


Angular¹⁸

AJ Arun Kumar

About me

- ▶ Senior Software Engineer working for  PITSolutions
- ▶ 7 years working with C# and JavaScript

History of Angular

- ▶ Framework created by Google to create web apps with rich interactive features
- ▶ Originally released in 2010 as AngularJS
 - ▶ Often known as Angular version 1
 - ▶ Discontinued on 2022
- ▶ A ground up re write of the framework was released in 2016 as “Angular”
 - ▶ Out of the box TypeScript support, improved Developer Experience using CLI

What is Angular ?

- ▶ Abstraction over DOM manipulating JavaScript API like `document.querySelector`, `document.createElement` etc.
- ▶ Simply put, Angular was created using JavaScript to make web dev easy just like FORTRAN/C++ was created using assembly to make programming easy

Abstraction

```
mov al, [num1] ; Load the first number into AL  
add al, [num2] ; Add the second number to AL  
mov [result], al ; Store the result in memory
```

VS

```
num1 = num1 + num2;
```

Angular is a similar type of thing. It abstracts over native JavaScript DOM APIs like `document.querySelector` and `document.createElement`

When should I avoid frameworks like Angular ?

- ▶ If your web app needs to squeeze out every drop of CPU performance, better **not** use Angular (or any other framework)
- ▶ E.g, you are building applications like VSCode

Why should I use Angular?

- ▶ Unless you are doing extremely performance sensitive web apps, which is 90+% of the time, let a JavaScript framework like Angular take care of your DOM manipulation
- ▶ It lets you build and ship rich UI features fast with way less bugs than writing plain JavaScript

How does Angular does that ?

- ▶ Special syntax and keywords over HTML
- ▶ Developers can express intend instead of writing step by step implementation
- ▶ Write code like “if password != confirmPassword, then hide register button” ^

Demo time !

- ▶ Create app, compile app
- ▶ HTML is not HTML (inspect output of compilation and analyse HTML)
- ▶ Dev server
- ▶ One way binding
- ▶ Attribute binding
- ▶ Event binding
- ▶ @if, @for
- ▶ Forms, Two way binding

Reactivity in Angular (how it has been working till now)

- ▶ `zone.js/ngZone` intercepts calls to DOM APIs like `document.addEventListener`, `setTimeout`, `fetch` etc and creates hooks.
 - ▶ In simple words - `ngZone` can detect even if an air molecule moved in UI (DOM). It tells angular runtime "something changed"
 - ▶ Angular runtime triggers "change detection". It needs check the whole app to see what changed. And then it updates the DOM/model depending upon the change. (If UI was changed by a click or keyboard the UI will be updated and if the model was changed by an async operation by `fetch/XHR` then specific portion of DOM will be updated)
 - ▶ Change detection is extremely optimised by the awesome angular engineers to be fast and efficient, but still, it cannot be efficient beyond a limit simply because `ngZone` cannot pin point the location in the component tree where the change happened.

Reactivity in Angular (how it will be in future)

- ▶ New reactive data structure has been created called 'signals'
- ▶ A signal can understand when its value is changed and can directly notify angular about the change. The notification is going to contain the exact location where the change happened, so no need to scan the entire app component tree to detect what changed.
- ▶ **ExpressionChangedAfterItHasBeenCheckedError** will not happen again because DOM is updated after all the changes to signals are done.

Demo time !

- ▶ Components
- ▶ Create signals
- ▶ Signals are guaranteed to be synchronous - you always have to specify a default value
- ▶ Computed
- ▶ Input, output, viewChild



Presented by
Arun