React Server Components

explained for backend developers

ABOUT ME

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

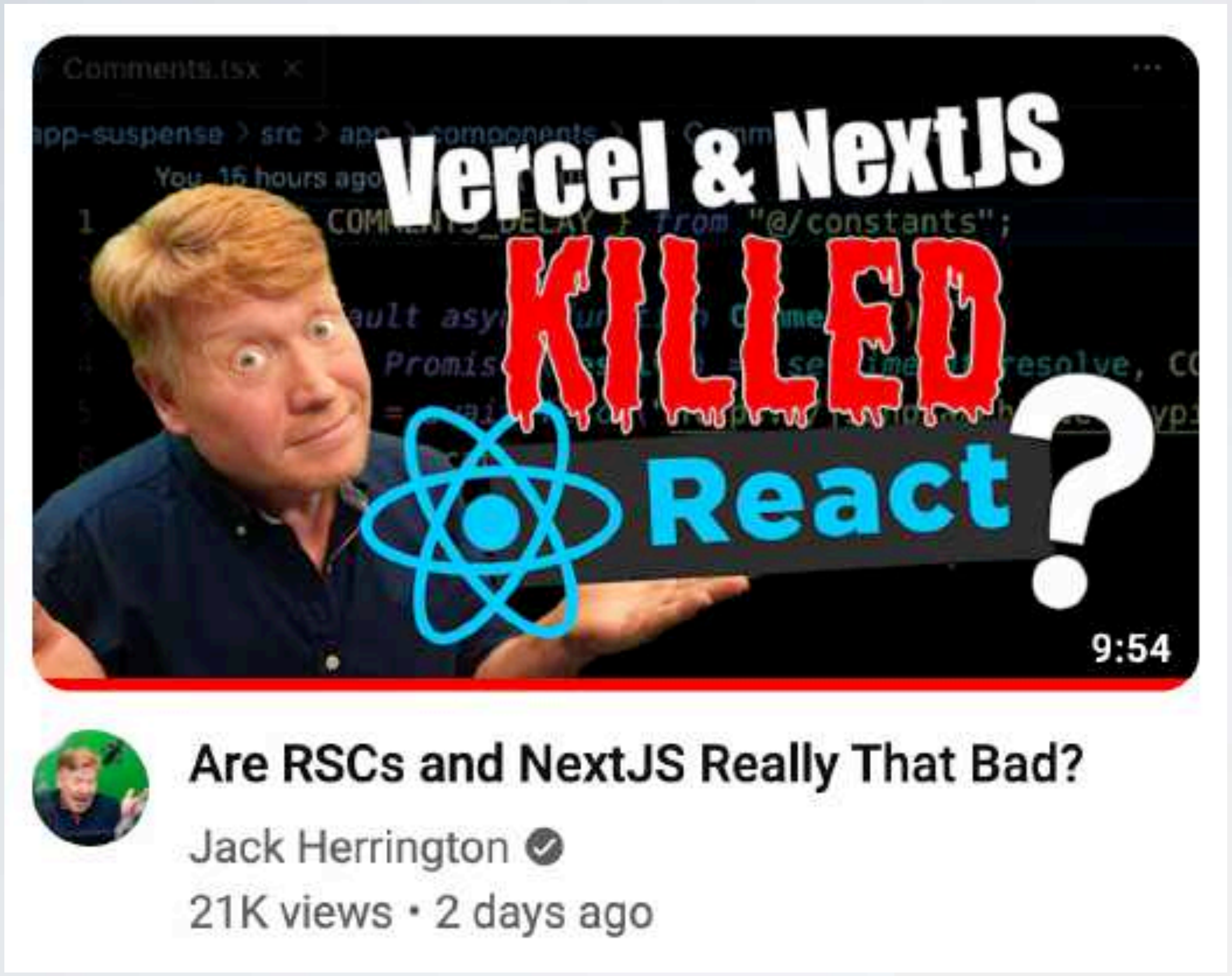
- Freelancer, in den letzten 9 Jahren vor allem in Projekten im Spannungsfeld zwischen modernen Webentwicklung und traditionellen Geschäftsanwendungen.
- Dozent an der Berner Fachhochschule seit 2007
- In-House Kurse & Beratungen zu Web-Technologien im Enterprise: UBS, Postfinance, Mobiliar, AXA, BIT, SBB, Elca, Adnovum, BSI ...

letsboot.ch
swiss dev training

digicomp

Berner Fachhochschule
Haute école spécialisée bernoise
Bern University of Applied Sciences

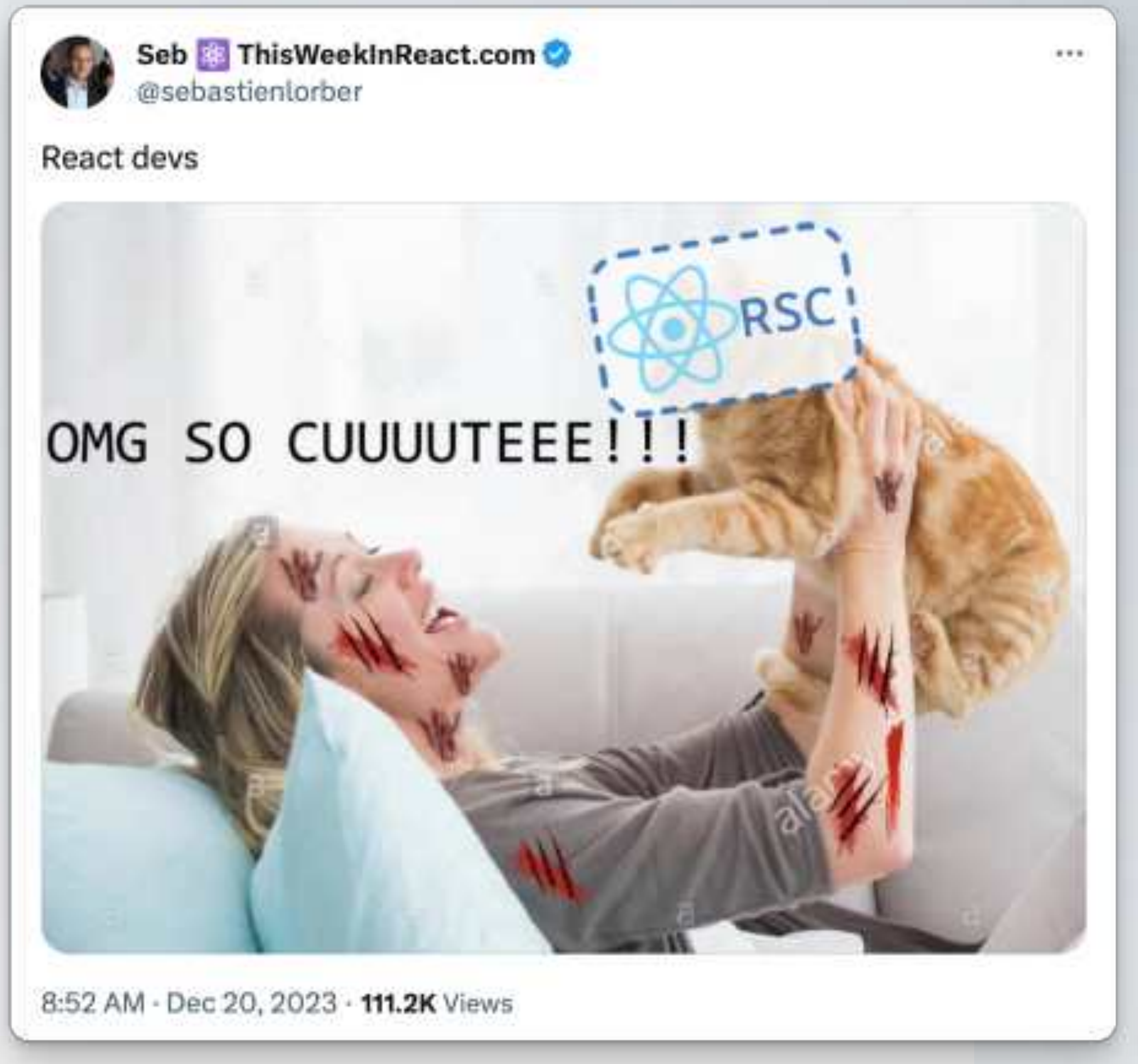
JavaScript / Angular / React / Vue / Vaadin
Schulung / Beratung / Coaching / Reviews
jonas.bandi@ivorycode.com



Are RSCs and NextJS Really That Bad?

Jack Herrington

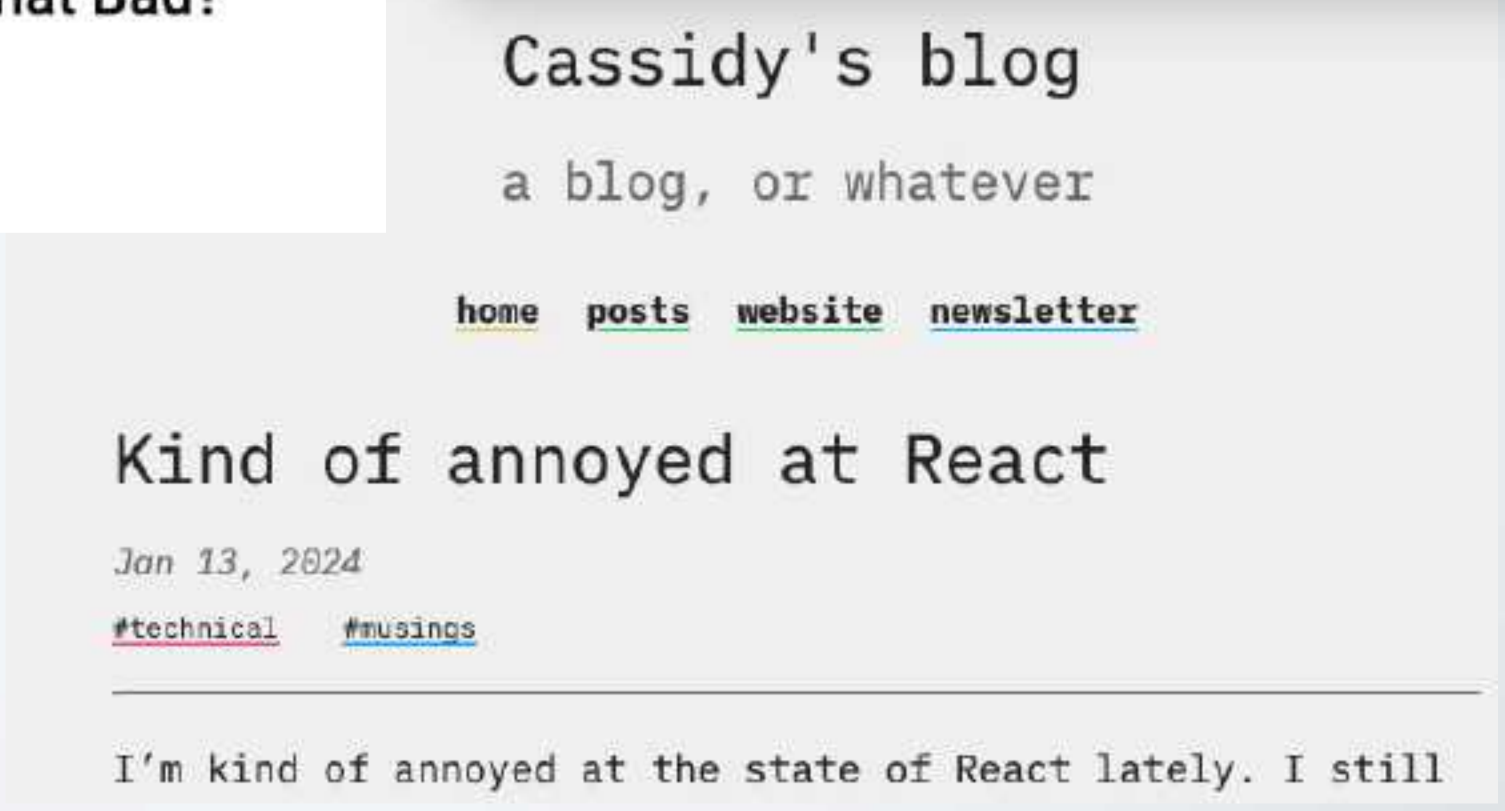
21K views • 2 days ago



Seb ThisWeekInReact.com @sebastienlorber

React devs

8:52 AM · Dec 20, 2023 · 111.2K Views



Cassidy's blog

a blog, or whatever

home posts website newsletter

Kind of annoyed at React

Jan 13, 2024

#technical #musings

I'm kind of annoyed at the state of React lately. I still

https://www.youtube.com/watch?v=u0OMdWJfdhg
https://blog.cassidoo.co/post/annoyed-at-react/

Refresher:

Good old React as we know it ...

React is now11years old! (released in May 2013).
jQuery was 7 years old, when React was released ...

React is Easy!

component

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

component

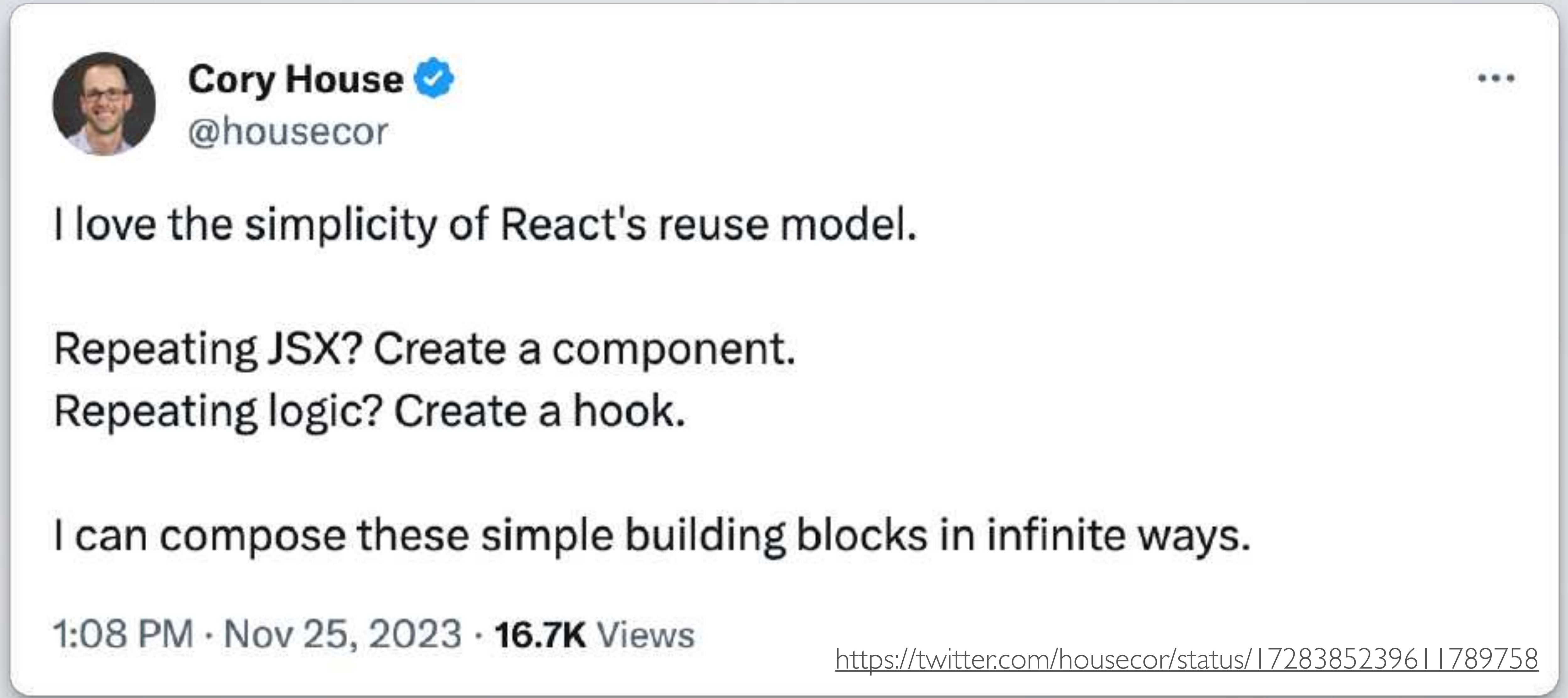
```
export function Greeter() {
   const useIdParamFromUrl();
   return <h1>Hello World!</h1>;
}
```

custom hook

```
export function useIdParamFromUrl() {
   const { userId } = useParams();
   return userId;
}
```

3rd party hook

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.

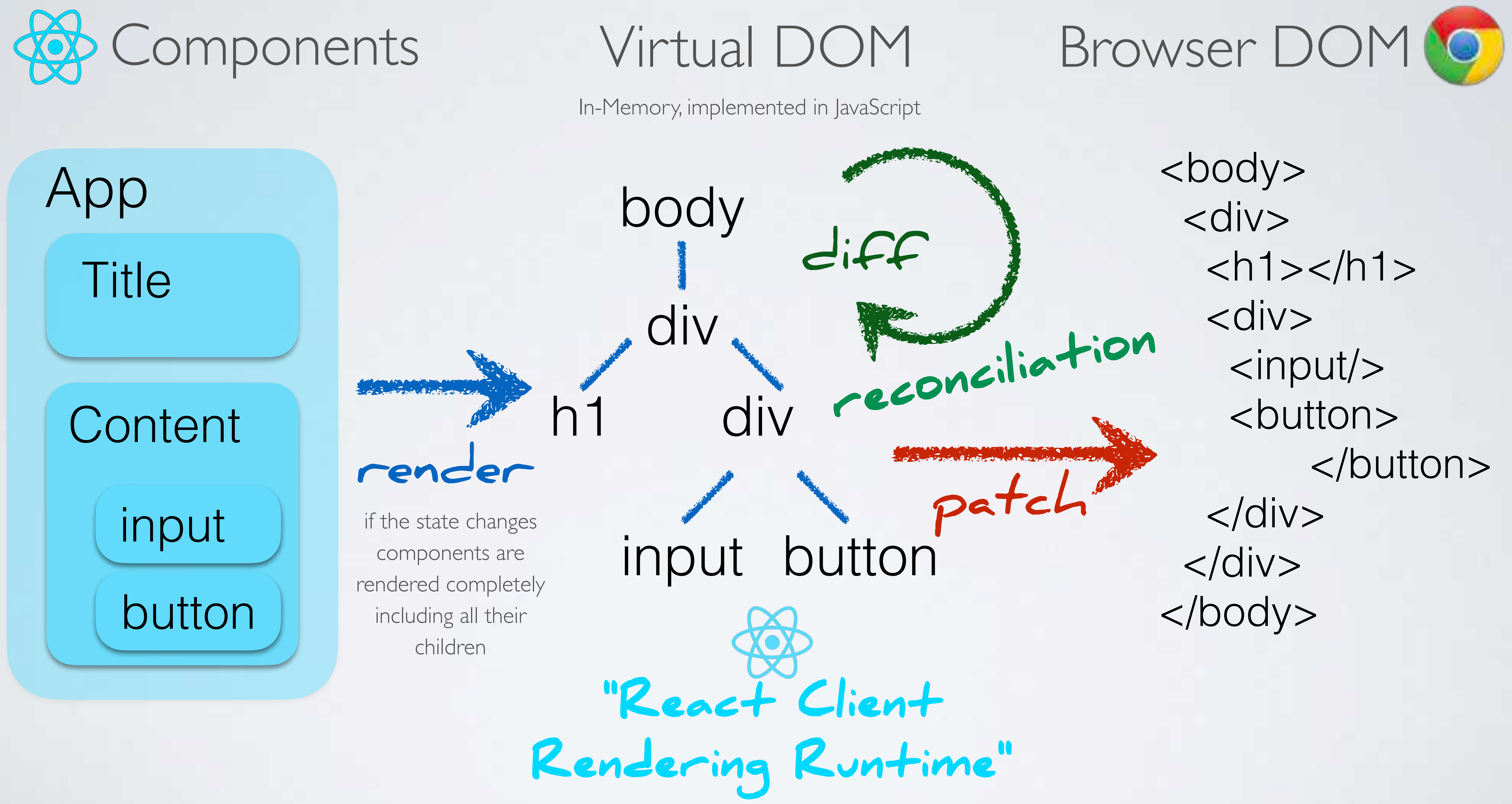
Repeating JSX? Create a component.
Repeating logic? Create a hook.

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 Virtual DOM



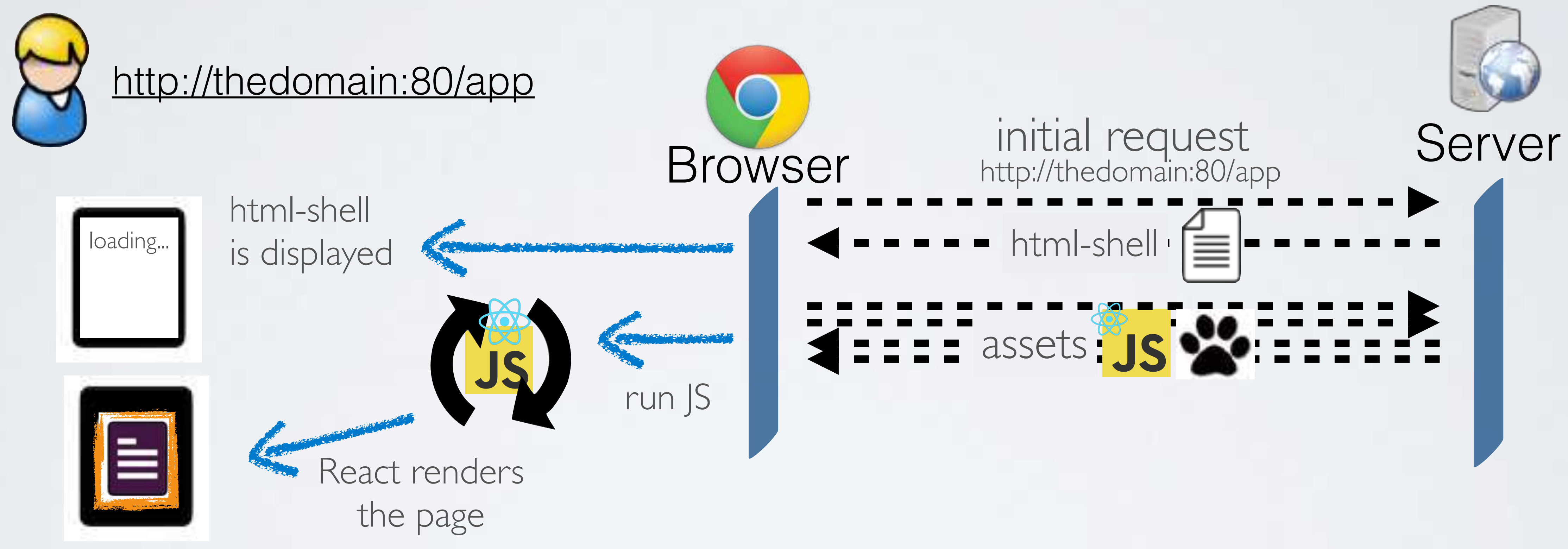
The Virtual DOM also enables server-side rendering and rendering to iOS/Android UIs.

React Data-Access

```
export function useApiData() {
  let ignore = false;
  const [data, setData] = useState("");

  useEffect(() => {
      async function fetchData() {
         const messageText = await fetchDataFromApi();
         if (!ignore) {
             setData(messageText);
         }
      }
      fetchData();

      return () => {
         ignore = true;
      };

  }, []);

  return data;
}
```

The sad face of React...

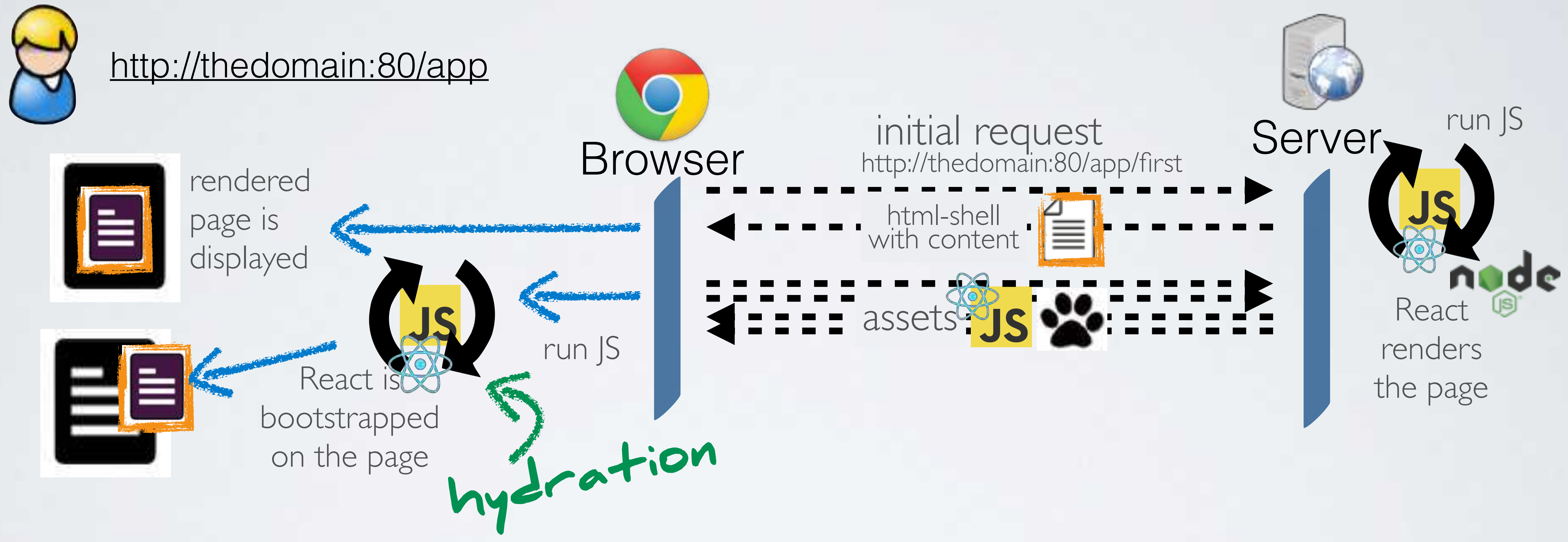
Traditional SPA: Client Side Rendering



Components are rendered on the client.

Server Side Rendering (SSR)

(initial rendering on the server - hydration on the client)



Components are rendered on the server and the client.

Advantages:
- SEO / social previews
- improving time to first contentful paint

This talk is not about Server Side Rendering!

Today every modern frontend framework is capable of server side rendering.

React Server Components is an Architecture!

I want you to forget everything you know about React!

Introducing:

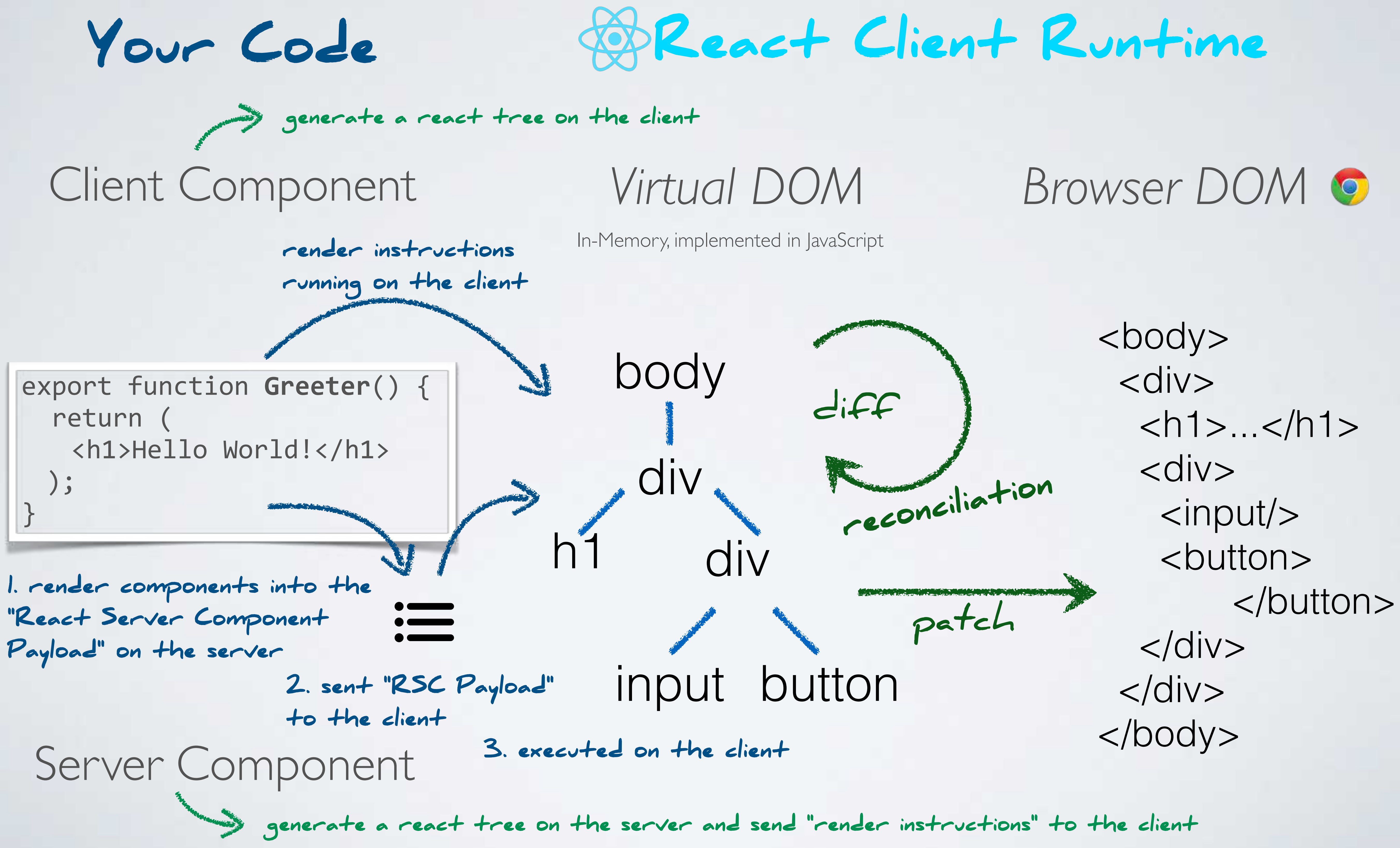
"React Server"

It's a React component ....

```
export function Greeter() {

  console.log("rendering Greeter");

  return (
    <div>
      <h1>Display of Greeter.</h1>
    </div>
  );
}
```

... but exclusively rendered on the server!

It is still a SPA!


Your Code
React Client Runtime
generate a react tree on the client
Client Component
Virtual DOM
In-Memory, implemented in JavaScript
Browser DOM
render instructions running on the client
body
div
h1
div
input
button
diff
reconciliation
patch
1. render components into the "React Server Component Payload" on the server
2. sent "RSC Payload" to the client
3. executed on the client
Server Component
generate a react tree on the server and send "render instructions" to the client

```
export function Greeter() {
  return (
    <h1>Hello World!</h1>
  );
}
```

```
<body>
  <div>
    <h1>...</h1>
    <div>
      <input/>
      <button>
          </button>
    </div>
  </div>
</body>
```

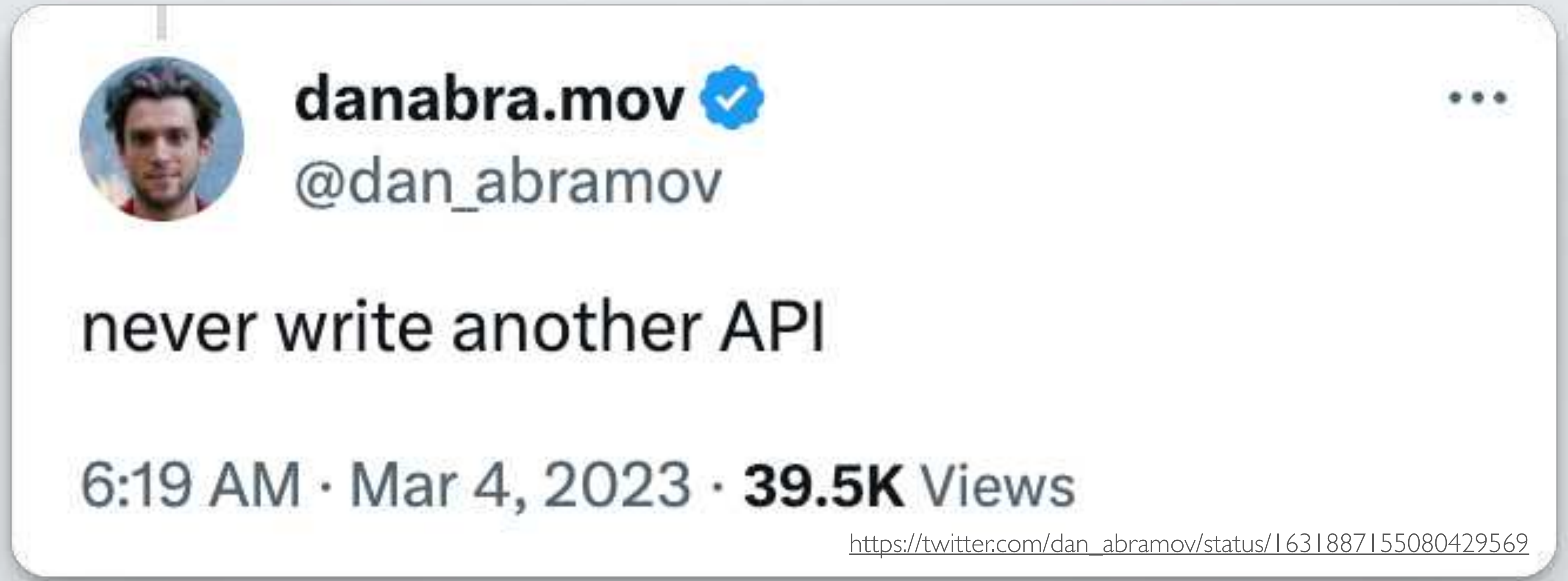
Load data on the server!

```
export async function Greeter() {

  const dataFromDb = await queryDataFromDb();

  return (
    <div>
      <h1>{dataFromDb}</h1>
    </div>
  );
}
```

Asynchronous rendering!

Making data fetching easy!

SPA data fetching without HTTP-API!


danabra.mov

@dan_abramov

never write another API

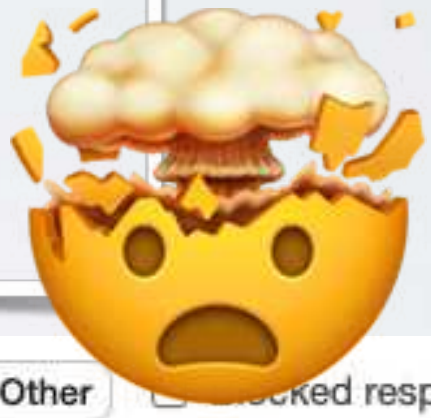
6:19 AM · Mar 4, 2023 · **39.5K** Views

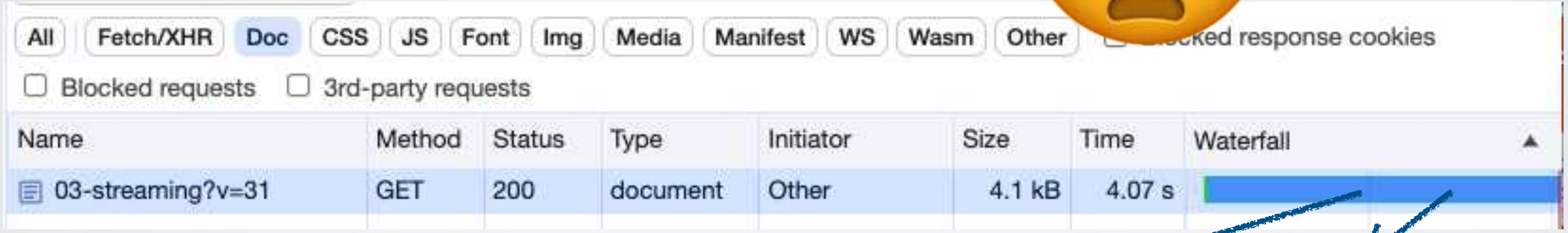
https://twitter.com/dan_abramov/status/1631887155080429569

Wouldn't that be tempting?

Out of Order Streaming

```
<h3>Server Data:</h3>
<Suspense fallback={<Spinner />}>
   <Backend messageId={1} />
</Suspense>
<Suspense fallback={<Spinner />}>
   <Backend messageId={2} />
</Suspense>
<Suspense fallback={<Spinner />}>
   <Backend messageId={3} />
</Suspense>
```

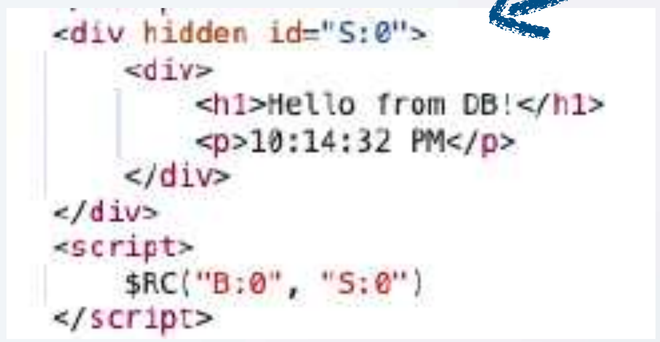




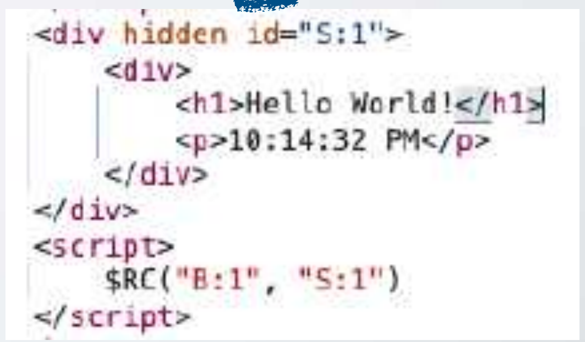
All | Fetch/XHR | Doc | CSS | JS | Font | Img | Media | Manifest | WS | Wasm | Other | ☐ [illegible]cked response cookies

☐ Blocked requests ☐ 3rd-party requests

Name	Method	Status	Type	Initiator	Size	Time	Waterfall ▲
03-streaming?v=31	GET	200	document	Other	4.1 kB	4.07 s	



```
<div hidden id="S:0">
    <div>
        <h1>Hello from DB!</h1>
        <p>10:14:32 PM</p>
    </div>
</div>
<script>
    $RC("B:0", "S:0")
</script>
```



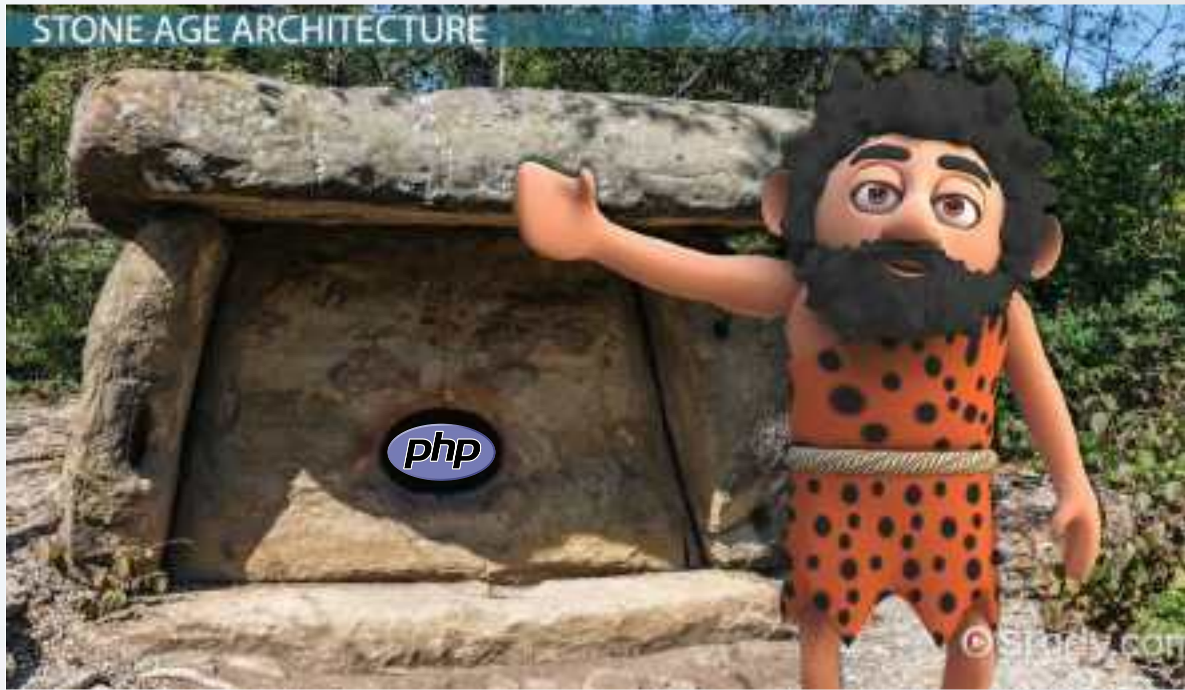
```
<div hidden id="S:1">
    <div>
        <h1>Hello World!</h1>
        <p>10:14:32 PM</p>
    </div>
</div>
<script>
    $RC("B:1", "S:1")
</script>
```

https://www.youtube.com/watch?v=23bHSDJD9y4

Wait!

🤯 ... 🤔

It looks like PHP from 25 years ago!



Prepare for more …

Limitations

React Server Components

... are rendered on the server only

- Can't use hooks:
  no state: **`useState`**, **`useReducer`**, **`useContext`**
  no lifecylce: **`useEffect`**
- Can't handle DOM events:
  `onClick`, **`onBlur`** ...
- Can't use browser APIs:
  `localstorage`, **`geolocation`** ...

React Client Components

... are rendered on the client and also initially on the server.

```
"use client"
export function Clock() {
  const [time, setTime] = useState(new Date())

   useEffect(() => {
     setInterval(() => setTime(new Date())}, 1000);
   }, []);

  return (
    <div>
      <h1>{time.toLocalTimeString()}</h1>
    </div>
  );
}
```

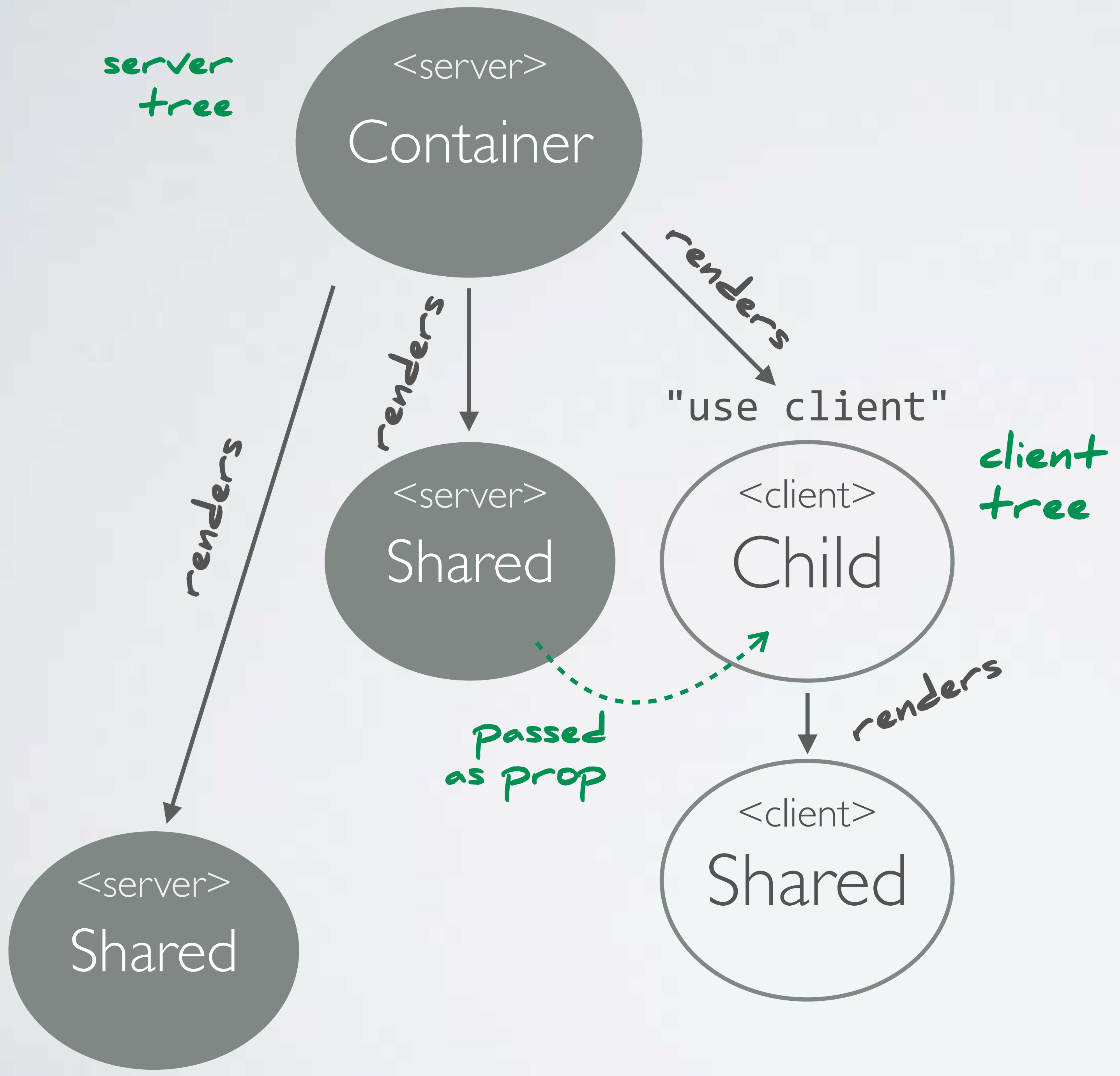
Client Components are "opt in".
Per default a component is a Server Component.

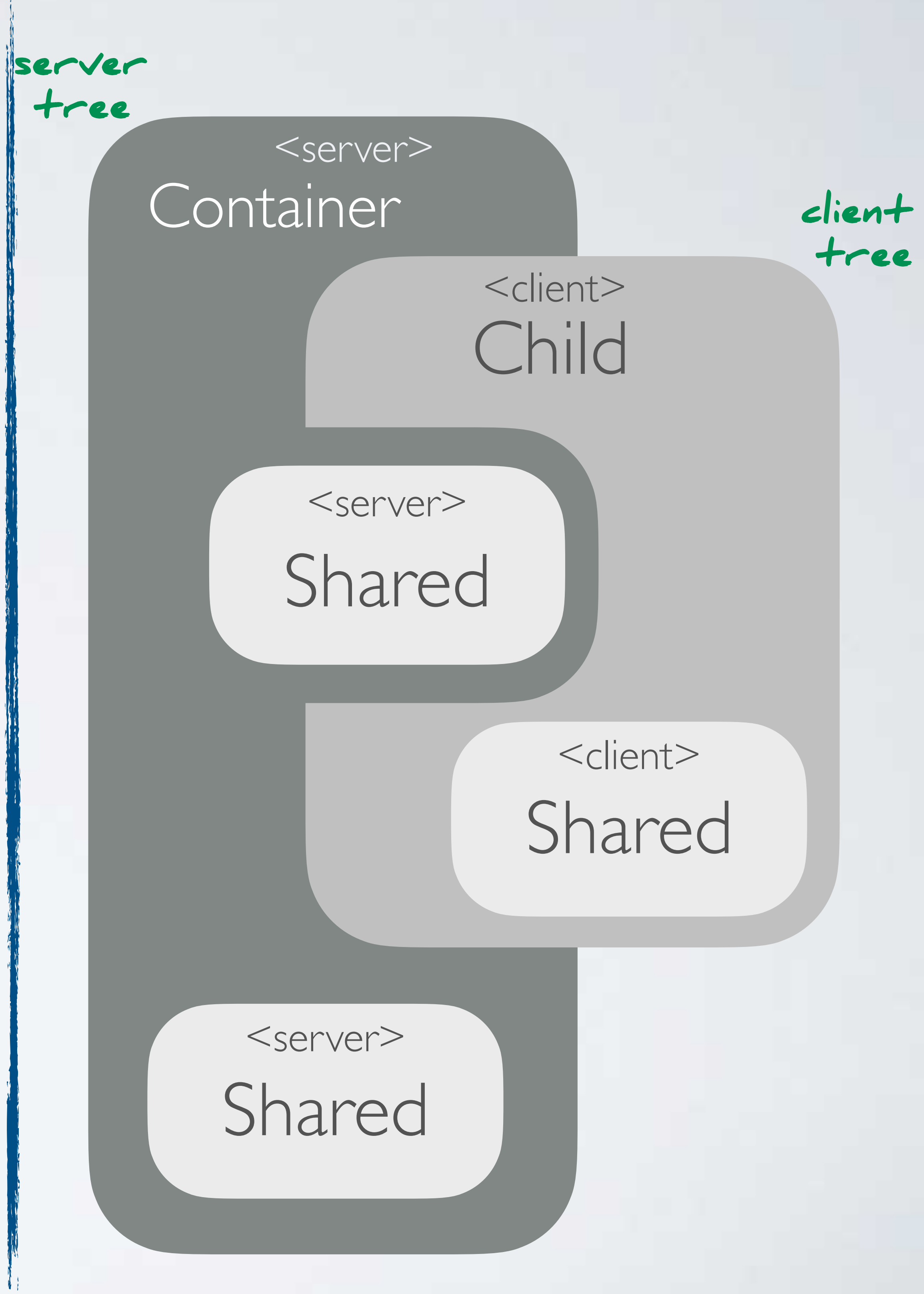
Introducing:

"React Server"

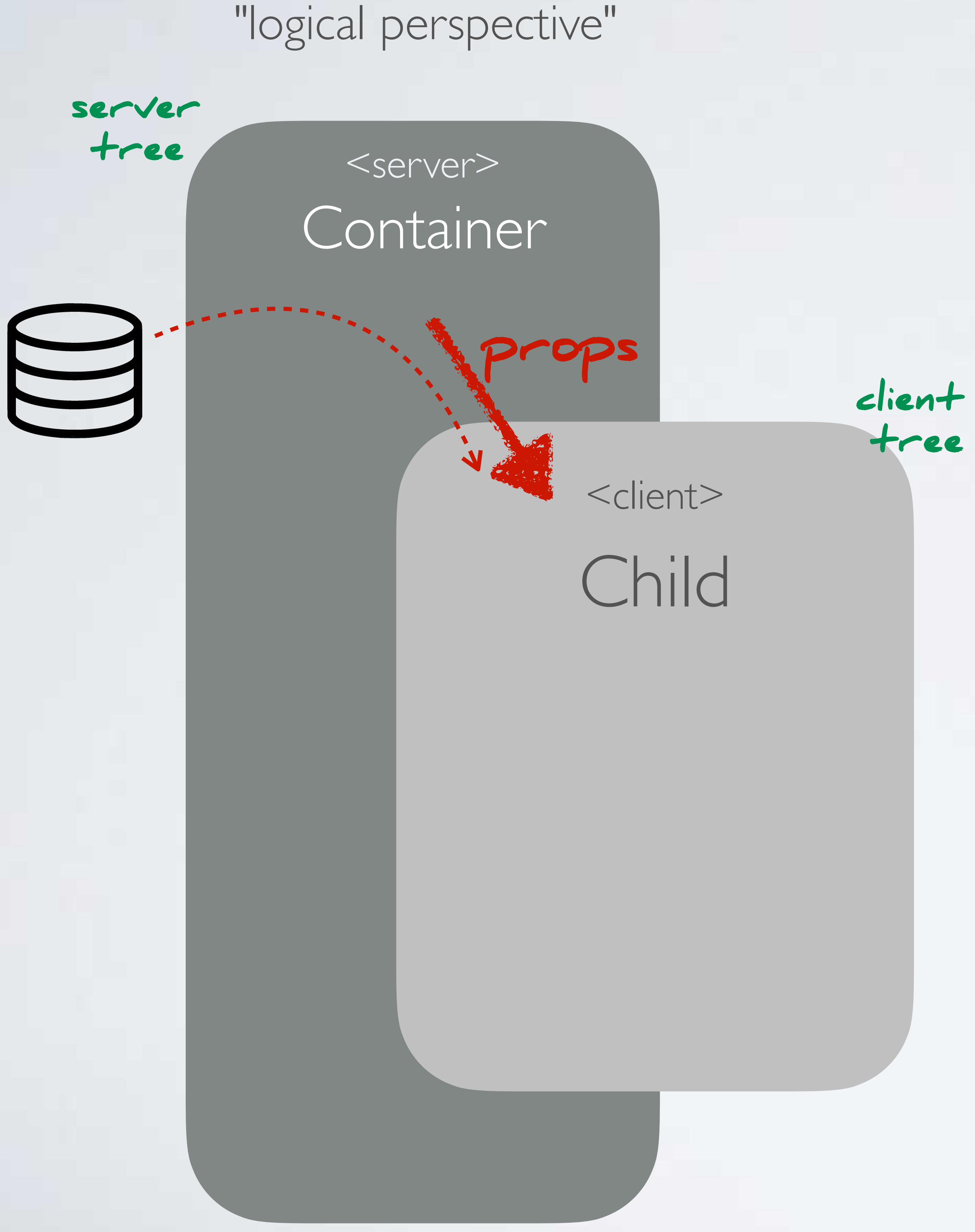
+ (traditional) Components

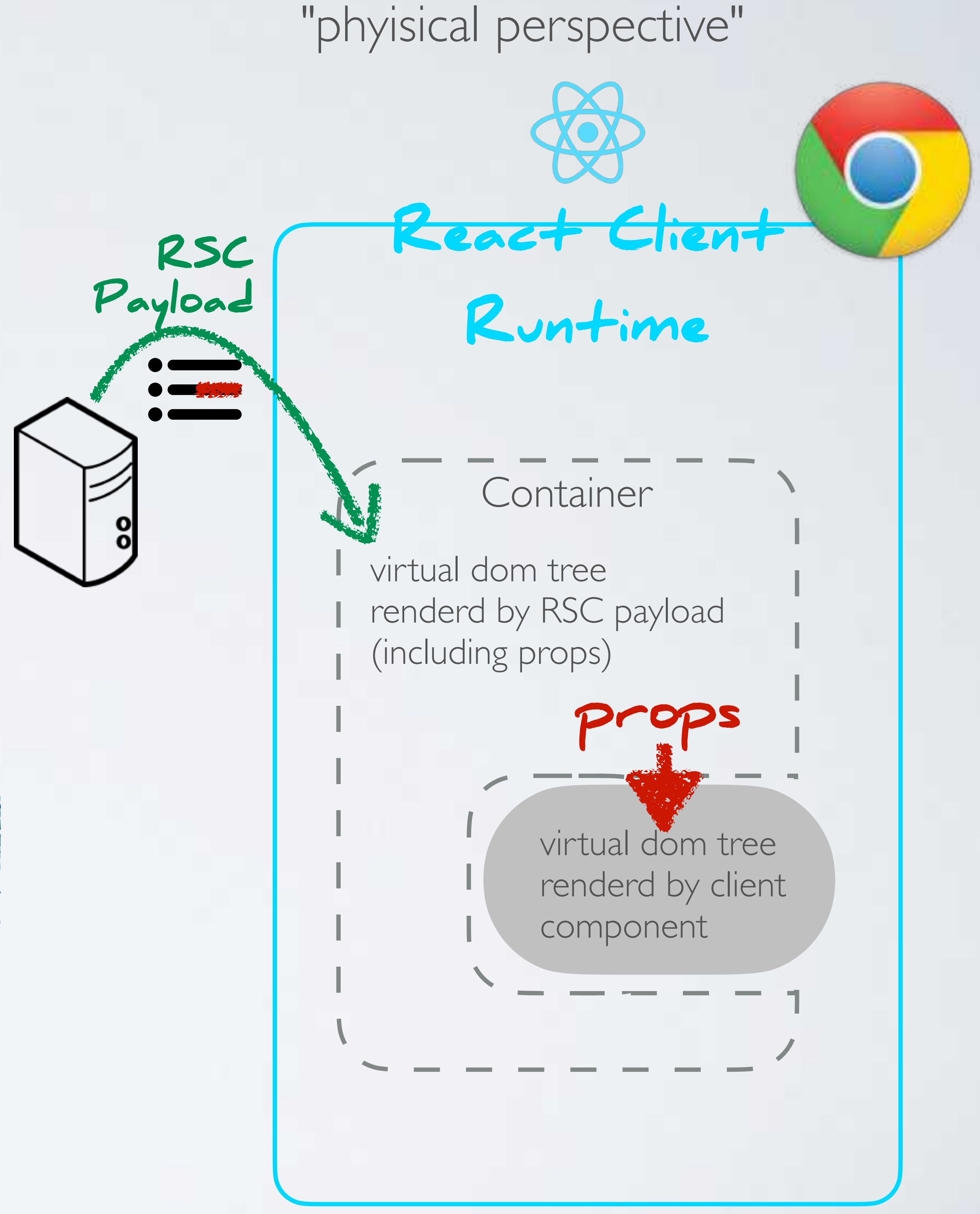
Composition

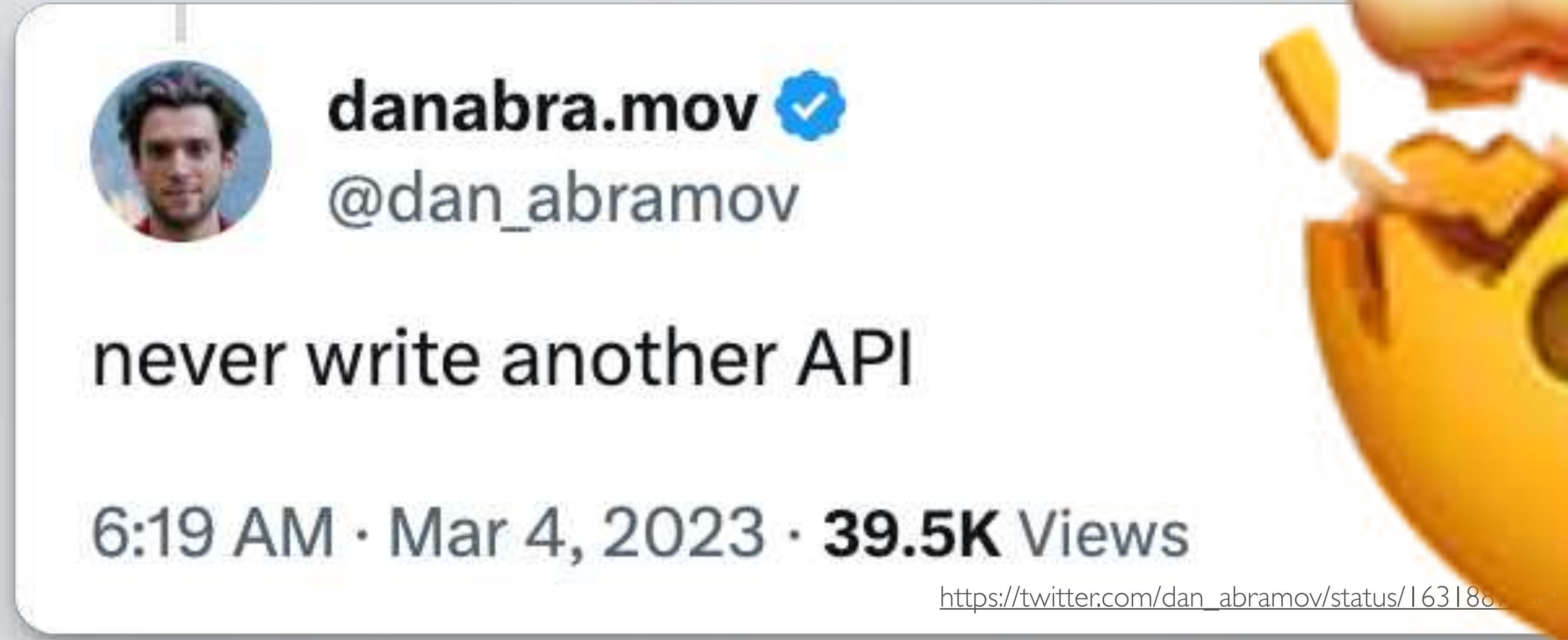




Full-Stack Data Flow




"phyisical perspective"
React Client Runtime
RSC Payload
Container
virtual dom tree renderd by RSC payload (including props)
props
virtual dom tree renderd by client component



danabra.mov
@dan_abramov

never write another API

6:19 AM · Mar 4, 2023 · **39.5K** Views

https://twitter.com/dan_abramov/status/163188

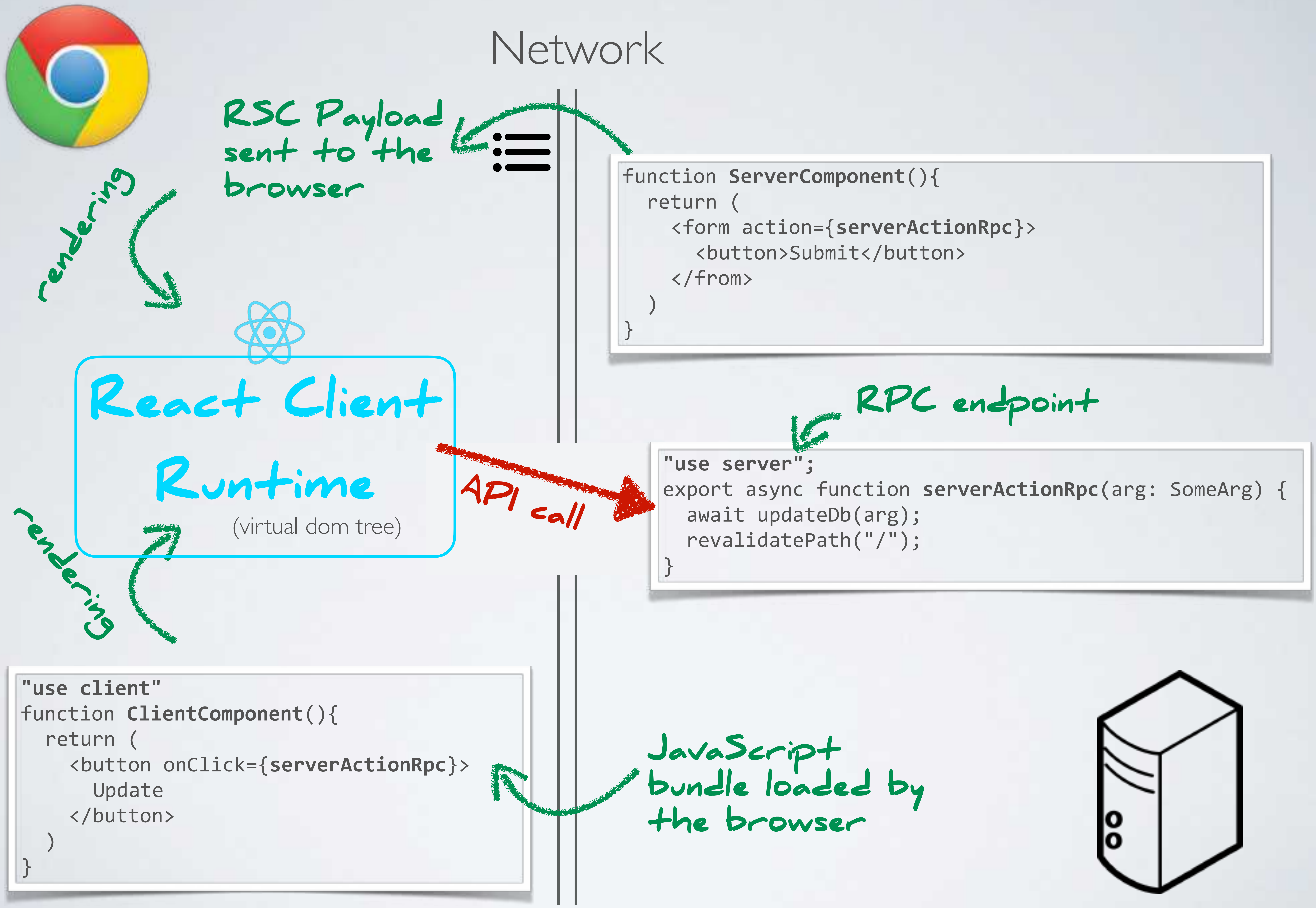
In case you did not believe it
the first time ...

Introducing:

"React Server"

+ Client Components

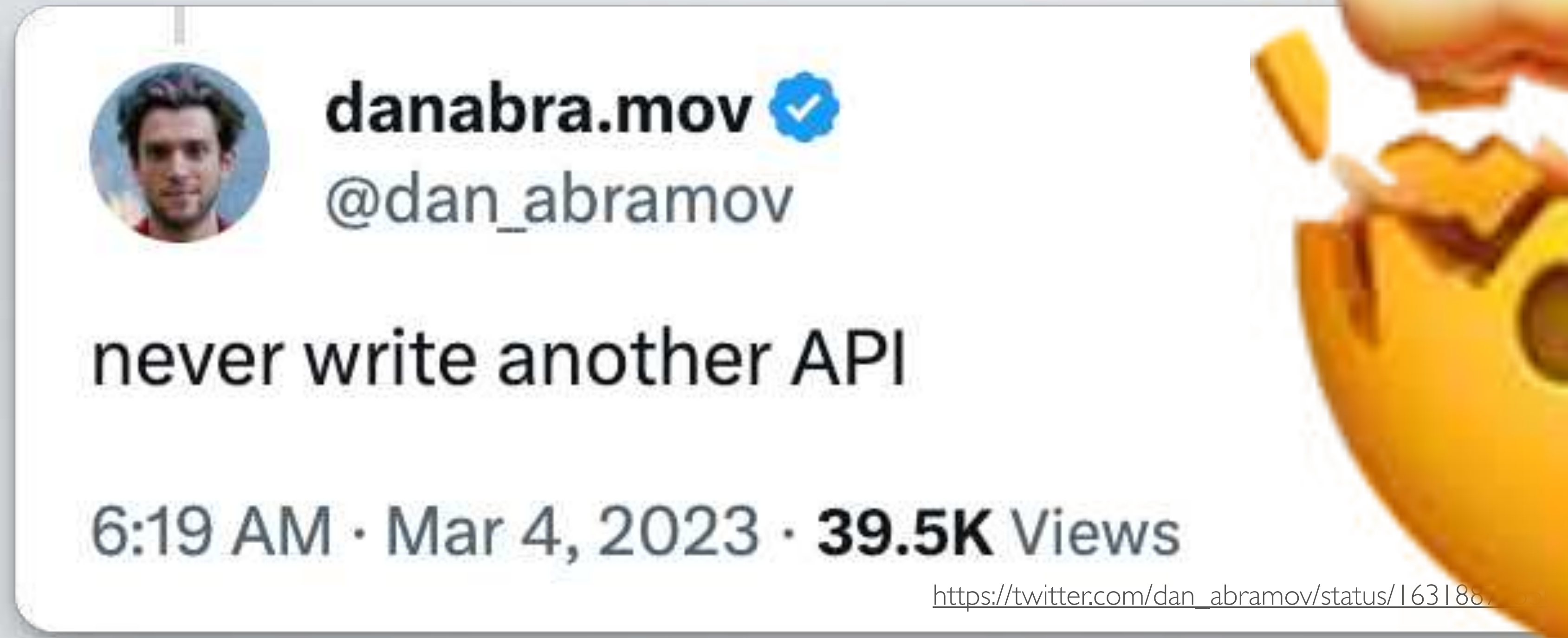
+ Server Actions



```
function ServerComponent(){
  return (
    <form action={serverActionRpc}>
      <button>Submit</button>
    </from>
  )
}
```

```
"use server";
export async function serverActionRpc(arg: SomeArg) {
  await updateDb(arg);
  revalidatePath("/");
}
```

```
"use client"
function ClientComponent(){
  return (
    <button onClick={serverActionRpc}>
      Update
    </button>
  )
}
```



danabra.mov

@dan_abramov

never write another API

6:19 AM · Mar 4, 2023 · **39.5K** Views

https://twitter.com/dan_abramov/status/163188

In case you need the repetition:

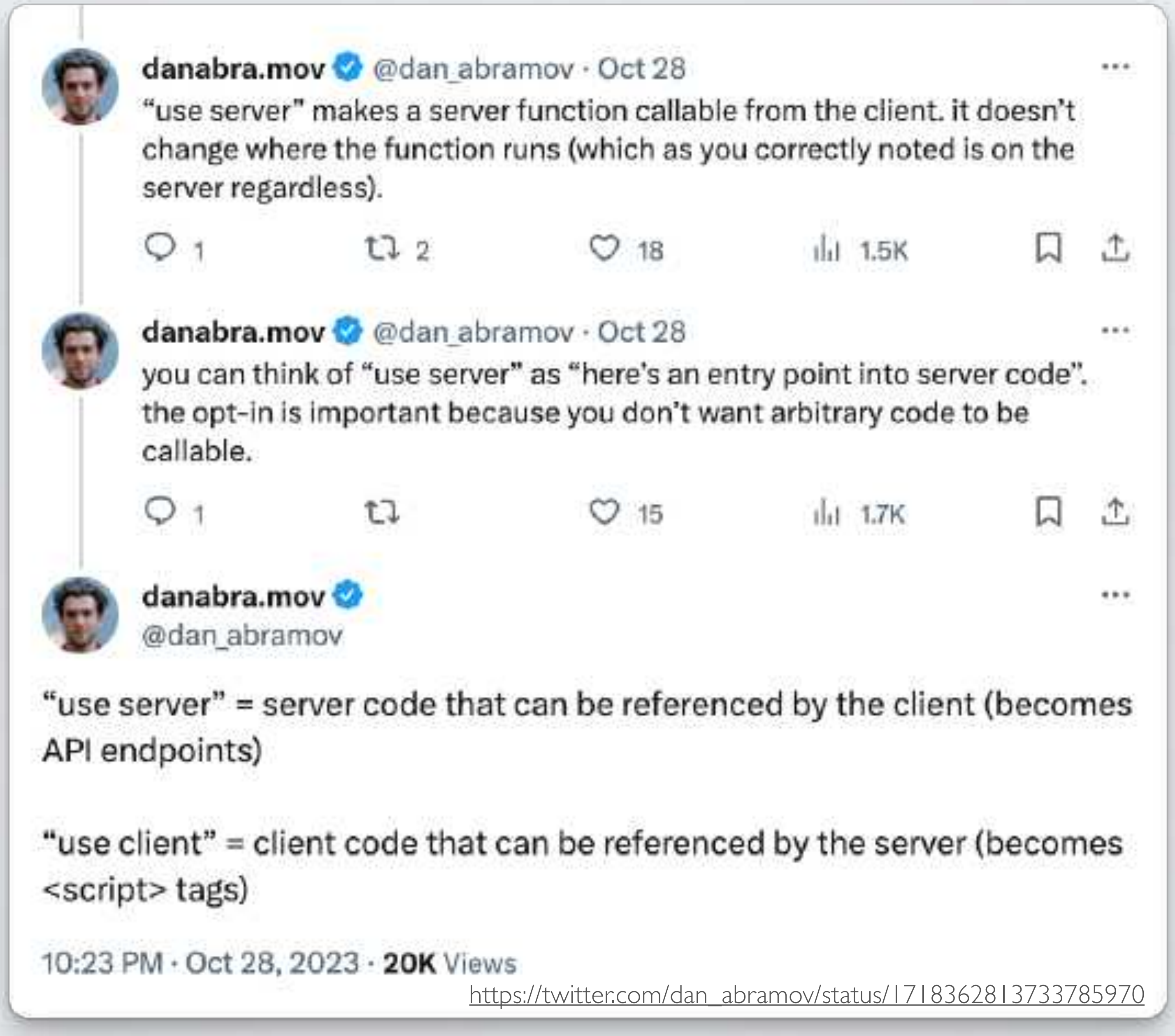
No HTTP-API for fetching nor mutating ....

Server … Client … it's confusing …

```
'use client'
'use server'
```

network boudary,
js bundle shipped to client

network boudary,
RPC endpoint called by client



danabra.mov @dan_abramov · Oct 28
"use server" makes a server function callable from the client. it doesn't change where the function runs (which as you correctly noted is on the server regardless).

1 2 18 1.5K

danabra.mov @dan_abramov · Oct 28
you can think of "use server" as "here's an entry point into server code". the opt-in is important because you don't want arbitrary code to be callable.

1 15 1.7K

danabra.mov
@dan_abramov

"use server" = server code that can be referenced by the client (becomes API endpoints)

"use client" = client code that can be referenced by the server (becomes <script> tags)

10:23 PM · Oct 28, 2023 · **20K** Views

https://twitter.com/dan_abramov/status/1718362813733785970

Summary

"React Server Components"

- is a full-stack architecture
- is based on the proven component model of React
- extends the composability patterns of React to the server-side
- solves client-server communication with a consistent programming model based on components transparent RPC
- is the answer for data-fetching and mutations in React
- has huge potential for an ecosystem of 3rd party full-stack components

- also improves performance by enabling smaller JS bundles and streaming server responses.

Disclaimer

NEXT.JS

The demos in this talk were based on Next.js. Next.js is currently the only mature framework that implements React Server Components.
https://nextjs.org/

In reality it is difficult (and frustrating) to draw the boundary between features of React Server Components Next.js.

Waku

Waku is an experimental framework that implements RSCs.
https://waku.gg/

Remix

Remix announced RSC integration in a future version.
https://remix.run/

Thank you!

Slides & Code: https://github.com/jbandi/voxxed-rsc

Questions? Discussions ...

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