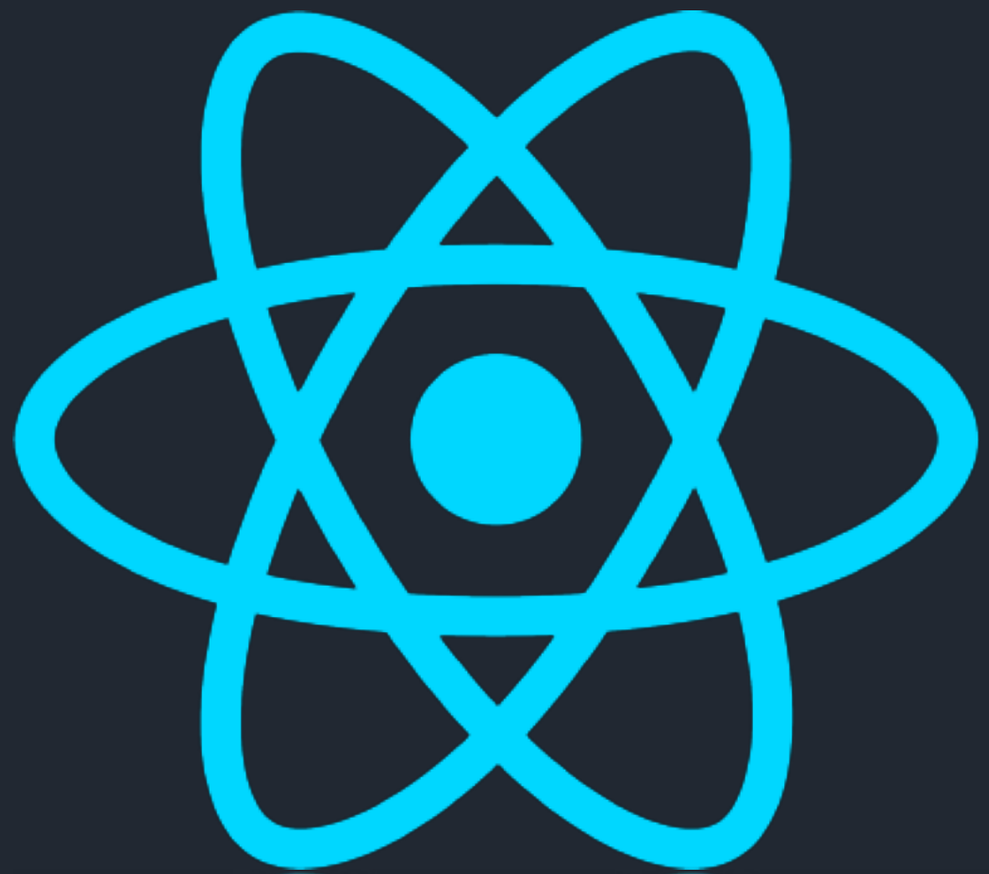
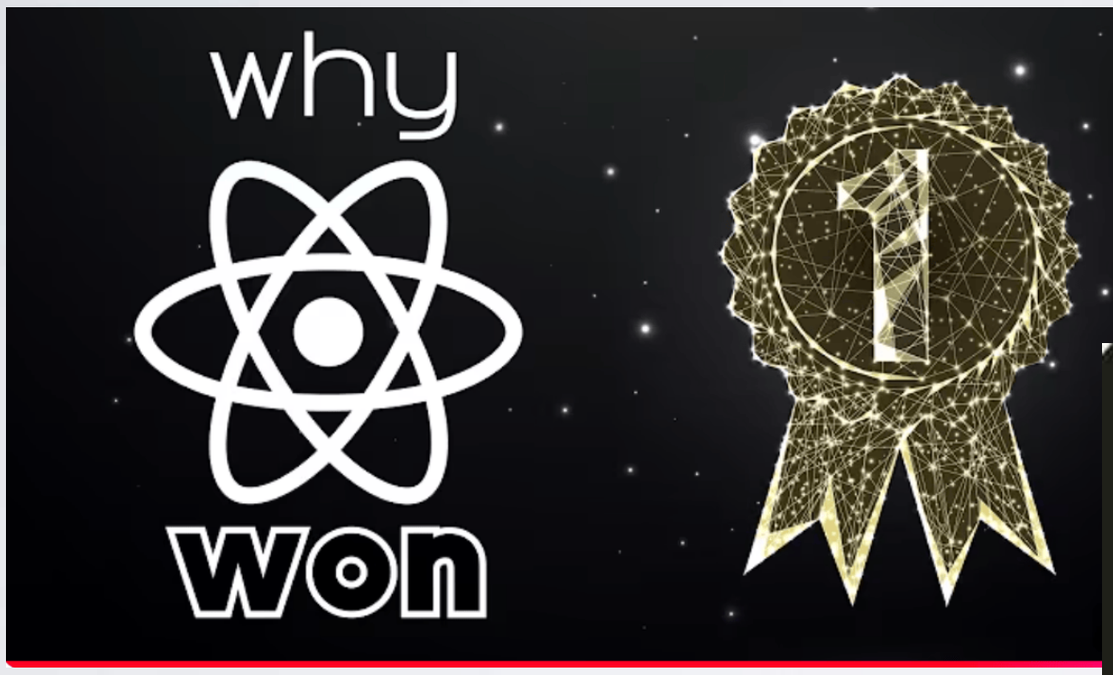


React
"The 30min Elevator Pitch"
(aka "Power Session")

jonas.bandi@ivorycode.com

React is by far the most popular Frontend Library



https://www.youtube.com/watch?v=Wm_xl7KntDs

https://www.youtube.com/watch?v=P1FLEnKZTAE

"The usefulness of a framework is proportional to its usage | community | ecosystem."

(a modern version of "Nobody ever gets fired for buying IBM")

ABOUT ME

Jonas Bandi
jonas.bandi@ivorycode.com

- Freelancer, in den letzten 12 Jahren vor allem in Projekten im Spannungsfeld zwischen modernen Webentwicklung und traditionellen Geschäftsanwendungen.
- Dozent an der Berner Fachhochschule seit 2007
- In-House Kurse & Beratungen zu Web-Technologien im Enterprise: UBS, Postfinance, Mobiliar, AXA, BIT, SBB, Elca, Adnovum, BSI ...

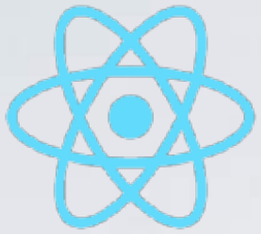
Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

letsboot.ch
swiss dev training

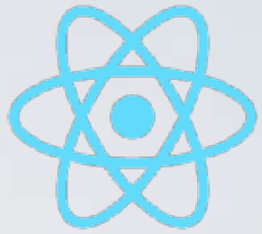
digicomp

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

jonas.bandi@ivorycode.com



AGENDA



React in the Frontend Ecosystem

React for Single Page Applications

Components - the Building Blocks of React

JSX - React Templating

The Virtual DOM

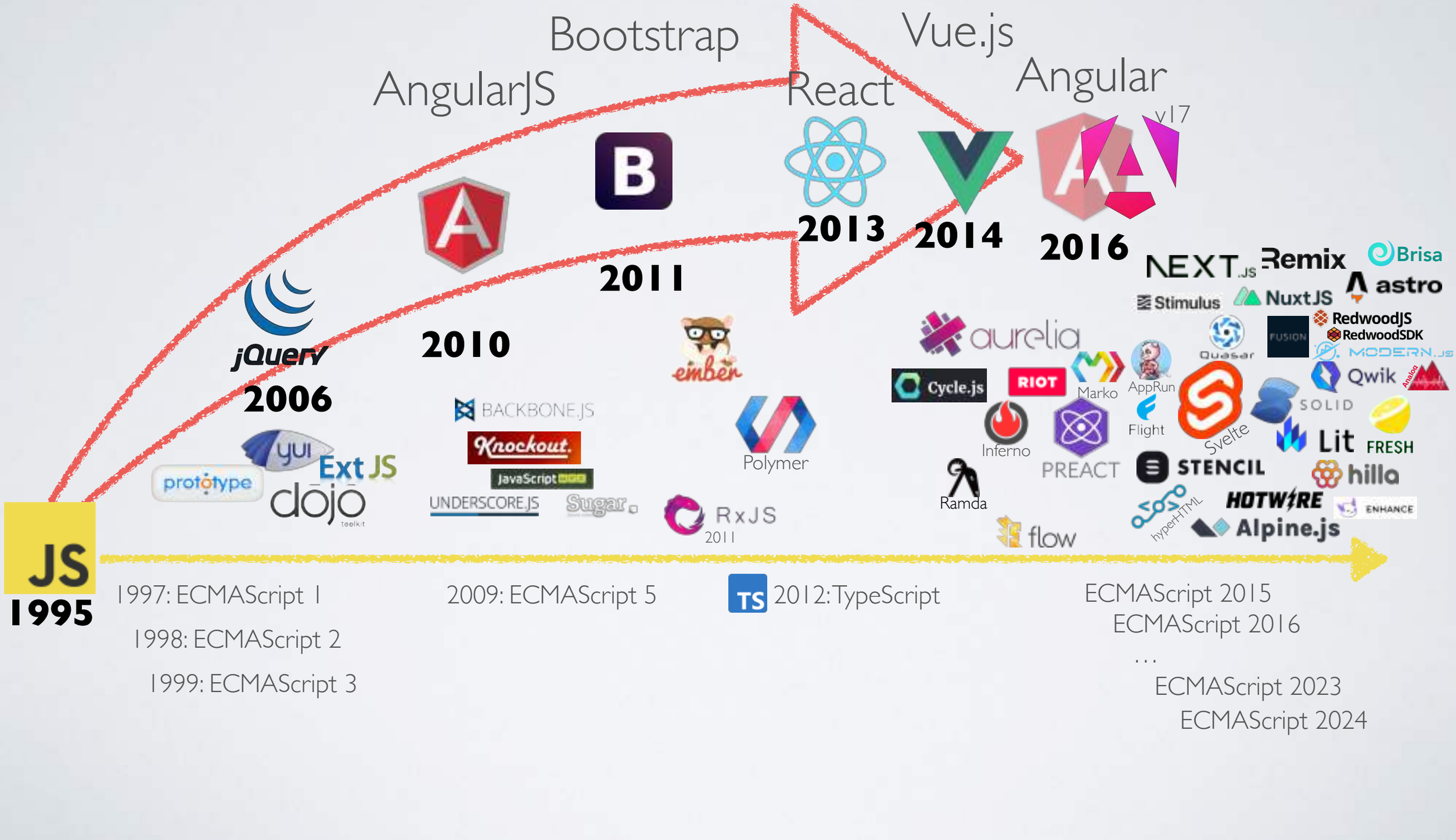
Reactivity

Functional Programming

The Evolution of React

Discussion & Questions

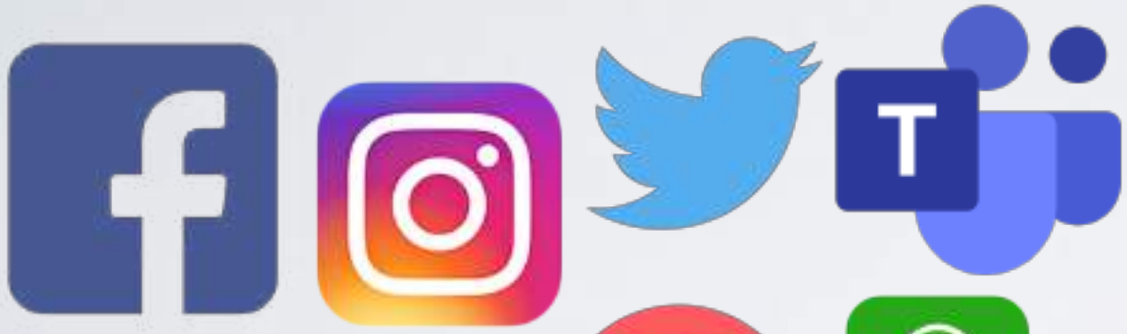
The JavaScript Frontend Ecosystem



The "Big 3" of SPA Frameworks

Framework Usage

React



Angular

"2000+ projects at google"

Google Analytics

die Mobiliar

Office 365

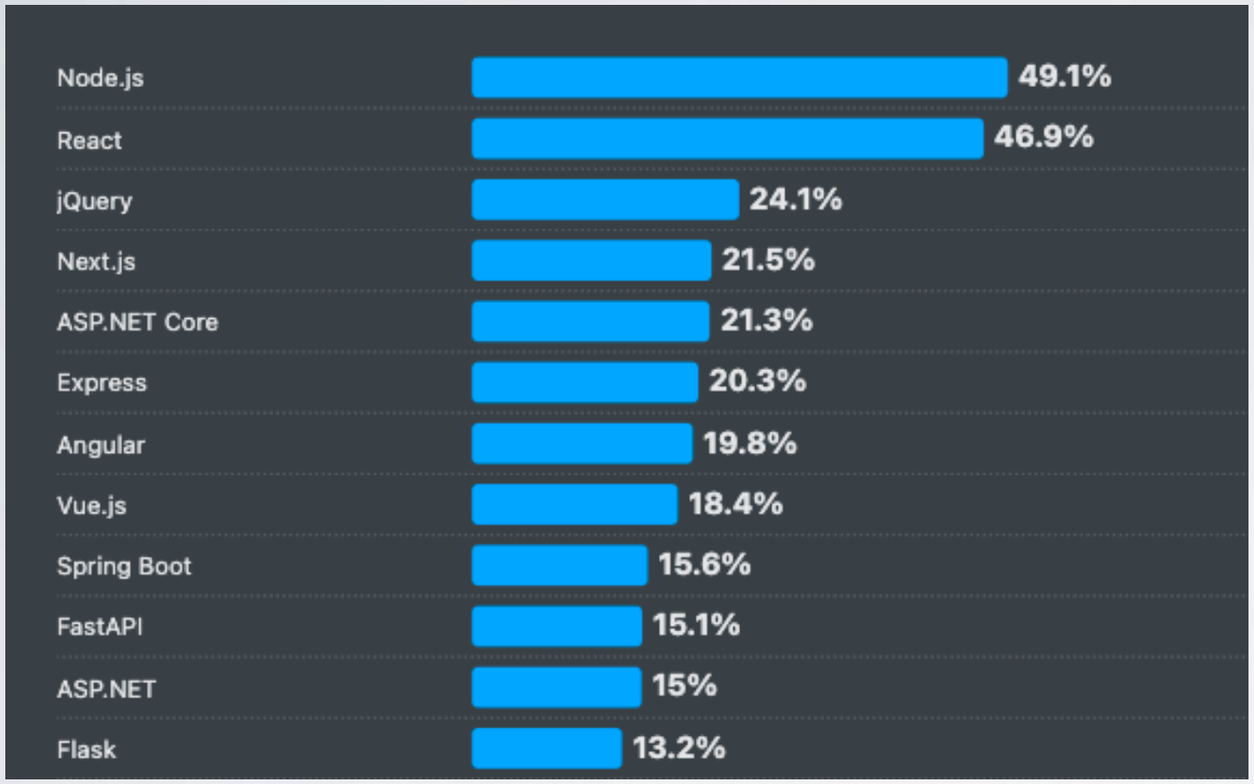
CapitalOne

Forbes

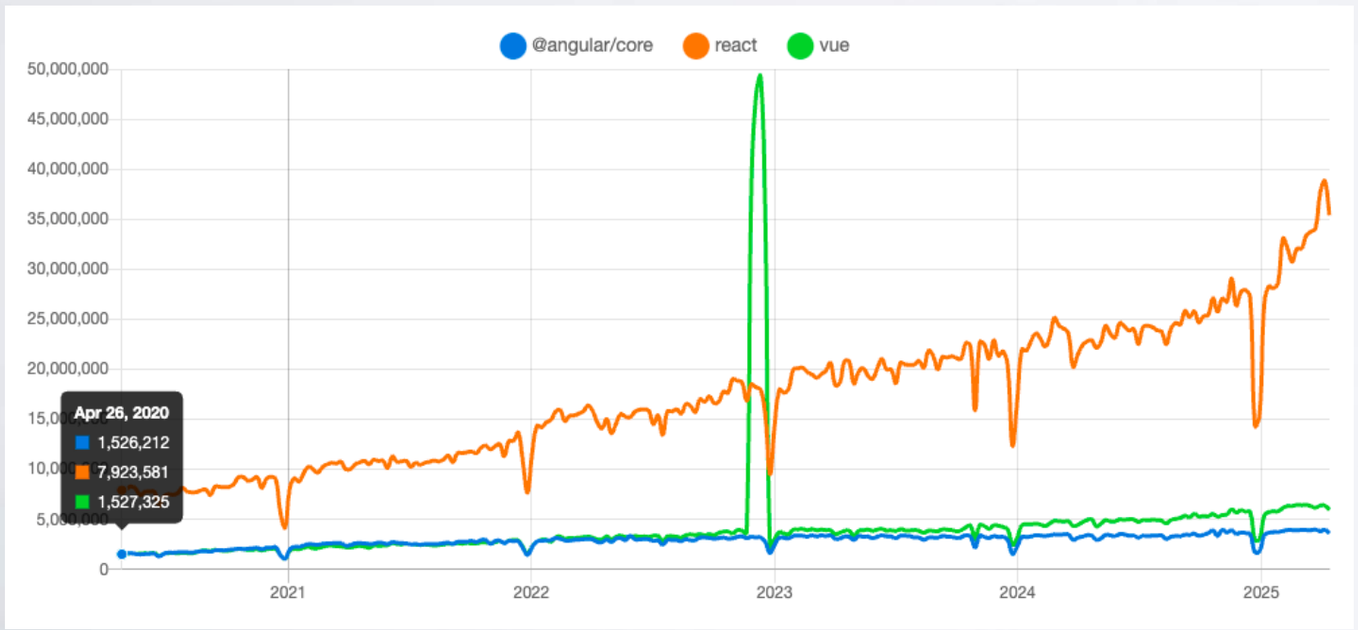
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Vue.js

Popularity: Angular vs. React vs. Vue

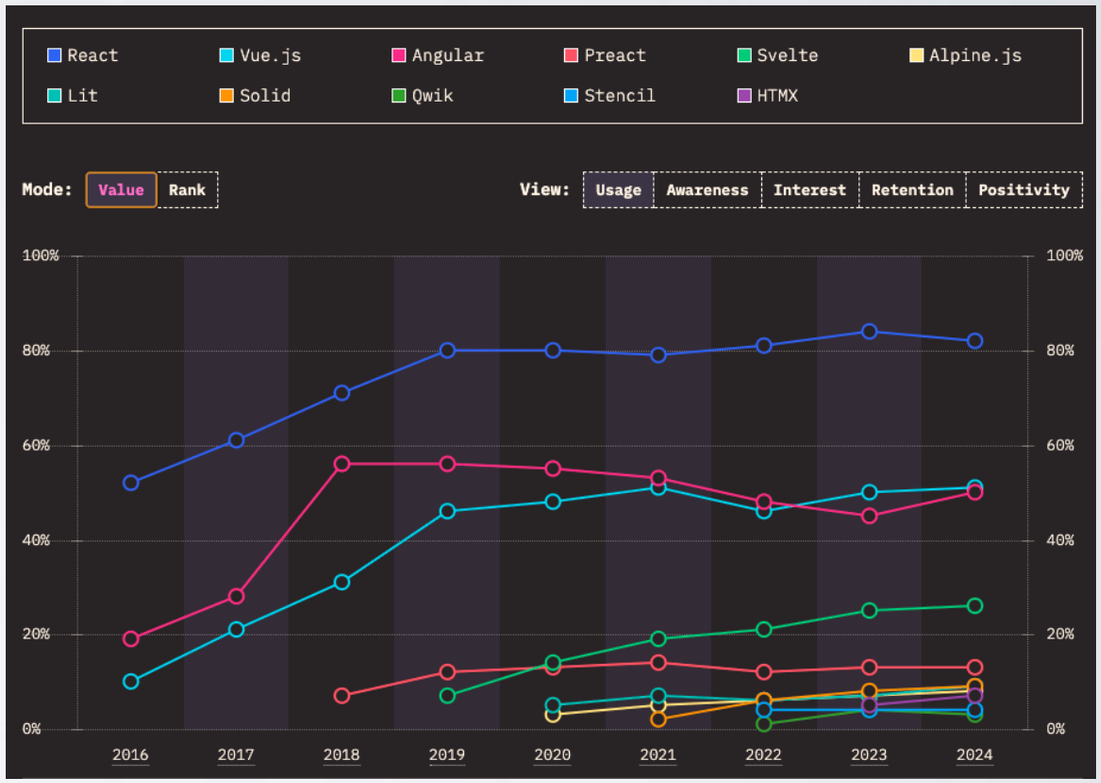


https://survey.stackoverflow.co/2025/technology#1-web-frameworks-and-technologies



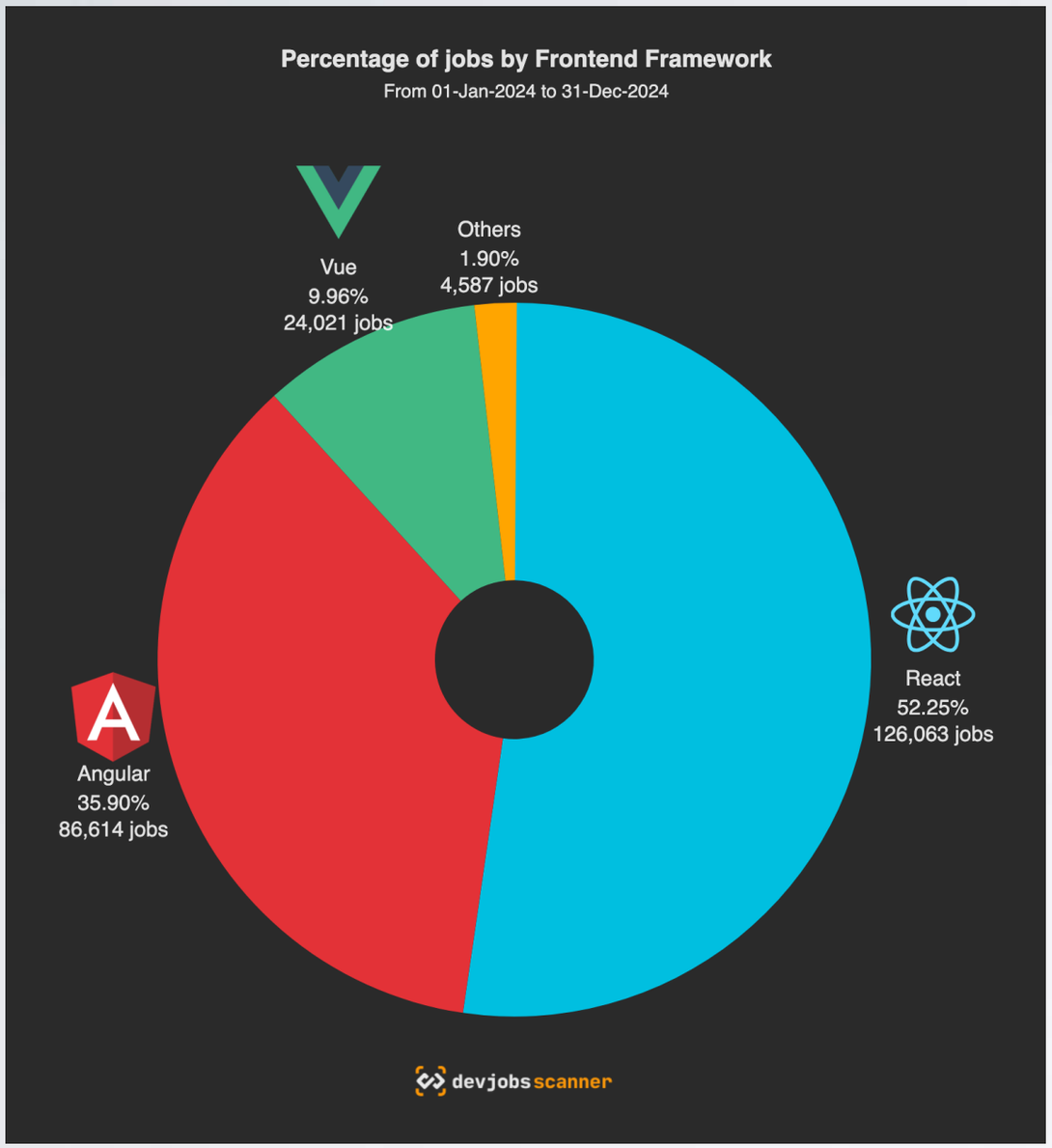
https://npmtrends.com/@angular/core-vs-react-vs-vue

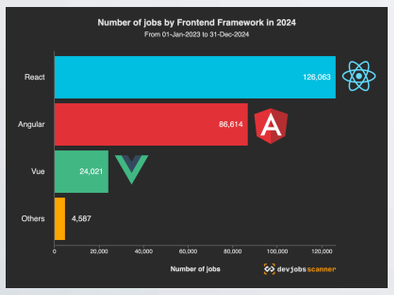
State of JavaScript Survey 2024

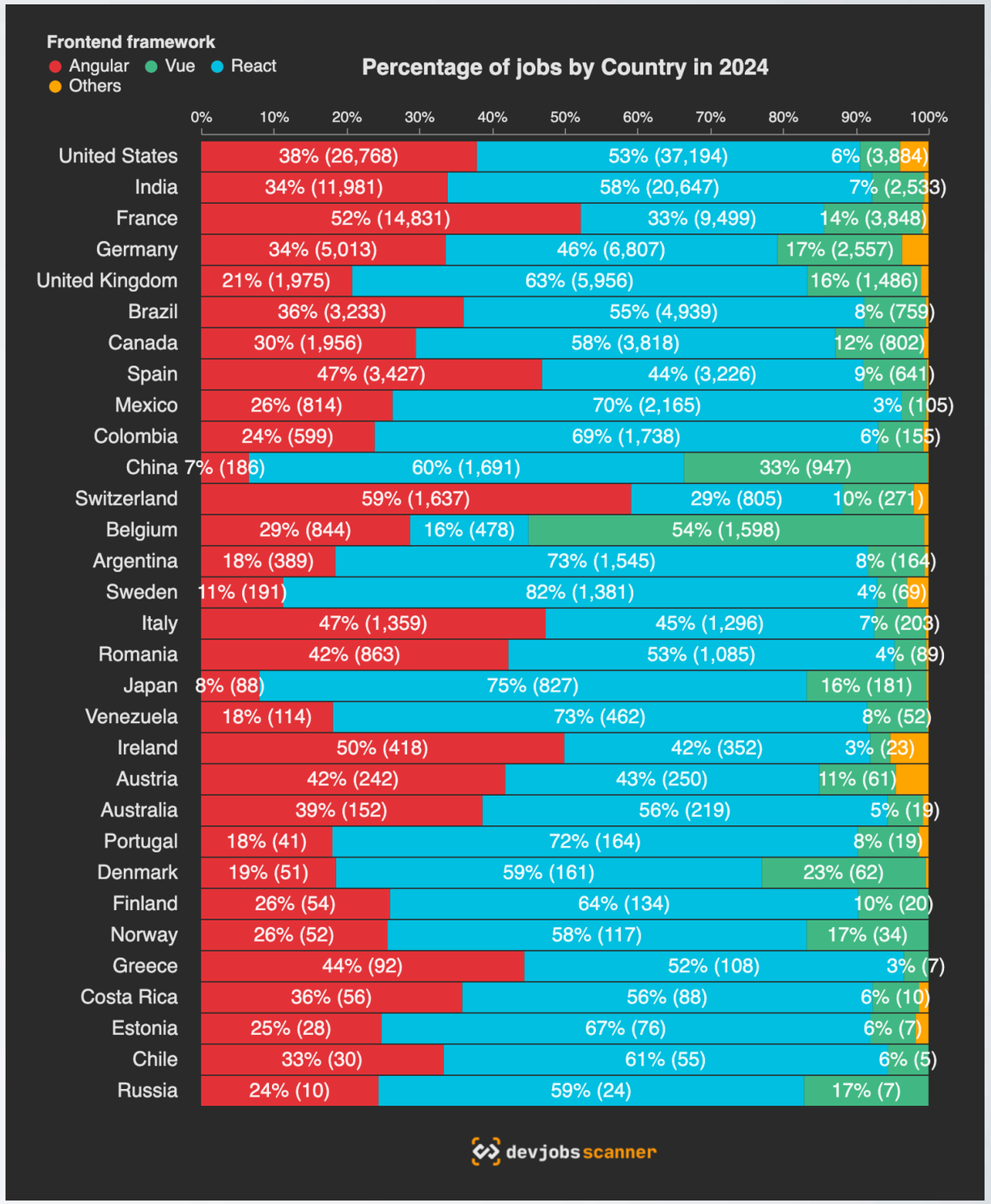


https://2024.stateofjs.com/en-US/libraries/front-end-frameworks/

Popularity: Angular vs. React vs. Vue



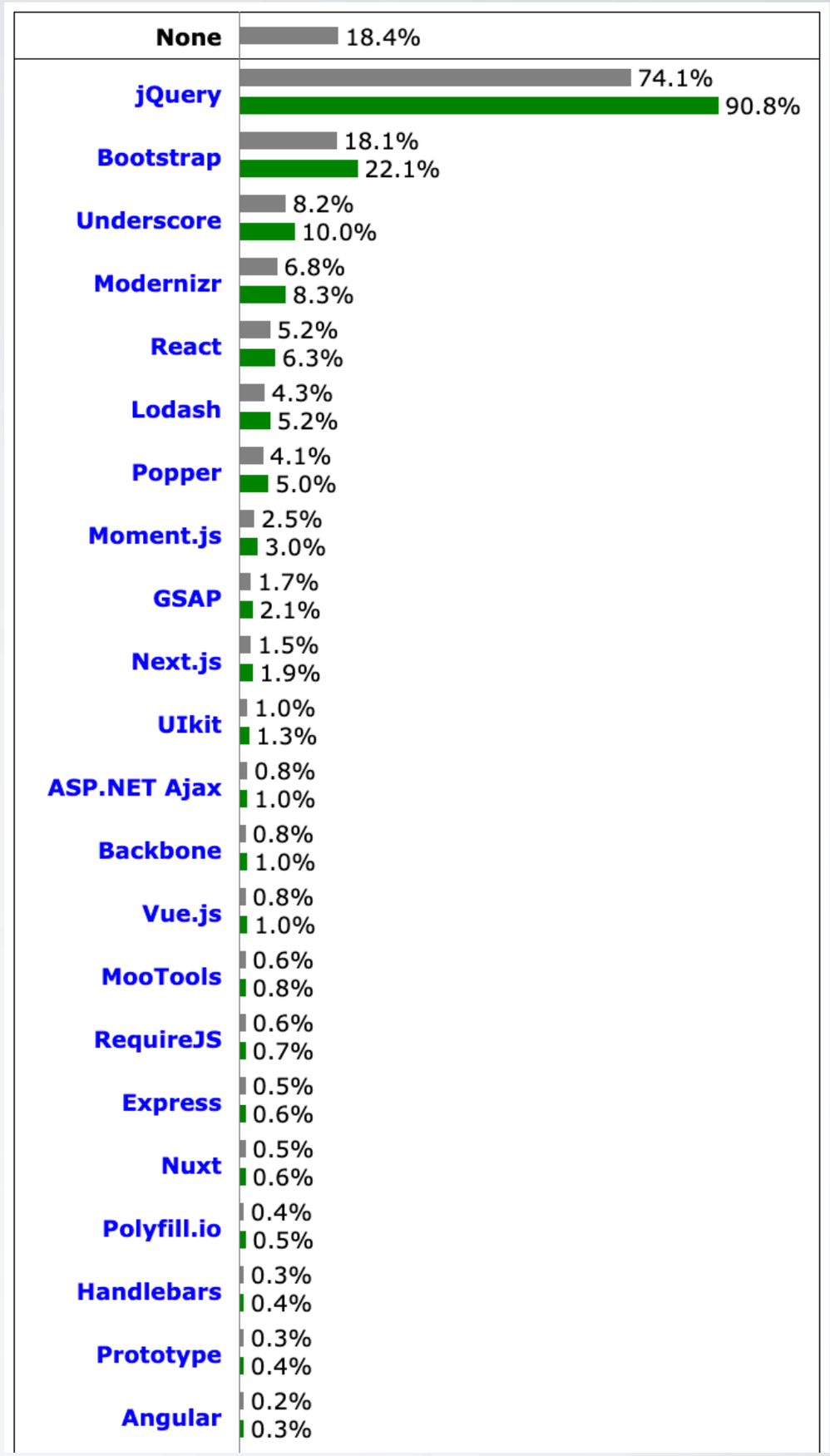




https://www.devjobsscanner.com/blog/the-most-demanded-frontend-frameworks/

By the way ...

...the web still runs on jQuery



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

Performance Comparison

Name Duration for...	vue-vapor-v3.6.0-alpha.2	svelte-v5.13.0	solid-v1.9.3	vue-v3.6.0-alpha.2	angular-cf-v20.0.1	angular-cf-signals-v20.0.1	react-hooks-v19.0.0	angular-ngfor-v20.0.1	react-compiler-hooks-v19.0.0	react-hooks-use-transition-v19.0.0
Implementation notes										
Implementation link	code	code	code	code	code	code	code	code	code	code
create rows creating 1,000 rows. (5 warmup runs).	24.0 ±0.1 (1.06)	23.6 ±0.1 (1.04)	23.8 ±0.1 (1.05)	26.9 ±0.1 (1.19)	32.3 ±0.2 (1.43)	32.1 ±0.2 (1.42)	28.0 ±0.4 (1.24)	32.1 ±0.2 (1.42)	27.7 ±0.4 (1.23)	35.2 ±0.7 (1.56)
replace all rows updating all 1,000 rows. (5 warmup runs).	27.5 ±0.2 (1.08)	27.8 ±0.2 (1.09)	27.6 ±0.2 (1.08)	30.9 ±0.1 (1.21)	37.7 ±0.3 (1.48)	37.3 ±0.2 (1.46)	33.6 ±0.3 (1.32)	37.0 ±0.1 (1.45)	33.7 ±0.3 (1.32)	36.5 ±1.2 (1.43)
partial update updating every 10th row for 1,000 row. (3 warmup runs). 4 x CPU slowdown.	11.1 ±0.3 (1.12)	10.8 ±0.4 (1.09)	10.8 ±0.4 (1.09)	12.9 ±0.3 (1.30)	12.0 ±0.4 (1.21)	12.3 ±0.2 (1.24)	14.7 ±0.2 (1.48)	11.5 ±0.4 (1.16)	14.7 ±0.5 (1.48)	34.5 ±3.9 (3.48)
select row highlighting a selected row. (5 warmup runs). 4 x CPU slowdown.	2.4 ±0.3 (1.09)	3.3 ±0.2 (1.50)	2.6 ±0.1 (1.18)	3.2 ±0.2 (1.45)	3.6 ±0.2 (1.64)	3.8 ±0.2 (1.73)	4.3 ±0.2 (1.95)	3.7 ±0.1 (1.68)	4.5 ±0.1 (2.05)	11.3 ±0.9 (5.14)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	13.7 ±0.4 (1.10)	13.8 ±0.3 (1.11)	14.1 ±0.3 (1.14)	14.5 ±0.2 (1.17)	15.1 ±0.3 (1.22)	15.7 ±0.3 (1.27)	104.5 ±0.5 (8.43)	123.4 ±1.0 (9.95)	104.4 ±0.7 (8.42)	122.1 ±3.6 (9.85)
remove row removing one row. (5 warmup runs). 2 x CPU slowdown.	10.4 ±0.4 (1.07)	10.5 ±0.1 (1.08)	10.8 ±0.1 (1.11)	12.7 ±0.1 (1.31)	11.5 ±0.1 (1.19)	13.5 ±0.2 (1.39)	11.8 ±0.1 (1.22)	11.7 ±0.1 (1.21)	11.8 ±0.2 (1.22)	14.5 ±0.4 (1.49)

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

Conclusion:

- Angular, React and Vue are very close when it comes to framework performance.
- The differences are not significant for typical "line of business" applications.
- There are many modern frameworks that are faster than Angular, React and Vue.

Key Differences

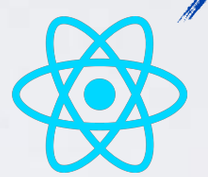
Templating

templates: enhancing html with dynamic constucts

separate files or inlined strings

single file components

(Vue also supports render functions with JSX)

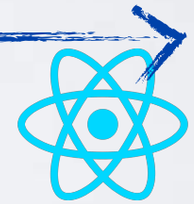


JSX: declaring UI with code

Framework vs Library

"complete" framework with opinionated choices

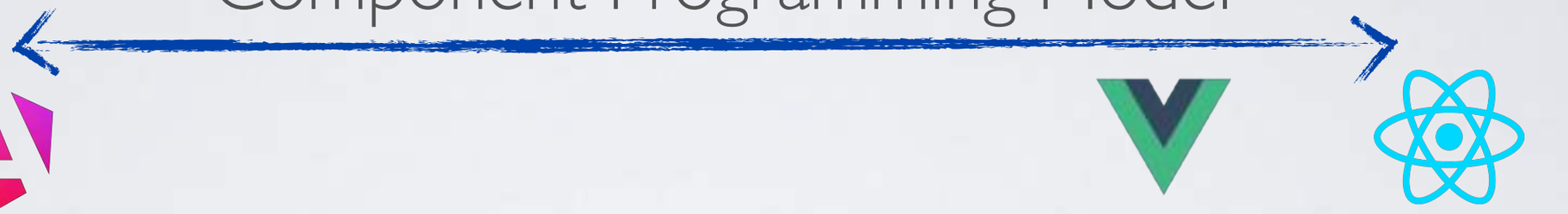
small & unopinionated library providing default options for important topics (router, centralized state management)



small & unopinionated library that must be combined with other libraries

Key Differences

Component Programming Model

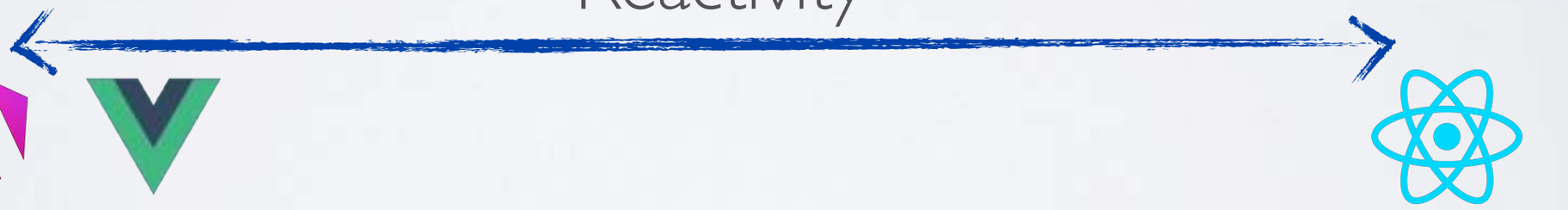


based on classes

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

functional API
("Hooks")

Reactivity



fine grained reactivity with signals

coarse grained render functions
(optimized by the React Compiler)

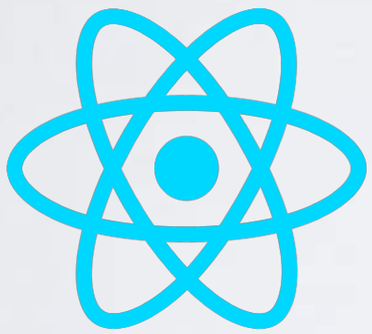
TypeScript Support

"TypeScript sees itself as the Switzerland for JavaScript Frameworks"

Anders Hejlsberg (2018)

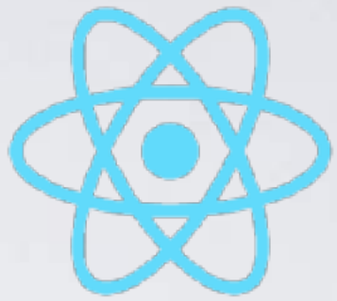
TypeScript: Static types for JavaScript: https://www.youtube.com/watch?v=ET4kT88JRXs

Angular is written in TypeScript and the whole Angular ecosystem and its community has fully embraced TypeScript.



TypeScript support is especially strong for React, since React does not have it's own templating "language". JSX is directly supported by TypeScript in a typesafe way.

Vue 3 was a full rewrite in TypeScript
One of the goals was to provide better support for TypeScript projects.

React 

React is a JavaScript library for building user interfaces.

Created by Facebook in 2013.

Current Version: 19 (released in December 2024)

Core Principles:

The DOM is created programmatically.

JSX enables declarative DOM programming.

A virtual DOM abstracts the real DOM.

Unidirectional data flow instead of two-way data binding.

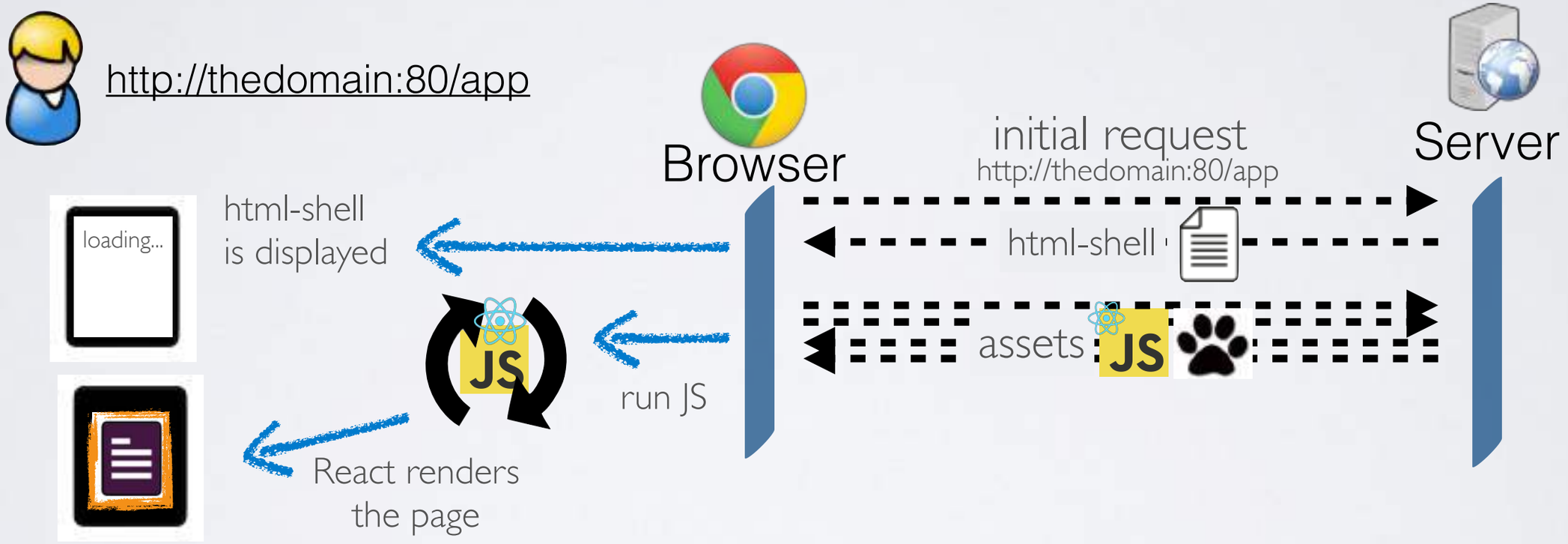
https://react.dev/
https://github.com/facebook/react/
https://react.dev/blog/2024/12/05/react-19

https://react.dev/versions

Single Page Applications

"Rich Clients running in the Browser"

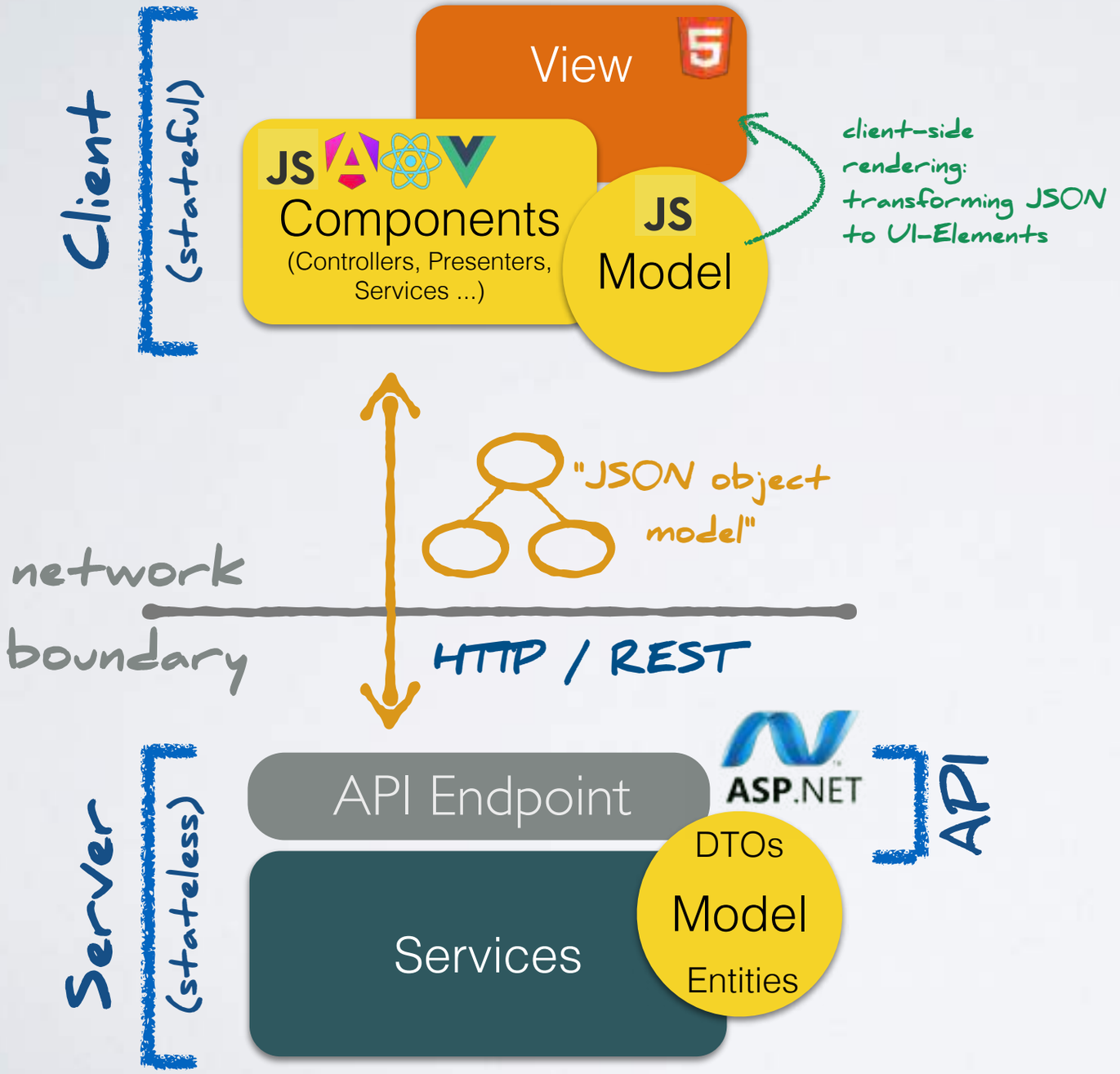
Traditional SPA:
Client Side Rendering



Components are rendered on the client.

Single Page Applications

SPA Architecture



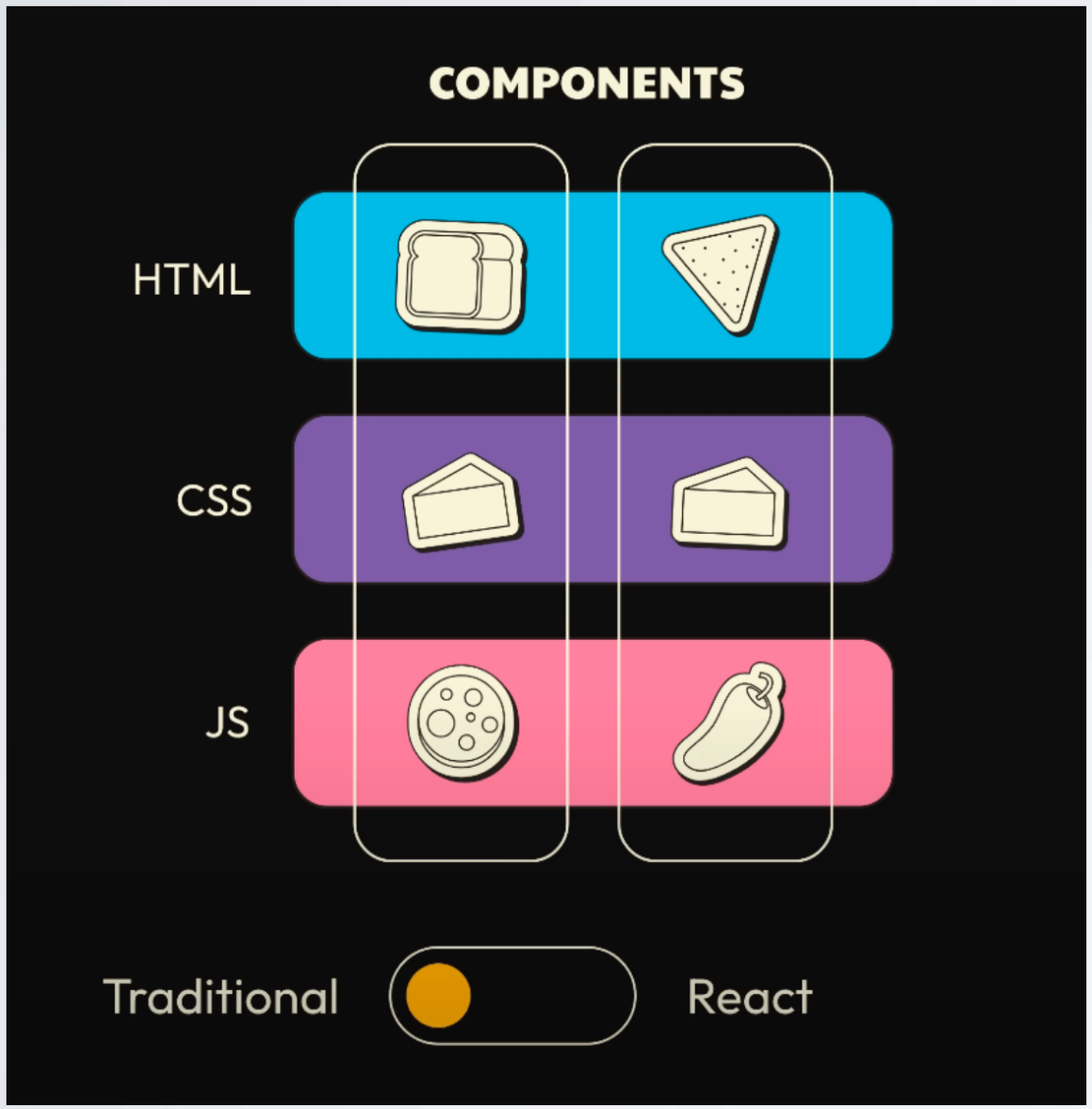
Rich client programming model in the browser.

Clear separation of concerns between client and server.

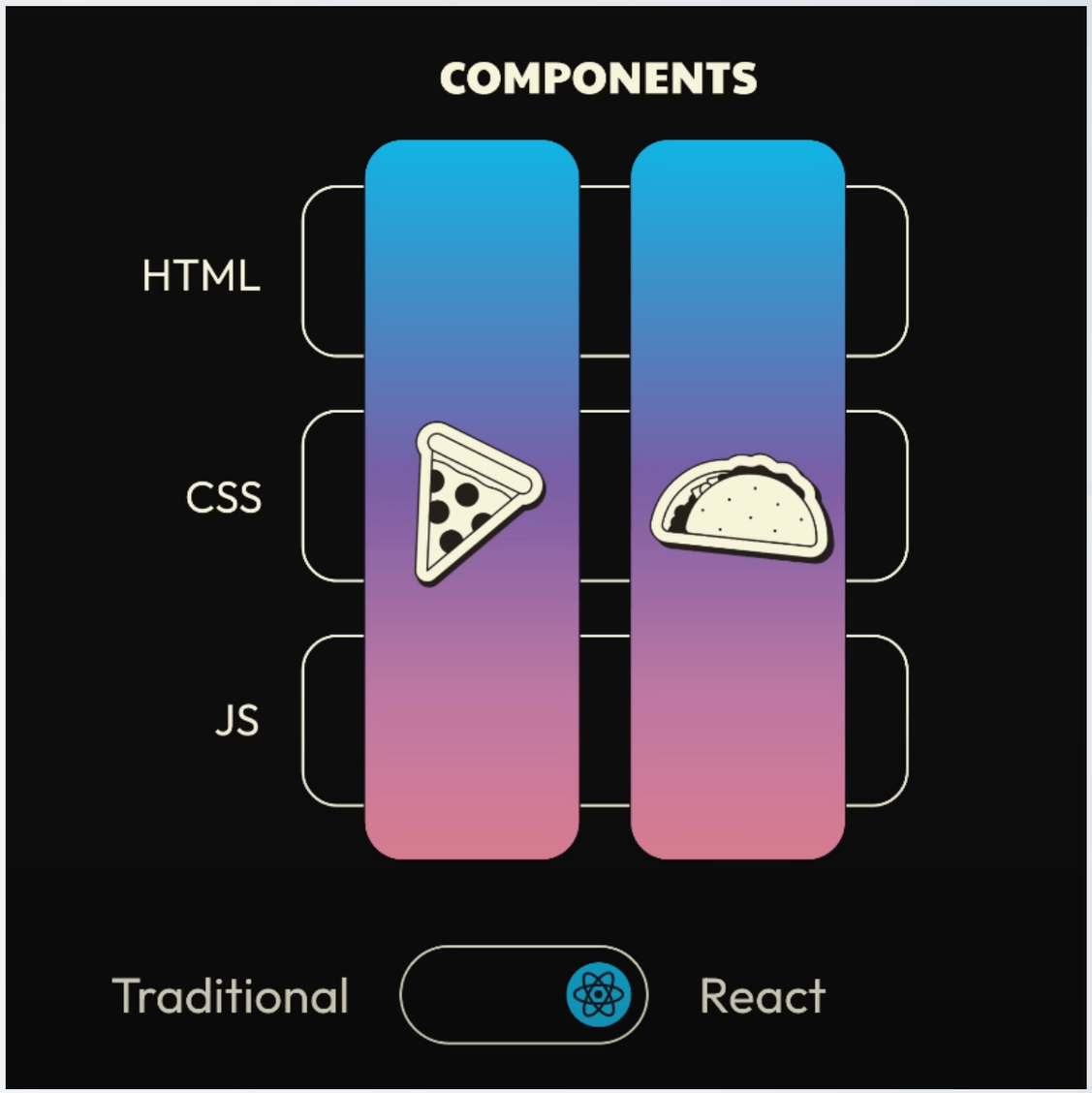
Everything is a Component

Separation of Concerns

Traditional Seperation of Concerns in Web Dev:



Seperation of Concerns in React:



Images from *ui.dev: The Story of react Query*
https://www.youtube.com/watch?v=OrliU0e09io

Dan Abramov
@dan_abramov

Following

Separating concerns by files is as effective as separating school friendships by desks. Concerns are “separated” when there is no coupling: changing A wouldn’t break B. Increasing the distance without addressing the coupling only makes it easier to add bugs.

9:37 PM - 16 Jun 2018

540 Retweets **1,817** Likes

40 540 1.8K

https://twitter.com/dan_abramov/status/1008131488481730561

JSX

XML-like extension for JavaScript

React uses JSX to describe the UI in a declarative way:

```
function Counter() {

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

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

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

JSX is not an "embedded string". It is compiled at build time into code that produces an element-tree at runtime.

JSX is an XML-like syntax extension to ECMAScript without any defined semantics: https://facebook.github.io/jsx/

JSX is used in many other frontend libraries: Preact, Solid.js, Qwik, Stencil, Vue.js, Inferno, Brisa …

Back to Separation of Concerns

Theoretically you could split a component into a "controller" and a "view":

Controller.ts

```
import {View} from './View';

export function Controller {

  // state & behavior
  const data = ... /* fetched from API */

  function doWork(){ ... }

  // delegate rendering
  return (
    <View data={data} onEvent={doWork} />
  );
}
```

View.js

```
export function View({data, onEvent}){
  return (
    <div>
      {data.message}
      <button onClick={()=>onEvent()}>
        Go!
      </button>
    </div>
  );
}
```

Note: This is an "academic" example that does not represent ideomatic usage of React.
But the patterns "lifting state up" and "*Container- and Presentation-Components*" is based on this concepts:

https://react.dev/learn/sharing-state-between-components

https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0

The Virtual DOM

The Virtual DOM

Components

Virtual DOM

In-Memory, implemented in JavaScript

Browser DOM

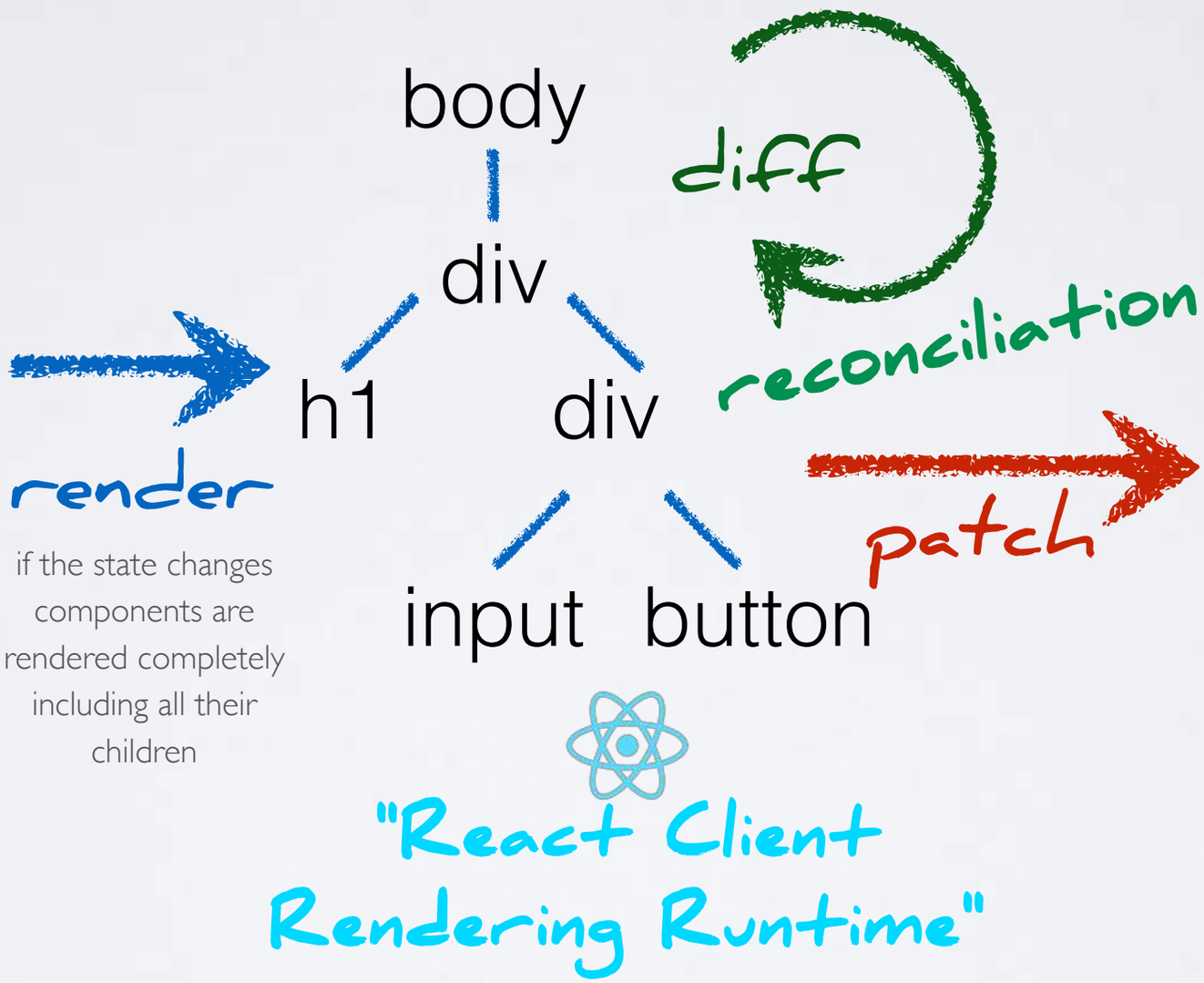
App

Title

Content

input

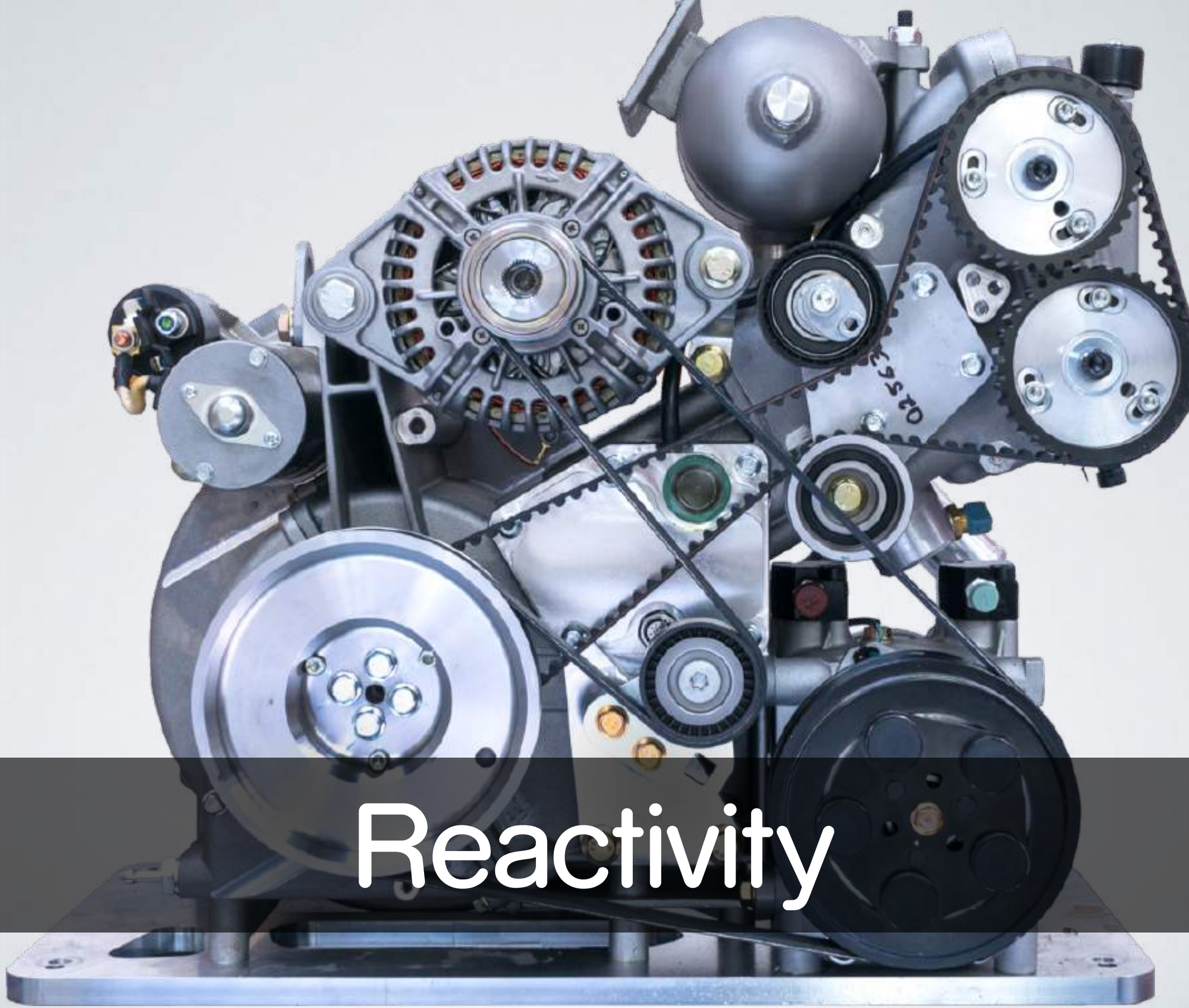
button



The Virtual DOM also "enables" server-side rendering and rendering to iOS/Android UIs.

In 2013 the virtual DOM was the reason why React was faster than other Frameworks.

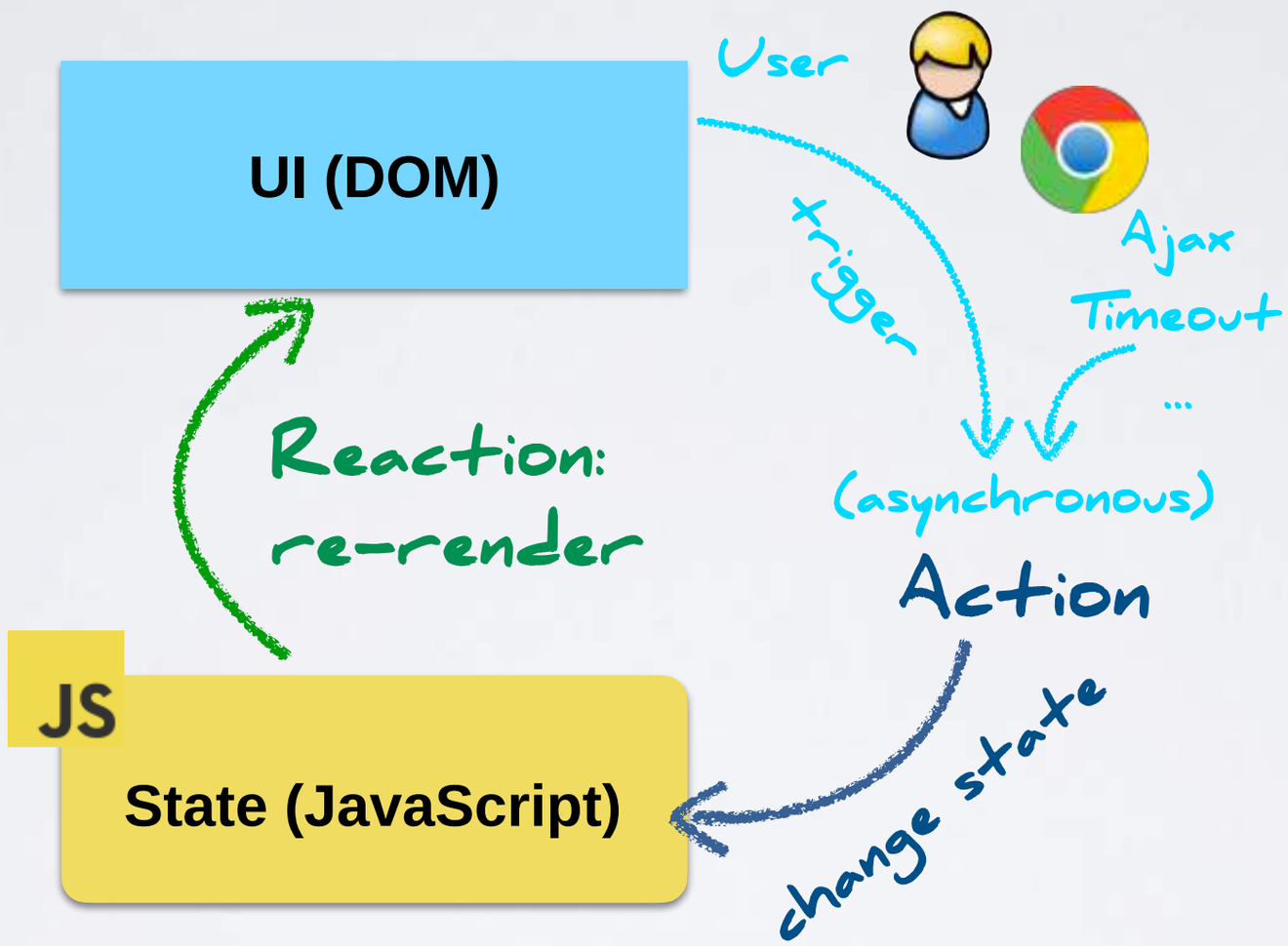
Today the Virtual DOM is the reason why React is slower than other modern frameworks.
https://svelte.dev/blog/virtual-dom-is-pure-overhead



Reactivity

State is Managed in JavaScript

The UI renders the state and "emits" events.



Reactivity in a SPA: The application reacts on state changes and updates the UI.

Evan You - Reactivity in Frontend JavaScript Frameworks: https://www.youtube.com/watch?v=r4pNEdIt_I4
http://teropa.info/blog/2015/03/02/change-and-its-detection-in-javascript-frameworks.html

Rich Harris
@Rich_Harris

The problem all frameworks are solving is *reactivity*. How does the view react to change?

- React: 'we re-render the world'
- Vue: 'we wrap your data in accessors'
- Svelte: 'we provide an imperative set() method that defeats TypeScript'
- Angular: 'zones' (actually idk 🤷‍♂️)

5:01 PM · Nov 3, 2018 · Twitter Web App

https://twitter.com/Rich_Harris/status/1058751085706383362

Thou shalt not manipulate the DOM!

UI = fn(state)

Reactivity: What and Why?

Traditional "DOM-centric" applications

UI = state

Browsers have "built-in" reactivity: If the DOM is changed, the UI is re-rendered.

Problem: the same state might be displayed at several places in the DOM.

With client-side Single-Page-Applications, the state is represented as JavaScript objects.

UI = fn(state)

The UI that you can see and manipulate on screen is the result of painting a visual representation of data.

Components:

Functional Programming Model

Function Components

Components are written as plain JavaScript functions.


```
function AppComponent(props) {
    return (
        <div>
            <h1>{props.title}</h1>
            <p>{props.message}</p>
        </div>
    );
}
```

The function is called each time the UI is rendered (i.e. with every "data-change")

React expects that the body of your component behaves like a *pure function*:
If the "inputs" (props, state, and context) are the same, it should return exactly the same JSX.
https://react.dev/learn/keeping-components-pure

A Visual Guide To React Mental Models:
https://obedparla.com/code/a-visual-guide-to-react-mental-models/

Legacy: Class Components

```
import React from 'react';

export class Greeter extends React.Component {
    render() {
        return <h1>Hello World!</h1>;
    }
}
```

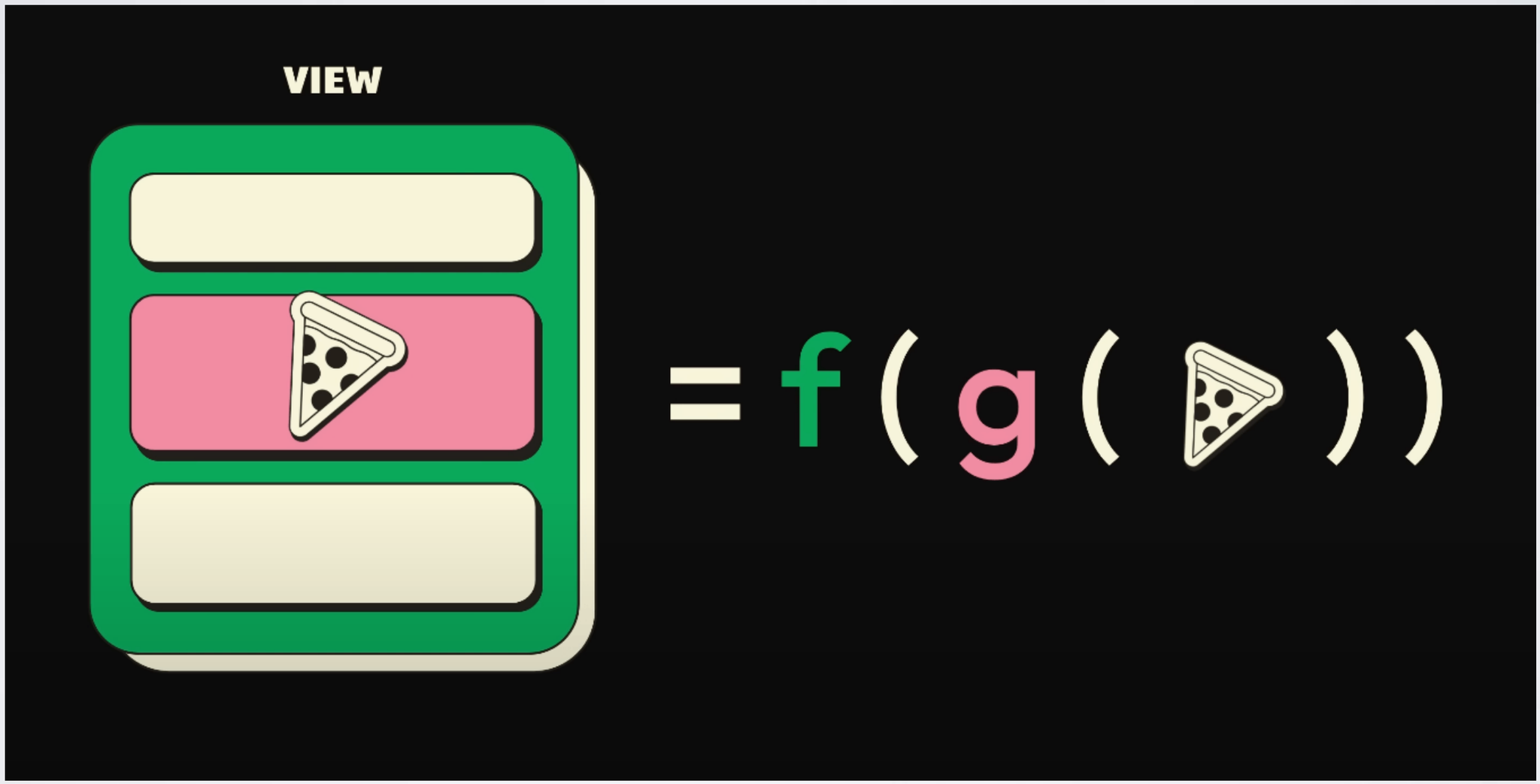
React 16.8 (February 2019) introduced "Hooks": a way to use state and lifecycle without writing a class.

This resulted in major changes in the React ecosystem.

https://playfulprogramming.com/posts/react-history-through-code

UI Composition

UI Composition *is* Function Composition.



Once embraced, we can apply a whole set of patterns from functional programming.

Image from *ui.dev: The Story of react Query*
https://www.youtube.com/watch?v=OrliU0e09io

The power of React is a component model which enables simple & elegant composition ...

Cory House
@housecor

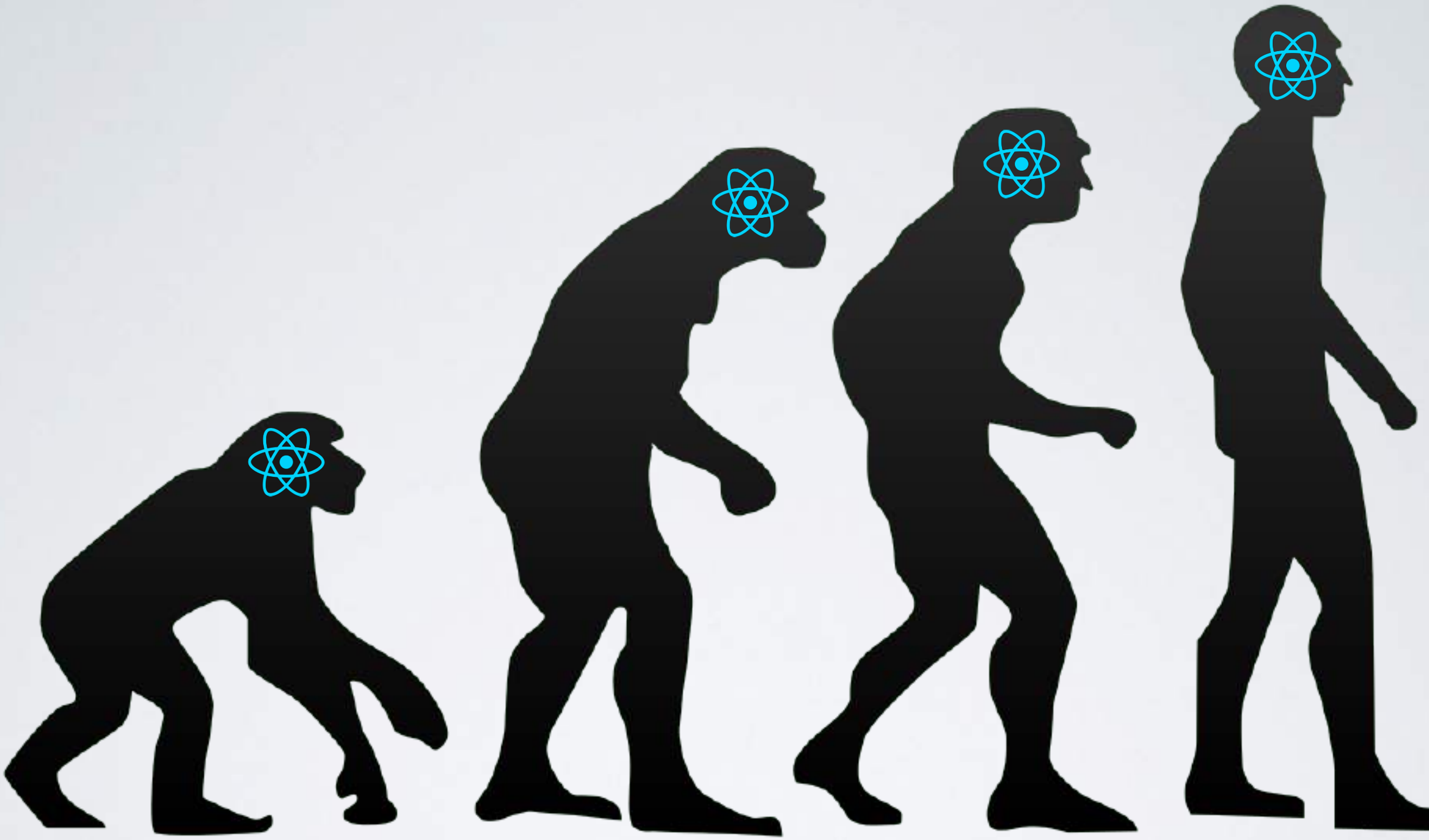
I love the simplicity of React's reuse model.

Repeating JSX? Create a component.
Repeating logic? Create a hook.

I can compose these simple building blocks in infinite ways.

1:08 PM · Nov 25, 2023 · **16.7K** Views

https://twitter.com/housecor/status/1728385239611789758



The Evolution of React

https://twitter.com/ralex1993/status/1234165541805281280

Evolution of React

2013 — React (Virtual Dom)

2015 — React Native components are based on ES2015 classes

2016 — splitting React and ReactDOM
React 0.14
Create React App
NEXT.JS
React 15

2019 — React 16.8: Hooks ("Function Components for everything")

2022 — Vite 2.0
React 18
Next.js App Router with RSCs

2024 — React 19
"official" release of RSCs

2025 — React Compiler

Cory House @housecor · Oct 30

React in 2014:
✅Make SPAs.
✅Use create-react-app.

React in 2022:
✅Make SPAs, native mobile, static, MPAs, and hybrid apps.
✅Use Vite, Next, Remix, Astro, etc.
✅HTTP via react-query, swr, Apollo, etc.
✅Routes via React Router, Remix, Next, Tanstack Router, etc…

38 119 1,131

Cory House @housecor · Oct 30

React 2014: Simple, but limited.

React 2022: Powerful and remarkably flexible, but intimidating and fragmented. A massive number of compelling options to consider. The hardest part is keeping up with the churn and choosing between multiple compelling options.

https://twitter.com/housecor/status/1586704800988684288

React Native

Learn once, write anywhere.

Written in JavaScript - rendered with native code.

https://reactnative.dev/

For Android + iOS by facebook.

React Native for Windows + macOS by Microsoft

- https://microsoft.github.io/react-native-windows/

React Native Apps

Meta

React Native is shaping mobile, web, and desktop experiences within Meta's product ecosystem, from Facebook Marketplace, Messenger Desktop, Ads Manager to the Meta Quest app and many more.

Facebook	Instagram	Facebook Ads Manager	Meta Horizon	Messenger Desktop
iOS · Android · Meta Quest	Meta Quest	iOS · Android	iOS · Android	Desktop
	Learn more			Learn more

Microsoft

Microsoft leverages the power of React Native to deliver excellent customer experiences in some of its most well known apps.

Microsoft doesn't stop at mobile platforms either -- Microsoft leverages React Native to target desktop too! Find out more in the dedicated showcase for React Native Windows and macOS.

Microsoft Office	Microsoft Outlook	Microsoft Teams	Xbox Game Pass	Skype
iOS · Android	iOS · Android	iOS · Android	iOS · Android	iOS · Android
Learn more	Learn more	Learn more	Learn more	Learn more

amazon

Amazon has used React Native to rapidly deliver new customer-facing features in some of its most popular mobile applications as early as 2016. Amazon also uses React Native to support customer-favorite devices such as the Kindle E-readers.

Amazon Shopping	Amazon Alexa	Amazon Photos	Amazon Kindle	Amazon Appstore
iOS · Android	iOS · Android	iOS · Android	Learn more	Learn more

https://facebook.github.io/react-native/showcase.html

React is not a Framework

- React is "only" a view library (however recently the scope is growing)

  Next.js, ReactRouter v7 (formerly Remix), RedwoodJS and TanStack Start are popular Frameworks based on React.

- React is often combined with other libraries/frameworks to build a complete stack for building a frontend-application
  - Routing
  - Data-Fetching (http, websocket ...)
  - i18n
  - Styling …
  - State-Management

Full-Stack React

(aka. Meta Frameworks)

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

https://react.dev/learn/creating-a-react-app#full-stack-frameworks

NEXT.JS

https://nextjs.org/

React Router v7

Remix "Framework Mode"

https://reactrouter.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).

RedwoodJS
RedwoodSDK

https://redwoodjs.com/
https://docs.rwsdk.com/

TanStack Start

https://tanstack.com/start/

MODERN.JS

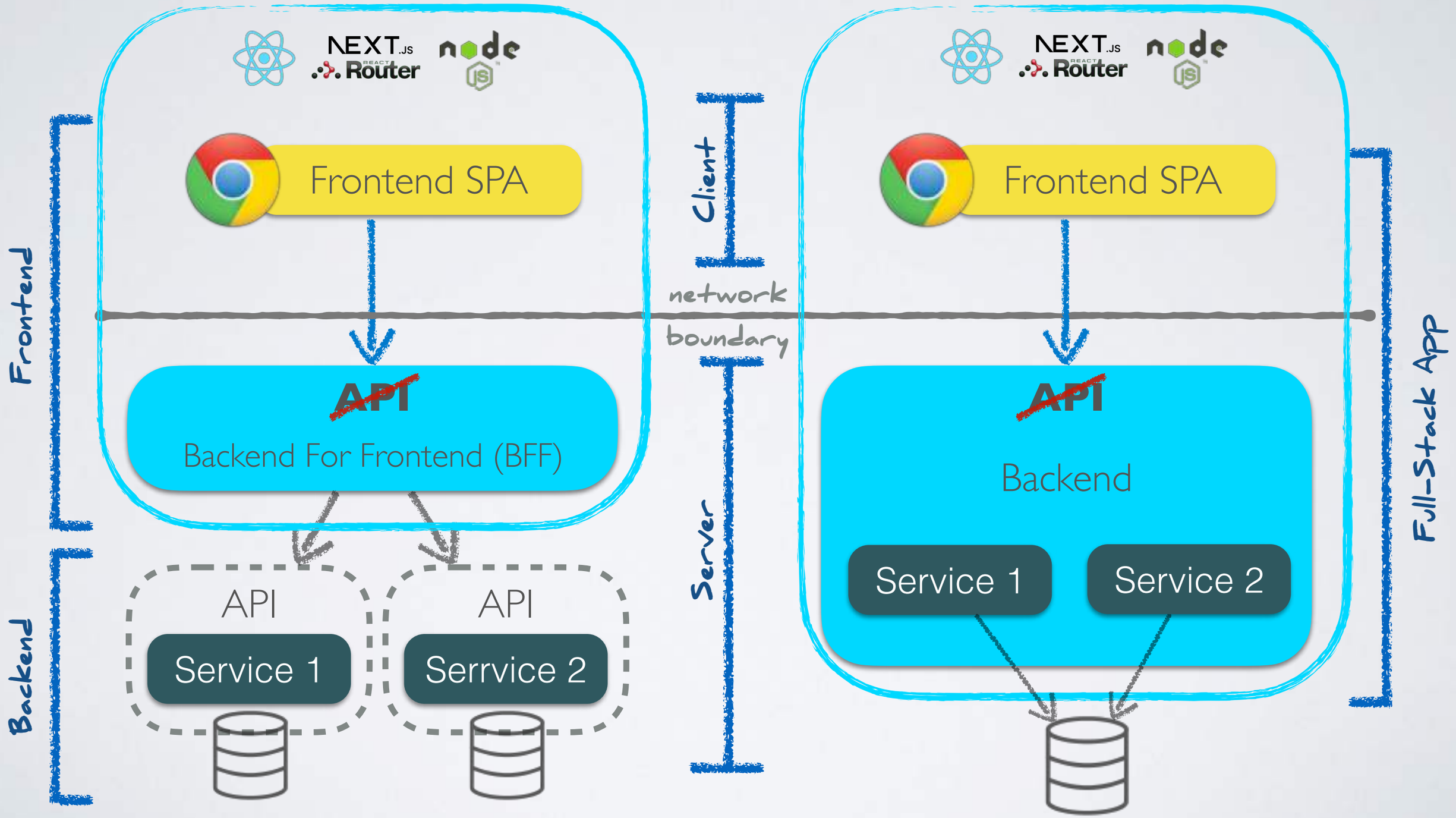
https://modernjs.dev/

Waku

https://waku.gg/

Remix was merged into React Router v7: https://remix.run/blog/react-router-v7

React Full-Stack Architectures



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

React Server Components (RSC)

https://twitter.com/sebastienlorber/status/1737380479383277636

https://twitter.com/AdamRackis/status/

Original Talk: https://www.youtube.com/watch?v=9CN9RCzznZc&t

The growing divide among React developers...

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

Are RSCs and NextJS Really That Bad?

Jack Herrington

21K views • 2 days ago

https://www.youtube.com/watch?v=u0OMdWlfdhg

Evan You @youyuxi

I don't think RSC itself being a React feature is a problem - it unlocks some interesting patterns.

The issue is the tradeoffs involved in making it work. It leaks into, or even demands control over layers that are previously not in scope for client-side frameworks. This creates heavy complexity (and associated mental overhead) in order to get the main benefits it promises.

React team made a bet that they can work with Next team to polish the DX to the extent that the benefit would essentially be free - so that RSC can be a silver bullet for all kinds of apps, and that it would become the idiomatic way to use React. IMO, that bet has failed.

4:06 PM · Mar 27, 2025 · **175.4K** Views

https://x.com/youyuxi/status/1905275294090662266

React Server Components is an Architecture!

React *as a library* provides the "foundation" for React Server Components but you need a Framework to use them (deep integration into Router and Bundler).

Frameworks that support React Server Components implement the "React Architecture"

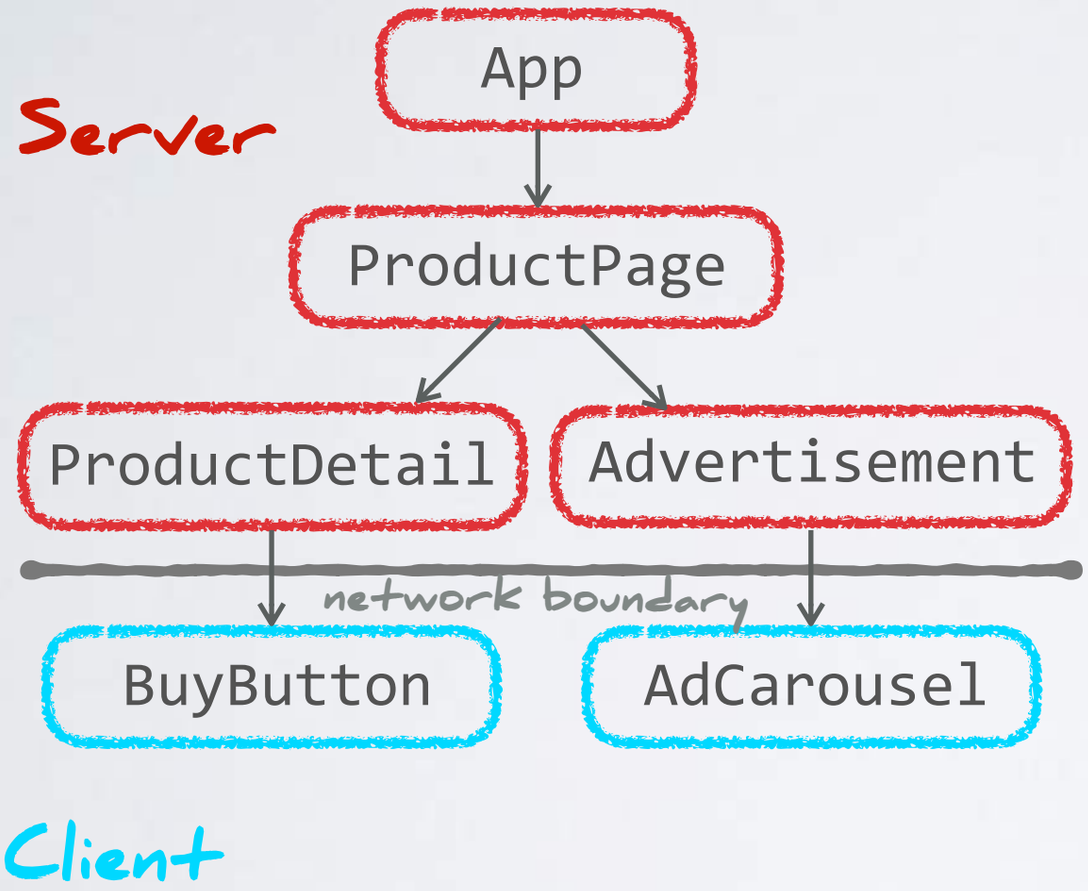
https://react.dev/

https://www.joshwcomeau.com/react/server-components/

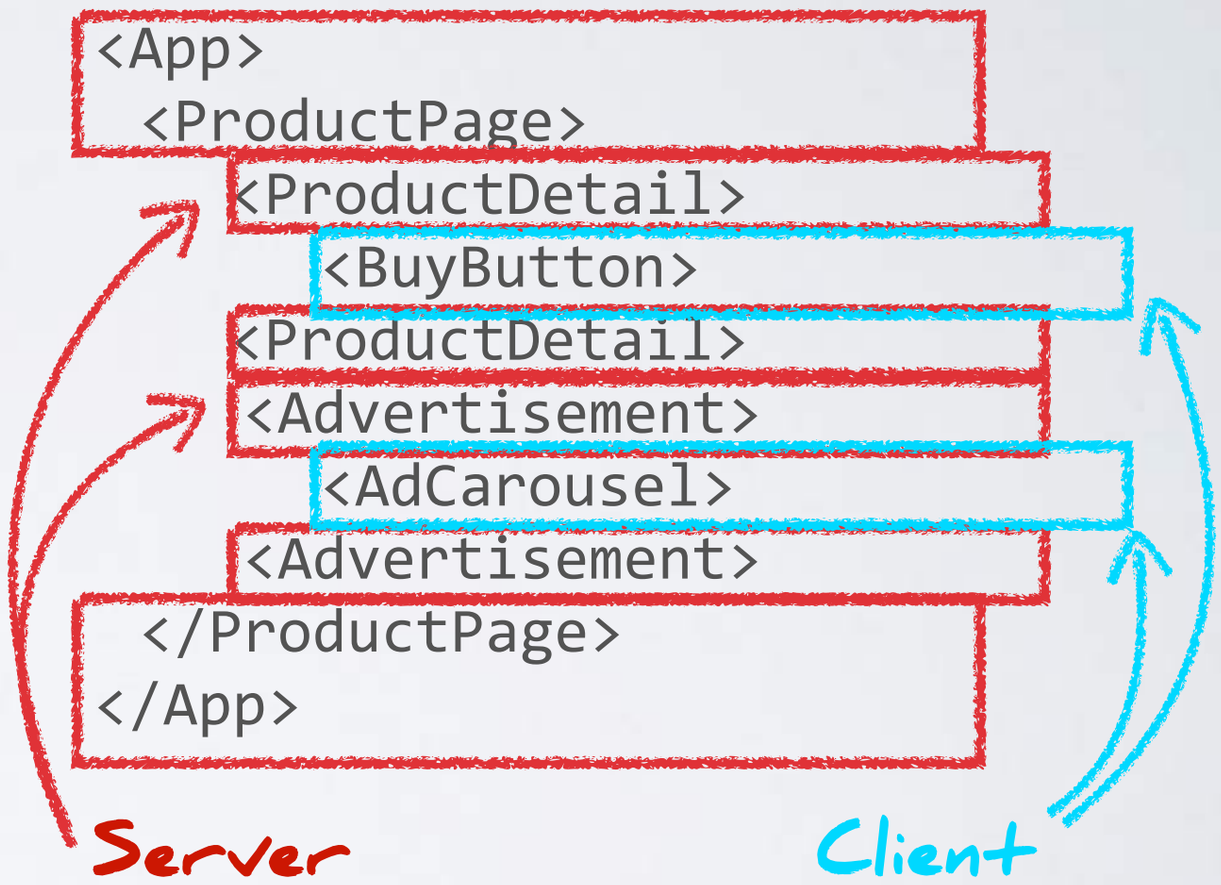
RSCs: Component Tree

React Server Component are executed on the server.

"logical perspective"



"composition perspective"

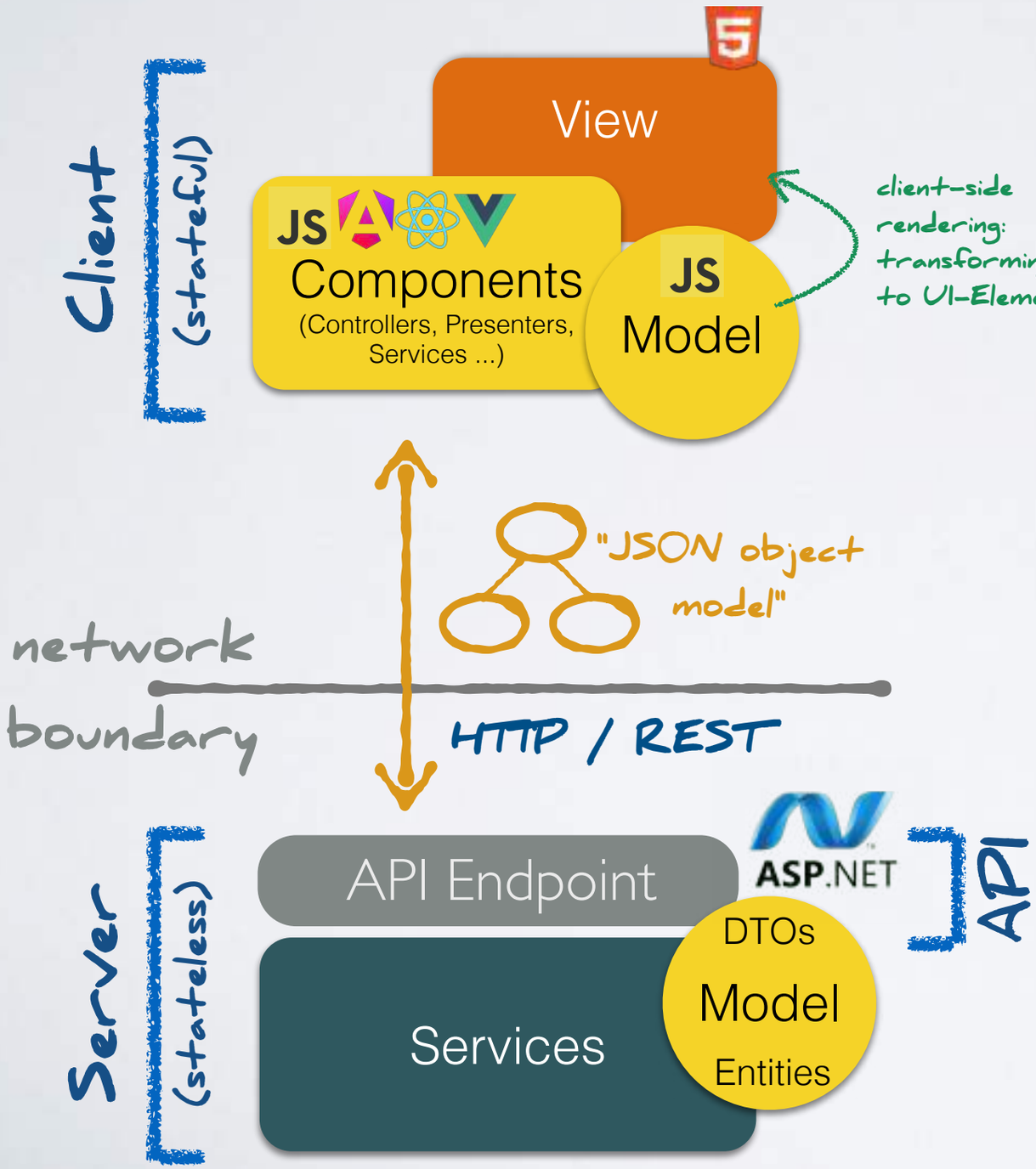


RSCs enable us to build a component-tree that spans between client and server while keeping the know component composition mechanisms of React components.
The network boundary becomes transparent.

But remember:

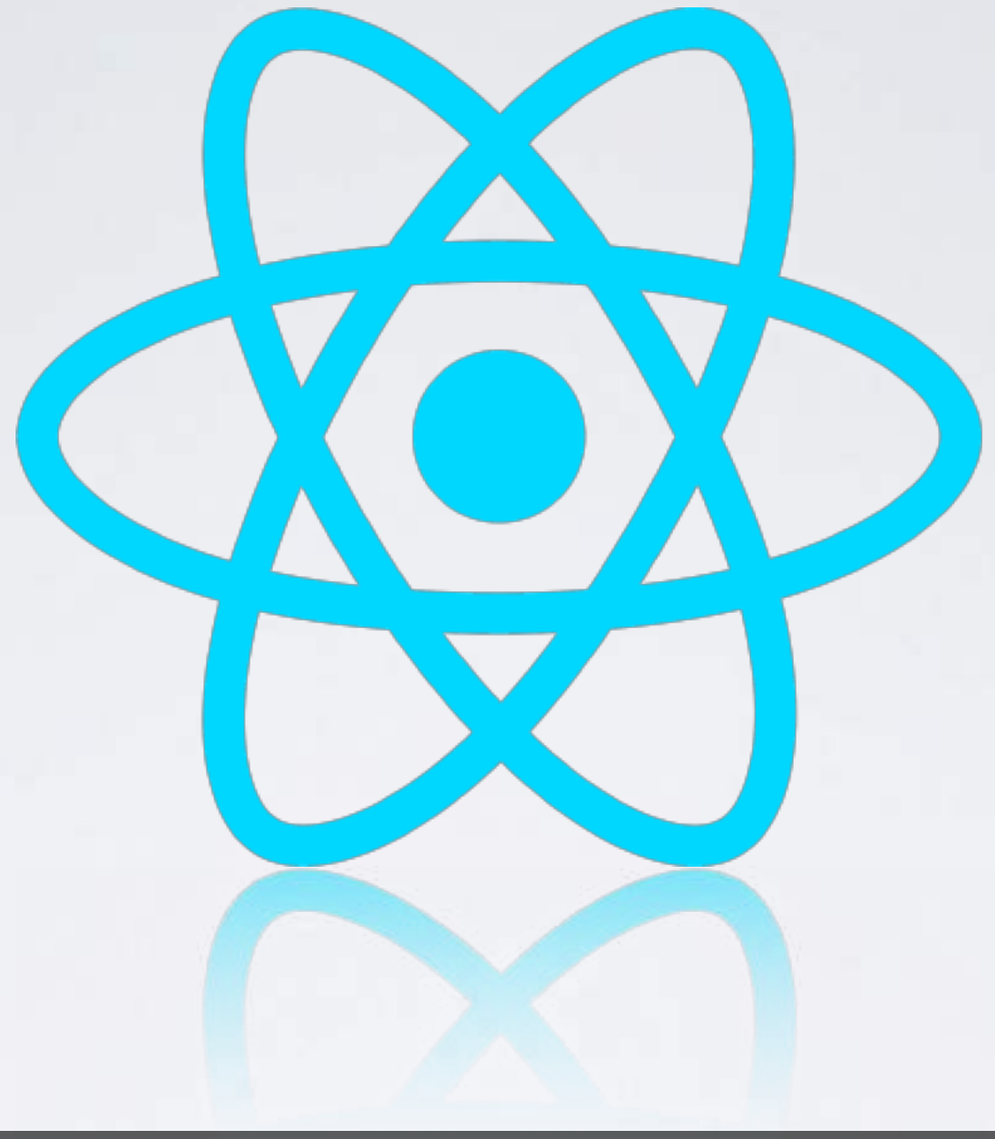
There is still much value in the traditional SPA Architecture.

Just because RSCs are the latest evolution of React, it does not mean that we have to use them for every project.



Rich client programming model in the browser.

Clear separation of concerns between client and server.



Have Fun with React!

... I hope you will have fun in my upcoming training :-)

TRAINING AGENDA

DAY 1

Intro

Getting started quickly with `create-vite`

JSX,React Components and Hooks

Component Architecture & Data Flow

Routing

DAY 2

Side Track: Async JavaScript & Ajax

Backend Access

In-Depth Topics:

More Hooks & Custom Hooks

State Management Concepts

Ecosystem Picks:
Tanstack-Query
nuqs

Performance Topics

Fullstack React

"SIDE-TRACK"

Modern JavaScript Development Tooling

Modern JavaScript (es2015 - es2024)

TypeScript

Thank you!

Slides & Code: https://github.com/ivorycode/react-galliker-2025

Discussion & Questions?

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