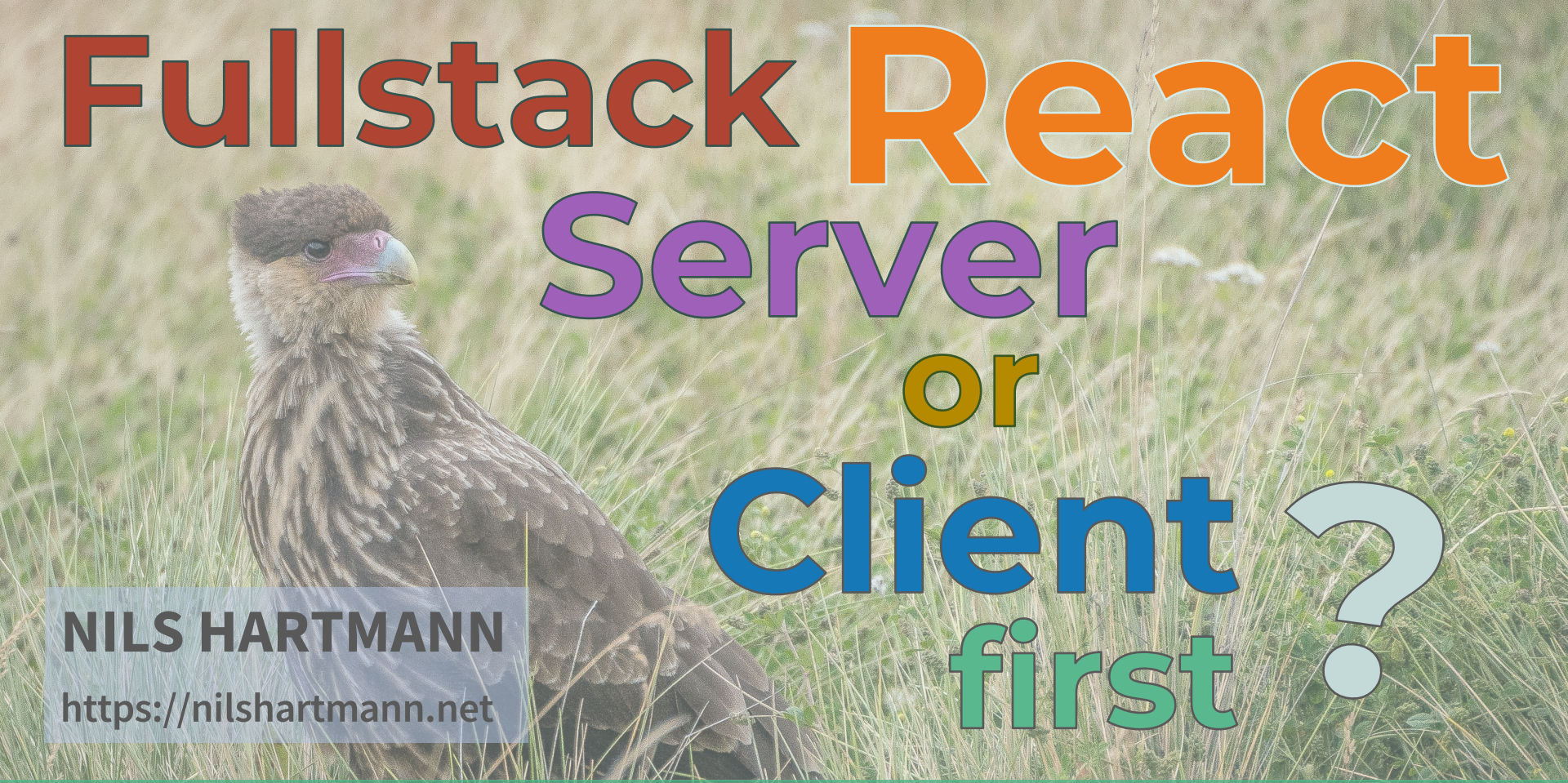
Fullstack React

Server or Client first?



NILS HARTMANN

https://nilshartmann.net

C‘T WEBDEV | COLOGNE | NOVEMBER 19, 2025

Cast your vote:

After Lunch Talk 

Talk 

or nap  ?

(Don‘t want to influence your decision, but you get the slides and source after the conference anyway…)

NILS HARTMANN

nils@nilshartmann.net

Freelance Software Developer

Java, Spring, GraphQL, React, TypeScript

https://graphql.schule/video-kurs

https://reactbuch.de

Fullstack React Server Client first ?

What is that? 🙀

NILS HARTMANN

https://nilshartmann.net

„Fullstack“ React

My definition

Client-first

„Fullstack“ React

My definition

Client-first

- An application contains UI/Rendering code on both **server** and **client** side

„Fullstack“ React

My definition

Client-first

- An application contains UI/Rendering code on both **server** and **client** side
- For both sides **a** framework is used

„Fullstack“ React

My definition

Client-first

- An application contains UI/Rendering code on both **server** and **client** side
- For both sides **a** framework is used
- Framework contains solutions for rendering, routing, data fetching, ...

„Fullstack“ React

My definition

Client-first

- An application contains UI/Rendering code on both **server** and **client** side
- For both sides **a framework** is used
- Framework contains solutions for rendering, routing, data fetching, ...

💡 React it self provides **low-level APIs** for server-side features

- But: you need a **framework** to make use of it

Our

Candidates


Client-first

Client-first

TanStack Start

https://tanstack.com/start

Currently RC

Vite Plug-in

Supports: serverside rendering (SSR), streaming, server functions

No React Server Components (yet)

Client-first

TanStack Start = Router + Server

Idea:

Build a „traditional“ SPA with TanStack Router

Move to serverside by adding TanStack Start

(ofc you can start with Start from the beginning)

Only parts of your logic run on serverside

Background

TanStack Router

https://tanstack.com/router

Alternative / replacement for React Router

Can be used for „traditional“ client-side SPAs

Vite Plug-in, file-based routing, exceptional TypeScript support

Server-first

Server-first

Next.js

https://nextjs.org

Currently Next.js 16, based on newest React 19.2

Support for all most all client and server React features

Including React Server Components (RSC) and Server Functions

Server-first

Next.js

Idea

(Almost*) all of your app runs on serverside

Only (small*) parts run on the client

Aggressive caching and performance optimizations for fastest UI

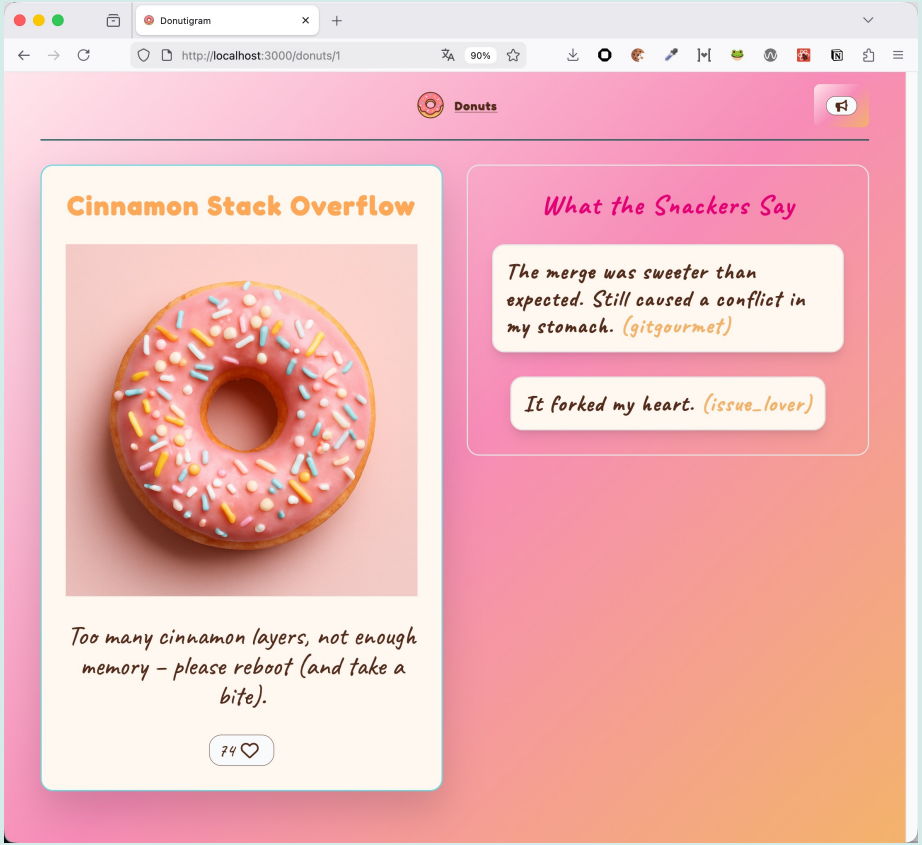
* My interpretation / opinion

Demo

Time

🔎 Our app...

http://localhost:3000



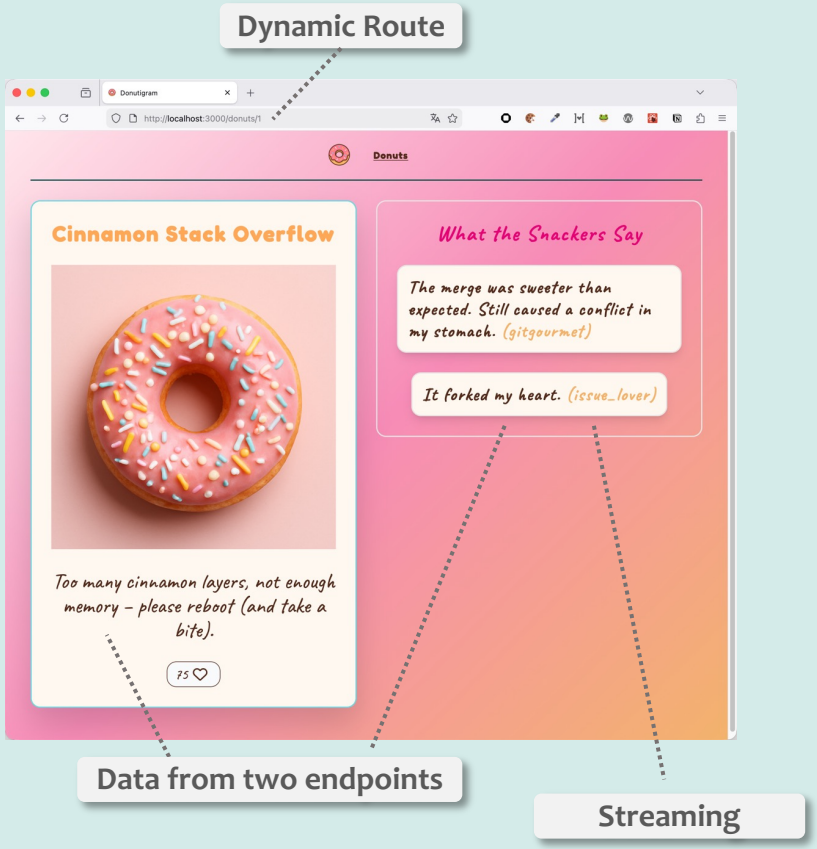
Demo 1

The „important“ features in action

Routing

Data Fetching

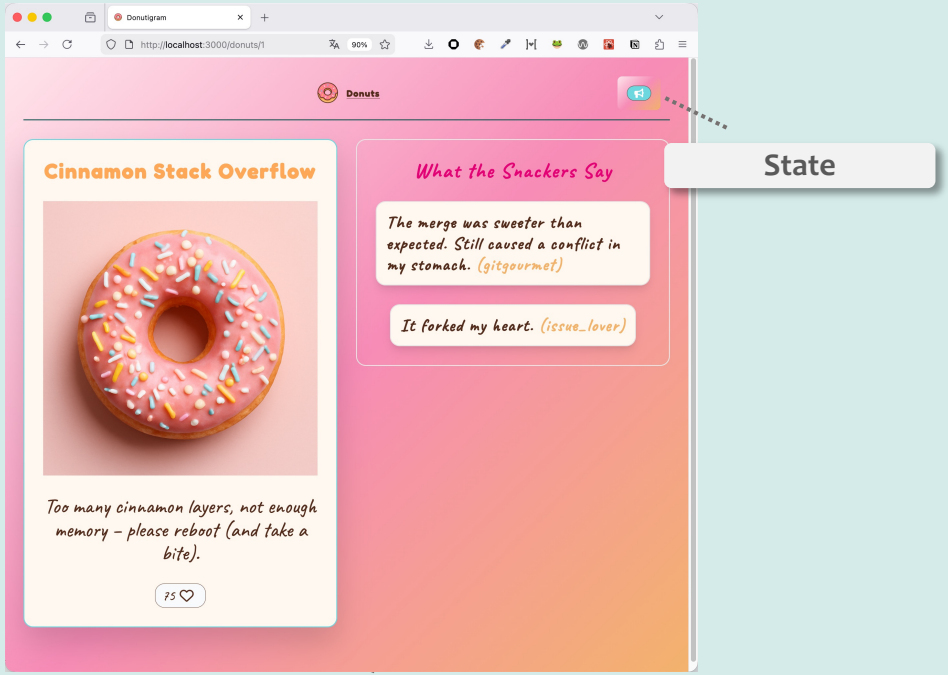
Serverside rendering



Demo 2

Interactivity

Client-side state

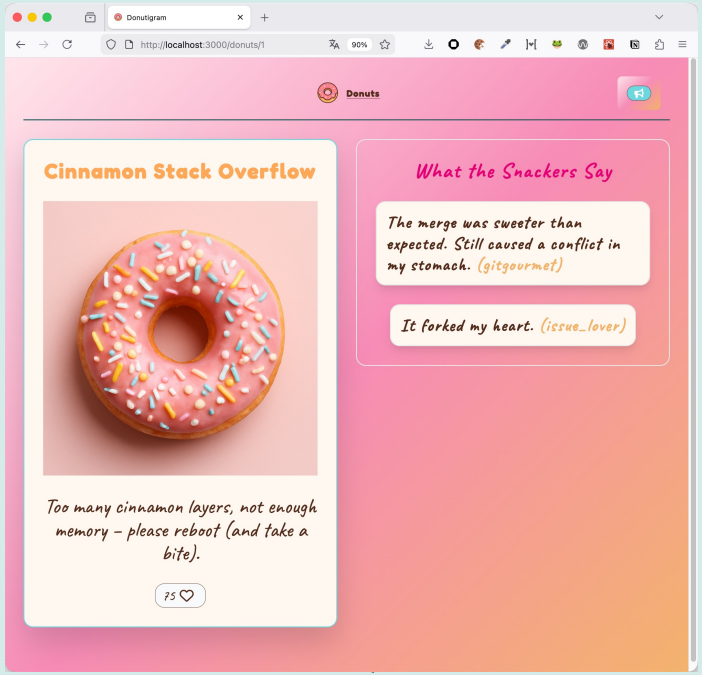


Demo 3

Server functions

aka

Server actions



Side

by

Side

COMPARISON

Server-side rendering (SSR)

TanStack Start

Client-first

- „Classic“ SSR
- Only initial request is server-side rendered
- HTML is returned
- Complete JavaScript Code of the App is returned and hydrated

Server-side rendering (SSR)

Next.js

Server-first

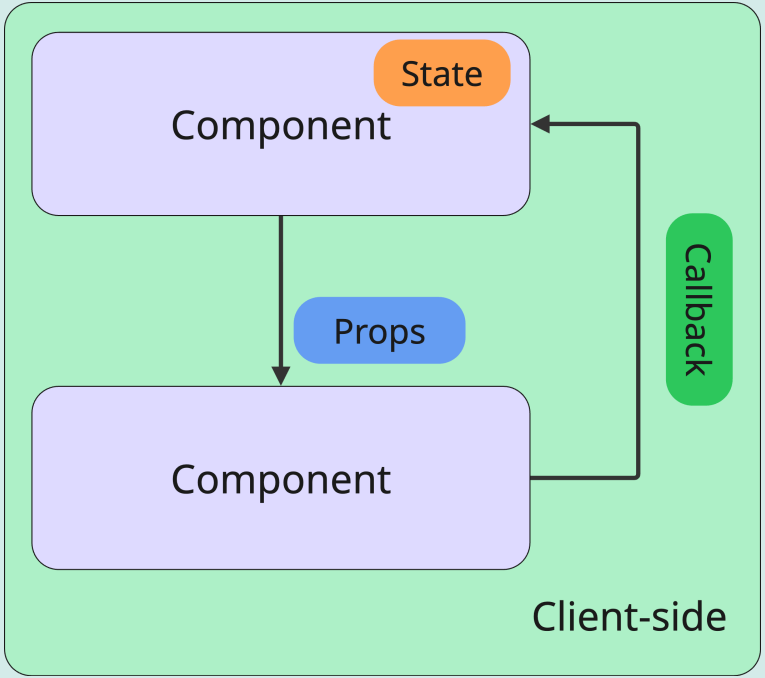
- „Classic“ SSR plus React Server Components (RSC)
- First request returns HTML
- RSC Payload is returned on all request
- Only Client Components are hydrated
- Any navigation (route change) is server request!
- State in Client Components survives

Architecture

TanStack Start

Client-first

- You **data** is available in the client
- You code follows „traditional“ React architecture
- Typical render cycle

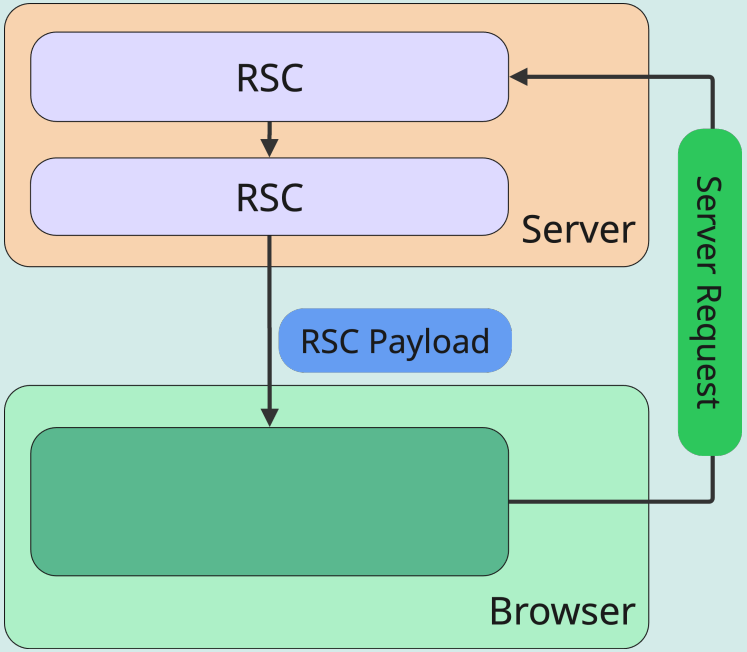


Architecture

Next.js

Server-first

- Typically you have „rendered UI“ on client
- No data (exception: client components)
- To re-render, server roundtrip might be neccessary
- Example:
  - Re-ordering a list



Global state

TanStack Start

Client-first

- Works with any global state solution, as state is only on the client
- For populating global state during SSR, your library must support it

Global state

Next.js

Server-first

- You can use global state for client components
- Somehow complicated: we have server round trips on any navigation
- Client-side global state can differ from server
- Example:
  - On navigation on server side new data is read and rendered
  - Client-side global state doesn‘t know about that
  - UI is inconsistent

Routing

TanStack Start

Client-first

- Based on TanStack Router (replacement for React Router)
- Vite plug-in
- Filebased routing
- Highly typesafe for routes, params and search params

Routing

Next.js

Server-first

- Built-in file-based Routing
- Params and search params and are promises passed as props

Data fetching

TanStack Start

Client-first

- Use library of your choice (typically: TanStack Query)
- Data can be pre-loaded in loader functions of each Route
- Support of Suspense and Streaming

Data fetching

Next.js

Server-first

- Typically in async React Server Components
- When using in Client Components, it‘s up to you
- You can load data for Client Components in RSC and pass data as props
  - A few restrictions for properties apply (serializable!)
  - You can even pass promises from server to client

Server functions

TanStack Start

Client-first

- Defines explicit API (function) for defining server functions
  - You can specify http method (post or get)
  - Handler for validating input arguments
- Almost all argument types are supports
- When calling your server function you need to pass a single argument
- You can call your server function wherever you want

Server functions

TanStack Start

Client-first

Server

```
const likeAction = createServerFn({ method: "POST" })

  .inputValidator((data) => {
      if (typeof data === "string") {
        return data;
      }

      throw new Error("Invalid data");
  })

  .handler(async ({ data: donutId }) => {
      const response = await ky.put(`...`)
      return DonutDto.parse(response);
  });
```

Client

```
function doLike(donutId: string) {
  return saveLikeAction({
    data: donutId,
  });
}
```

Server functions

Next.js

Server-first

- „Official“ React Server Function (aka Server Actions)
- Server functions can be inlined in Server Components or files
- Defined using a „use server“ directive
- Only for write operations (do not call while rendering)

Static Side Generation (SSG)

TanStack Start

Client-first

- Supported
- Does not seem to be a first-class use case
- For generation, your complete side is crawled

Static Side Generation (SSG)

Next.js

Server-first

- Can be used to fully generate static pages in build
- You can even partially generate pages
- Aggressive (and complex...) caching

Pre-rendering/pre-loading

TanStack Start

Client-first

- Loaders are executed before rendering
- <Link /> components determines pre-loading behaviour
- Default behaviour can also be set globally

Pre-rendering/pre-loading

Next.js

Server-first

- Per-default enabled for all internal links
- Is that a good choice?
- To change behaviour you have to set the property on every <Link />-component

Summary

Summary

- First: Do you really need server-side React at all?

Summary

- First: Do you really need server-side React at all?
- Next.js:
  - Probably nearest to React ideas and APIs
  - Shines if you need aggressive caching, prerendering etc.
  - Can be tricky with traditional React patterns, like often changing global state

Summary

- First: Do you really need server-side React at all?
- Next.js:
  - Probably nearest to React ideas and APIs
  - Shines if you need aggressive caching, prerendering etc.
  - Can be tricky with traditional React patterns, like often changing global state
- TanStack Start:
  - APIs are more explizit (createServerFn)
  - Concept easier (?) without RSC (not „two worlds“)
  - Suitable for interaktive apps that needs SSR (SEO etc.)

NILS HARTMANN
https://nilshartmann.net

Thanks a lot!

Code & Slides: https://react.schule/ctwebdev2025-fullstack

Questions and contact: nils@nilshartmann.net

My workshops (de): https://nilshartmann.net/workshops

will publish the source code later or tomorrow (please be patient)