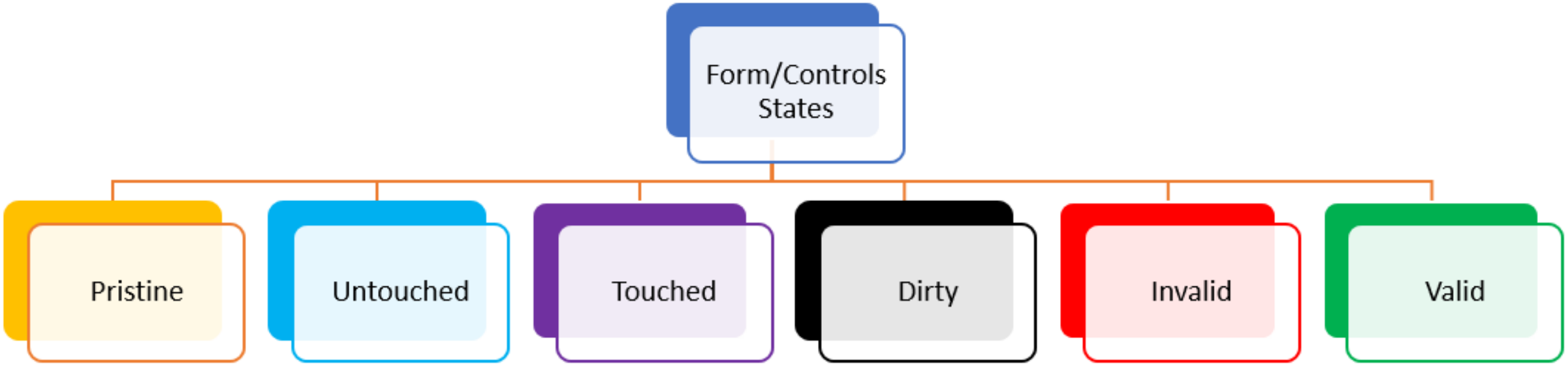
- HTML forms are an essential part of a web application
- Angular provides two ways to create form – Template Driven and Model Driven



Angular Form Building Blocks



Angular Form and Form Controls States

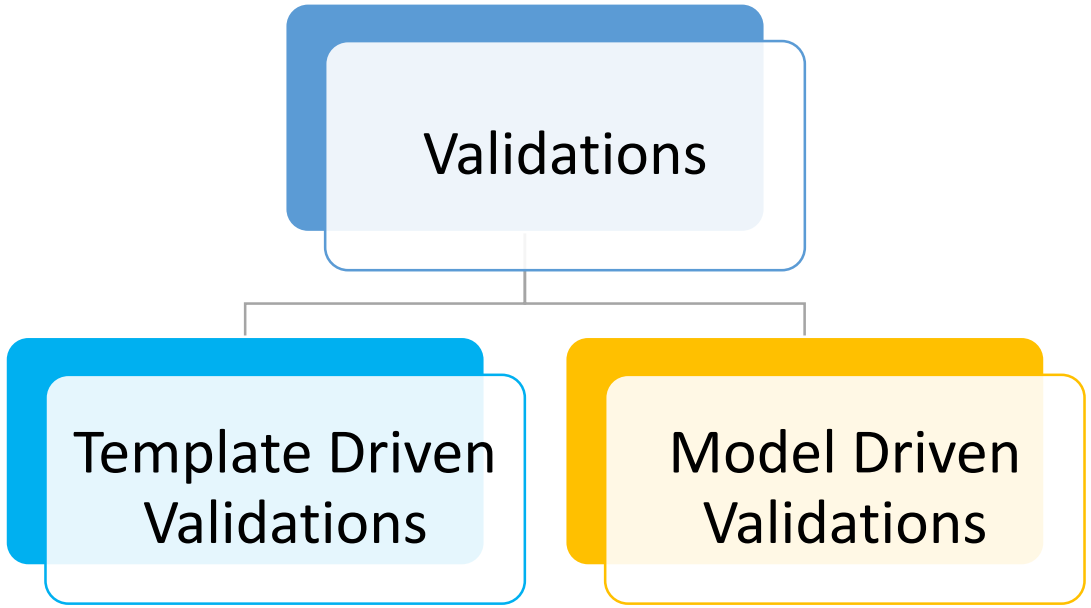


States Transition

- untouched => touched
- invalid => valid
- pristine => dirty

Angular Form Validations

- Validate user input for accuracy and completeness
- Performs on client side

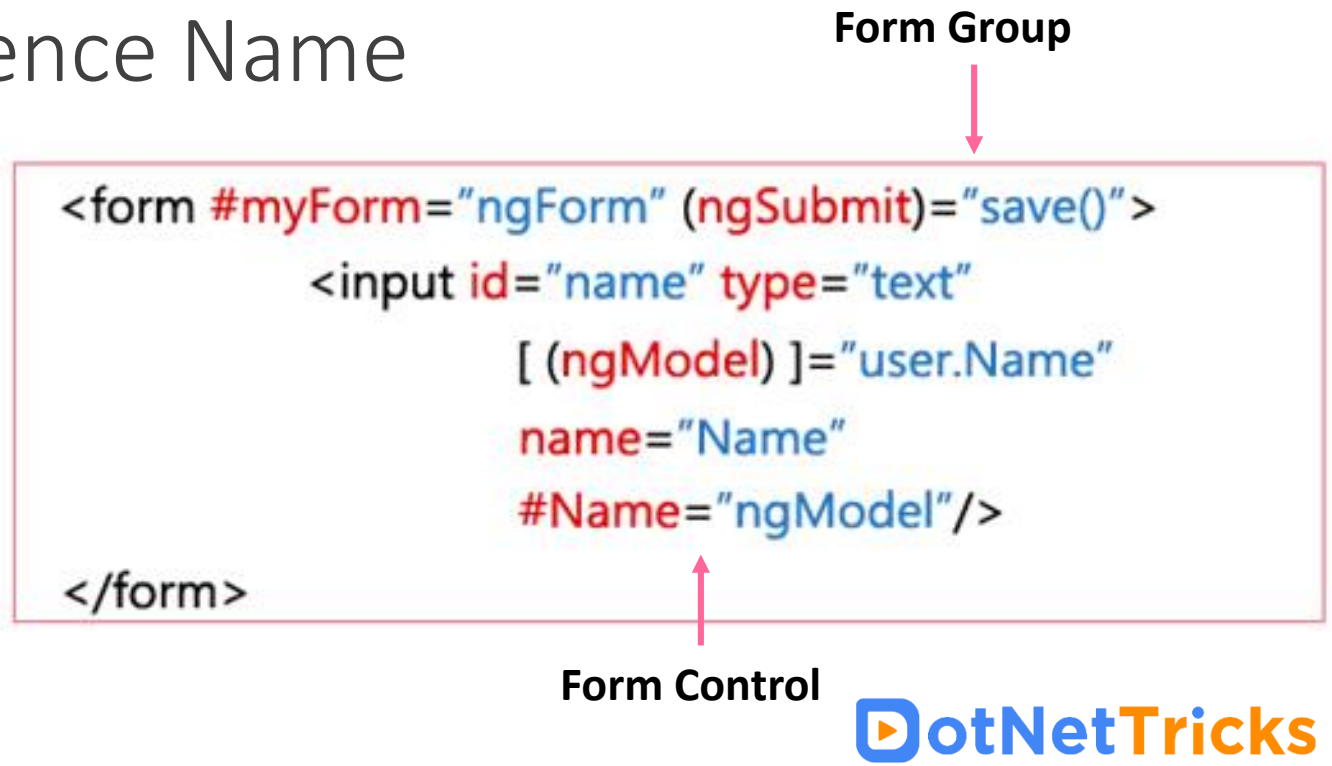


Angular Forms Built-In Validation

- required
- minlength
- maxlength
- pattern
- email – supports in Ng4
- min
- max

Template Driven Form

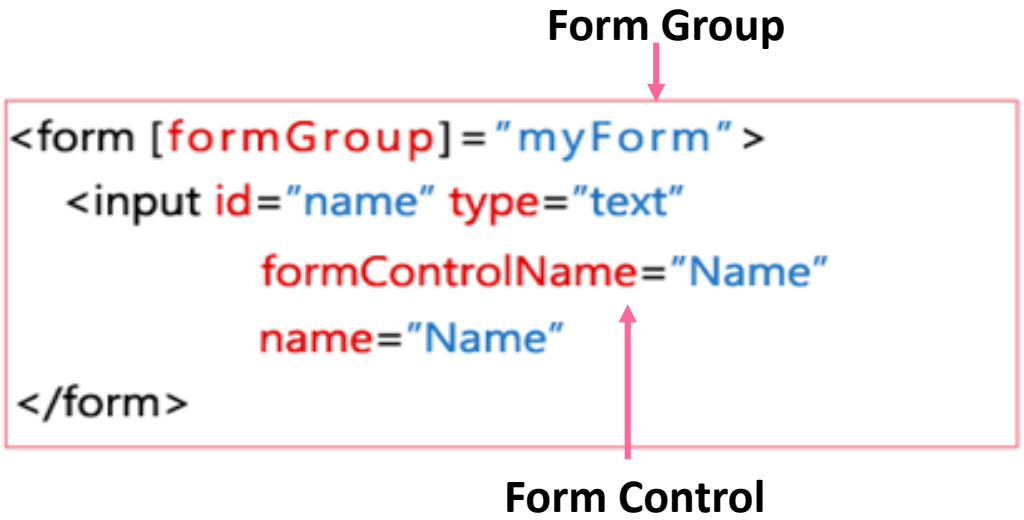
- Form is setup and configured in HTML Code
- Easy and suits for simple form
- Use Directives and Reference Name
  - ngForm
  - ngModel
  - #refName



DotNetTricks

Model Driven (Reactive) Form

- Form is setup and configured in component class
- Based on *reactive style* of programming where you use the underlying APIs FormControl and FormGroup that track the value and validation status
- Reactive forms offer the ease of testing, and validation



DotNetTricks

Template Driven vs. Model Driven Form

- Form is setup and configured in HTML Code
- Easy and suits for simple form
- Two-way data binding
- Automatically track form and input element state
- Form is passed to component class via *ngSubmit*()
- Unit Testing is complex

- Form is setup and configured in component class using FormBuilder
- Flexible and suits for complex form
- No data binding(immutable data model)
- Reactive Transformation that can react to changes in data across the app
- Form can be accessed in component class without passing it via ngSubmit()
- Unit Testing is easy