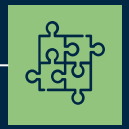
About Angular

2020.5.

TABLE OF CONTENTS

01

History



02

Angular concept

03

Fundamentals

04

Pros & cons,
Angular 9 Features

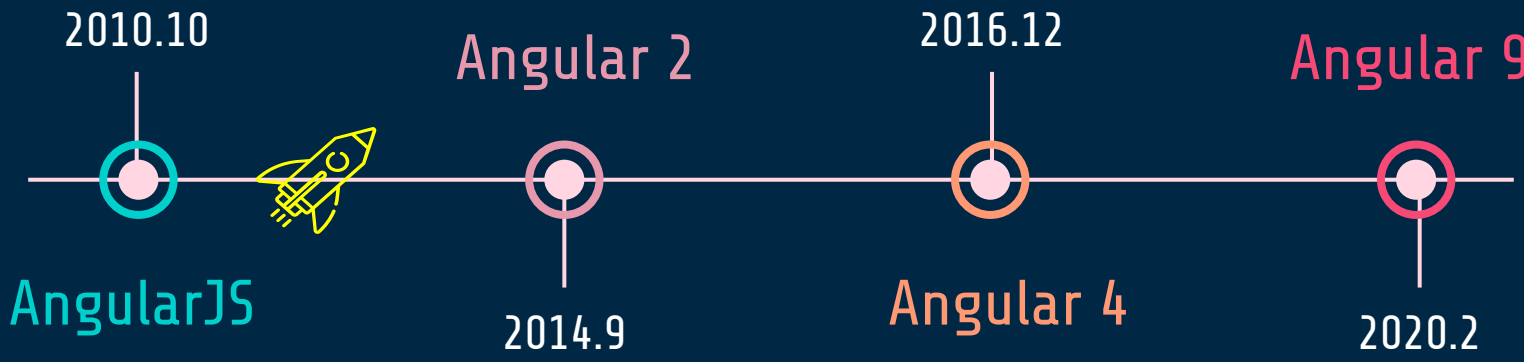
History

When, How, Why...

01

History

Why? To provide framework for client-side MVC, MVVM architecture along with components commonly used in rich internet applications



AngularJS vs Angular

AngularJS

JavaScript

Scope

Old Template engine

No CLI

...

Angular

TypeScript

Component Hierarchy

New Template engine

Angular CLI

...

Angular Concept

Modules, Components,
Templates, Services, DI...

02

1) Modules

Angular has its own modularity system called NgModules.

They can contain components, service providers, and other code files whose scope is defined by the containing NgModule.

- Definition

```
import { NgModule }      from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
@NgModule({
  imports:      [ BrowserModule ],
  providers:    [ Logger ],
  declarations: [ AppComponent ],
  exports:      [ AppComponent ],
  bootstrap:    [ AppComponent ]
})
export class AppModule { }
```

2) Components

A component controls a patch of screen called a view.

It's the logic of the application - interacts with the view through an API of properties and methods.

- Definition (metadata, class)

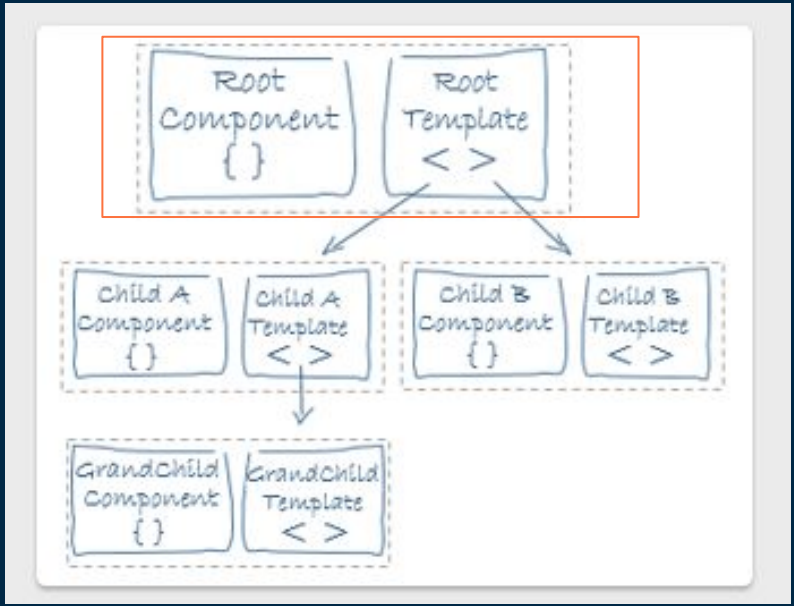
```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

3) Templates and Views

A template is a form of HTML that tells Angular how to render the component.

Views are typically arranged hierarchically, allowing you to modify or show and hide entire UI sections or pages as a unit



– Template syntax (data binding, pipes, directives)

A template looks like regular HTML, except that it also contains Angular template syntax, which alters the HTML based on your app's logic and the state of app and DOM data.

```
<h2>Hero List</h2>

<p><i>Pick a hero from the list</i></p>
<ul>
  <li *ngFor="let hero of heroes" (click)="selectHero(hero)">
    {{hero.name}}
  </li>
</ul>

<app-hero-detail *ngIf="selectedHero" [hero]="selectedHero"></app-hero-detail>
```

Data binding

Angular supports two-way data binding, a mechanism for coordinating the parts of a template with the parts of a component

```
<li>{{hero.name}}</li>
<app-hero-detail [hero]="selectedHero"></app-hero-detail>
<li (click)="selectHero(hero)"></li>
```

```
<input [(ngModel)]="hero.name">
```

- Interpolation {{}}: displays component's value
- Property binding: passes the value from the parent to child component
- Event binding: calls the event handler once the event occurs
- Two-way binding (used mainly in template-driven forms) : [()]

Pipes

Angular pipes let you declare display-value transformations in your template HTML.

{{interpolated_value | pipe_name}}

```
<!-- Default format: output 'Jun 15, 2015'-->
 <p>Today is {{today | date}}</p>

<!-- fullDate format: output 'Monday, June 15, 2015'-->
<p>The date is {{today | date:'fullDate'}}</p>

 <!-- shortTime format: output '9:43 AM'-->
 <p>The time is {{today | date:'shortTime'}}</p>
```

Directives

Angular pipes let you declare display-value transformations in your template HTML.

Types: structural and attribute

```
<li *ngFor="let hero of heroes"></li>
<app-hero-detail *ngIf="selectedHero"></app-hero-detail>
```

```
<input [(ngModel)]="hero.name">
```

4) Services

Service is a broad category encompassing any value, function, or feature that an app needs. A service is typically a class with a narrow, well-defined purpose. It should do something specific and do it well.

Definition and Usage

```
export class Logger {
  log(msg: any)   { console.log(msg); }
  error(msg: any) { console.error(msg); }
  warn(msg: any)  { console.warn(msg); }
}
```

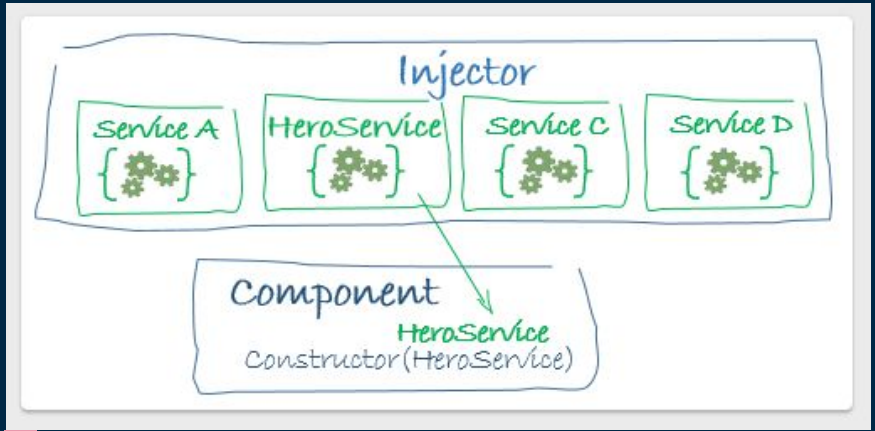
```
export class HeroService {
  private heroes: Hero[] = [];

  constructor(
    private backend: BackendService,
    private logger: Logger) { }

  getHeroes() {
    this.backend.getAll(Hero).then( (heroes: Hero[]) => {
      this.logger.log(`Fetched ${heroes.length} heroes.`);
      this.heroes.push(...heroes); // fill cache
    });
    return this.heroes;
  }
}
```

5) Dependency injection (DI)

Components consume services; that is, you can inject a service into a component, giving the component access to that service class.

```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
```

```
constructor(private service: HeroService) { }
```



Fundamentals

Forms, Observables & RxJS, ...

03

1) Forms

Angular provides two different approaches to handling user input through forms: reactive and template-driven

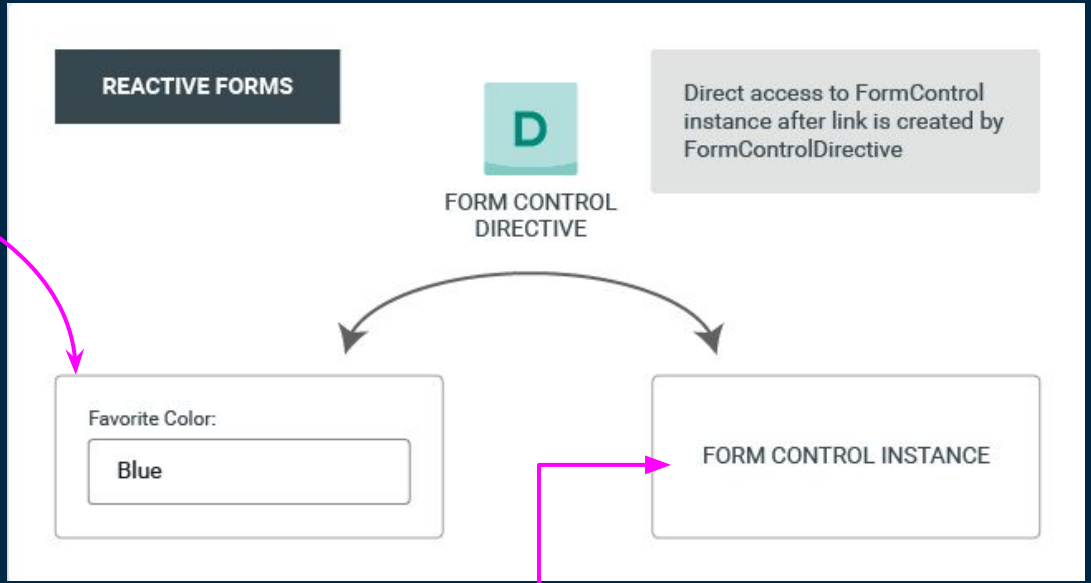
- Differences between reactive form and template-driven form

	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

- Reactive form: model driven

```
<label>
  Name:
  <input type="text" [formControl]="name">
</label>
```

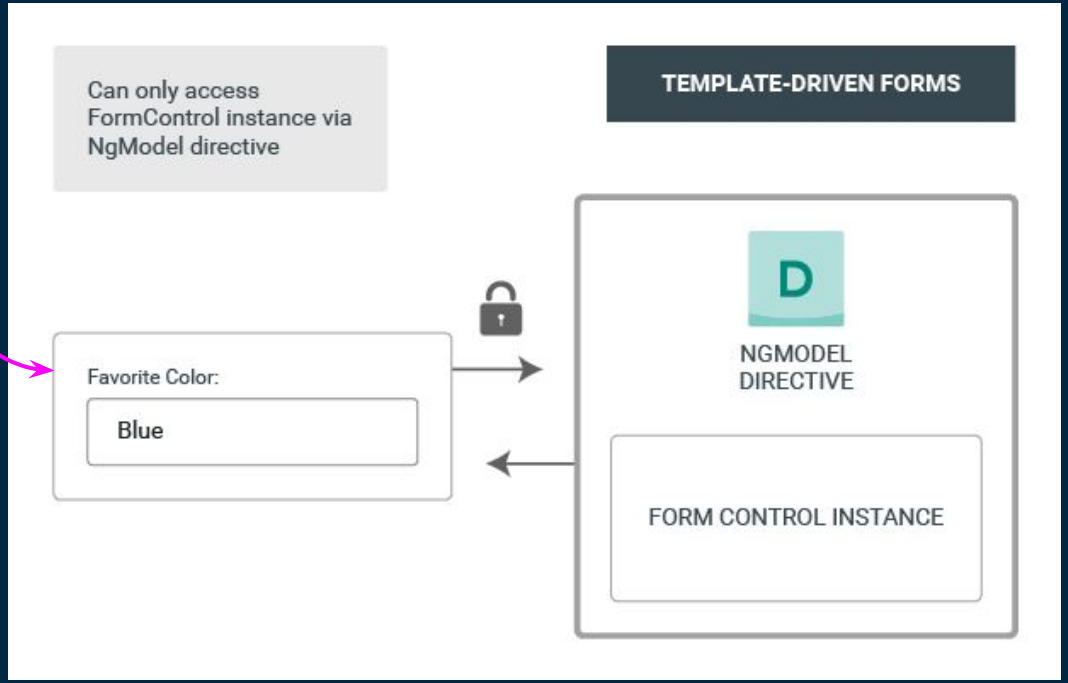
```
import { Component } from '@angular/core';
import { FormControl } from '@angular/forms';

@Component({
  selector: 'app-name-editor',
  templateUrl: './name-editor.component.html',
  styleUrls: ['./name-editor.component.css']
})
export class NameEditorComponent {
  name = new FormControl('');
}
```



- Template-driven form: model driven

```
<label>
  Name:
  <input type="text" [formControl]="name">
</label>
```

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

@Component({
  selector: 'app-name-editor',
  templateUrl: './name-editor.component.html',
  styleUrls: ['./name-editor.component.css']
})
export class NameEditorComponent {

}
```



2) RxJS Library

RxJS (Reactive Extensions for JavaScript) is a library for reactive programming using observables that makes it easier to compose asynchronous or callback-based code.

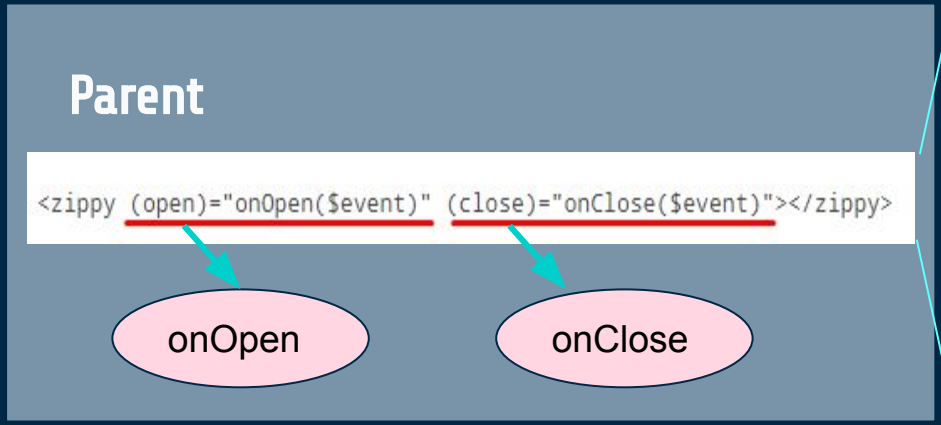
Observables provide support for passing messages between parts of your application.

- From a promise (from)
- From a counter (interval)
- From an event (fromEvent)
- From an AJAX request (ajax)
- ...

```
import { from } from 'rxjs';

// Create an Observable out of a promise
const data = from(fetch('/api/endpoint'));
// Subscribe to begin listening for async result
data.subscribe({
  next(response) { console.log(response); },
  error(err) { console.error('Error: ' + err); },
  complete() { console.log('Completed'); }
});
```

Observable in Angular

Observable is used in Event Emitter.




```
@Component({
  selector: 'zippy',
  template: `
  <div class="zippy">
    <div (click)="toggle()">Toggle</div>
    <div [hidden]="!visible">
      <ng-content></ng-content>
    </div>
  </div>`})

export class ZippyComponent {
  visible = true;
  @Output() open = new EventEmitter<any>();
  @Output() close = new EventEmitter<any>();

  toggle() {
    this.visible = !this.visible;
    if (this.visible) {
      this.open.emit(null);
    } else {
      this.close.emit(null);
    }
  }
}
```

Pros & Cons, When, Why, Angular 9 Features

04

1) Pros & Cons

Pros

- Component-based architecture that provides a higher quality of code

  Reusability, Readability, Unit-test friendly, Maintainability
- TypeScript
- RxJS
- Google Long-Term Support
- Seamless updates using Angular CLI

Cons

- Variety of different structures

  Injectables, components, pipes, modules, ...
- Migrating legacy systems from AngularJS to Angular requires time
- Relatively slower performance
- Compilation speed

2) When Angular

Angular is suitable for:

- Enterprise web apps (thanks to TypeScript)
- Apps with dynamic content

Angular is not suitable for:

- Lightweight websites with static content
- SEO-optimized websites
- Short-term projects
- Apps with a microservice design approach

3) Angular 9 Features compared to prior versions

the latest version of Angular - Angular 9

- Smaller bundle sizes and augmented performance
- AOT(compiles at build time) compilation everywhere

  AOT is better than JIT (compiles at run time on browser)

4) Angular vs React vs Vue

Framework or Library?

- Angular is a framework, while React and Vue are libraries

Flexibility

- React and Vue are more flexible and universal than Angular

Performance

- Vue, React and then Angular

Native App Development

- React Native, Ionic, Weex

Learning Curve

- Vue, React and then Angular

Data Binding

- React: one-way data binding
- Angular: two-way data binding
- Vue: both

4) Angular vs React vs Vue

	Angular	React	Vue
Type	A Framework	Library to build UI	A library
Why Choose	If you want to use TypeScript	If you want to go for "everything-is-JavaScript" approach	Easy JavaScript and HTML
Founders	Powered by Google	Maintained by Facebook	Created by Former Google Employee
Initial Release	September 2016	March 2013	February 2014
Application Types	If you want to develop Native apps, hybrid apps, and web apps	If you want to develop SPA and mobile apps	Advanced SPA and started supporting Native apps
Ideal for	If you want to focus on large-scale, feature-rich applications	Suitable for modern web development and native-rendered apps for iOS and Android	Ideal for web development and single-page applications
Learning Curve	A steep learning curve	A little bit easier than Angular	A small learning curve
Developer-friendly	If you want to use the structure-based framework	If you want to have flexibility in the development environment	If you want to have separation of concerns
Model	Based on MVC (Model-View-Controller) architecture	Based on Virtual DOM (Document Object Model)	Based on Virtual DOM (Document Object Model)
Written in	TypeScript	JavaScript	JavaScript
Community Support	A large community of developers and supporters	Facebook developers community	Open-source project sponsored through crowd-sourcing
Language Preference	Recommends the use of TypeScript	Recommends the use of JSX – JavaScript XML	HTML templates and JavaScript
Popularity	Widely popular among developers	More than 27,000 stars added over the year	More than 40,000 stars added on GitHub during the year
Companies Using	Used by Google, Forbes, Wix, and weather.com	Used by Facebook, Uber, Netflix, Twitter, Reddit, Paypal, Walmart, and others	Used by Alibaba, Baidu, GitLab, and others

4) Angular vs React vs Vue

What Should I Choose?

- If you like **flexibility** more than other features, use React.
- If you love coding in **TypeScript**, go for Angular.
- If you are a **JavaScript lover**, use React because it is all about JavaScript.
- If you are a **fan of clean code**, use Vue in your application.
- If you want **separation of concerns** in your application, use Vue.
- If you are a **fond of object-oriented programming**, Angular is definitely the pick for you.
- For **cross-platform app development**, React Native is an ideal choice

“Let’s learn Angular”

Thank you