The State of SPA Frameworks

ABOUT ME

Jonas Bandi
jonas.bandi@ivorycode.com
Twitter: @jbandi

- Freelancer, in den letzten 10 Jahren vor allem in Projekten im Spannungsfeld zwischen modernen Webentwicklung und traditionellen Geschäftsanwendungen.
- In-House Kurse & Beratungen zu Web-Technologien im Enterprise: UBS, Postfinance, Mobiliar, AXA, BIT, SBB, Elca, Adnovum, BSI ...
- Dozent an der Berner Fachhochschule seit 2007

Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

letsboot.ch
swiss dev training

digicomp

JavaScript / Angular / React / Vue / Vaadin
Schulung / Beratung / Coaching / Reviews

jonas.bandi@ivorycode.com

What are you using … ?

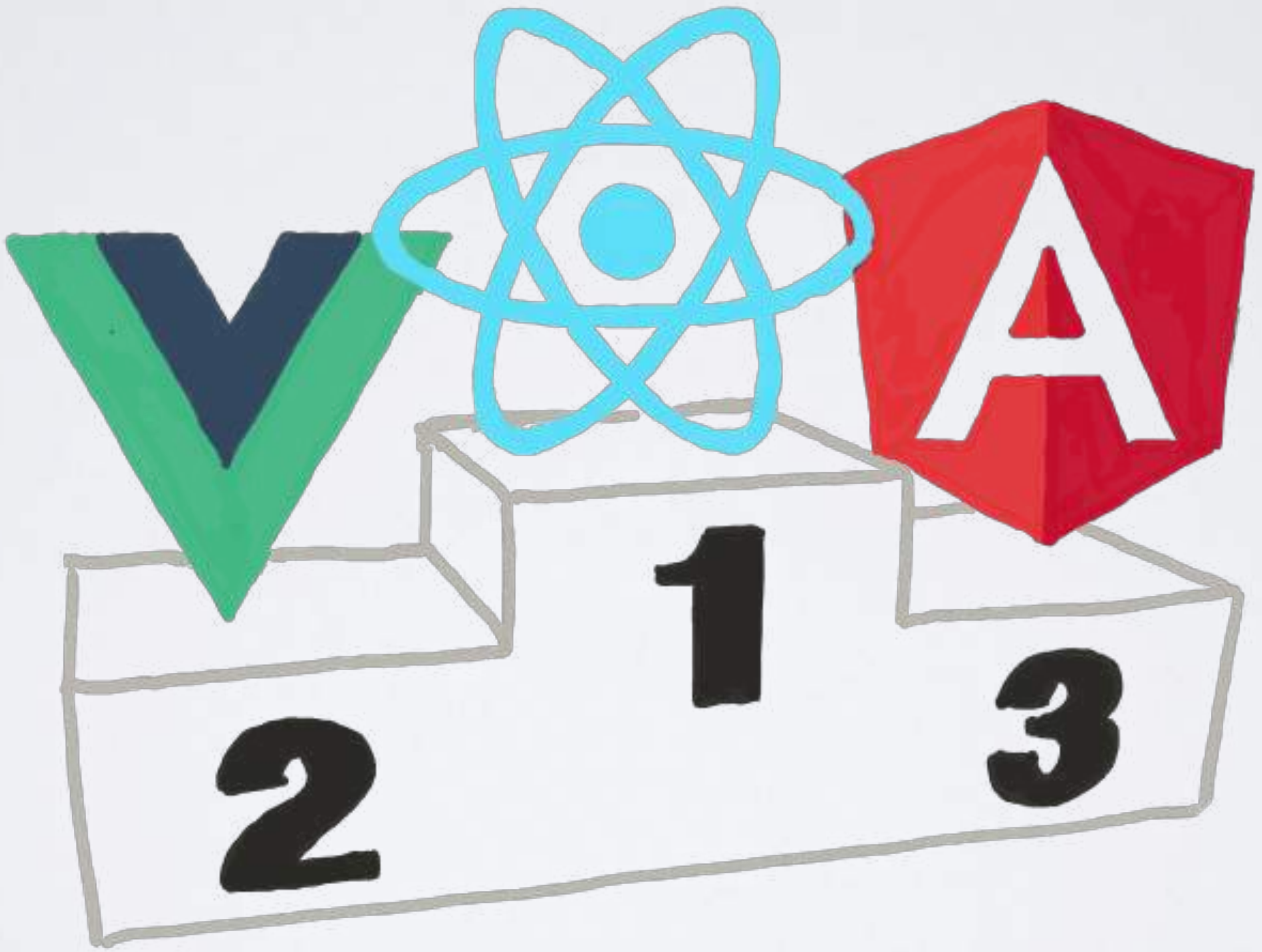
Agenda

Big Picture: Angular, React, Vue

The Framework Convergence

Latest Trends and Anecdotes: Angular, React, Vue

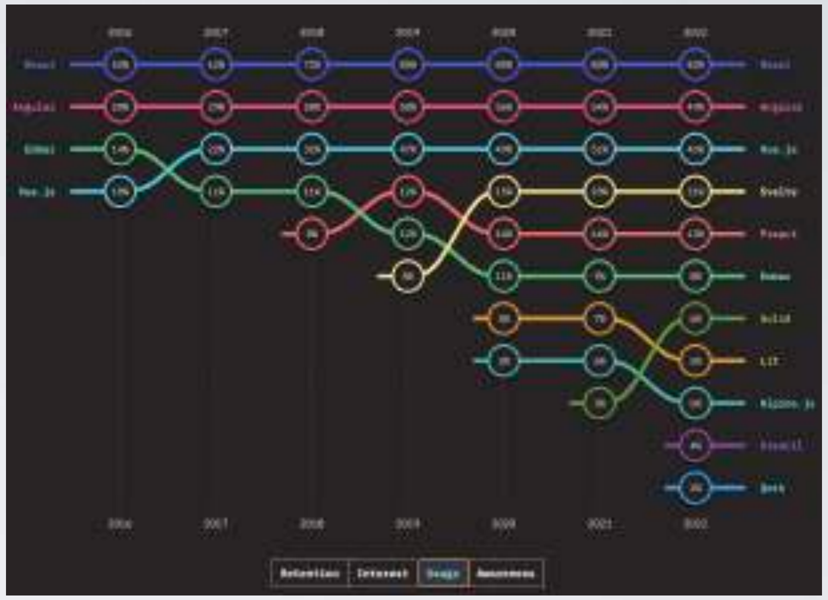
Which one is The Best™ ?



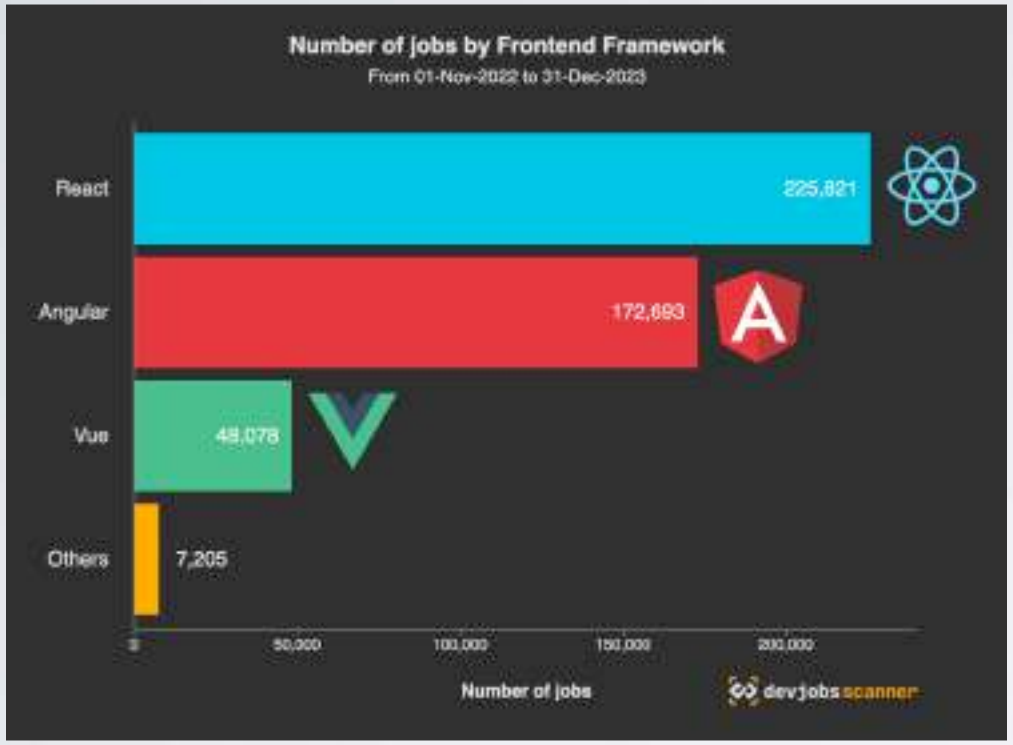
React

The most popular SPA framework!

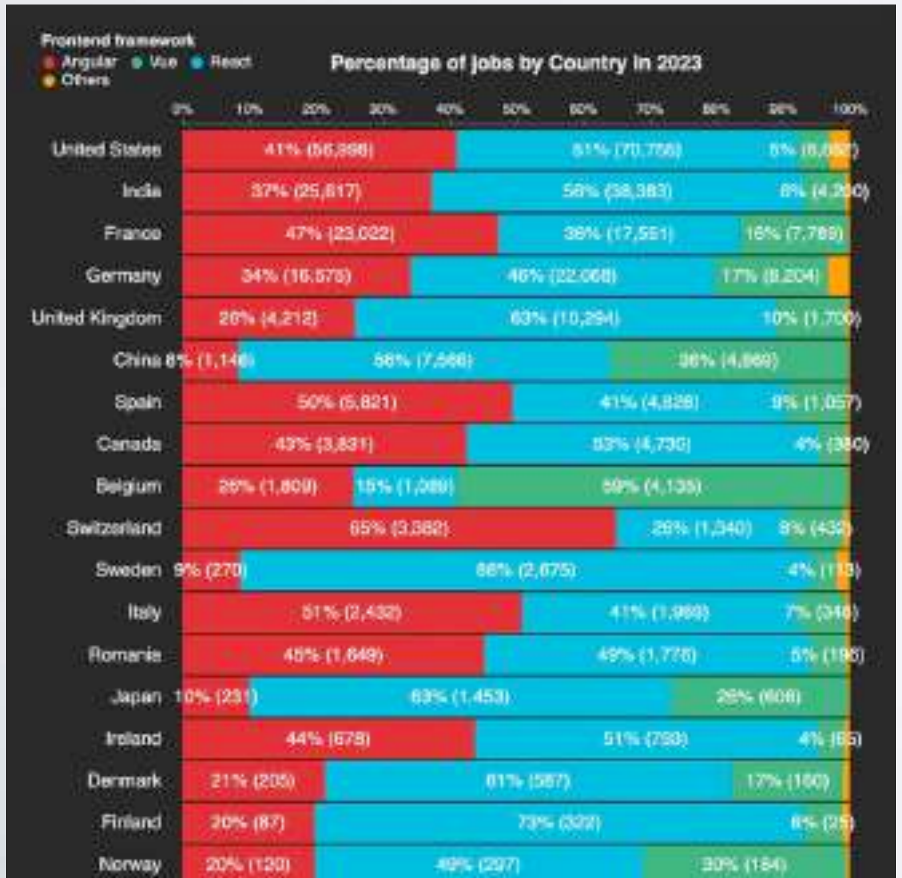
State of JavaScript Survey 2022: Usage



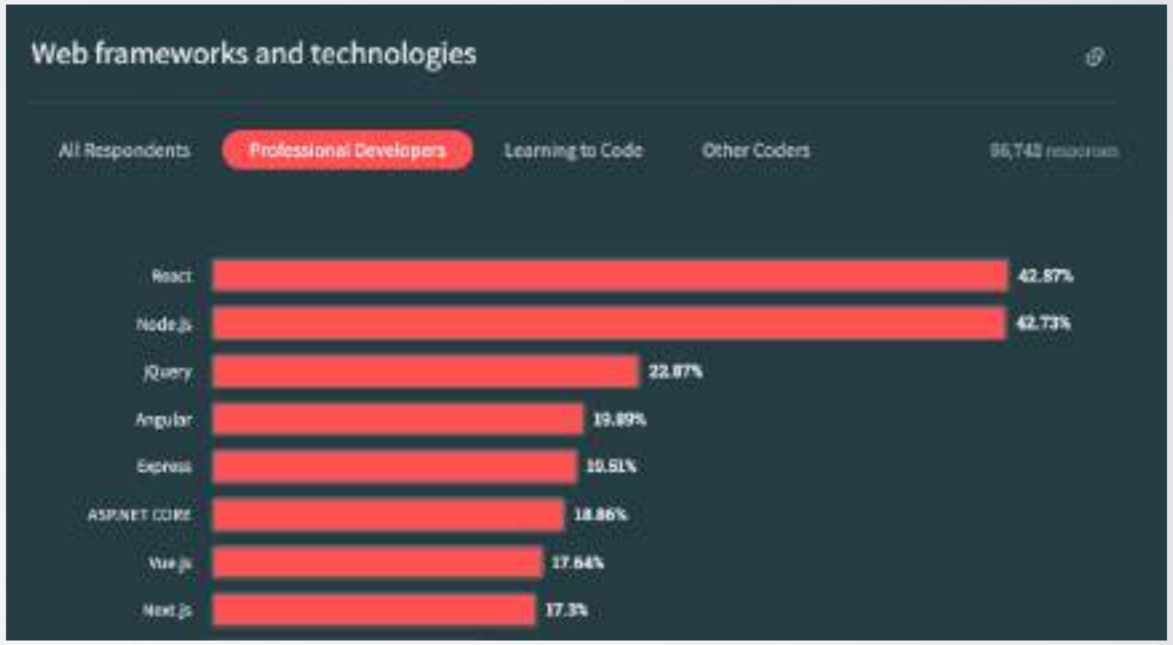
https://2022.stateofjs.com/en-US/libraries/front-end-frameworks/



https://www.devjobsscanner.com/blog/the-most-demanded-frontend-frameworks/



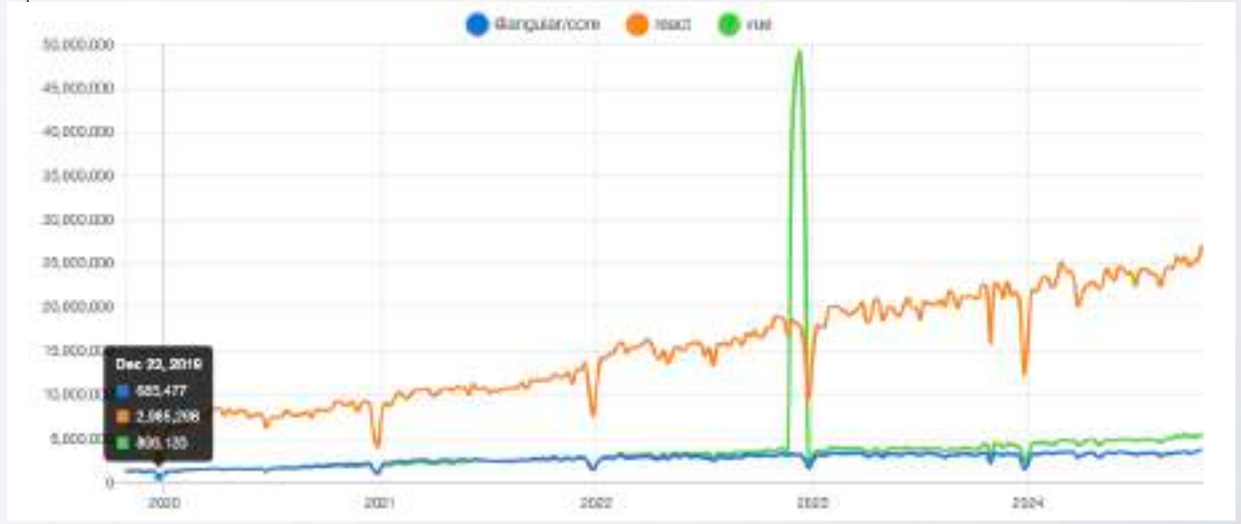
The most compelling reason to learn Angular in 2023:
https://www.youtube.com/watch?v=WC3hiMiH5kI



React	42.87%
Node.js	42.73%
jQuery	22.87%
Angular	19.89%
Express	19.51%
ASP.NET CORE	18.86%
Vue.js	17.64%
Next.js	17.3%

https://survey.stackoverflow.co/2023/#most-popular-technologies-webframe-prof

npm downloads:



https://npmtrends.com/@angular/core-vs-react-vs-vue

Framework Usage

React

ChatGPT Claude

NETFLIX

airbnb WhatsApp

Dropbox amazon

zalando

Outlook.com

Microsoft JIRA

BBC Spectrum Grafana

Angular

Google Analytics Gemini

"2000+ projects at google"

Office 365 die Mobiliar

Capital One

Forbes

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Vue.js

Adobe

MediaWiki

WIKIPEDIA
The Free Encyclopedia

upwork Alibaba.com

Nintendo

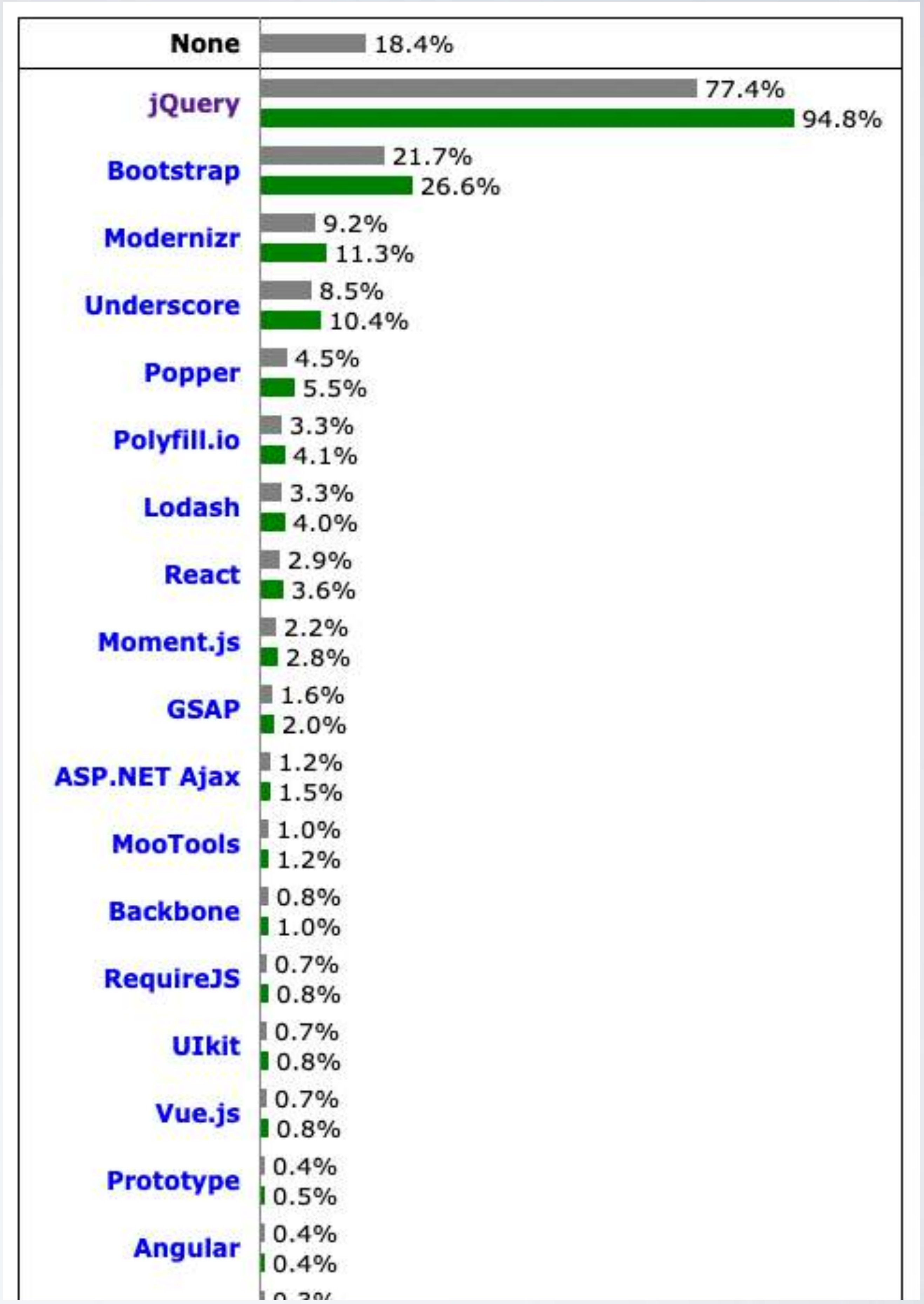
GitLab Baidu 百度 腾讯 Tencent

Framework usages of Angular, React and Vue are all still growing.

At the moment it does not look like one of the frameworks is "killing" another framework.

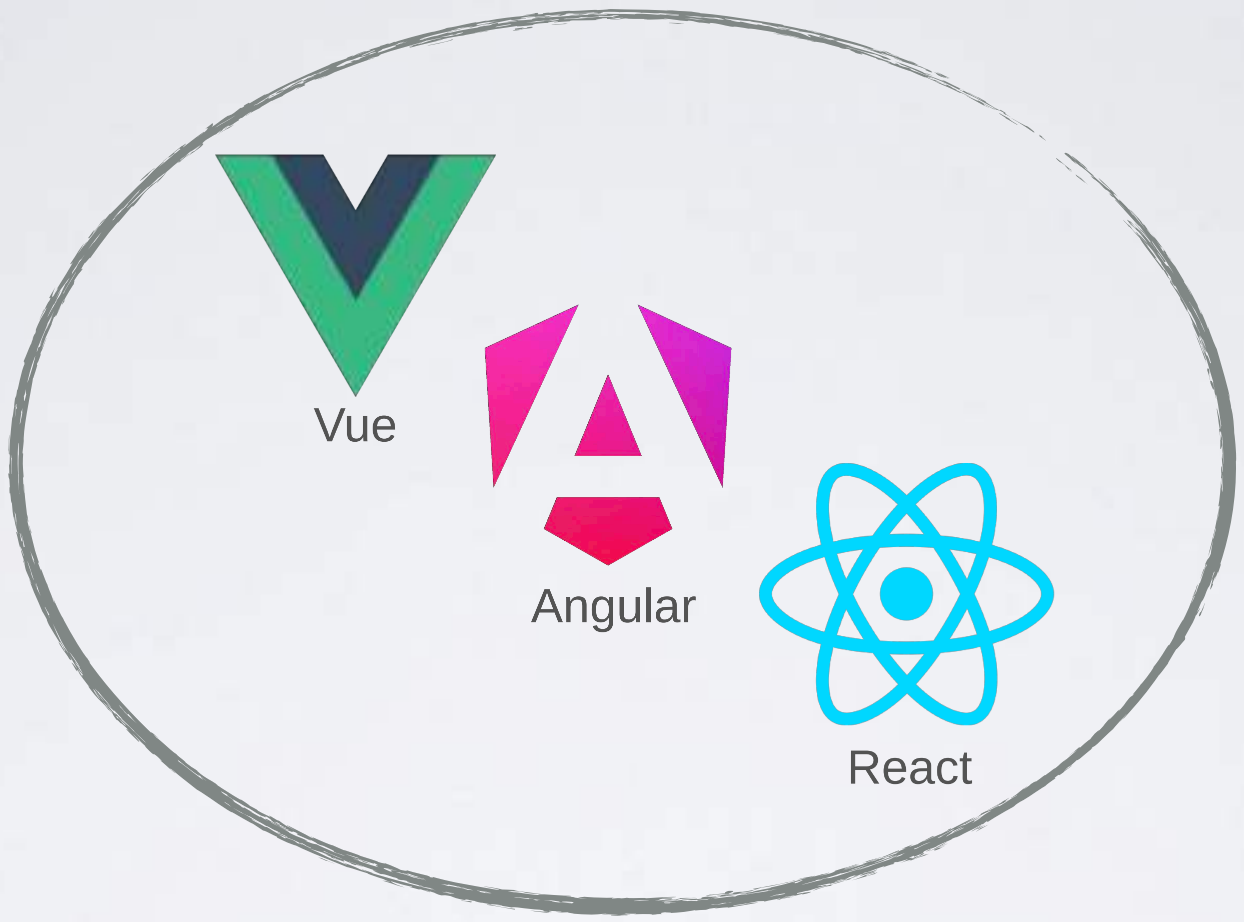
🤫 by the way …

… the web still runs on jQuery



https://w3techs.com/technologies/overview/javascript_library

Typical Single Page Applications



Angular, React and Vue Projects are very similar:

- based on modern JavaScript (incl. TypeScript support)
- built on top of the npm ecosystem
- require Node.js & npm tooling at build-time
- heavy conceptual separation of frontend and backend
- "unopinionated" about backend technologies
- embracing the Browser as "application platform"

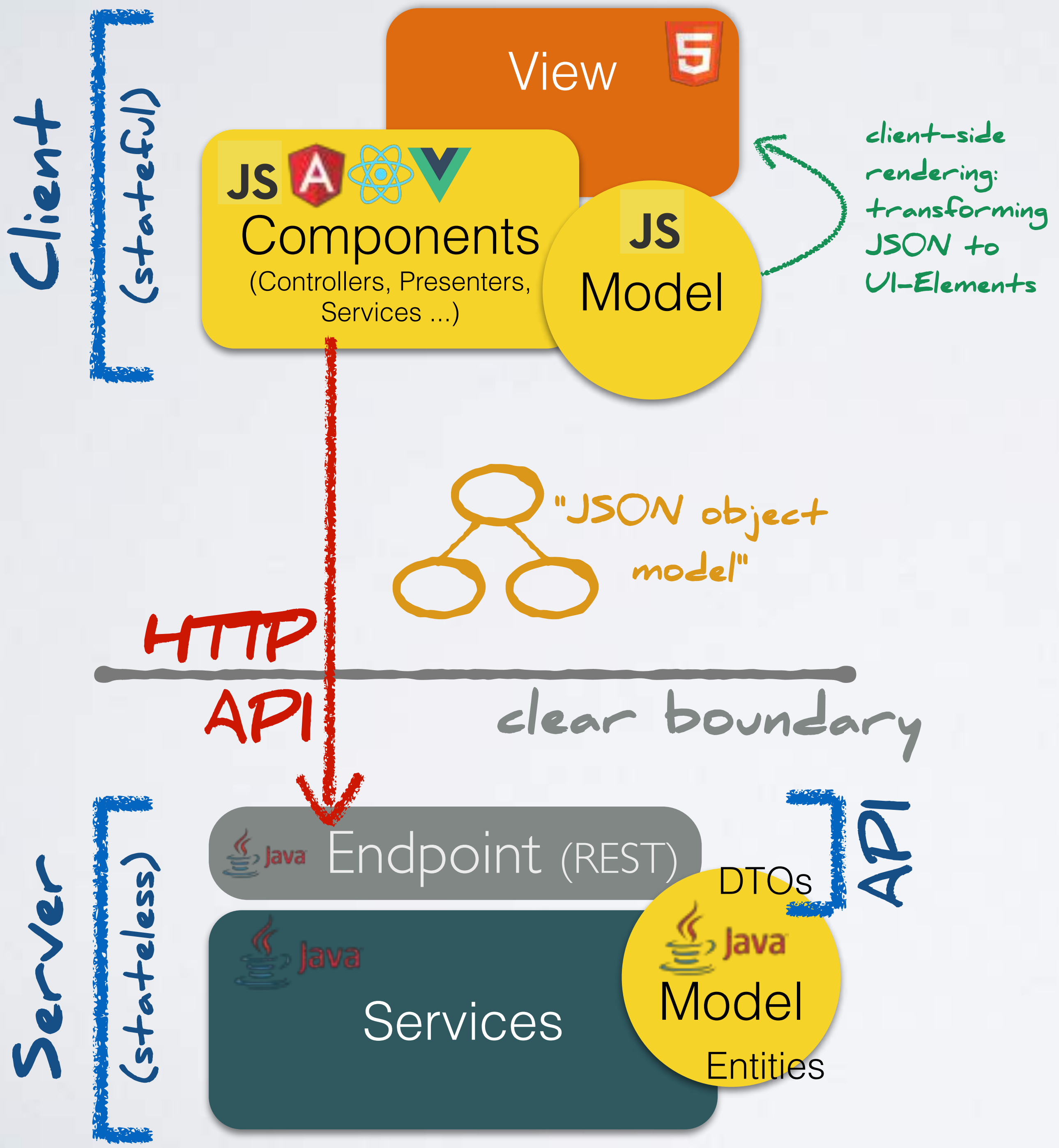
Does it even matter?

Angular, React and Vue are very similar.

Choosing the flavor of "state-of-the-art" SPA frontend framework is *not a critical success factor* for your project.
Architecture, tooling, development process and organizational factors have *an order of magnitude more influence* on the success of your project.

The typical SPA Architecture

the "traditional" client-server boundary



The rise of SPA development caused a "de-facto" architecture of formalized HTTP/REST-APIs.

Symptoms:
- "API-First" Design
- "The central role of API-Gateways" (the return of ESBs)

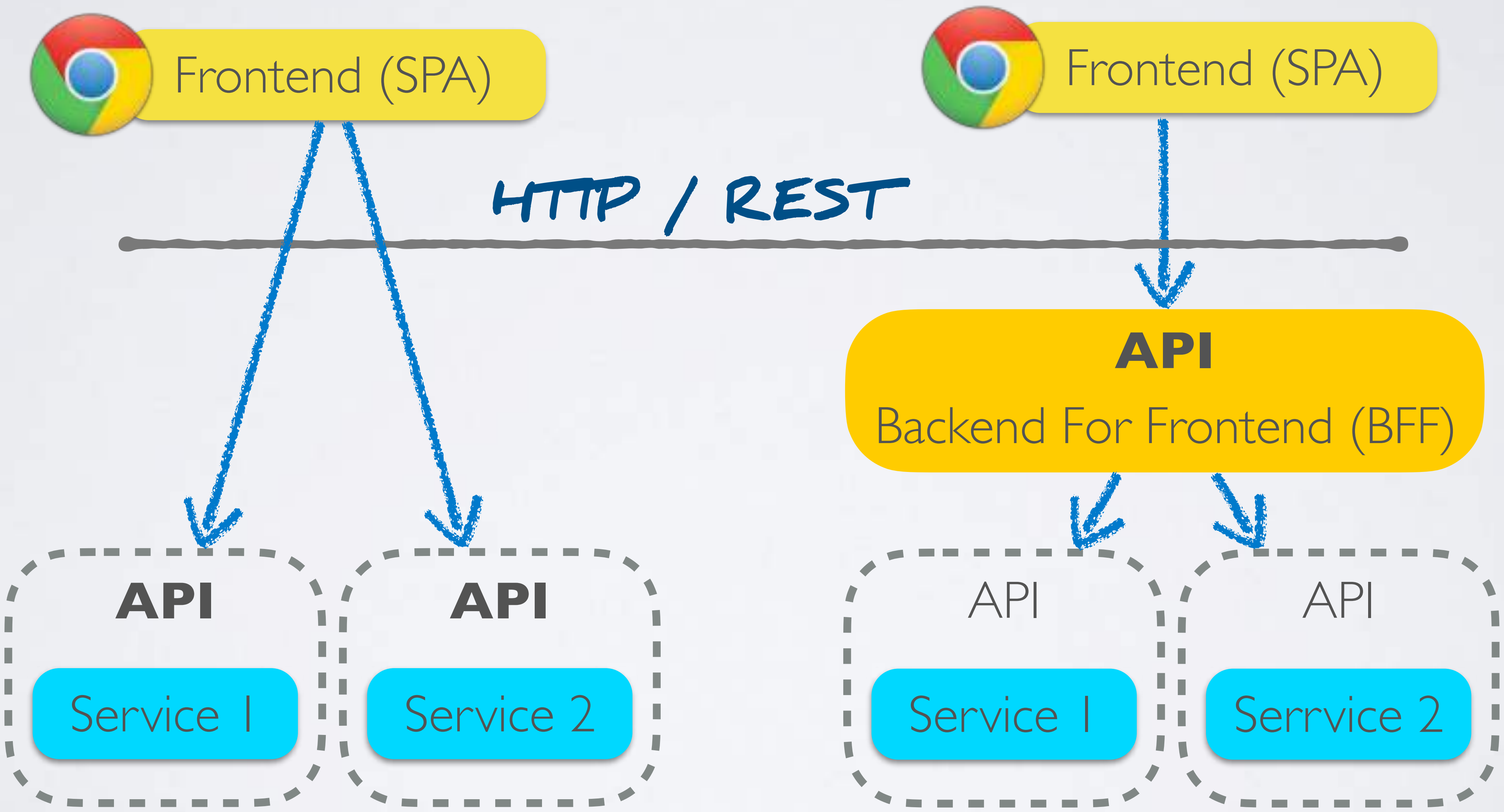
...

Creating a formalized API is a non-trivial effort: Design of URLs, Mapping, Serialization, Security ...

There are advantages in a formalized HTTP API: separation of concern, clearly specified and testable boundaries, reuse, team separation ...

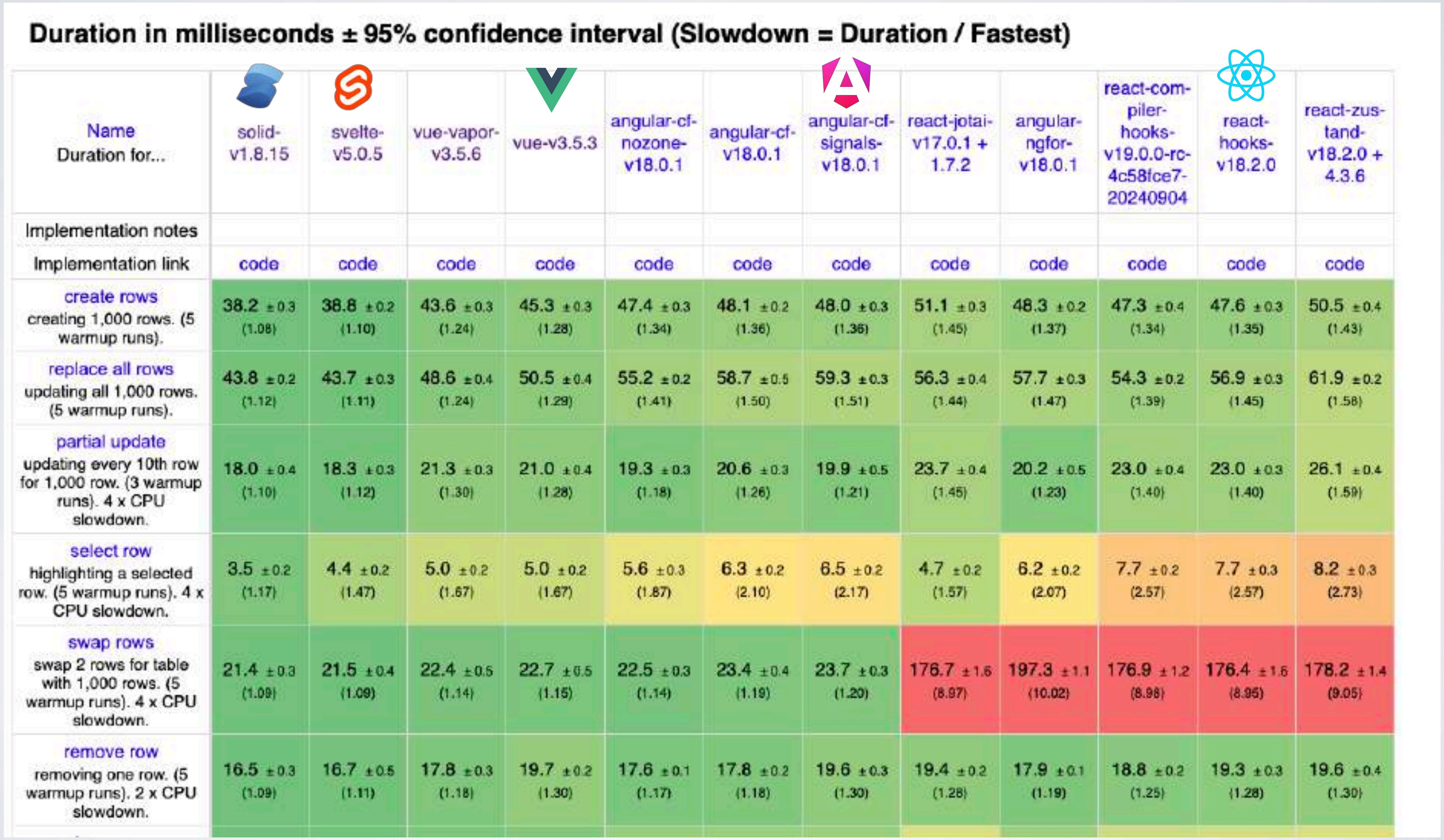
Traditional Architectures for SPAs

Client - API - Server



https://samnewman.io/patterns/architectural/bff/

Performance?

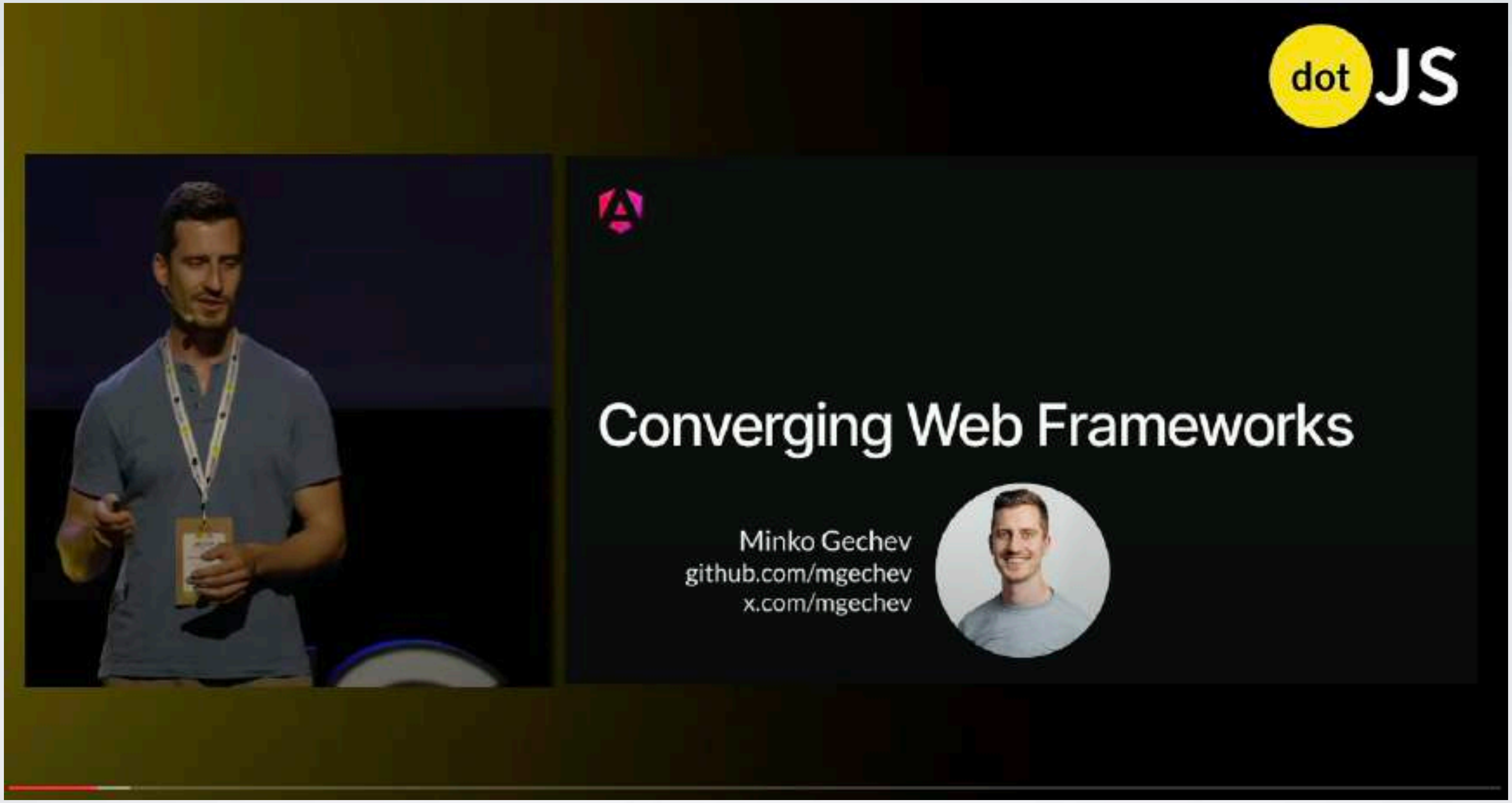


Duration in milliseconds ± 95% confidence interval (Slowdown = Duration / Fastest)

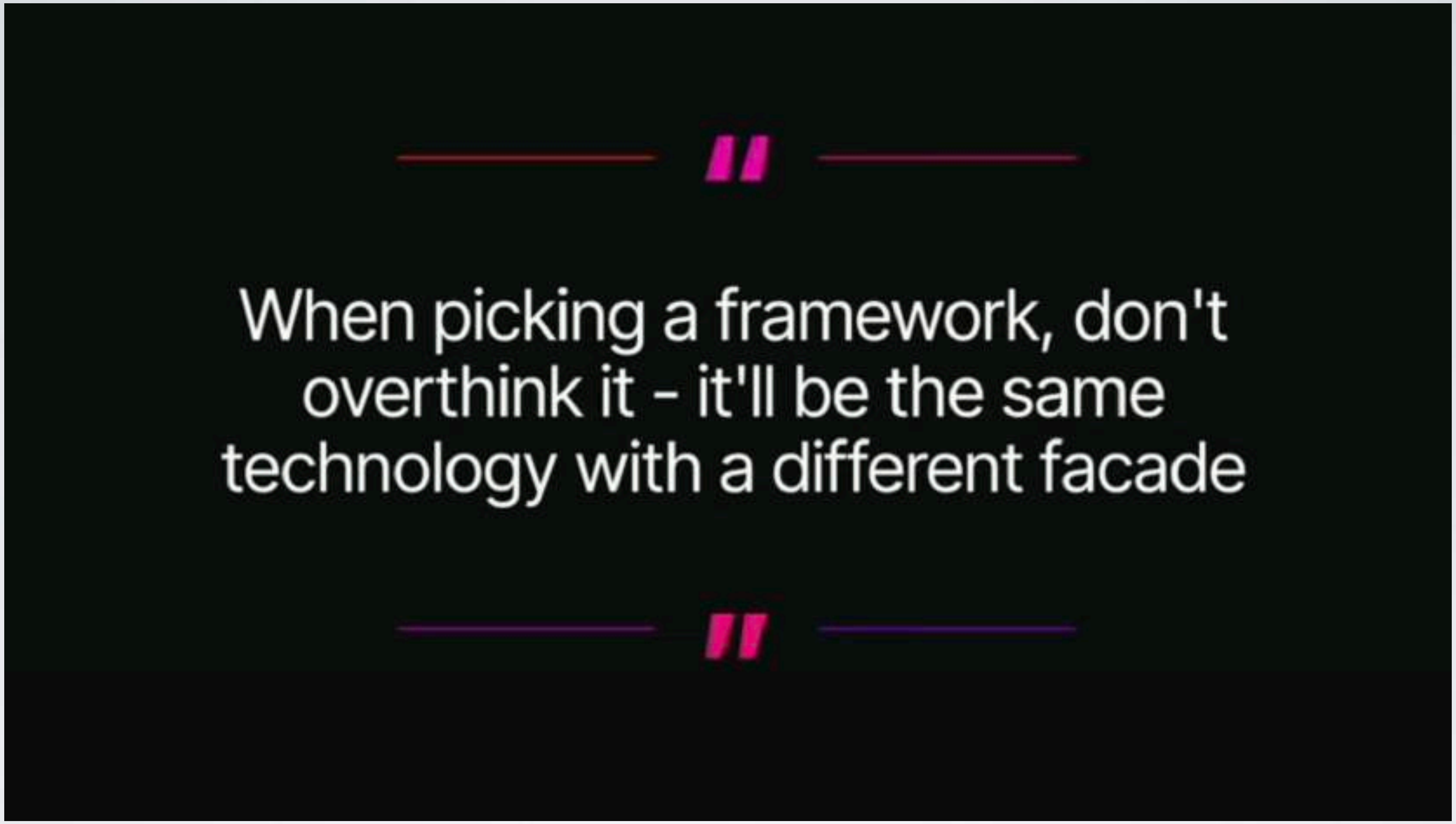
Name Duration for...	solid-v1.8.15	svelte-v5.0.5	vue-vapor-v3.5.6	vue-v3.5.3	angular-cf-nozone-v18.0.1	angular-cf-v18.0.1	angular-cf-signals-v18.0.1	react-jotai-v17.0.1 + 1.7.2	angular-ngfor-v18.0.1	react-compiler-hooks-v19.0.0-rc-4c58fce7-20240904	react-hooks-v18.2.0	react-zustand-v18.2.0 + 4.3.6
Implementation notes												
Implementation link	code	code	code	code	code	code	code	code	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	38.2 ±0.3 (1.08)	38.8 ±0.2 (1.10)	43.6 ±0.3 (1.24)	45.3 ±0.3 (1.28)	47.4 ±0.3 (1.34)	48.1 ±0.2 (1.36)	48.0 ±0.3 (1.36)	51.1 ±0.3 (1.45)	48.3 ±0.2 (1.37)	47.3 ±0.4 (1.34)	47.6 ±0.3 (1.35)	50.5 ±0.4 (1.43)
replace all rows updating all 1,000 rows. (5 warmup runs).	43.8 ±0.2 (1.12)	43.7 ±0.3 (1.11)	48.6 ±0.4 (1.24)	50.5 ±0.4 (1.29)	55.2 ±0.2 (1.41)	58.7 ±0.5 (1.50)	59.3 ±0.3 (1.51)	56.3 ±0.4 (1.44)	57.7 ±0.3 (1.47)	54.3 ±0.2 (1.39)	56.9 ±0.3 (1.45)	61.9 ±0.2 (1.58)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	18.0 ±0.4 (1.10)	18.3 ±0.3 (1.12)	21.3 ±0.3 (1.30)	21.0 ±0.4 (1.28)	19.3 ±0.3 (1.18)	20.6 ±0.3 (1.26)	19.9 ±0.5 (1.21)	23.7 ±0.4 (1.45)	20.2 ±0.5 (1.23)	23.0 ±0.4 (1.40)	23.0 ±0.3 (1.40)	26.1 ±0.4 (1.59)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	3.5 ±0.2 (1.17)	4.4 ±0.2 (1.47)	5.0 ±0.2 (1.67)	5.0 ±0.2 (1.67)	5.6 ±0.3 (1.87)	6.3 ±0.2 (2.10)	6.5 ±0.2 (2.17)	4.7 ±0.2 (1.57)	6.2 ±0.2 (2.07)	7.7 ±0.2 (2.57)	7.7 ±0.3 (2.57)	8.2 ±0.3 (2.73)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	21.4 ±0.3 (1.09)	21.5 ±0.4 (1.09)	22.4 ±0.5 (1.14)	22.7 ±0.5 (1.15)	22.5 ±0.3 (1.14)	23.4 ±0.4 (1.19)	23.7 ±0.3 (1.20)	176.7 ±1.6 (8.97)	197.3 ±1.1 (10.02)	176.9 ±1.2 (8.98)	176.4 ±1.6 (8.95)	178.2 ±1.4 (9.05)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	16.5 ±0.3 (1.09)	16.7 ±0.5 (1.11)	17.8 ±0.3 (1.18)	19.7 ±0.2 (1.30)	17.6 ±0.1 (1.17)	17.8 ±0.2 (1.18)	19.6 ±0.3 (1.30)	19.4 ±0.2 (1.28)	17.9 ±0.1 (1.19)	18.8 ±0.2 (1.25)	19.3 ±0.3 (1.28)	19.6 ±0.4 (1.30)

https://krausest.github.io/js-framework-benchmark/current.html

Thesis:
Current Frameworks have become very similar



dotJS 2024 - Minko Gechev - Product Lead for Angular at Google
Converging Web Frameworks: https://www.youtube.com/watch?v=grRH8e46Pso

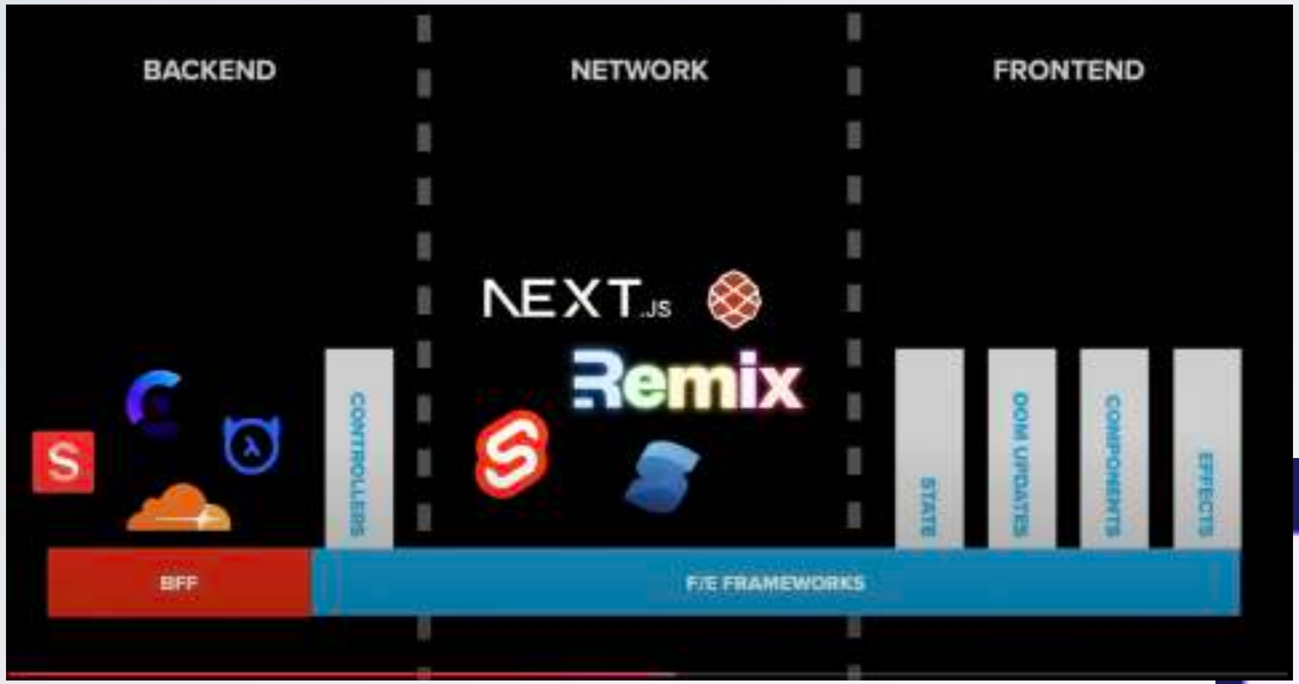


"When picking a framework, don't overthink it - it'll be the same technology with a different facade"

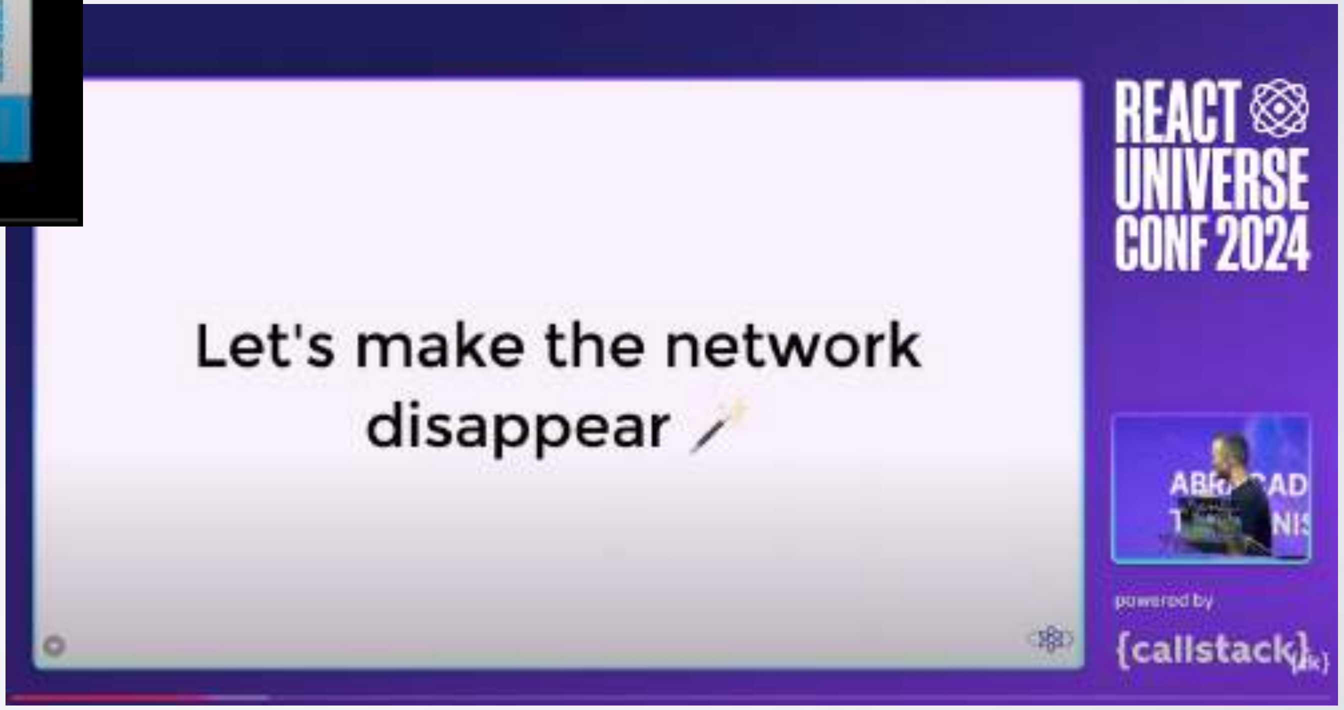
dotJS 2024 - Minko Gechev - Product Lead for Angular at Google
Converging Web Frameworks: https://www.youtube.com/watch?v=grRH8e46Pso

Consider non-technical factors:
Stability & Reliability, Community, Enjoyment

Thesis:
Innovation in frontend frameworks is moving towards the server-side and the full-stack perspective of web development.

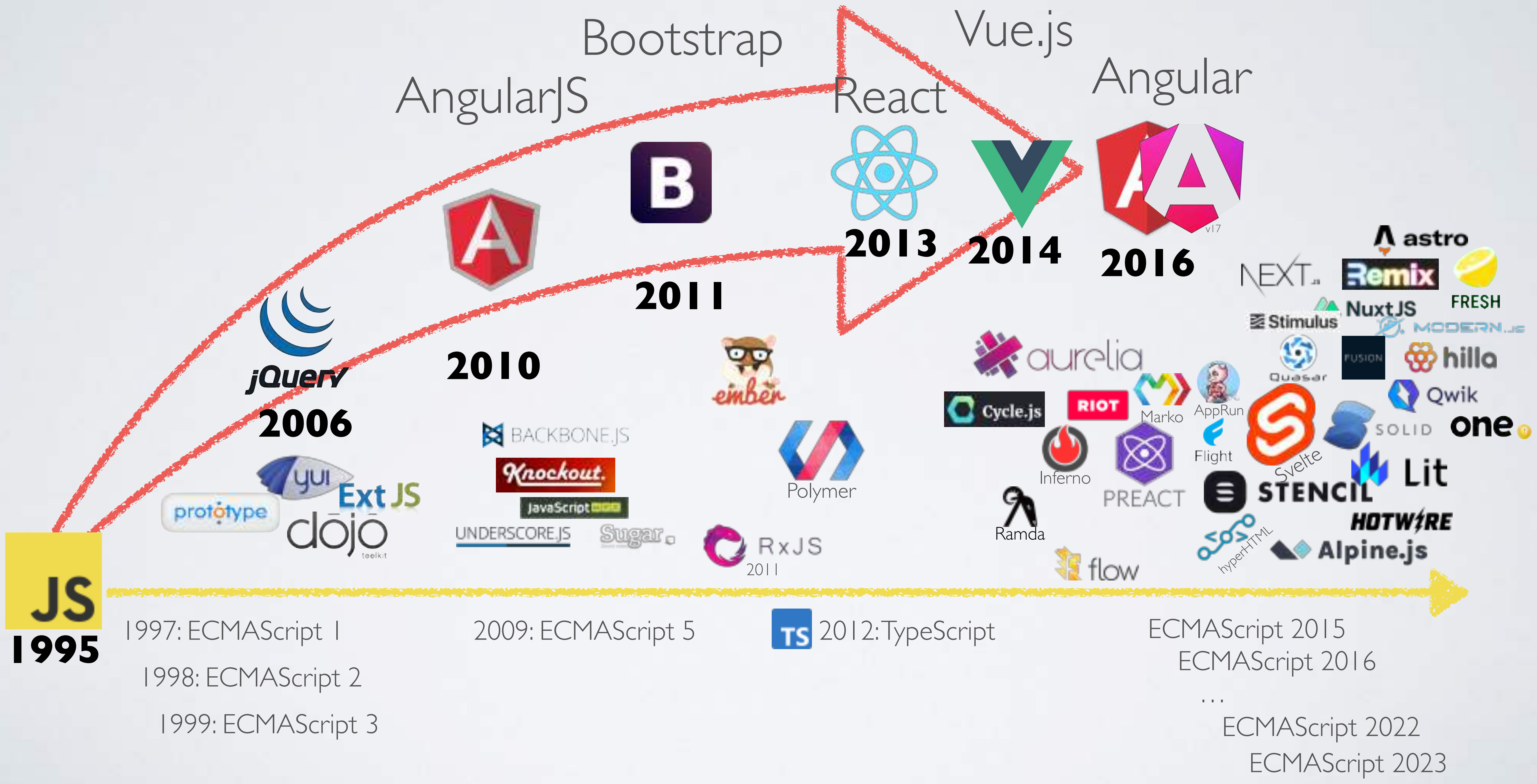


"Mind The Gap" by Ryan Florence
at Big Sky Dev Con 2024
https://www.youtube.com/watch?v=zqhE-CepH2g


Let's make the network disappear

Abracadabra: The vanishing network — Kent C. Dodds | React Universe Conf 2024
https://www.youtube.com/watch?v=E8LLty9rTWw

The Frontend JavaScript Ecosystem



Framework Convergence by Example

SIGNALS

Signals for fine grained reactivity

Build Tooling based on Vite

Signals for fine-grained Reactivity

Signals

Signals are reactive wrappers around state.
They track the access of the state and notify changes.

```
const count = signal(0);

console.log(count());

count.set(count + 1);
```

```
const count = ref(0);

console.log(count.value);

count.value += 1;
```

```
const [count, setCount] = createSignal(0);

console.log(count());

setCount(count() + 1)
```

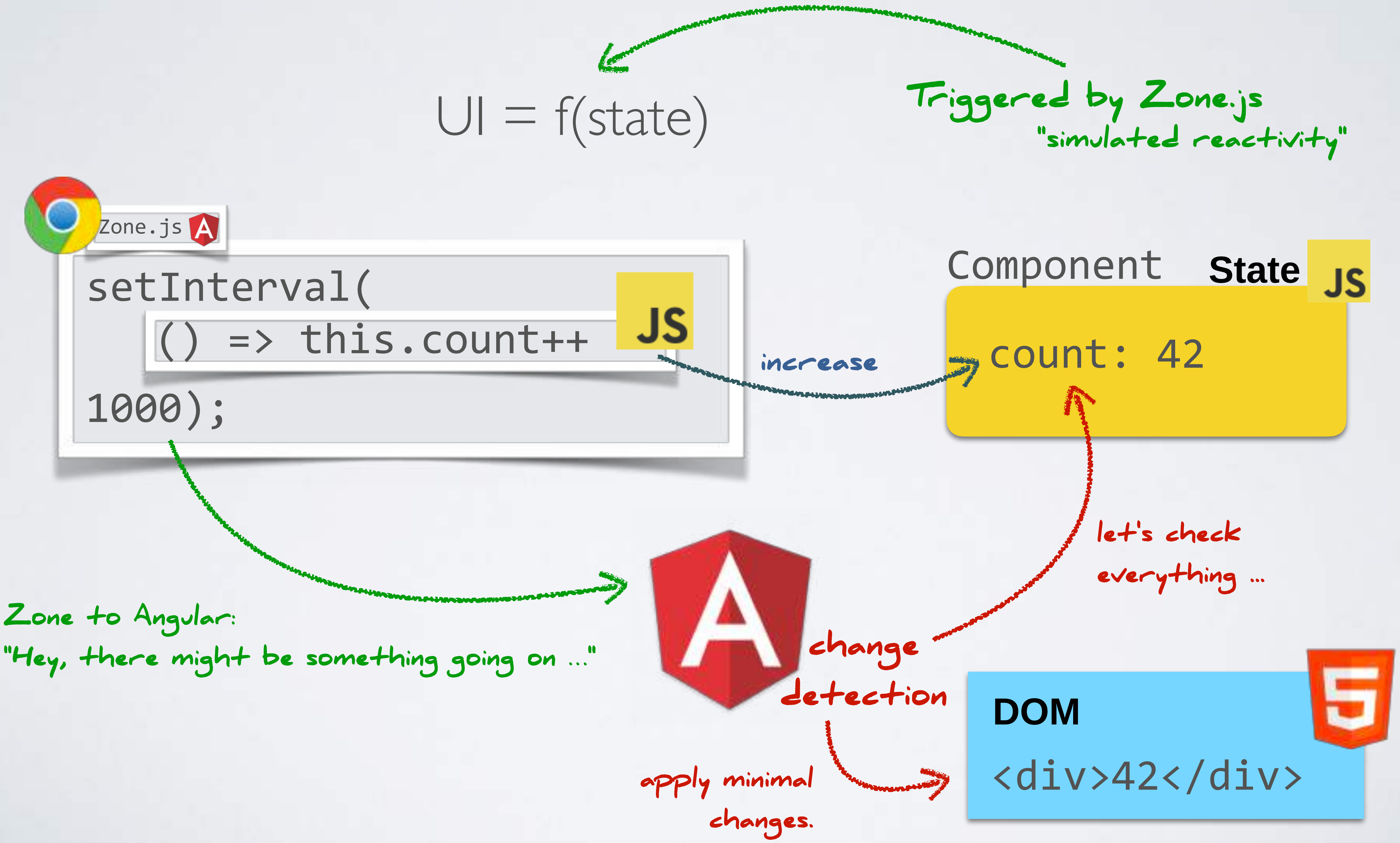
```
const count = signal(0);

console.log(count.value);

count.value += 1;
```

Default Reactivity in Angular

"simulated reactivity"



```
setInterval(
    () => this.count++
1000);
```

```
count: 42
```

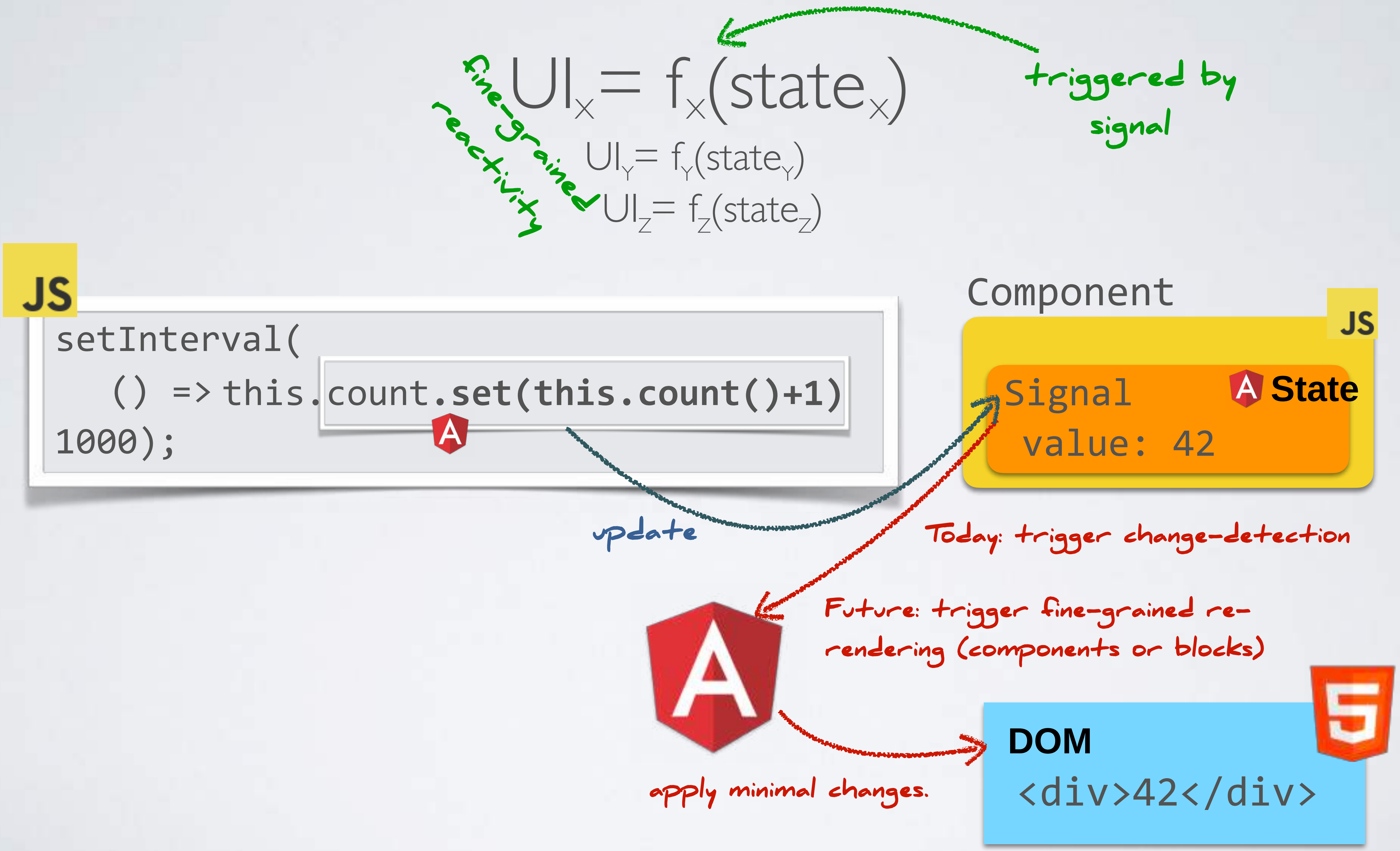
```
<div>42</div>
```

Reactivity with Angular Signals

$$UI_X = f_X(state_X)$$

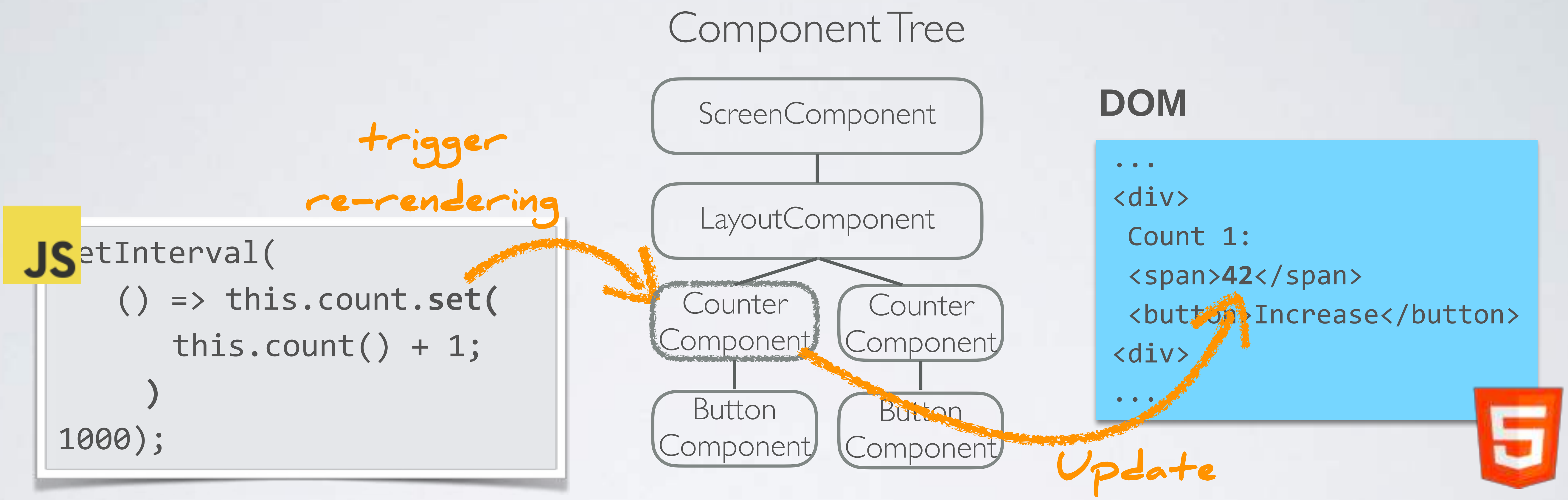
$$UI_Y = f_Y(state_Y)$$

$$UI_Z = f_Z(state_Z)$$



```
setInterval(
   () => this.count.set(this.count()+1)
1000);
```

Fine Grained Reactive State



```
JS setInterval(
    () => this.count.set(
        this.count() + 1;
    )
1000);
```

```
...
<div>
 Count 1:
 <span>42</span>
 <button>Increase</button>
<div>
...
```

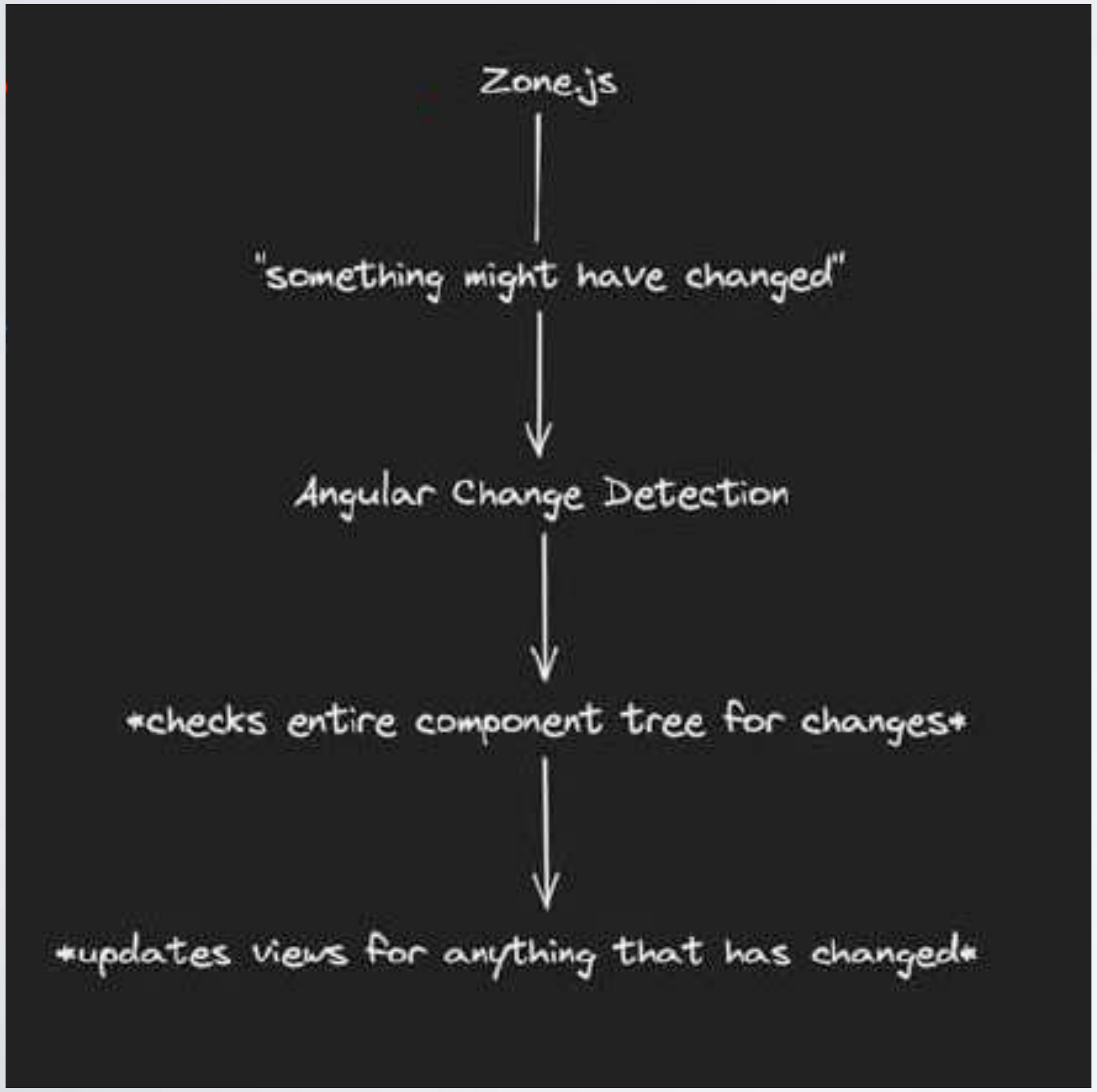
Fine grained reactive state only triggers re-rendering on components (or blocks) that depend on changed state.

Many frameworks already implement fine grained reactivity: Vue, Solid, Svelte, Qwik ...

Signals and Change Detection

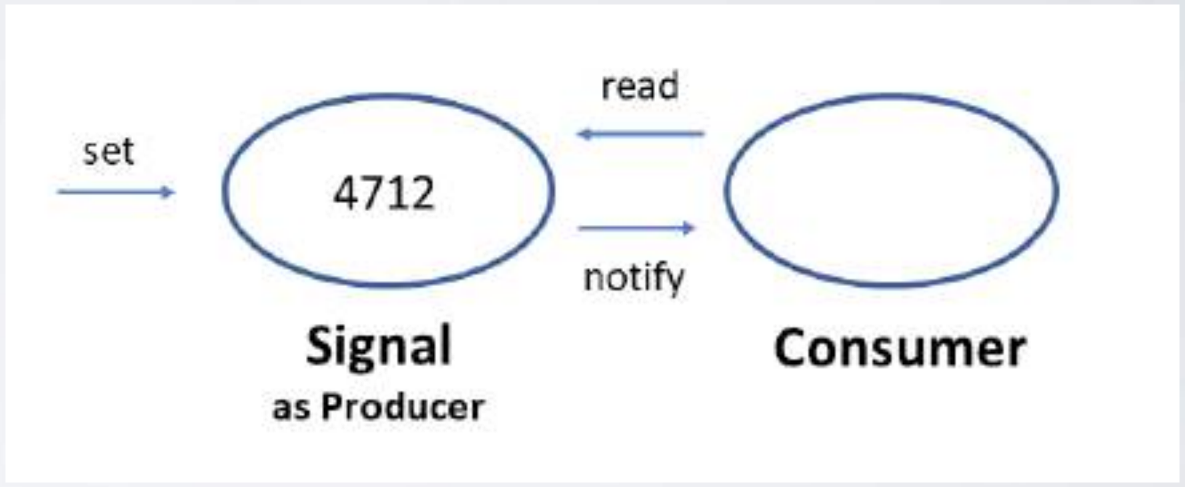
Signals will enable fine grained reactivity

Change Detection with Zone.js today:


Zone.js
"something might have changed"
Angular Change Detection
checks entire component tree for changes
updates views for anything that has changed

https://modernangular.com/articles/what-are-signal-based-components

Reactivity with signal based components in the future:



Signals enable a granular event-driven architecture for precise and isolated ui updates.

https://dev.to/this-is-learning/react-vs-signals-10-years-later-3k71

https://www.angulararchitects.io/aktuelles/angular-signals/

```
import { Component, signal, computed, effect, OnInit }
                                   from '@angular/core';
@Component({
  selector: 'app-counter',
  template: `
    <h3>Count {{ count() }}</h3>
    <h3>Double {{ doubleCount() }}</h3>
    <button (click)="increment()">Increment</button>
  `,
})
export class CounterComponent {

  count = signal(0);

  doubleCount = computed(() => {
    console.log('Computing double count');
    return this.count() * 2;
  });

  constructor() {
    effect(() => {
      console.log('Effect: Count is now: ', this.count());
    });
  }

  increment() {
    this.count.set(this.count() + 1);
  }

}
```

Angular Signals: https://angular.dev/guide/signals

```
<template>
    <h3>Count: {{ count }}</h3>
    <h3>Double: {{ doubleCount }}</h3>
    <button @click="increment">Increment</button>
</template>
<script setup lang="ts">
  import { computed, effect, ref } from "vue";

  const count = ref(0);

  const doubleCount = computed(() => {
    console.log('Computing double count');
    count.value * 2:
  });


  watchEffect(() => {
    console.log("count changed", count.value);
  });


  function increment() {
    count.value = count.value + 1;
  }

</script>
```

Signals ... old wine in new skins?
(aka Fine-Grained Reactivity)

Vue (Composition API in 2020)	`ref` / `reactive`
Solid (2019)	`createSignal`
Svelte v5 (2024)	Runes: `$state`,
Preact (signals in 2022)	`signal`
Angular (v16 in 2023)	`signal`
Jotai (for React) (2020)	`atom`
Recoil (for React) (2020)	`atom`
MobX (2016)	`observable`
Knockout (2010)	`observable`

A hands-on Introduction to Fine-Grained Reactivity: https://dev.to/ryansolid/a-hands-on-introduction-to-fine-grained-reactivity-3ndf

FrontendMasters - Reactivity with SolidJS: https://frontendmasters.com/courses/reactivity-solidjs/

Future: Native Signals in JavaScript

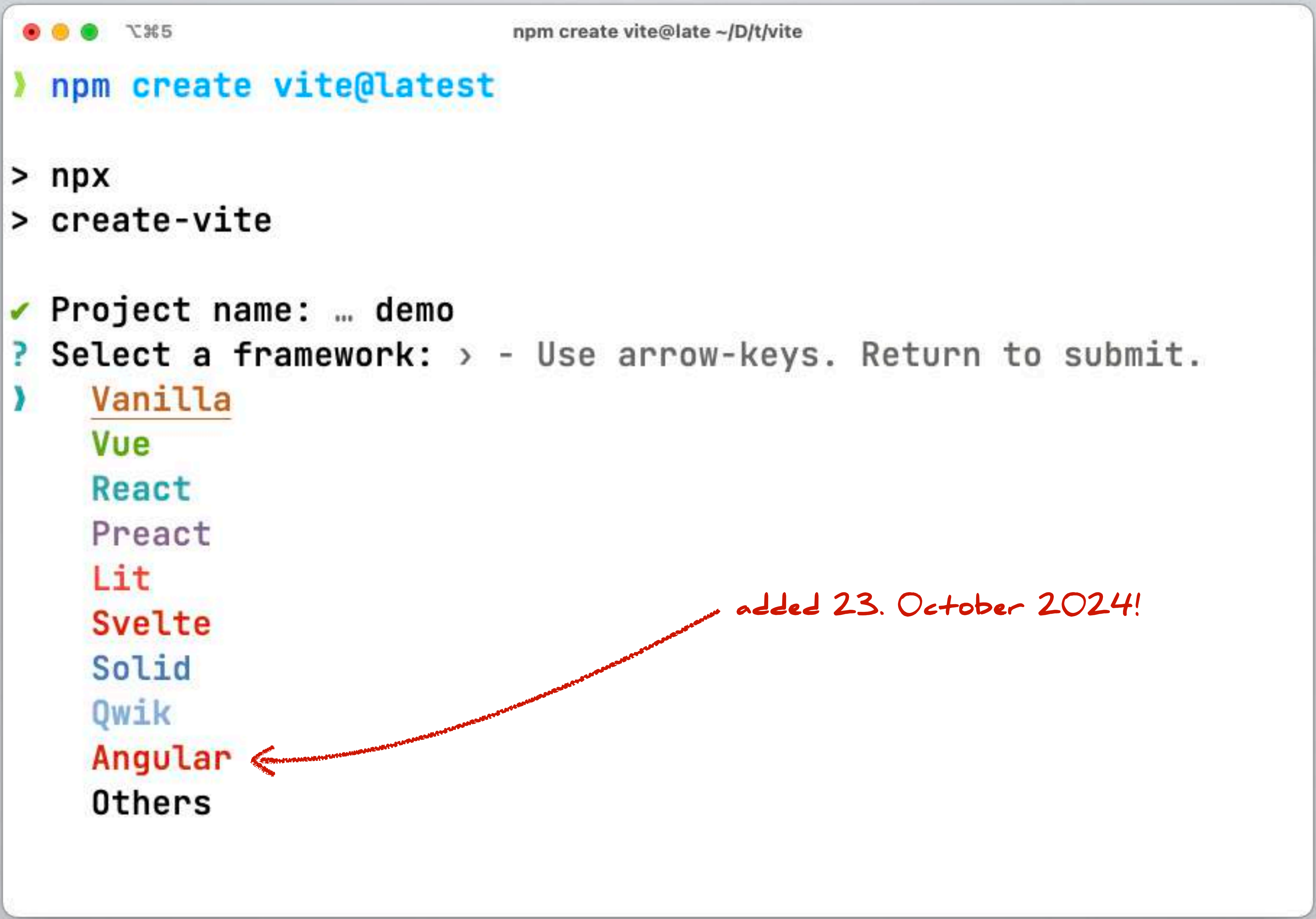
SIGNALS

https://github.com/tc39/proposal-signals

Vite for Build-Tooling

"The Build Tool for The Web"

https://vite.dev/



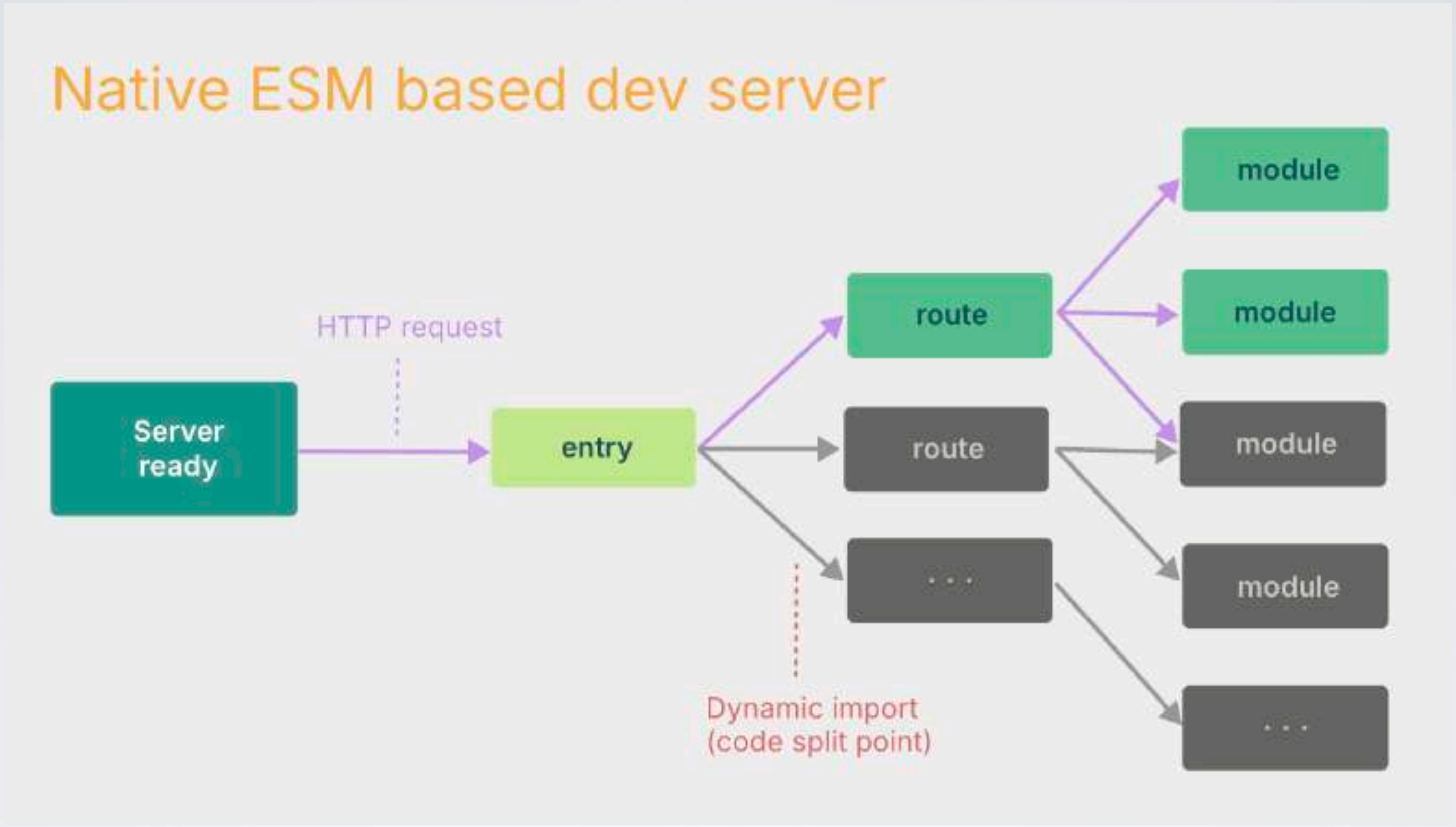
```
npm create vite@late ~/D/t/vite

❯ npm create vite@latest

> npx
> create-vite

✔ Project name: … demo
? Select a framework: › - Use arrow-keys. Return to submit.
❯   Vanilla
    Vue
    React
    Preact
    Lit
    Svelte
    Solid
    Qwik
    Angular
    Others
```

The unbundled development workflow



The browser takes over the job of the bundler by loading code as esm modules. Vite just transforms single resources on demand.

Vite uses esbuild and rollup for the production build.

https://vite.dev/guide/why.html

Vite is the default tooling for most modern frontend framework setups:

PREACT

Remix

RedwoodJS

TanStack Start

Nuxt

Analog

qwik

SVELTE KIT

SVELTE

SOLIDSTART

SOLIDJS

https://viteconf.org/

Alternative Modern Frontend Build Toolchains:

TURBOPACK

https://turbo.build/
https://areweturboyet.com/

Rspack

https://rspack.dev/

`void(0)` - Next Generation Toolchain for JavaScript

Evan You - October 1, 2024:

I have founded VoidZero Inc., a company dedicated to building an open-source, high-performance, and unified development toolchain for the JavaScript ecosystem. We have raised $4.6 million in seed funding.

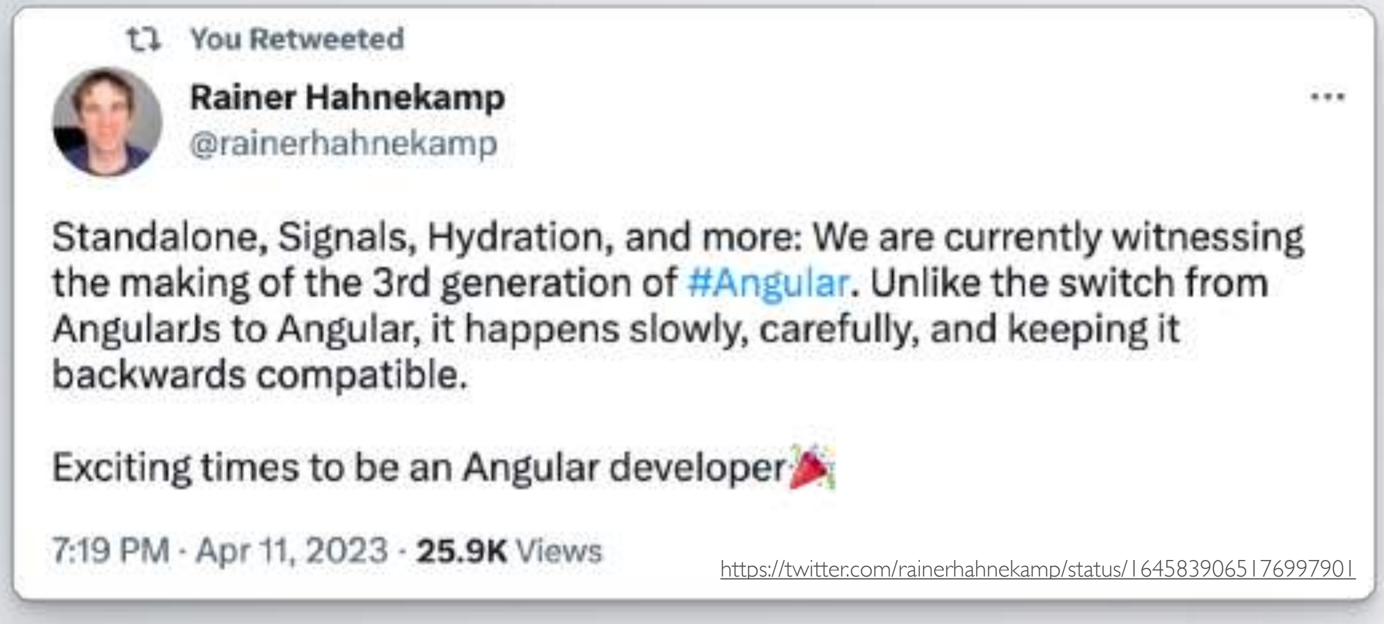
Vite Vitest Oxc Rolldown

https://voidzero.dev/posts/announcing-voidzero-inc
https://www.youtube.com/watch?v=EKvvptbTx6k

https://voidzero.dev/

Latest Development in specific Frameworks

Angular is currently changing (radically?)



You Retweeted

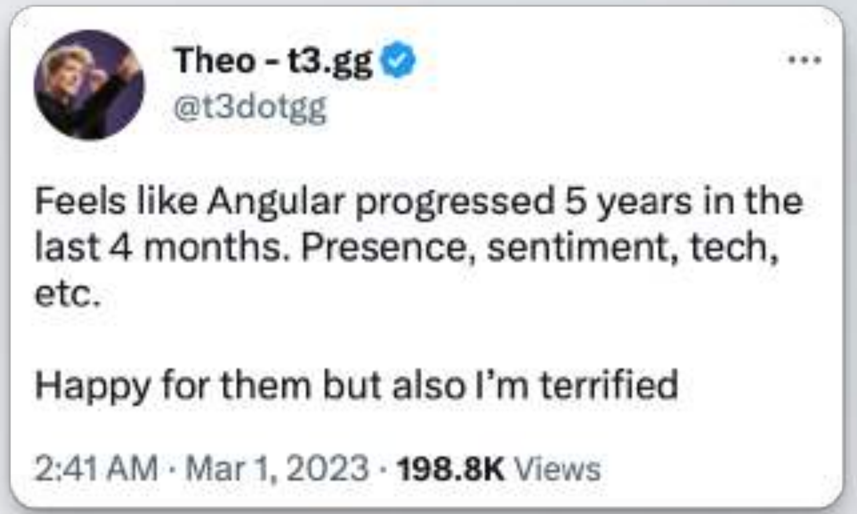
Rainer Hahnekamp
@rainerhahnekamp

Standalone, Signals, Hydration, and more: We are currently witnessing the making of the 3rd generation of #Angular. Unlike the switch from AngularJs to Angular, it happens slowly, carefully, and keeping it backwards compatible.

Exciting times to be an Angular developer🎉

7:19 PM · Apr 11, 2023 · **25.9K** Views

https://twitter.com/rainerhahnekamp/status/1645839065176997901



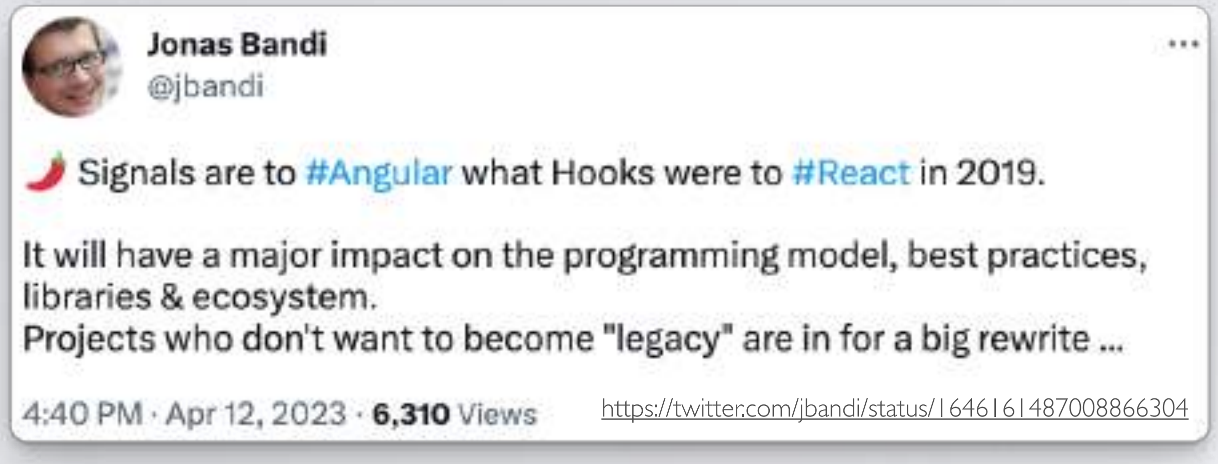
Theo - t3.gg
@t3dotgg

Feels like Angular progressed 5 years in the last 4 months. Presence, sentiment, tech, etc.

Happy for them but also I'm terrified

2:41 AM · Mar 1, 2023 · **198.8K** Views

https://twitter.com/t3dotgg/status/1630744988823982082



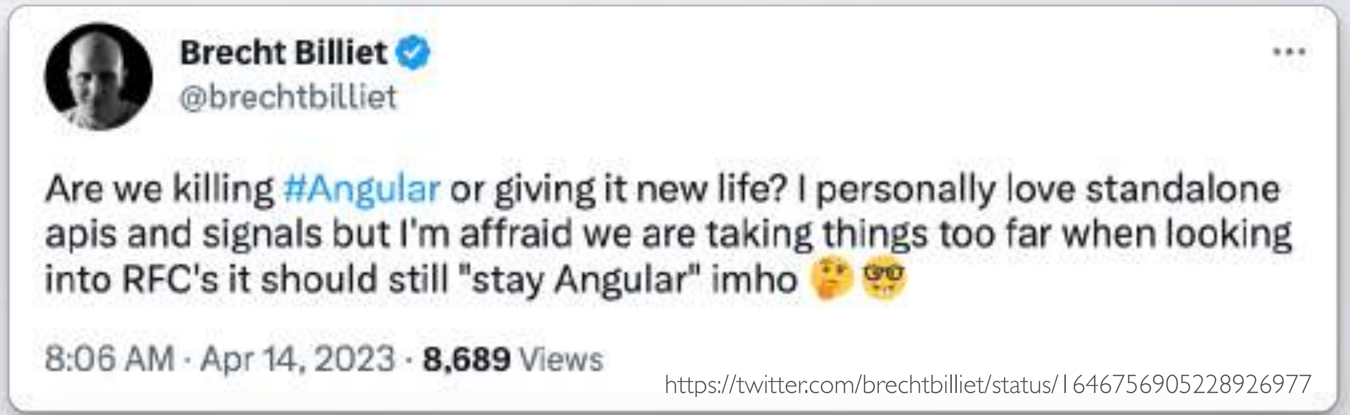
Jonas Bandi
@jbandi

🌶️ Signals are to #Angular what Hooks were to #React in 2019.

It will have a major impact on the programming model, best practices, libraries & ecosystem.
Projects who don't want to become "legacy" are in for a big rewrite ...

4:40 PM · Apr 12, 2023 · **6,310** Views

https://twitter.com/jbandi/status/1646161487008866304

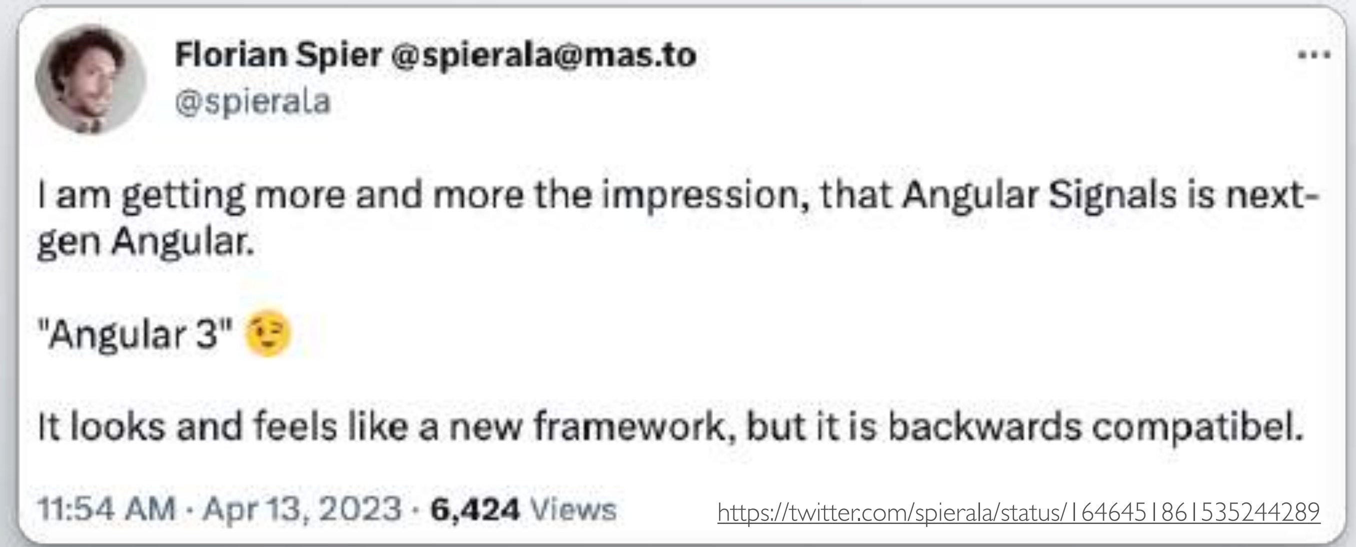


Brecht Billiet
@brechtbilliet

Are we killing #Angular or giving it new life? I personally love standalone apis and signals but I'm affraid we are taking things too far when looking into RFC's it should still "stay Angular" imho 🤔🤓

8:06 AM · Apr 14, 2023 · **8,689** Views

https://twitter.com/brechtbilliet/status/1646756905228926977



Florian Spier @spierala@mas.to
@spierala

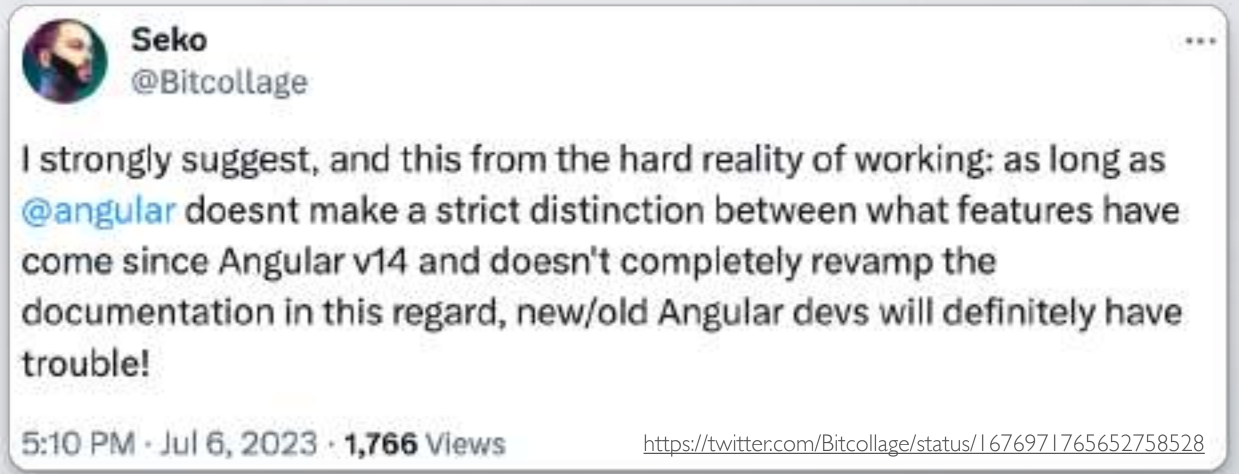
I am getting more and more the impression, that Angular Signals is next-gen Angular.

"Angular 3" 😉

It looks and feels like a new framework, but it is backwards compatibel.

11:54 AM · Apr 13, 2023 · **6,424** Views

https://twitter.com/spierala/status/1646451861535244289



Seko
@Bitcollage

I strongly suggest, and this from the hard reality of working: as long as @angular doesnt make a strict distinction between what features have come since Angular v14 and doesn't completely revamp the documentation in this regard, new/old Angular devs will definitely have trouble!

5:10 PM · Jul 6, 2023 · **1,766** Views

https://twitter.com/Bitcollage/status/1676971765652758528

https://github.com/angular/angular/discussions/49685

Angular 17 - The Angular Renaissance

November 6
10am Pacific

goo.gle/angular-event

Say hello {{again}}
to Angular

https://www.youtube.com/live/Wq6GpTZ7AX0?si=tD6RO6rFQvZMNW5O

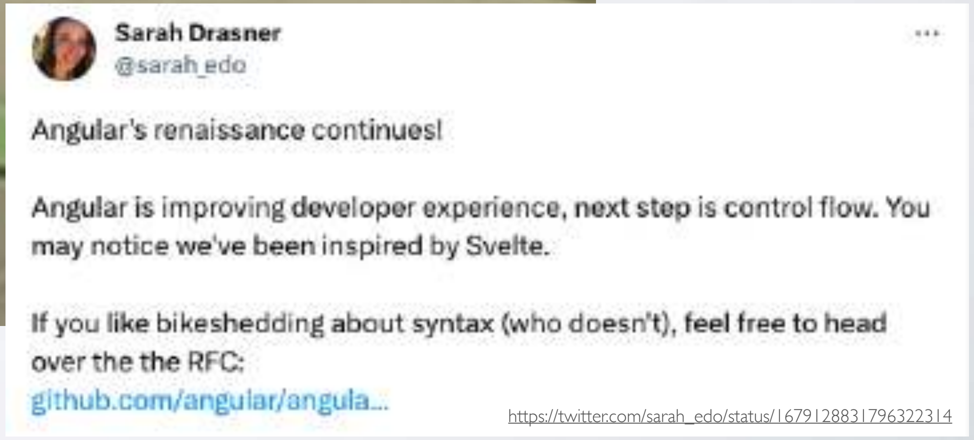
New Homepage:
https://angular.dev/

New Logo:

Angular is changing ...

"The Angular Renaissance"

https://twitter.com/Enea_Jahollari/status/1669008447042473985



Sarah Drasner
@sarah_edo

Angular's renaissance continues!

Angular is improving developer experience, next step is control flow. You may notice we've been inspired by Svelte.

If you like bikeshedding about syntax (who doesn't), feel free to head over the the RFC:
github.com/angular/angula...

https://twitter.com/sarah_edo/status/1679128831796322314

Angular merges with Wiz

WIZ is an internal framework at Google that powers consumer apps like Search, Meet, Photos, Youtube ...

https://blog.angular.dev/angular-and-wiz-are-better-together-91e633d8cd5a
https://twitter.com/sarah_edo/status/1770478763253379488
https://www.youtube.com/watch?v=A0RQ0RCNF2U

Modern Angular

Standalone Components	introduced in v14	Simplifying the structure of Angular projects.	Getting rid of `NgModule`.
`inject()` function	introduced in v14	New mechanism for dependency injection.	No more constructor injection. "Decoupling" DI from class syntax.
Control Flow (template syntax)	introduced in v17	New template syntax: `@if`, `@for` ...	Getting rid of structural directives `*ngIf`, `*ngFor`
Signals (state & reactivity)	developer preview, introduced in v16	New primitives for modeling reactive state. Eventually enabling "fine-grained" reactivity.	State is explicitely modeled with Angular constructs. It is no longer "just JavaScript".
Making RxJS optional	future	Avoiding the complexity of RxJS for simple scenarios.	Offering alternatives to observable-based APIs.

https://angular.dev/roadmap#explore-modern-angular

inject() added in v14

https://angular.io/api/core/inject

```
export class ToDoScreenComponent implements OnInit {
  constructor(private todoService: TodoService) {
  }
  ...
}
```

```
export class ToDoScreenComponent implements OnInit {
  private todoService = inject(ToDoService);
  ...
}
```

```
@Injectable({ providedIn: 'root' })
export class TodoService {
  constructor(private http: HttpClient) {}
  ...
}
```

```
@Injectable({ providedIn: 'root' })
export class TodoService {
  private http = inject(HttpClient);
  ...
}
```

The `inject` function only works during "constructor phase"!
- during constructor call
- during field initialization

- enables "non-class" scenarios (i.e. functional router guards)
- Can be combined with ECMAScript private fields!
- Better compatibility with ECMAScript.

https://codereacter.medium.com/why-angular-14s-new-inject-function-is-so-amazing-ac281e7148d1
https://angular-buch.com/blog/2022-11-use-define-for-class-fields

DI Functions with **`inject()`**

`inject()` enables sharing functionality via functional composition

```
import { inject } from '@angular/core';
import { ActivatedRoute } from '@angular/router';

export function getRouteQueryParam(key: string) {
  return inject(ActivatedRoute).snapshot.queryParams[key];
}
```

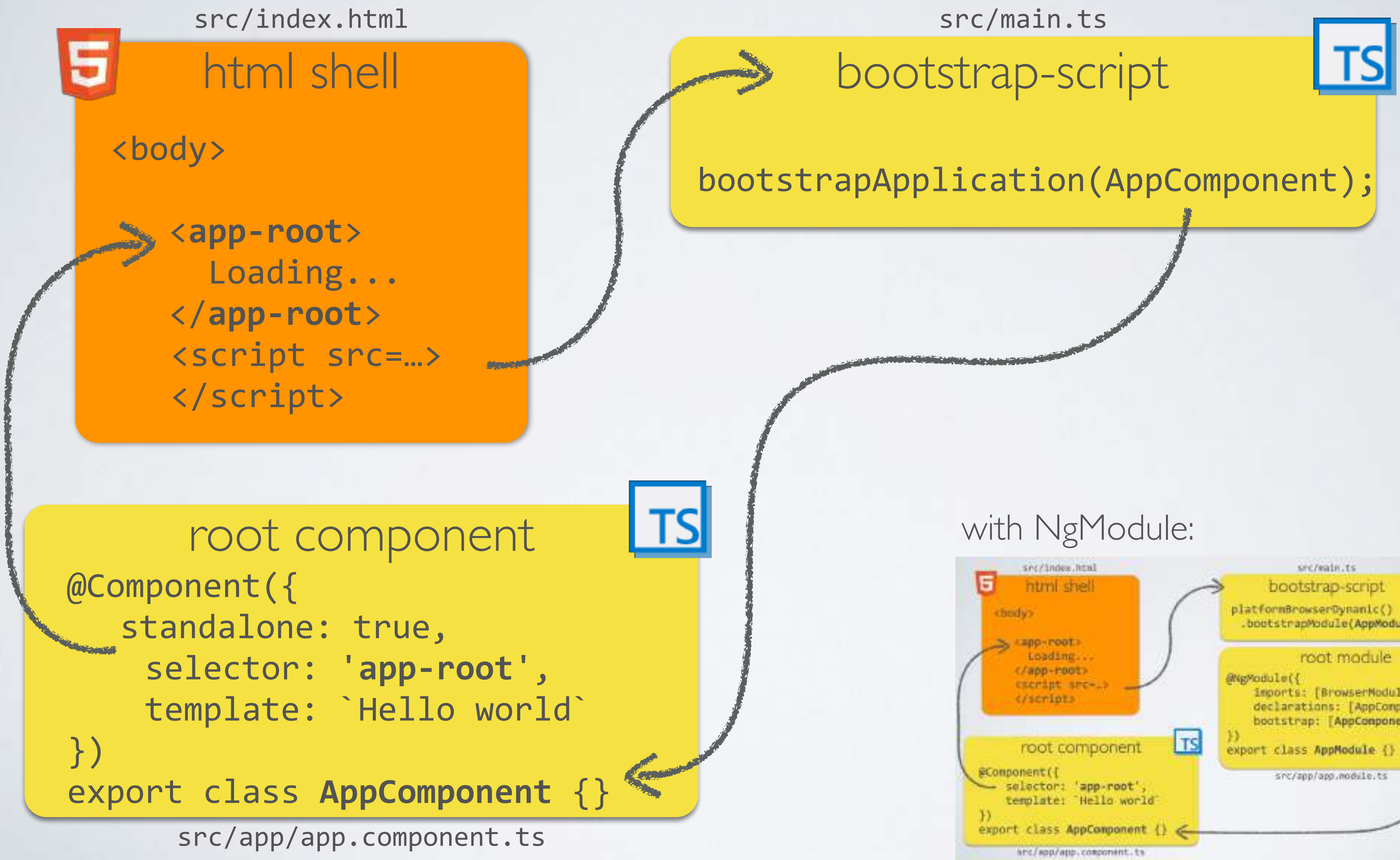
```
@Component({
  selector: 'app-todo-page',
  templateUrl: './todo-page.component.html',
})
export class TodoPageComponent {
  id = getRouteQueryParam('id');

  ngOnInit(){
    console.log('Id Query Parameter', this.id);
  }
}
```

https://netbasal.com/unleash-the-power-of-di-functions-in-angular-2eb9f2697d66

Standalone Components

introduced in v15 / default in CLI v17 / default in v19



src/index.html

html shell

```html
<body>

    <app-root>
      Loading...
    </app-root>
    <script src=…>
    </script>
```

src/main.ts

bootstrap-script

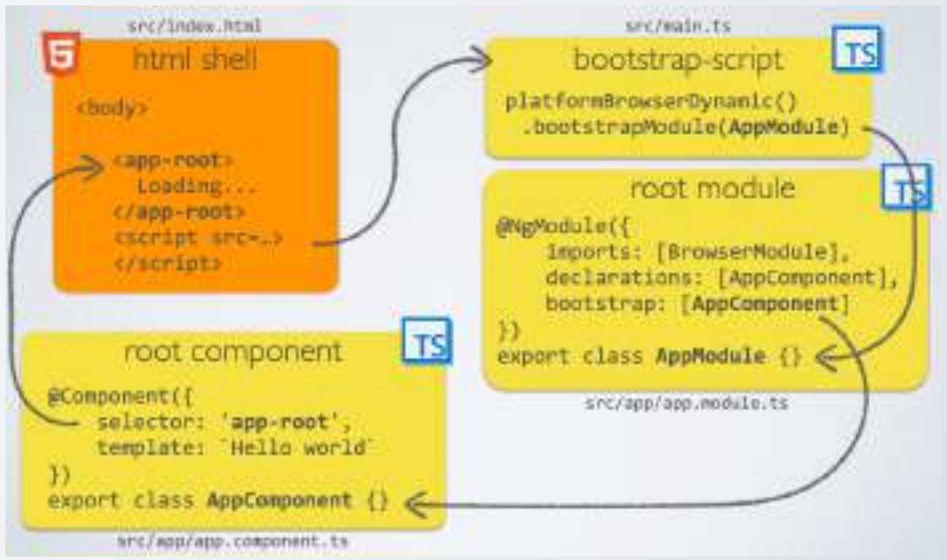
```ts
bootstrapApplication(AppComponent);
```

root component

```ts
@Component({
   standalone: true,
    selector: 'app-root',
    template: `Hello world`
})
export class AppComponent {}
```

src/app/app.component.ts

with NgModule:



src/index.html

html shell

```html
<body>

  <app-root>
    Loading...
  </app-root>
  <script src=…>
  </script>
```

src/main.ts

bootstrap-script

```ts
platformBrowserDynamic()
  .bootstrapModule(AppModule)
```

root module

```ts
@NgModule({
    imports: [BrowserModule],
    declarations: [AppComponent],
    bootstrap: [AppComponent]
})
export class AppModule {}
```

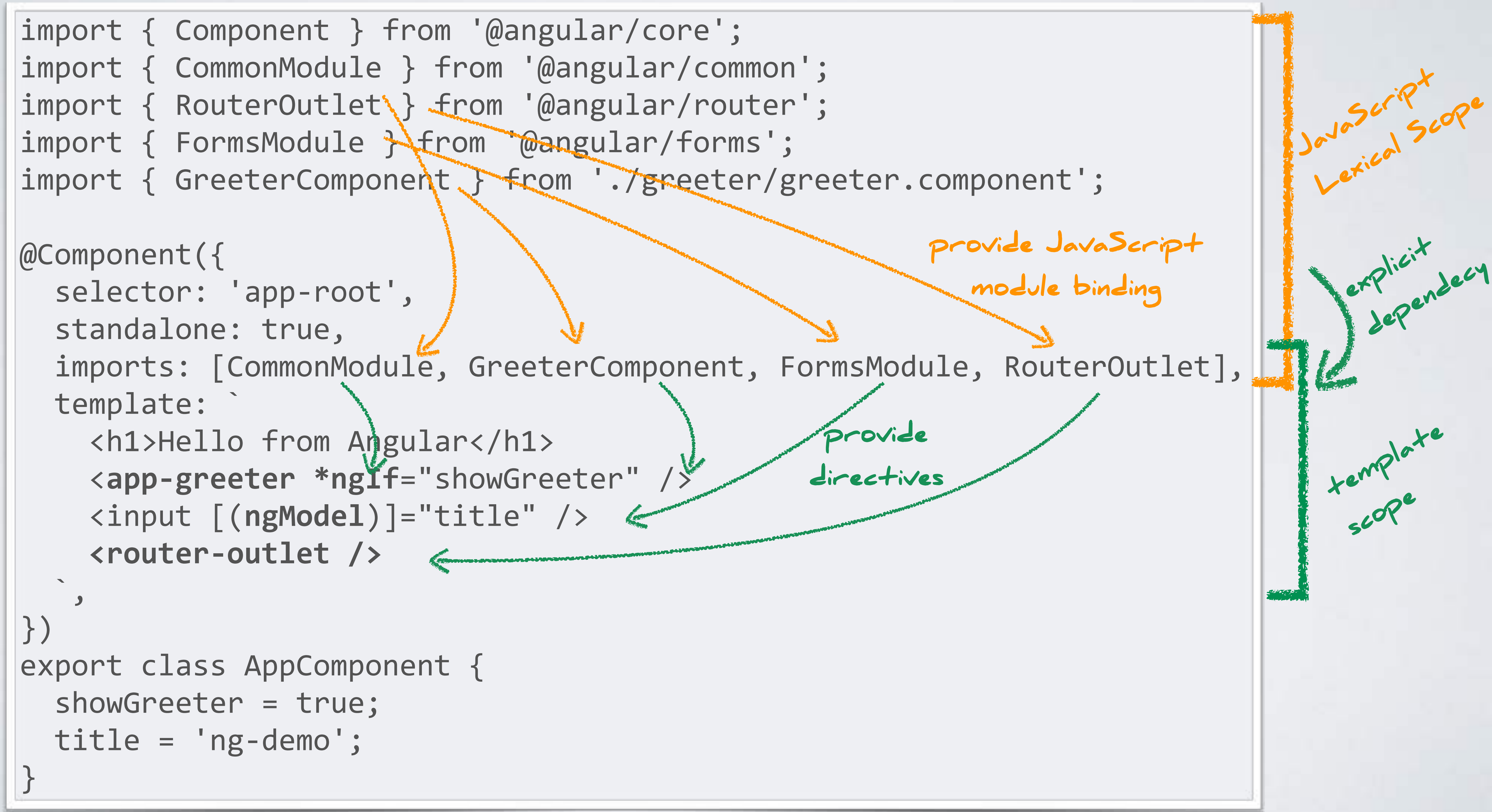
src/app/app.module.ts

root component

```ts
@Component({
   selector: 'app-root',
   template: `Hello world`
})
export class AppComponent {}
```

src/app/app.component.ts

Standalone Template Context

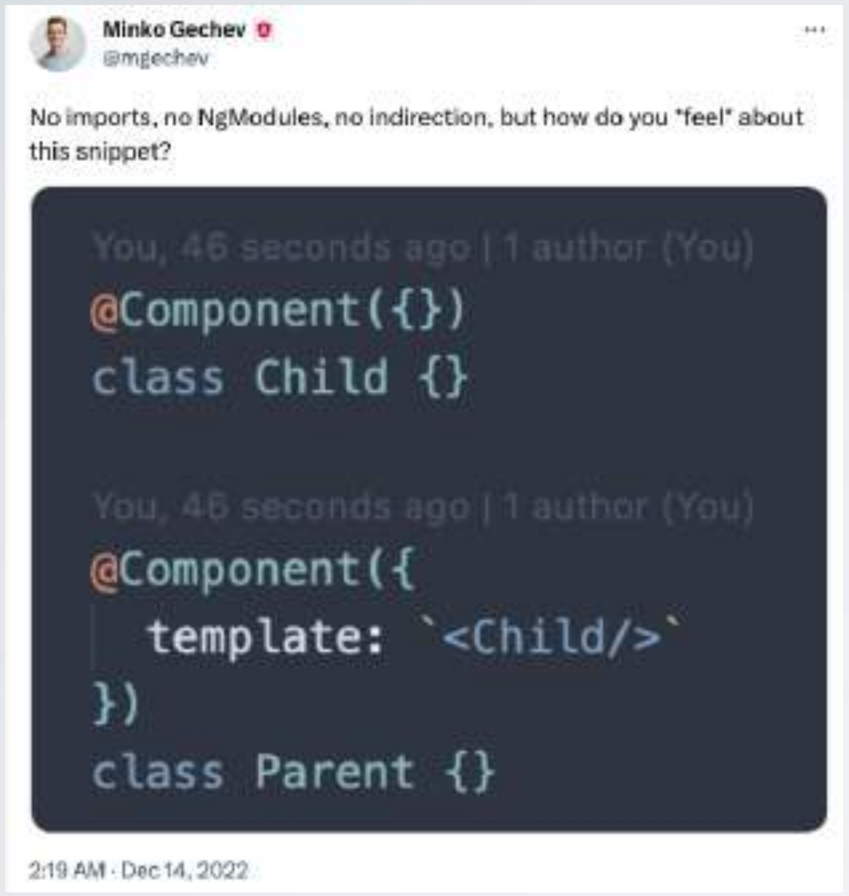


```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterOutlet } from '@angular/router';
import { FormsModule } from '@angular/forms';
import { GreeterComponent } from './greeter/greeter.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule, GreeterComponent, FormsModule, RouterOutlet],
  template: `
    <h1>Hello from Angular</h1>
    <app-greeter *ngIf="showGreeter" />
    <input [(ngModel)]="title" />
    <router-outlet />
  `,
})
export class AppComponent {
  showGreeter = true;
  title = 'ng-demo';
}
```

Next Step: Lexical Scope in Templates?

... this is not officially planned!



Minko Gechev
@mgechev

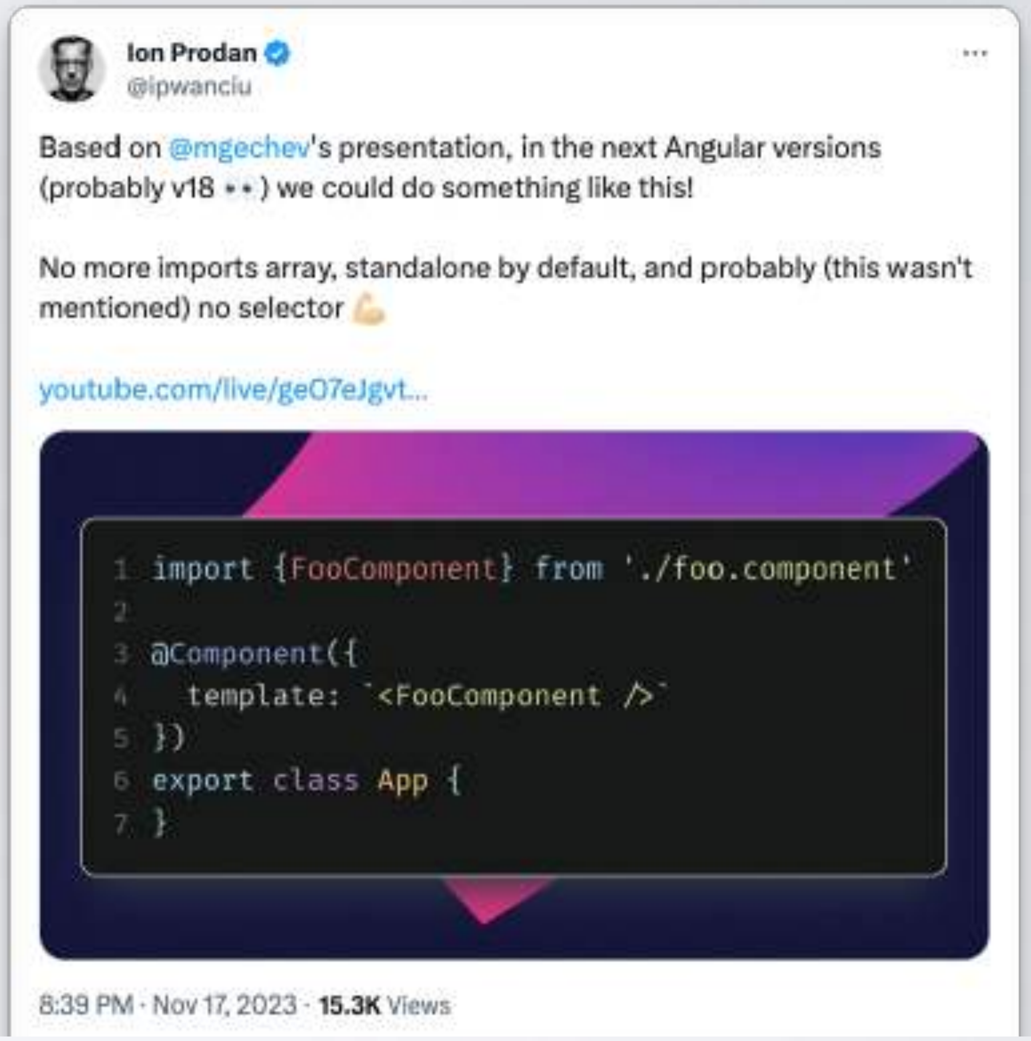
No imports, no NgModules, no indirection, but how do you "feel" about this snippet?

```
You, 46 seconds ago | 1 author (You)
@Component({})
class Child {}

You, 46 seconds ago | 1 author (You)
@Component({
  template: `<Child/>`
})
class Parent {}
```

2:19 AM · Dec 14, 2022

https://twitter.com/mgechev/status/1602835672041017344



Ion Prodan
@ipwanciu

Based on @mgechev's presentation, in the next Angular versions (probably v18 👀) we could do something like this!

No more imports array, standalone by default, and probably (this wasn't mentioned) no selector 💪

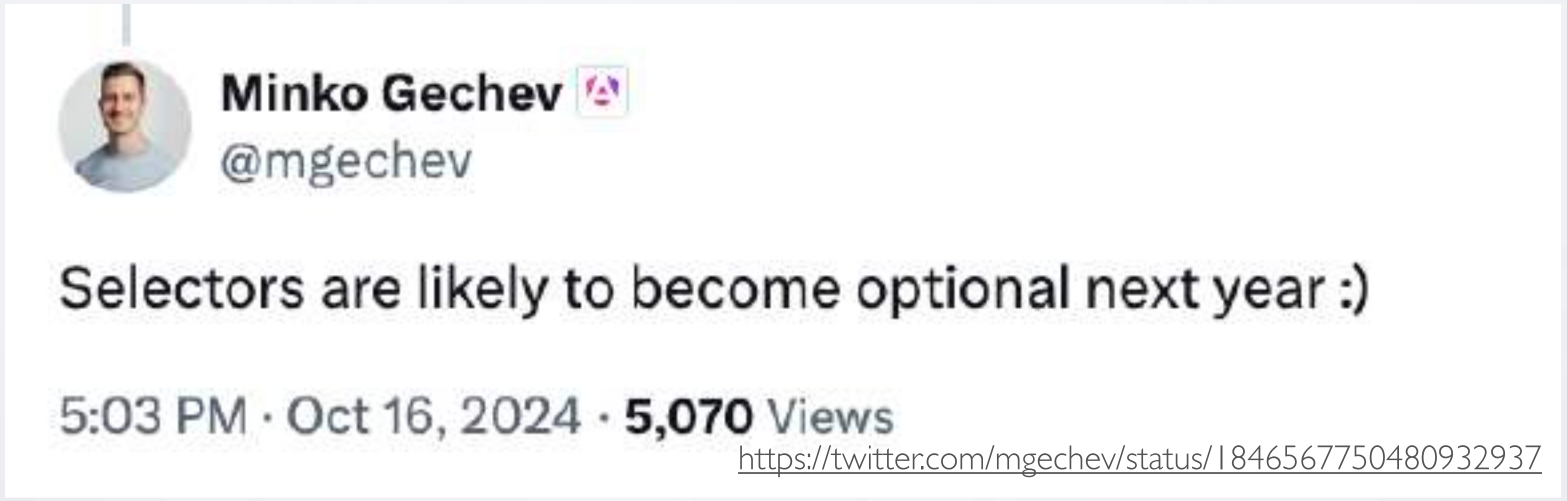
youtube.com/live/geO7eJgvt...

```
import {FooComponent} from './foo.component'

@Component({
  template: `<FooComponent />`
})
export class App {
}
```

8:39 PM · Nov 17, 2023 · **15.3K** Views

https://twitter.com/ipwanciu/status/1725599609546784925



Minko Gechev
@mgechev

Selectors are likely to become optional next year :)

5:03 PM · Oct 16, 2024 · **5,070** Views

https://twitter.com/mgechev/status/1846567750480932937

Control-Flow

Angular templates support control flow blocks that let you conditionally show, hide, and repeat elements.

```
<h2>IF..ELSE Statement</h2>
@if (show) {
  <span>Inside if</span>
} @else if (showAnotherIf) {
  <span>Inside else if</span>
} @else {
  <span>Inside else</span>
}
```

```
<h2>SWITCH CASE</h2>
@switch (caseNo) {
  @case (1) {
    <span>Rendering case 1</span>
  }
  @case (2) {
    <span>Rendering case 2</span>
  }
  @default {
    <span>Rendering default</span>
  }
}
```

```
<h2>FOR LOOP</h2>
<ul>
  @for (item of skills;
        track item.id;
        let i = $index, f = $first,
        l = $last, ev = $even,
        o = $odd, co = $count) {
    <li>{{item}}
      <ul>
        <li>Index - {{i}}</li>
        <li>Is First - {{f}}</li>
        <li>Is Last - {{l}}</li>
        <li>Is even - {{ev}}</li>
        <li>Is odd - {{o}}</li>
        <li>Count - {{co}}</li>
      </ul>
    </li>
  } @empty {
    <li>No item</li>
  }
</ul>
```

optional contextual variables

https://angular.dev/guide/templates/control-flow

Structural Directives

Structural directives shape or reshape the DOM's structure.
The host element of the directive is used as a template that is instantiated by Angular.

The three common, built-in structural directives:
`*ngFor, *ngIf, *ngSwitchCase`

```
<ul>
  <li *ngFor="let character of characters">
    <span>{{character.firstName}}</span>
    <span>{{character.lastName}}</span>
    <span *ngIf="showDistrict">{{character.district}}</span>
    <span [ngSwitch]="character.emotion">
      <span *ngSwitchCase="Emotion.InLove">❤️</span>
      <span *ngSwitchCase="Emotion.Angry">🔪</span>
      <span *ngSwitchCase="Emotion.Sad">💧</span>
      <span *ngSwitchDefault>⁉️</span>
    </span>
  </li>
</ul>
```

https://angular.io/guide/structural-directives

*ngIf & else

```
<span *ngIf="character.isInLove; else elseBlock">❤️</span>
<ng-template #elseBlock>⁉️</ng-template>
```

```
<span *ngIf="character.isInLove; then thenBlock; else elseBlock"></span>
<ng-template #thenBlock>❤️</ng-template>
<ng-template #elseBlock>⁉️</ng-template>
```

https://angular.io/api/common/NgIf

Deferrable Views

(added in v17)

Lazy Loading "done right"

```
@defer {
  <large-component />
} @loading (after 100ms; minimum 1s) {
  <img alt="loading..." src="loading.gif" />
} @placeholder (minimum 500ms) {
  <p>Placeholder content</p>
} @error {
  <p>Failed to load the component</p>
}
```

A powerful API with triggers, conditions and prefetching:

```
@defer(on hover){ ...
```

```
@defer(on viewport) { ...
```

```
@defer (on interaction; prefetch on idle) { ...
```

https://angular.dev/guide/defer#defer

https://angularexperts.ch/blog/angular-defer-lazy-loading-total-guide

Signals for State

In Angular its easy to model state with JavaScript class properties.
This is the "hands-off" approach to reactivity.
In modern Angular it is recommended to model state explicitly with Signals instead. This is the new reactivity system of Angular.

template

```
Greetings: {{name()}}
<div [innerText]="name()"></div>
<input #nameInput (input)="onNameChange(nameInput.value)"/>
```

```
@Component(...)
export class GreeterComponent {
  name = signal('Tyler Durden');
  onNameChange(name: string){
      this.name.set(name);
  }
}
```

component

```
@Component({
    selector: 'app-greeter',
    standalone: true,
    imports: [FormsModule],
    template: `
          <input [(ngModel)]="name"/>
          Greetings: {{name()}}
      `
})
export class GreeterComponent {
  name = signal('John');
}
```

https://angular.dev/guide/signals
https://www.youtube.com/watch?v=CoZD3fyMAtk

Angular Signals

https://angular.dev/guide/signals

```
@Component({
   template: `<div>fullName()</div>`
})
export class App {
  firstName = signal('Jane');
  lastName = signal('Doe');

  fullName = computed(() => `${this.firstName()} ${this.lastName()}`);

  constructor() {
    effect(() => console.log('Name changed:', this.fullName()));
  }

...
```

```
signal.set(newVal);
signal.update(val => newVal);
```

```
const signal = signal.asReadOnly();
```

Angular Signals

"New Reactive Primitives"

A new programming model for managing state:
- new data flow concepts
- proper concept for derived state

Superior technical implementation of "reactivity"
- reactive state instead of change detection
- fine-grained reactivity

The Impact of Signals

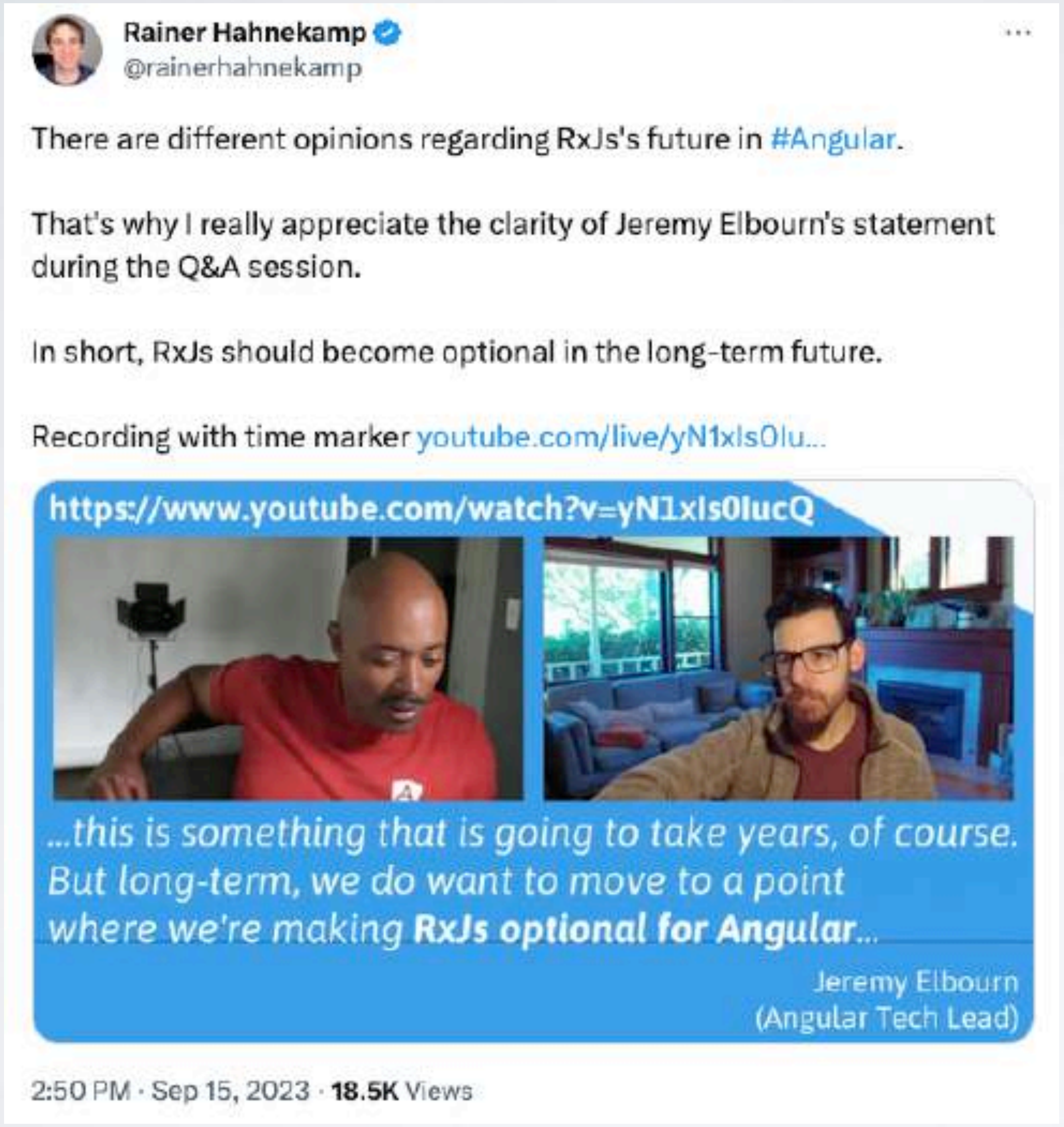
"rethinking best-practices in Angular"

- signals will replace many usage scenarios of RxJS
- signals make the `async` pipe obsolete
- signals will make `ngOnChanges` obsolete
- signals will make Zone.js obsolete
- signals will make `OnPush` change detection obsolete
- signals might be changing the concept of *unidirectional dataflow* ...

Angular Signals Ecosystem

- Angular will expose Signals via its APIs
- More APIs: linkedSignal, deepSignal
- Libraries will be built around Signals
  - https://ngrx.io/guide/signals/signal-store
  - https://github.com/timdeschryver/ng-signal-forms

The future of RxJs in Angular?



Rainer Hahnekamp
@rainerhahnekamp

There are different opinions regarding RxJs's future in #Angular.

That's why I really appreciate the clarity of Jeremy Elbourn's statement during the Q&A session.

In short, RxJs should become optional in the long-term future.

Recording with time marker youtube.com/live/yN1xIs0lu...

https://www.youtube.com/watch?v=yN1xIs0lucQ

...this is something that is going to take years, of course. But long-term, we do want to move to a point where we're making ***RxJs optional for Angular...***

Jeremy Elbourn
(Angular Tech Lead)

2:50 PM · Sep 15, 2023 · **18.5K** Views

https://twitter.com/rainerhahnekamp/status/1702666071977980413

"Signals for state.
RxJs for async
scenarios"

https://www.youtube.com/watch?v=yN1xIs0lucQ&t=2090s

Signals RxJS Interop

https://angular.dev/guide/signals/rxjs-interop

```
counterObservable = interval(1000);

// Get a signal representing the value of the observable
counter = toSignal(this.counterObservable, {initialValue: 0});
```

```
query = Signal('test');

// Get an observable that tracks the value of the signal
query$ = toObservable(this.query);
```

The Resource API

will be in v19

Async Loading with Signals

```
todosResource = resource({
    request: this.searchCriteria,
    loader: (param) => {
      return this.#todoService.find(param.request);
    }
});


todos = computed(() => this.todosResource.value() ?? []);
```

https://riegler.fr/blog/2024-10-18-resources-as-signals
https://push-based.io/article/everything-you-need-to-know-about-the-resource-api
https://www.angulararchitects.io/en/blog/asynchronous-resources-with-angulars-new-resource-api/

Migrating to Modern Angular

https://angular.dev/reference/migrations

```
ng generate @angular/core:standalone
```

```
@Component({
  selector: 'greeter',
  template: '<div *ngIf="showGreeting">Hello</div>',
})
export class GreeterCompon
  showGreeting = true;
}
```

automatic
migration

```
@Component({
  selector: 'greeter',
  template: '<div *ngIf="showGreeting">Hello</div>',
  standalone: true,
  imports: [NgIf]
})
export class GreeterComponent {
  showGreeting = true;
}
```

But wait ...
... you said no innovation in frontend frameworks recently?

Is it innovation?

The Recent History of React

- React 16.8 "Hooks" - February 2018
- React 17 - October 2020

  (no new features)
- React 18 was released in March 2022

  (concurrent rendering, transitions, streaming SSR ...)

React Hooks

introduced with React 16.8 in 2019.

https://react.dev/reference/react/hooks

Class Component:

```
class Counter1 extends React.Component {

  state = {
    count: 0
  };

  increase = () => {
    this.setState({count: this.state.count + 1})
  };

  render() {
    return (
      <div>
        <h1>Count {this.state.count}</h1>
        <button onClick={this.increase}>
           Increase
         </button>
      </div>
    );
  }
}
```

using ES2015 classes
using JSX
using proposed class fields

method derived from base class

the "old" way

Function Component with Hook:

```
function Counter2() {

  const [count, setCount] = useState(0);

  function increase() {
    setCount(count + 1);
  }

  return (
    <div>
      <h1>Count {count}</h1>
      <button onClick={increase}>Increase</button>
    </div>
  );
}
```

Hook

using JavaScript functions
using JSX

the "new" way

Note : functional components and class components can be mixed in the same application.

Motivation: https://legacy.reactjs.org/docs/hooks-intro.html

Why Hooks (2019)?

https://reactjs.org/docs/hooks-intro.html#motivation

"Hooks solve a wide variety of seemingly unconnected problems of class components."

Hooks reduce complexity in components and enable better patterns for decoupling application logic and ui-logic.

Hooks enable better structure, better reuse and better composition of application logic.

Hooks are often an elegant replacement for previous reusability patterns like "higher order components" and "render props"

Hooks are not undisputed:

- https://stevenkitterman.com/posts/the-catch-with-react-hooks/
- https://blog.logrocket.com/frustrations-with-react-hooks/
- https://dillonshook.com/a-critique-of-react-hooks/
- https://medium.com/swlh/the-ugly-side-of-hooks-584f0f8136b6
- https://medium.com/better-programming/why-react-hooks-are-the-wrong-abstraction-8a44437747c1
- https://danielrotter.at/2022/01/16/some-reasons-for-disliking-react-hooks.html

"Why React Hooks": https://www.youtube.com/watch?v=eX_L39UvZes

Why Hooks (2023)?

Because the React ecosystem has embraced Hooks.

(and React without it's ecosystem is like a car without seats, wheels, windows, pedals ...)

All modern React libraries expose their APIs via Hooks:

- ReactRouter / TanStack Router
- Jotai, Zustand, MobX, ReactRedux, Recoil ...
- TanStackQuery, SWR
- react-i18next
- ...

Full-Stack React

(aka. Meta Frameworks)

The official React documentation is recommending to use a "production-grade framework".

https://react.dev/learn/start-a-new-react-project#production-grade-react-frameworks

https://nextjs.org/

https://remix.run/

https://www.gatsbyjs.com/

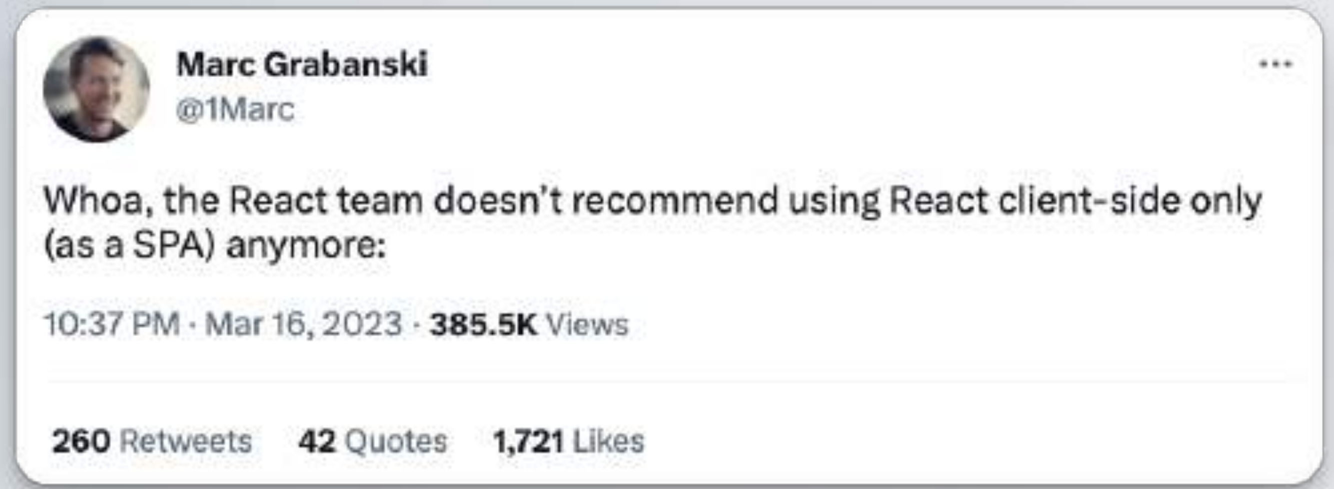
The common goal of those meta-frameworks is to simplify the project setup of frontend applications.
They include concepts for server-side-rendering, routing, data-fetching, mutations ...
But they come with their own conceptual overhead and learning curve!
These frameworks typically require running JavaScript on the server (like Node.js).

https://redwoodjs.com/

https://waku.gg/

https://tanstack.com/start/

Remix will be merged with React Router v7: https://remix.run/blog/merging-remix-and-react-router

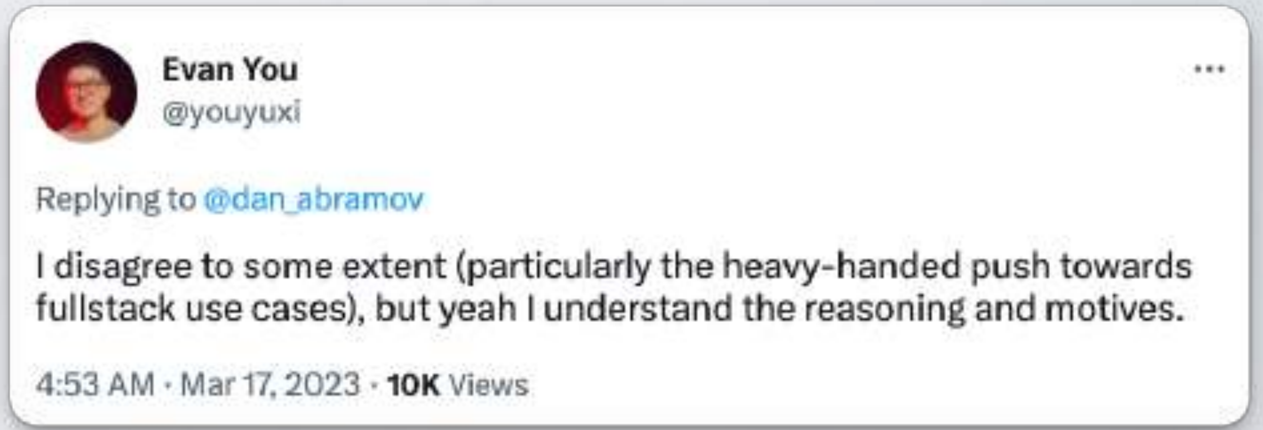

Marc Grabanski
@1Marc

Whoa, the React team doesn't recommend using React client-side only (as a SPA) anymore:

10:37 PM · Mar 16, 2023 · **385.5K** Views

260 Retweets **42** Quotes **1,721** Likes

https://twitter.com/1Marc/status/1636481900381388802

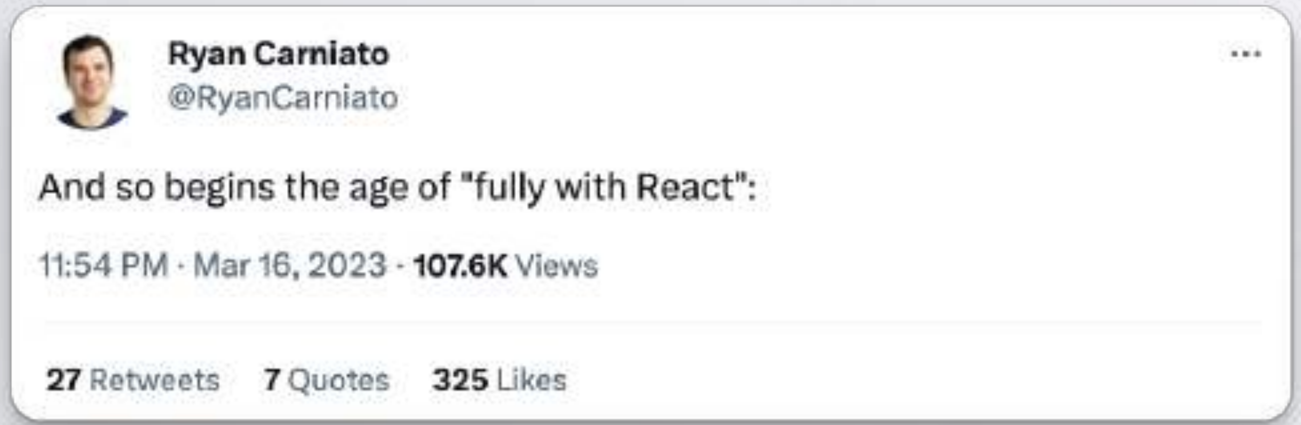

Evan You
@youyuxi

Replying to @dan_abramov

I disagree to some extent (particularly the heavy-handed push towards fullstack use cases), but yeah I understand the reasoning and motives.

4:53 AM · Mar 17, 2023 · **10K** Views

https://twitter.com/youyuxi/status/1636576506574176258

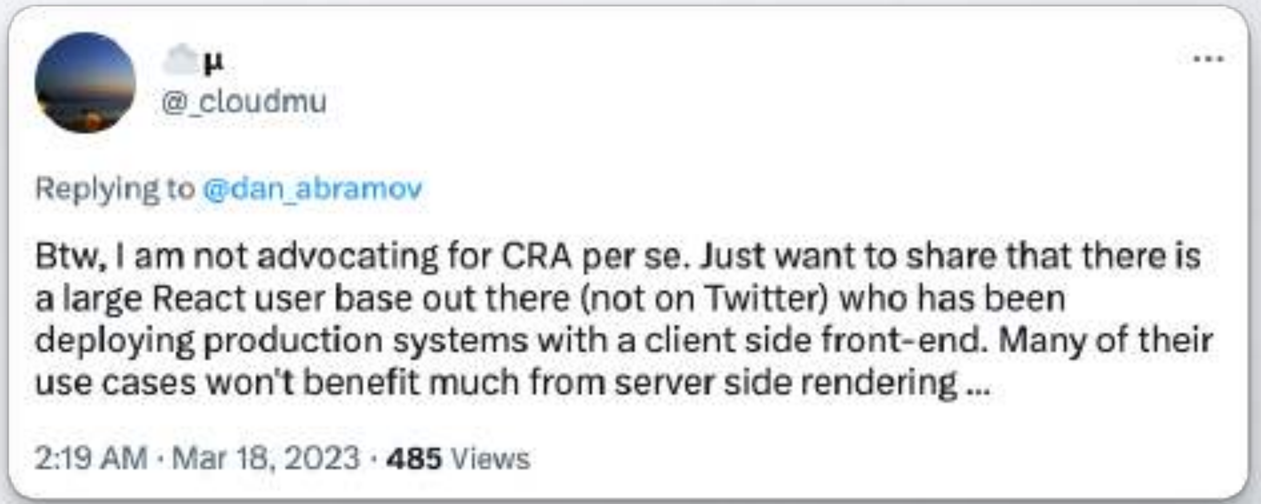

Ryan Carniato
@RyanCarniato

And so begins the age of "fully with React":

11:54 PM · Mar 16, 2023 · **107.6K** Views

27 Retweets **7** Quotes **325** Likes

https://twitter.com/RyanCarniato/status/1636501181039276032


μ
@_cloudmu

Replying to @dan_abramov

Btw, I am not advocating for CRA per se. Just want to share that there is a large React user base out there (not on Twitter) who has been deploying production systems with a client side front-end. Many of their use cases won't benefit much from server side rendering ...

2:19 AM · Mar 18, 2023 · **485** Views

https://twitter.com/_cloudmu/status/1636900018643775488

The React/Vercel Controversy

Vercel is a company that provides cloud infrastructure:
https://vercel.com/

Vercel is funding Next.js, the most popular React metaframework.

Vercel hired several React core developers.

The React documentation started to recommend "production grade frameworks".

Next.js started to use unreleased features of React.
(bundling "canary version" of React)

Next.js started promoting React Server Components, which run on the backend / in the cloud 🤔.

Next.js is not easy to self-host, some features only run on Vercel Infrastructure.

The Wild West of React

React is just a "library for creating user interfaces".

The ecosystem around React is the wild wild west.

Metaframeworks:

- Next.js
- Remix
- Redwood
- TanStack Start

Routers:

- React Router
- TanStack Router
- Chicane
- Wouter

Data Fetching Libraries:

- TanStack Query
- SWR
- Apollo
- axios
- ky
- ...

State Management Libraries:

- Zustand
- Jotai
- Legend State
- Recoil
- Redux
- MobX
- ...

Component Libraries:

- MUI
- Chakra UI
- Ant Design
- Radix UI
- shadcn
- Next UI
- ...

Styling Libraries:

- CSS Modules
- Emotion.js
- StyleX
- TSS React
- Pigment CSS
- Vanilla Extract
- Tailwind
- ...

Form Libraries:

- Formik
- Rect Hook Forms
- HouseForm
- Felte
- Conform
- ...

i18n Libraries:

- i18next
- React-intl
- Lingui
- FBT

React 19 RC was released in April 2024

... we are still waiting on the official release ...

- a concept for "Actions"
  (async, transitions, pending & error state,, optimistic updates, ...)
  - form-handling
  - event handling
  - server communication
- simplifications: use, ref passing, Context ...

- React Server Components 🤯
  (there was a first demo of RSCs in Dezember 2020)

https://react.dev/blog/2024/04/25/react-19
https://vercel.com/blog/whats-new-in-react-19

React 19 and Suspense - A Drama in 3 Acts:
https://tkdodo.eu/blog/react-19-and-suspense-a-drama-in-3-acts

React Compiler

currently in beta, available in Next v15

https://react.dev/learn/react-compiler

In the future the "React Compiler" will hopefully make manual optimization with `memo`, `useMemo` and `useCallback` obsolete ...

The result should be fine grained reactivity, similar to frameworks based on signals. But the mental model of "coarse-grained re-rendering" remains.
"UI as a function of state"

The Recent History of React

Vue 2 - September 2016
(options api)

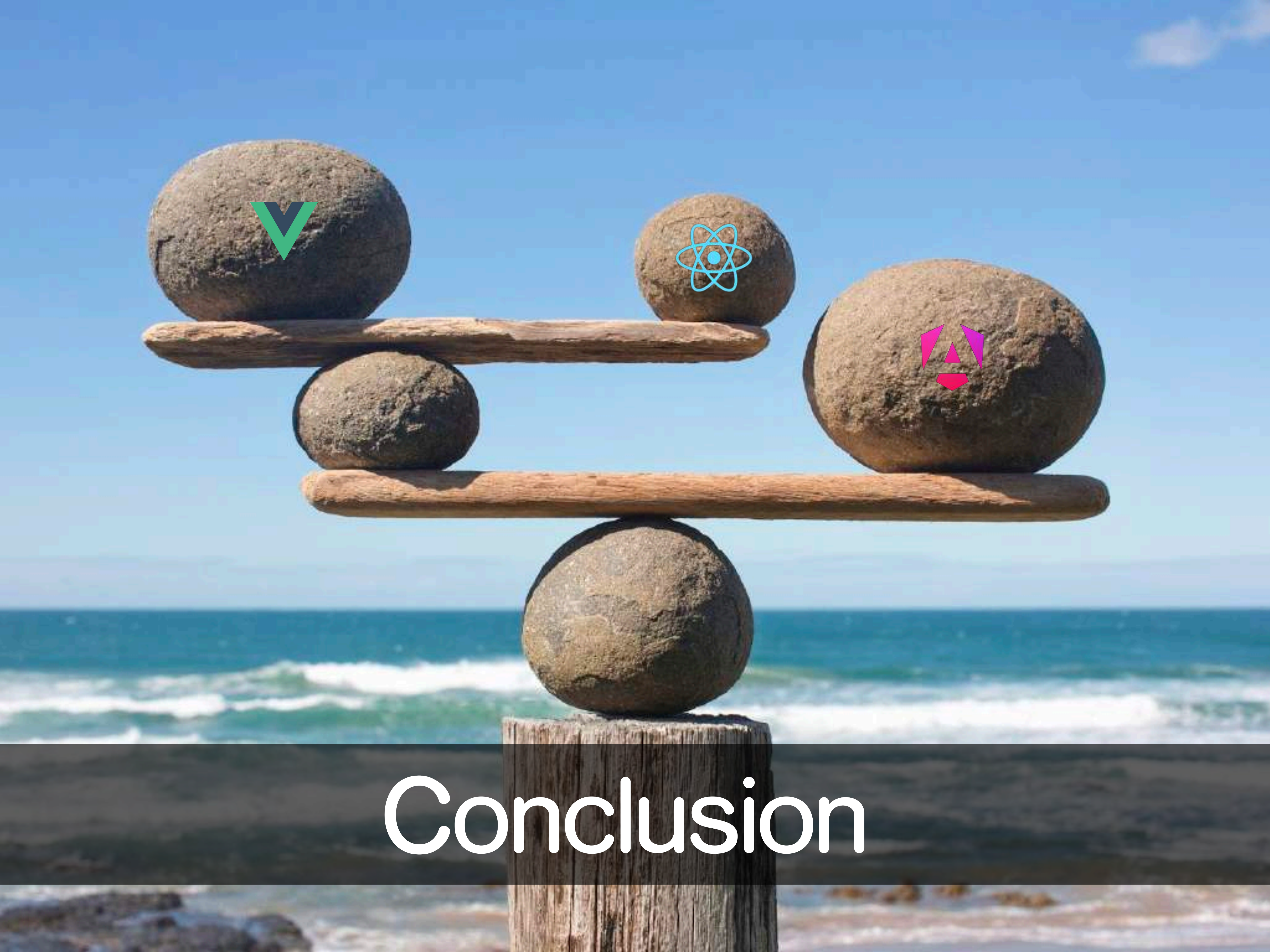
Vue 3 - September 2020
(rewrite in TypeScript, composition api)

Vue 2 End of Life - December 2023

Vue: less Hype, less Drama

It is often overlooked that:

- Vite was create in the Vue ecosystem
    - also `unjs` and `Nitro`, which are widely used outside of the Vue ecosystem
- Vue was the first mainstream framework which provided signal-style reactivity (fine grained reactivity)
- Pionieered single-file components processed by build tools
- Vue is a progressive framework enabling a wide range of usage scenarios
- Vue remains independent (i.e. not dominantly owned by a single company)



Conclusion

- Angular tries to be the "stable and reliable" framework. But right now it is in a phase of heavy changes.

- React is the wild west:
  You have to choose a framework on top of React and/or choose & maintain a stack of libraries.

- Vue is currently the most stable framework and many innovations came out of the Vue ecosystem.

Many Similarities!

There are *many similarities* between Angular, React & Vue:

Single Page Application
Component Architecture
State managed in JavaScript
Binding the DOM to JavaScript
Dataflow Architecture
Typically "complex" frontend build setup based on Node & npm
Support for TypeScript

The *main differences* are:

- The mechanism to declare the UI (template vs. render function)
- Scope of the framework/library (which functionality is "built in" and how flexible is it to combine with other libraries)
- "Reactivity" concept: change detection which triggers UI updates

High-Level Differentiation

User-Experience:

- There is no difference between Angular, React & Vue.
- From a technical point of view the frameworks are very similar (JavaScript- & DOM-based). Any project that can be implemented wit one of the frameworks can be implemented with the others.
- Performance differences are very small.

Developer-Experience:

- There are notable differences between Angular, React & Vue
  - Scope & 3rd-Party Libs, Template vs. JSX, Class vs. functional, State-Management ...

Key Differences

Templating

templates: enhancing html with dynamic constucts

separate files or inlined strings

single file components

(Vue also supports render functions with JSX)

JSX: declaring UI with code

Framework vs Library

"complete" framework with opinionated choices

small & unopinionated library providing default options for important topics (router, centralized state management)

small & unopinionated library that must be combined with other libraries

Key Differences

Component Programming Model

based on classes

functional API (composition API)
object based API is still available.

new functional API
("Hooks")

Reactivity

fine grained reactivity with
signals

coarse grained render
functions
(optimized by the React compiler)

Who do you (want to) trust?

Google

facebook®

Open Source Community

Have Fun with the Framework of your choice!

Thank you!

Slides & Code: https://github.com/jbandi/modern-web-cudos

Questions? Discussions …

Jonas Bandi
JavaScript / Angular / React / Vue / Vaadin
Schulung / Beratung / Coaching / Reviews
jonas.bandi@ivorycode.com