
# Angular

## Introduction to Angular framework for SPAs

Đorđe Kalanj
Miljan Veljović

MATF, Beograd, 19.12.2019.

# AGENDA SLIDE

01 What is Angular?

02 Components, Directives & Pipes

03 Services and Modules

04 Routing and Navigation

05 Template Driven Forms & Reactive Forms

# 01

# WHAT IS ANGULAR?

Angular in a nutshell.

# What is Angular?

- Angular is JavaScript framework for building user interfaces as a SPA (Single Page Application).
- Available at npmjs.com/package/@angular/core
- On of the most popular JavaScript frameworks (along with React).
- Angular uses TypeScript and Observables.

# AngularJS vs Angular

- AngularJS is JavaScript framework for developing SPA web applications, released in 2010.
- Angular is a completely rewrite version of AngularJS framework, from version 2.0.0 until now (currently, 8.2.14 version is the latest one).
- Both frameworks are MVC frameworks.
- We will cover Angular in this course



# 02

# COMPONENTS, DIRECTIVES & PIPES

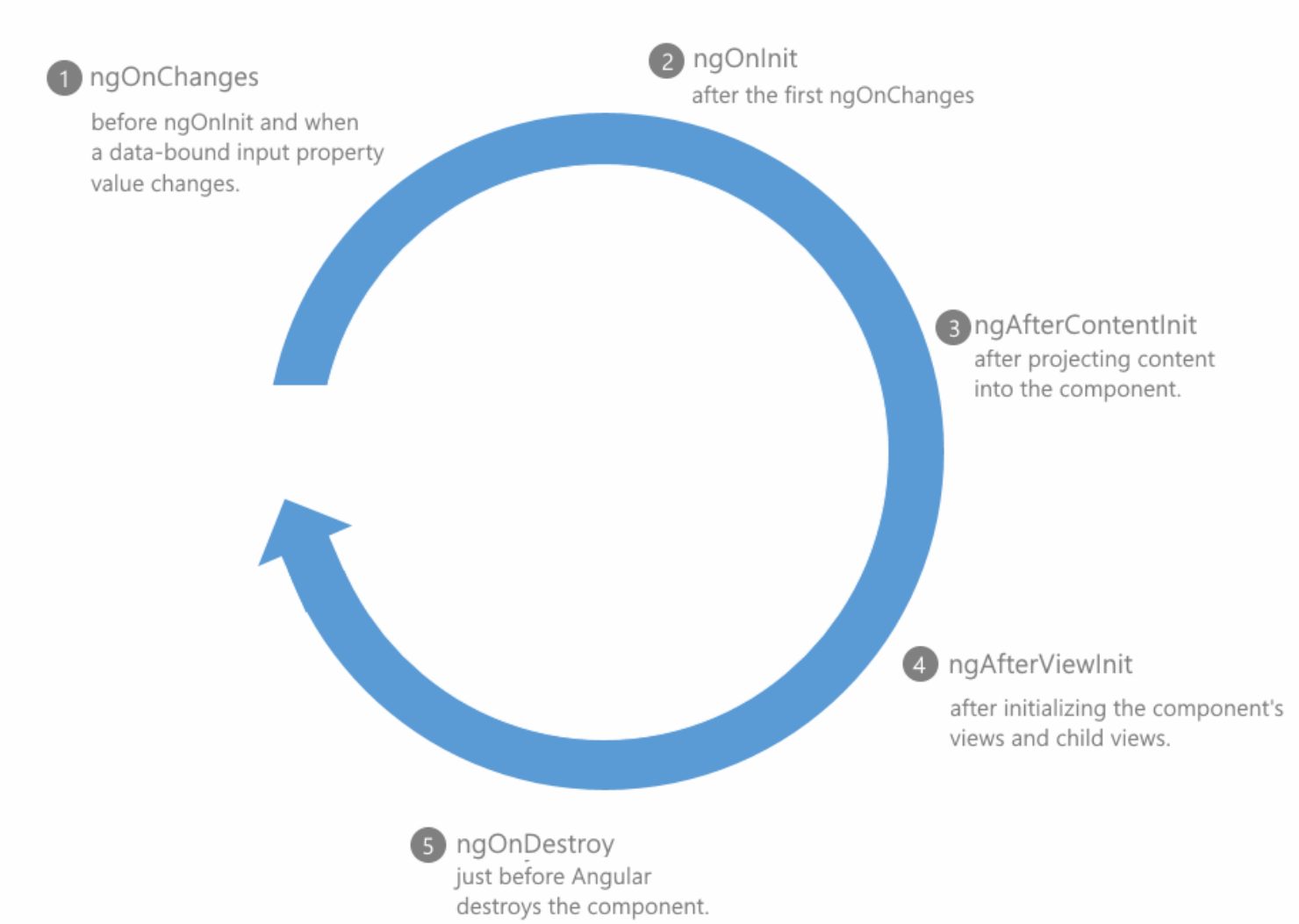
Core of the Angular.

# Components

- Building blocks of Angular applications.
- Made of HTML, TypeScript and CSS (or Sass/Less).
- Definition of component is placed in TypeScript file **.component.ts**, where HTML and CSS files are linked as well.
- Components have lifecycle hooks methods such as **ngOnInit**, **ngOnChanges**, **ngOnDestroy**, etc.



# Component lifecycle



# Data binding

- Way to pass the data from parent to child component.
- Component's input is just a regular class field that has **@Input** decorator.
- Parent component passes the data to child component by adding **[childInputName]="parentComponentData"** as an attribute to the child component's selector.
- Child component is re-rendered every time some input changes.

# Event binding

- Child component can notify parent component by dispatching an event to the parent.
- Commonly, it's called **output** and it's just a regular class field with **@Output** decorator.
- Class field that is component's output should be instance of **EventEmitter,** special class in Angular that provides functionality for sending the data.
- Parent component listens to child component's output by passing **(childOutputName)="parentComponentMethod($event)"**



# Two-way binding

- Data binding and event binding can be combined into one single biding called two-way binding.
- For example, HTML input element's change event should change component's field, and component's field value should be passed to HTML input element.
- Two-way binding synax is **[(ngModel)]="propertyName"** (Banana in a Box)



# Directives

- Angular building blocks that change the DOM.
- There are three types of directives:
- **Components** - directives with a template.
- **Structural directives** - change the DOM layout by adding and removing DOM elements.
- **Attribute directives** - change the appearance or behavior of an element, component, or another directive.



# Structural Directives

- Directives that add or remove elements from the DOM.
- Built in structural directives: **ngIf, ngFor, ngSwitch**, etc.
- Angular provides a way to define your own structural directives.



# Attribute Directives

- Directives that modify existing DOM elements by changing their appearance
- Built in structural directives: **ngStyle, ngClass,** etc.
- Angular provides a way to define your own attribute directives.



# Pipes

- Utility for modifying templates that uses some rule defined by developer.
- For example, pipe for transforming the timestamp into the human-readable string is the common one.

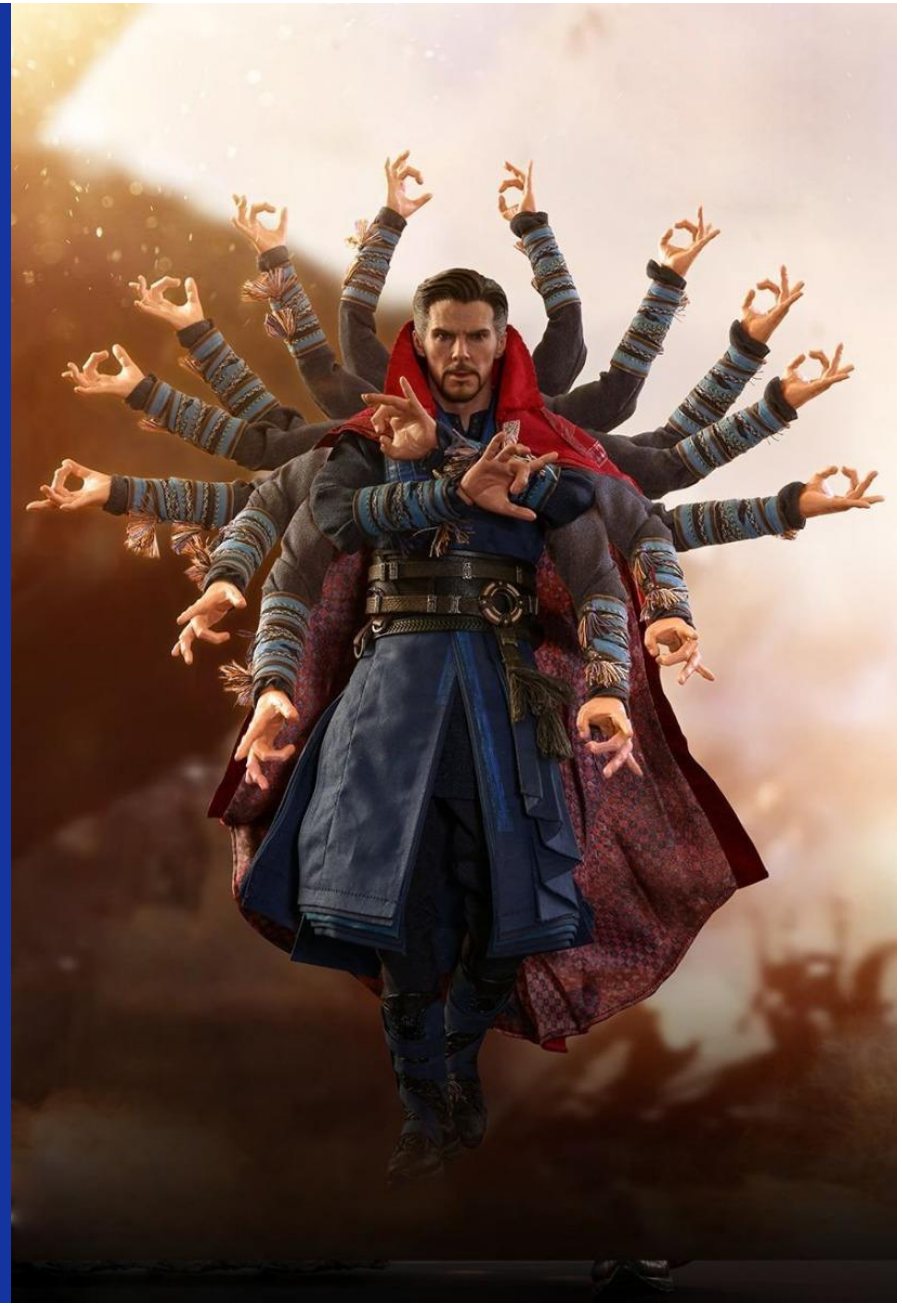
```
import { Pipe, PipeTransform } from '@angular/core';
import { DatePipe } from '@angular/common';

@Pipe({ name: 'stringify-date' })
export class StringifyDate implements PipeTransform {
  transform(timestamp: number): string {
    return new Date(timestamp).toLocaleString();
  }
}
```



# 03

# SERVICES AND MODULES

Non-UI part of Angular.



# Services

- Services are part of Angular application which contain business logic.
- Services don't have templates and can only communicate with other services.
- Services are used by components via DI (Dependency Injection).
- There's a special service called **HttpClient** for making HTTP calls.

```
import { Injectable } from '@angular/core';

@Injectable({ providedIn: 'root' })
export class HeroService {
  heros = [{ name: 'Doctor Who' }, { name: 'Sherlock' }];

  constructor() {}

  getHeros() {
    return this.heros;
  }
}
```

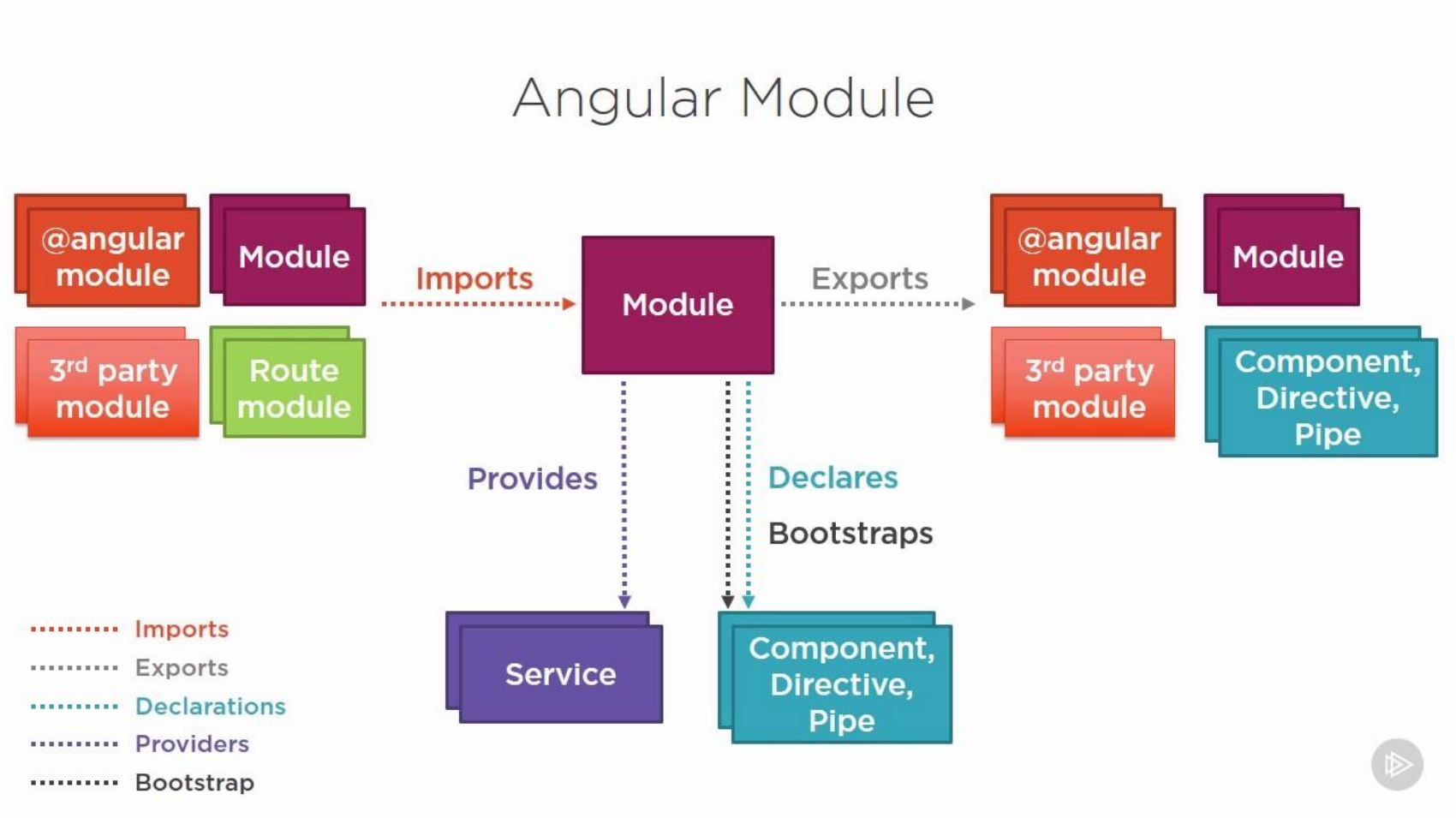


# Modules

- Angular applications are usually made of multiple modules that represent different independent parts of the application.
- Usually, the main module is called **AppModule**, and other modules are called Feature Modules.
- Every Component, Directive, Service, Pipe, etc. **must** be declared in some module in application.
- If declared in Feature Module, it is available **only** in that module.
- If declared in AppModule, it is available in **entire** application.
- Special kind of Module that defines routing rules is called **Routing Module**.



# Module diagram



# 04

# ROUTING AND NAVIGATION

SPA does not mean only one URL.

# Routing and Navigation

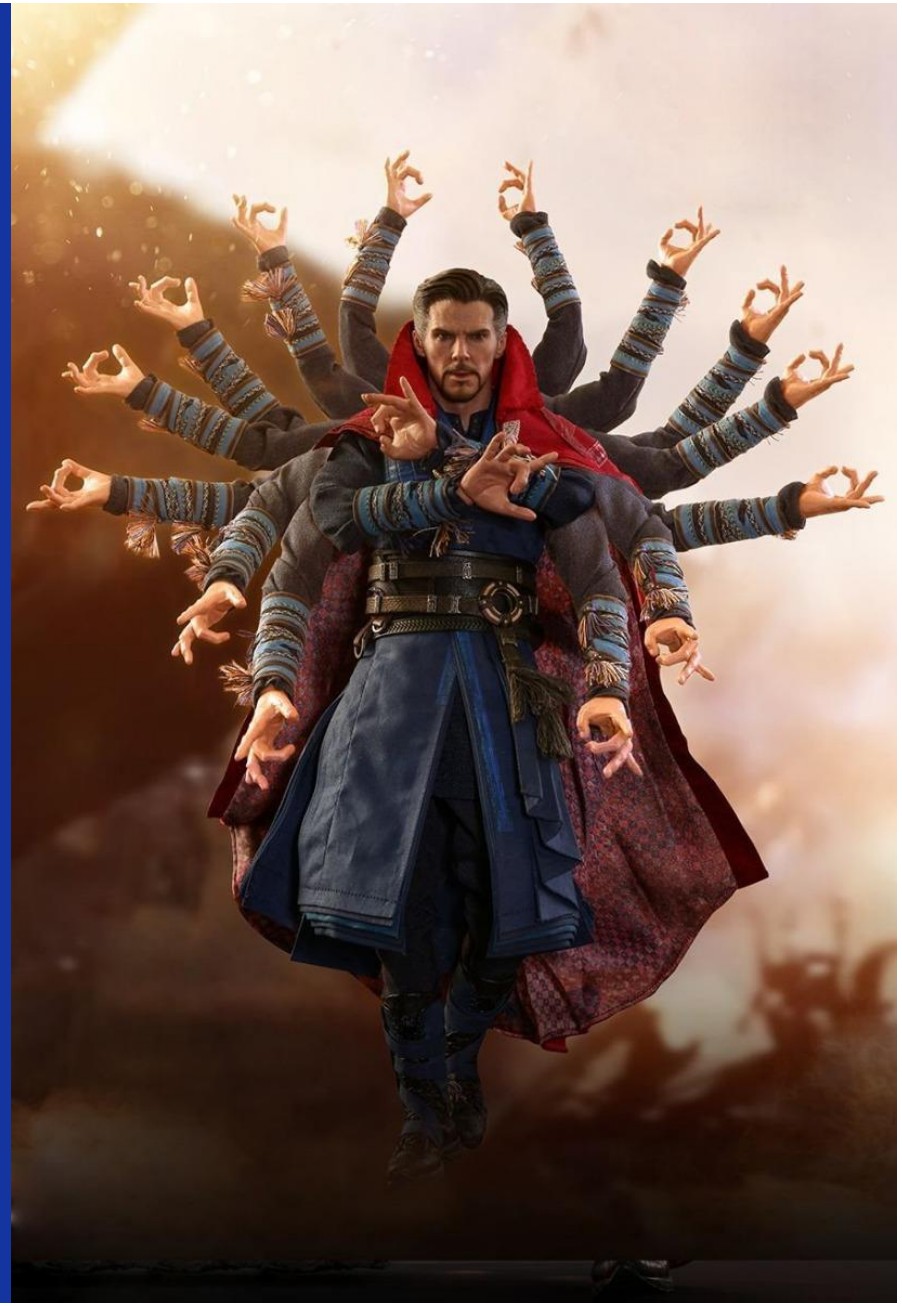
- For routing, special service called **Router** is used.
- It provides methods for programmatically navigating user within the application.
- Routes are defined in ***-routing.module**.
- Angular provides directives for navigating through the application which can be used within HTML, so there's no need for using Router Service.



# 05

# TEMPLATE DRIVEN & REACTIVE FORMS

If you'll ever need user interaction.



# Template Driven Forms

- Uses two-way binding to map component's fields to the HTML form and change component's fields values accoring to HTML form changes.
- Form structure is defined in the component's template.
- Suitable for small forms.
- Not so powerful.
- Validation must be implemented on your own.



# Reactive Forms

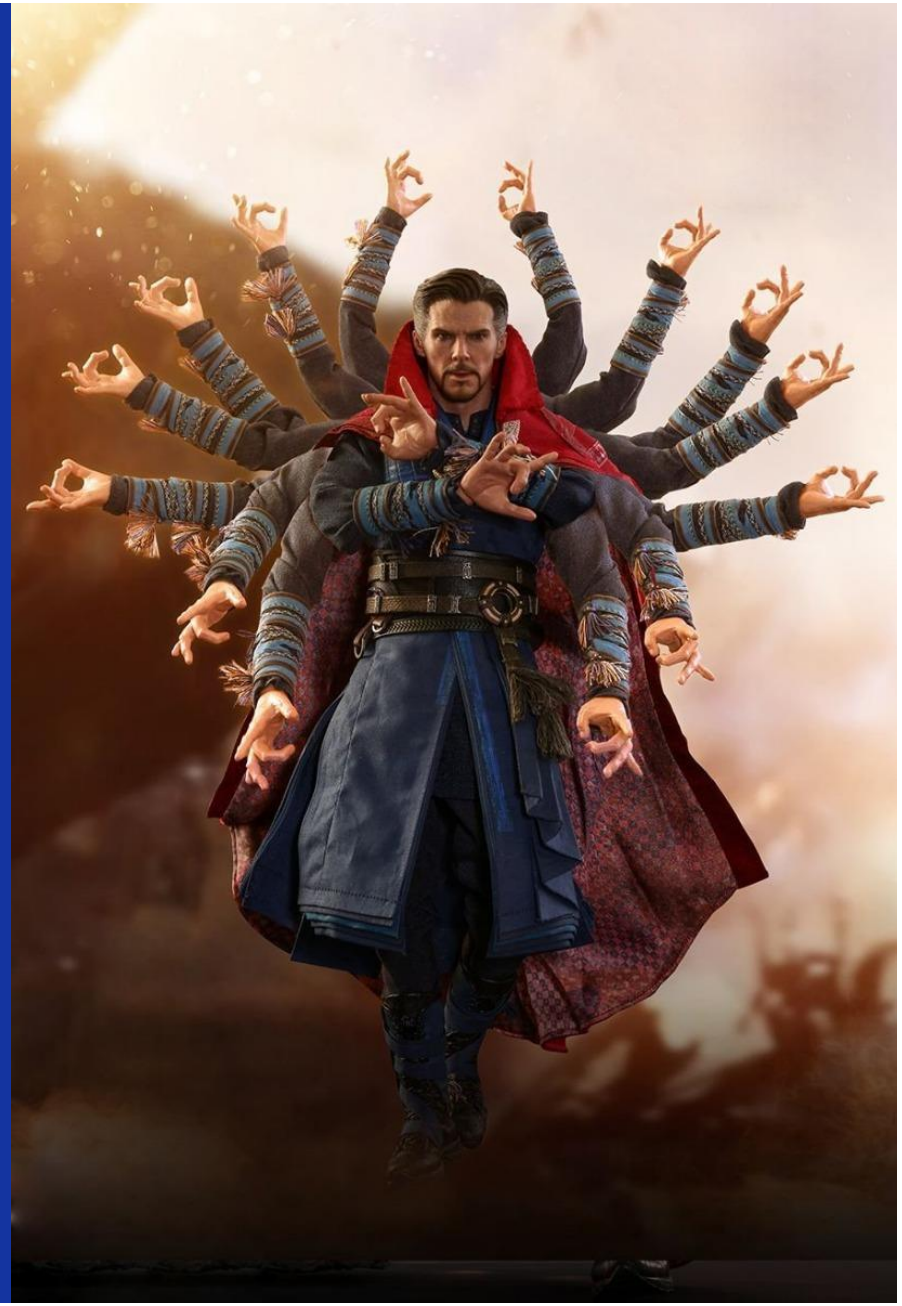
- Form structure is defined in TypeScript file of the component.
- HTML form is connected to TypeScript via [**formGroup**] and [**formControlName**] inputs.
- More flexible than Template Driven Forms.
- More powerful.
- Provides a built-in functionality for validation.



# 06

# REDUX

Manage state like a pro.



# QUESTIONS ?

levi nine
Technology Services

# THANK YOU

levi nine

Technology Services