
Reacting to Angular
an introduction to the best enterprise framework

In this presentation

- A basic introduction to Angular
- Comparing the concept/meaning to React
- Version 17 and above
- A lot of new features because of the Angular Renaissance
- No “Signal” explanation
- No “ngZone” explanation
- No complicated stuff

What is Angular?

- Initial release in Sep 14, 2016 by the evil Google
  - We don't talk about version 1.x
- Enforced MVC-S (module-view-controller-service)
- Heavily used in enterprises
- Very opinionated approach
- Best in-class CLI
- Latest version: 18
- With the renaissance in version 17
- Best meta framework: AnalogJS
- Best UI library: Angular Material

You might not know about Angular!

- The reason why Typescript popular
- The unspoken father of Vue
  - **Evan You** was a former Angular team member
  - **Therefore, it's not a coincidence that Vue is heavily inspired by Angular**
- The inspiration of React and their fanboy
- Faster than both React and Vue
  - In very specific conditions
- The motivation for ESBuild, Module Federation, Dependencies Injection, and many more…
- You (normally) have ONE way of doing things correctly

Template Language

Both Angular and React has their own way of writing templates.

Angular

- HTML-based template language
- Adding Angular-specific syntax (like `*ngFor`, `*ngIf`, or new control flow syntax `@if { }`, etc.)
- ✨NEW✨ Now you can also write HTML in your Angular TypeScript files!

React

- JSX-based template language
- Mostly JavaScript syntax for handling control flow (eg: `[].map(() => <Component />)`, `true ? <Component /> : null`, etc.)

React

```tsx
// src/app.tsx
import Welcome from './welcome';

export function App() {
  return (
    <div>
      <Welcome />
    </div>
  );
}

export default App;
```

Angular

```ts
// src/app.component.ts
import { Component } from '@angular/core';
import { RouterModule } from '@angular/router';
import { WelcomeComponent } from './welcome.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [WelcomeComponent, RouterModule],
  template: `
    <div>
      <app-welcome />
    </div>
  `,
})
export class AppComponent {}
```

Angular Single-File Component

```ts
// src/app.component.ts
@Component({
  selector: 'app-root',
  standalone: true,
  imports: [WelcomeComponent, RouterModule],
  template: `
    <div>
      <app-welcome />
    </div>
  `,
})
export class AppComponent {}
```

Angular Traditional Component

```ts
// src/app.component.ts
@Component({
  standalone: true,
  imports: [WelcomeComponent, RouterModule],
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent {}
```

```html
<!-- src/app.component.html -->
<div>
  <app-welcome></app-welcome>
</div>
```

```css
/* src/app.component.css */
```

React

```tsx
// src/app.tsx
export function App() {
  return (
    <div>
      Hello World
    </div>
  );
}
```

Angular

```ts
// src/app.component.ts
@Component({
  standalone: true,
  selector: 'app-root',
  template: `
    <div>
      Hello World
    </div>
  `,
})
export class AppComponent {}
```

React

- Most control flow is done with JavaScript syntax
- Annoying restrictions:
  - No `if` statements - replaced with ternary operators or logical `&&` / `||` operators
  - No `for` loops - replaced with `map`
  - No `switch` statements - replaced multiple ternary operators
  - Have to return a single element - wrap in a `<Fragment>` if needed
  - Can't use `class` - use `className` instead
  - Can't use `style` - use `style` object instead

Angular

- Most control flow is done with Angular-specific syntax
- Annoying restrictions:
  - Have to remember new syntax (`*ngIf` , `*ngFor` , etc., or `@if` , `@for` , … with new control flow syntax)
  - `<ng-template>` , `<ng-container>` , `<ng-content>` , and other Angular-specific elements
  - `[<name>]` and `(<name>)` syntax for binding and events (`[(ngModel)]` for two-way binding)

React

```tsx
// src/app.tsx
export function App() {
  return (
    <div>
      Hello World
    </div>
  );
}
```

Angular

```ts
// src/app.component.ts
@Component({
  selector: 'app-root',
  template: `
    <div>
      Hello World
    </div>
  `,
})
 export class AppComponent {}
```