



NILS HARTMANN

<https://nilshartmann.net>

React

2023

Beginn einer

neuen

Ära?

Slides: <https://react.schule/ejs23>

NILS HARTMANN

nils@nilshartmann.net

Freiberuflicher Entwickler, Architekt, Trainer aus Hamburg

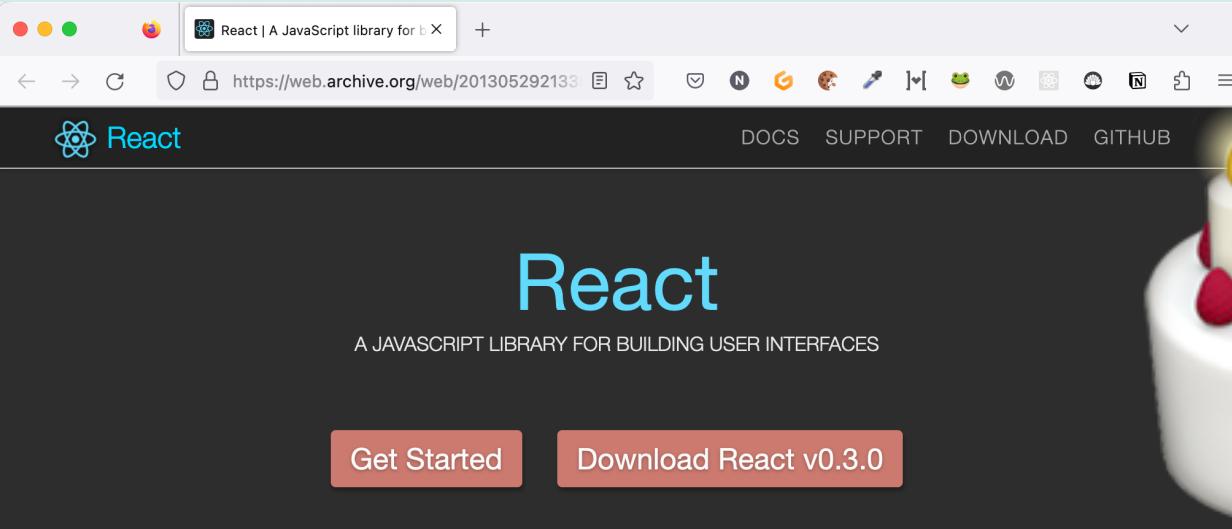
Java, Spring, GraphQL, React, TypeScript



<https://graphql.schule/video-kurs>

<https://reactbuch.de>

[HTTPS://NILSHARTMANN.NET](https://nilshartmann.net)



<https://web.archive.org/web/201305292133/http://facebook.github.io/react/>

React
A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

Get Started Download React v0.3.0

DECLARATIVE
React uses a declarative paradigm that makes it easier to reason about your application.

EFFICIENT
React minimizes interactions with the DOM by using a mock representation of the DOM.

FLEXIBLE
React works with the libraries and frameworks that you already know.



29.05.2013

VOR ZEHN JAHREN...

React v16.8: The One With Hooks

February 06, 2019 by [Dan Abramov](#)

With React 16.8, [React Hooks](#) are available in a stable release!

<https://legacy.reactjs.org/blog/2019/02/06/react-v16.8.0.html>

Go full-stack with a framework

React is a library. It lets you put components together, but it doesn't prescribe how to do routing and data fetching. To build an entire app with React, we recommend a full-stack React framework like [Next.js](#) or [Remix](#).

<https://react.dev/>

A screenshot of a web browser displaying a blog application titled "React Training Blog". The URL is "localhost:4001". The page shows three blog posts:

- Post 1:** Date: 10.01.2021, Title: **Keep calm and learn React!**, Preview: Pommy ipsum air one's dirty linen fork out plum pu..., Action: **Read this Blog Post**
- Post 2:** Date: 17.04.2020, Title: **Increasing React developer experience**, Preview: Tweeting a baseball.Sit on human they not getting ..., Action: **Read this Blog Post**
- Post 3:** Date: 02.04.2020, Title: **Using Redux with care**, Preview: Duis autem vel eum iriure dolor in hendrerit in vu..., Action: **Read this Blog Post**

To the right of the posts is a sidebar titled "Tags" containing the following categories:

- React
- Tutorial
- Bootstrap
- JavaScript
- Best Practice
- DX
- Context
- Redux
- State
- URL
- CSS
- Routing
- WebDev
- Marzipan

Code: <https://github.com/nilshartmann/fullstack-react-playground>

EIN BEISPIEL...

EIN BEISPIEL

Was macht die Beispiel-Anwendung aus?

- Viel statischer Content

Blog Posts

Create new Post

[Order by date Desc](#) [Order by date Asc](#)

10.01.2021

Keep calm and learn React!

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

[Read this Post](#)

Latest comment:

Disrupt inspire and think tank, social entrepreneur but preliminary thinking think tank compelling.

17.04.2020

Increasing React developer experience

Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

[Read this Post](#)

Latest comment:

Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

Tags

WebDev State

JavaScript Tutorial Context

DX Marzipan React

URL Redux Bootstrap

Best Practice Routing CSS

EIN BEISPIEL

Was macht die Beispiel-Anwendung aus?

- Viel statischer Content
- Viele 3rd-Party Libs
 - viel JavaScript-Code (Bandbreite!)

MomentJS!

The screenshot shows a blog application interface. At the top, there's a header with "Blog Posts" and a "Create new Post" button. Below the header, there are two blog post cards.

- Post 1:** Date: 10.01.2021, Title: **Keep calm and learn React!**, Preview: Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b..., Action: **Read this Post**.
- Post 2:** Date: 17.04.2020, Title: **Increasing React developer experience**, Preview: Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg..., Action: **Read this Post**.

To the right of the posts is a sidebar titled "Tags" containing a tag cloud. Red arrows point from the text labels to specific parts of the sidebar:

- A red arrow points from the text "React!" to the "React" tag in the sidebar.
- A red arrow points from the text "tag-cloud.js" to the entire tag cloud area.
- A red arrow points from the text "Marked!" to the "Marked" tag in the sidebar.
- A red arrow points from the text "MomentJS!" to the date "17.04.2020" of the second post, which uses MomentJS for date parsing.

EIN BEISPIEL

Was macht die Beispiel-Anwendung aus?

- Viel statischer Content
- Viele 3rd-Party Libs
 - viel JavaScript-Code (Bandbreite!)
- ...aber nur minimale Benutzer-Interaktionen (PostEditor)

Blog Posts

[Order by date Desc](#) [Order by date Asc](#)

10.01.2021

Keep calm and learn React!

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

[Read this Post](#)

Latest comment:
Disrupt inspire and think tank, social entrepreneur but preliminary thinking think tank compelling.

17.04.2020

Increasing React developer experience

Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

[Read this Post](#)

Latest comment:
Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

[Create new Post](#)

Tags

WebDev State
JavaScript Tutorial Context
DX Marzipan React
URL Redux Bootstrap
Best Practice Routing CSS

EIN BEISPIEL

Was macht die Beispiel-Anwendung aus?

- Data Fetching
 - Laden der Posts von externen Services



Probleme von klassischen Single-Page Anwendungen (lt. React-Team)

Even if you don't need routing or data fetching at first, you'll likely want to add some libraries for them. As your JavaScript bundle grows with every new feature, you might have to figure out how to split code for every route individually. As your data fetching needs get more complex, you are likely to encounter server-client network waterfalls that make your app feel very slow. As your audience includes more users with poor network conditions and low-end devices, you might need to generate HTML from your components to display content early—either on the server, or during the build time. Changing your setup to run some of your code on the server or during the build can be very tricky.

<https://react.dev/learn/start-a-new-react-project>

Routing und Data Fetching benötigen Bibliotheken

Even if you don't need routing or data fetching at first, you'll likely want to add some libraries for them. As your JavaScript bundle grows with every new feature, you might have to figure out how to split code for every route individually. As your data fetching needs get more complex, you are likely to encounter server-client network waterfalls that make your app feel very slow. As your audience includes more users with poor network conditions and low-end devices, you might need to generate HTML from your components to display content early—either on the server, or during the build time. Changing your setup to run some of your code on the server or during the build can be very tricky.

<https://react.dev/learn/start-a-new-react-project>

JavaScript-Code (im Browser) wächst mit jedem Feature

Even if you don't need routing or data fetching at first, you'll likely want to add some libraries for them. As your JavaScript bundle grows with every new feature, you might have to figure out how to split code for every route individually. As your data fetching needs get more complex, you are likely to encounter server-client network waterfalls that make your app feel very slow. As your audience includes more users with poor network conditions and low-end devices, you might need to generate HTML from your components to display content early—either on the server, or during the build time. Changing your setup to run some of your code on the server or during the build can be very tricky.

<https://react.dev/learn/start-a-new-react-project>

Laden von Daten kann Eindruck langsamer App erzeugen

Even if you don't need routing or data fetching at first, you'll likely want to add some libraries for them. As your JavaScript bundle grows with every new feature, you might have to figure out how to split code for every route individually. As your data fetching needs get more complex, you are likely to encounter server-client network waterfalls that make your app feel very slow. As your audience includes more users with poor network conditions and low-end devices, you might need to generate HTML from your components to display content early—either on the server, or during the build time. Changing your setup to run some of your code on the server or during the build can be very tricky.

<https://react.dev/learn/start-a-new-react-project>

Frühe Darstellung auch bei schlechtem Netzwerk/Hardware

Even if you don't need routing or data fetching at first, you'll likely want to add some libraries for them. As your JavaScript bundle grows with every new feature, you might have to figure out how to split code for every route individually. As your data fetching needs get more complex, you are likely to encounter server-client network waterfalls that make your app feel very slow. As your audience includes more users with poor network conditions and low-end devices, you might need to generate HTML from your components to display content early—either on the server, or during the build time. Changing your setup to run some of your code on the server or during the build can be very tricky.

<https://react.dev/learn/start-a-new-react-project>

Ausführung von React-Code auf Server/im Build ist kompliziert

"Fullstack Architektur-Vision"

<https://react.dev/learn/start-a-new-react-project#which-features-make-up-the-react-teams-full-stack-architecture-vision>

"Fullstack Architektur-Vision"

<https://react.dev/learn/start-a-new-react-project#which-features-make-up-the-react-teams-full-stack-architecture-vision>

- **React Server Components (RSC):**

- Komponenten, die auf dem Server, Client und im Build gerendert werden können
- Data Fetching "integriert"

"Fullstack Architektur-Vision"

<https://react.dev/learn/start-a-new-react-project#which-features-make-up-the-react-teams-full-stack-architecture-vision>

- **React Server Components (RSC):**

- Komponenten, die auf dem Server, Client und im Build gerendert werden können
- Data Fetching "integriert"

- **Suspense:**

- Loading Indikatoren deklarativ direkt im React-Tree, Server-Client übergreifend

"Fullstack Architektur-Vision"

<https://react.dev/learn/start-a-new-react-project#which-features-make-up-the-react-teams-full-stack-architecture-vision>

- **React Server Components (RSC):**

- Komponenten, die auf dem Server, Client und im Build gerendert werden können
- Data Fetching "integriert"

- **Suspense:**

- Loading Indikatoren deklarativ direkt im React-Tree, Server-Client übergreifend

- **Problem:** Dafür braucht es Infrastruktur (Server, Build, Bundling, ...)

👉 Deswegen "Fullstack-Framework"

Zero-Bundle-Size

Server

Components

Introducing Zero-Bundle-Size React Server Components

December 21, 2020 by [Dan Abramov](#), [Lauren Tan](#), [Joseph Savona](#), and [Sebastian Markbåge](#)

2020 has been a long year. As it comes to an end we wanted to share a special Holiday Update on our research into zero-bundle-size React Server Components.

<https://legacy.reactjs.org/blog/2020/12/21/data-fetching-with-react-server-components.html>

SERVER COMPONENTS

Idee: Komponenten können auf dem Server und im Build gerendert werden

- Sie stehen nicht auf dem Client zur Verfügung
- Der Server schickt lediglich eine *Repräsentation der UI*, aber *keinen Code*

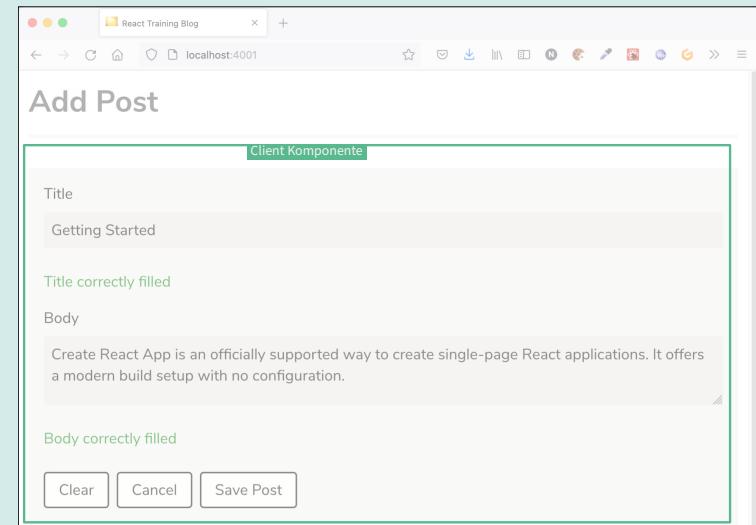
👉 "Zero-Bundle-Size"

SERVER COMPONENTS

Arten von Komponenten

Client-Komponenten (wie bisher)

- Werden auf dem Client ausgeführt
- JavaScript-Code wird zum Client gesendet
- Können auf dem Server vorgerendert werden



Neu: Server-Komponenten

- werden *nur* auf dem Server (oder zur Build-Zeit) ausgeführt
- liefern UI (!) zum React-Client zurück (kein JavaScript-Code)
- API: "normale" React-Komponenten (JS/TS, JSX, ...)

Neu: Server-Komponenten

- werden *nur* auf dem Server (oder zur Build-Zeit) ausgeführt
- liefern UI (!) zum React-Client zurück (kein JavaScript-Code)
- API: "normale" React-Komponenten (JS/TS, JSX, ...)

```
import { marked } from "marked";
import { dateToString } from "./components/date-formatter";

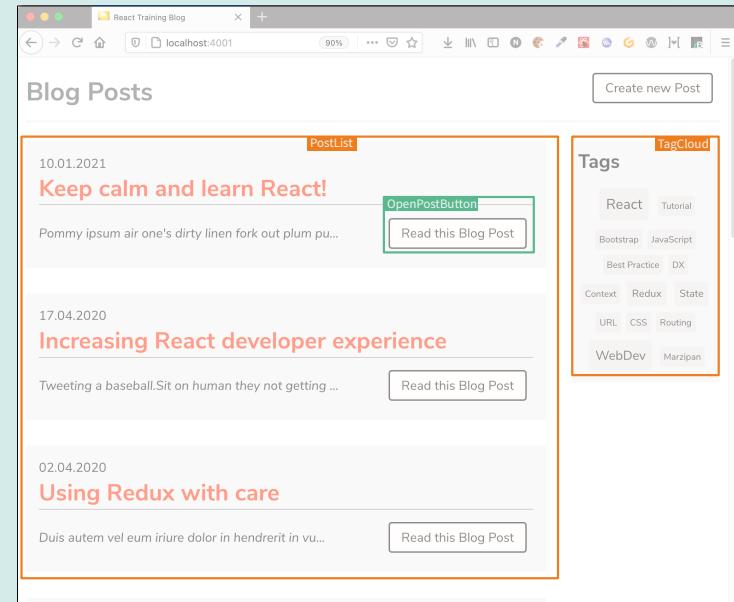
export default function Post({ post }) {
  const date = dateToString(post.date);
  const body = marked.parse(post.body);

  return (
    <article className="Container">
      <p className="Date">{date}</p>
      <h1>{post.title}</h1>
      <div dangerouslySetInnerHTML={{ __html: body }} />
    </article>
  );
}
```

ARTEN VON KOMPONENTEN

Weiterhin ein Komponenten-Baum

- Ein Teil der Komponenten kommt jetzt jetzt vom Server...
- **Server Komponenten sind nicht auf dem Client vorhanden!**



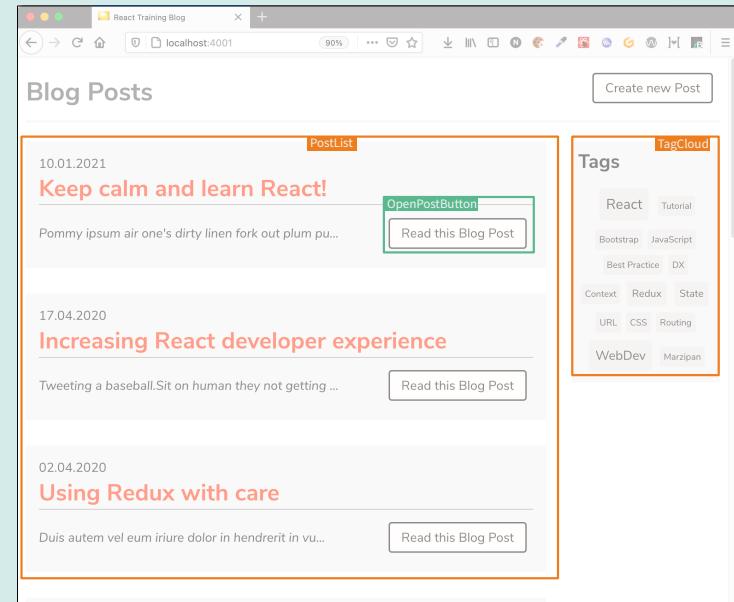
ARTEN VON KOMPONENTEN

Weiterhin ein Komponenten-Baum

- Ein Teil der Komponenten kommt jetzt vom Server...
- **Server Komponenten sind nicht auf dem Client vorhanden!**

Demo

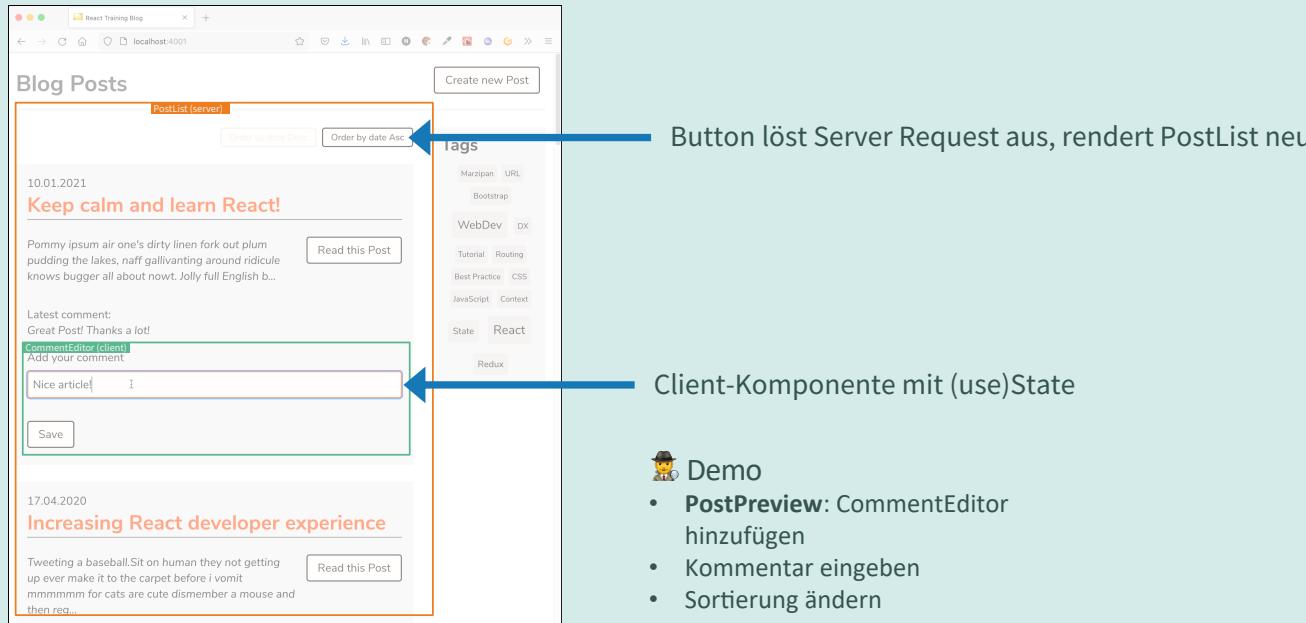
- PostListPage im Code:
- Server-Komponenten "PostList" und "TagCloud" gibt es als Komponenten, aber nicht auf dem Client (-> React Dev Tools)
- Netzwerktab:
 - Client Komponenten wie gewohnt



ARTEN VON KOMPONENTEN

Weiterhin ein Komponenten-Baum

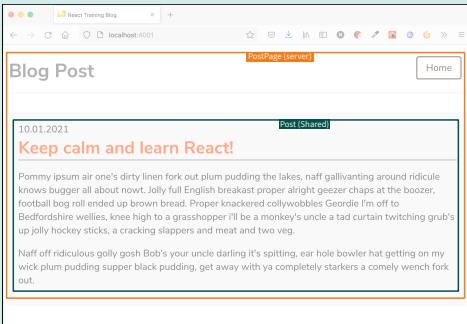
- Client-State bleibt beim neu-rendern von Server-Komponenten erhalten



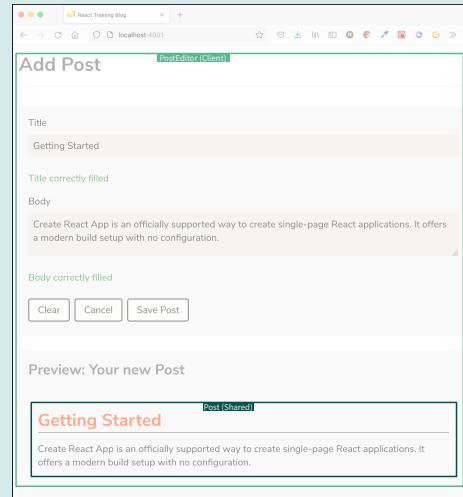
ARTEN VON KOMPONENTEN

"Mixed" Components

- Komponenten können sowohl auf dem Client als auch auf dem Server gerendert werden
- JS-Code wird erst bei Bedarf auf den Client geladen (ansonsten nur UI)



Verwendung "Post"-Komponente 1:
innerhalb einer Server-Komponente



Verwendung "Post"-Komponente 2:
innerhalb einer Client-Komponente



Demo

- Post-Seite: keine "Post-Komponente"
- PostEditor: Post-Komponente wird geladen (-> Netzwerk-Tab) und als Komponente gerendert (-> Dev Tools)

Data Fetching

DATEN LADEN

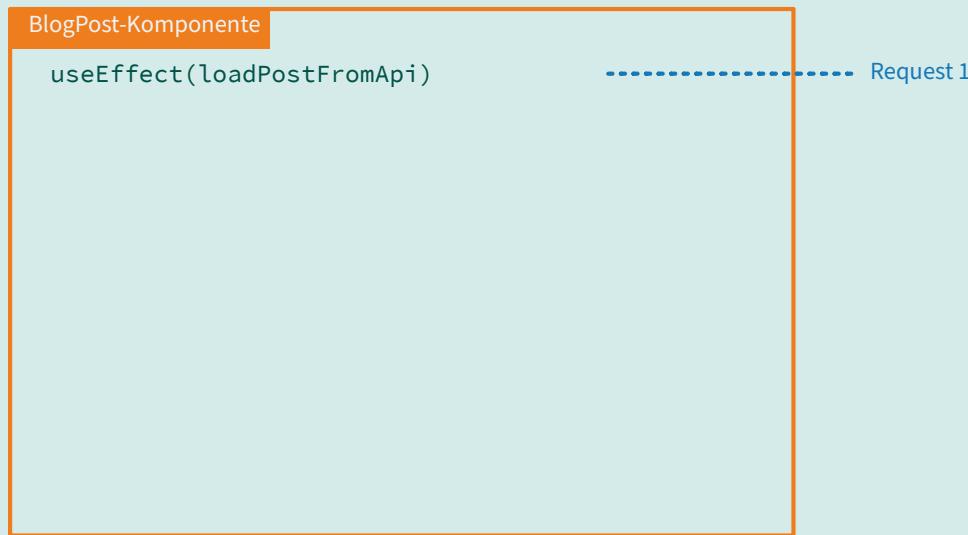
Mögliches Problem: Laden von Daten auf dem Client

- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten

DATEN LADEN

Laden von Daten auf dem Client

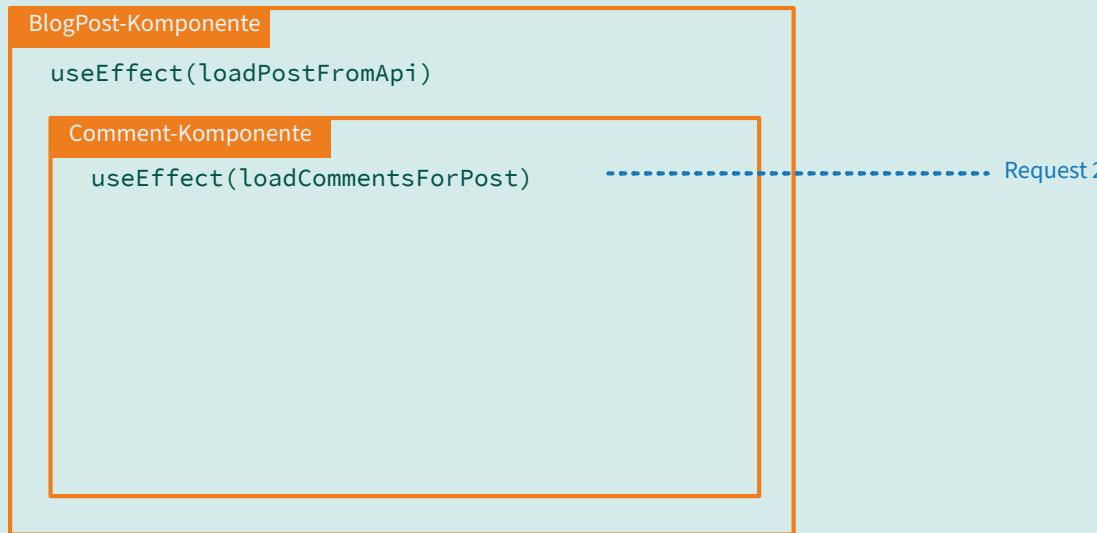
- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten



DATEN LADEN

Laden von Daten auf dem Client

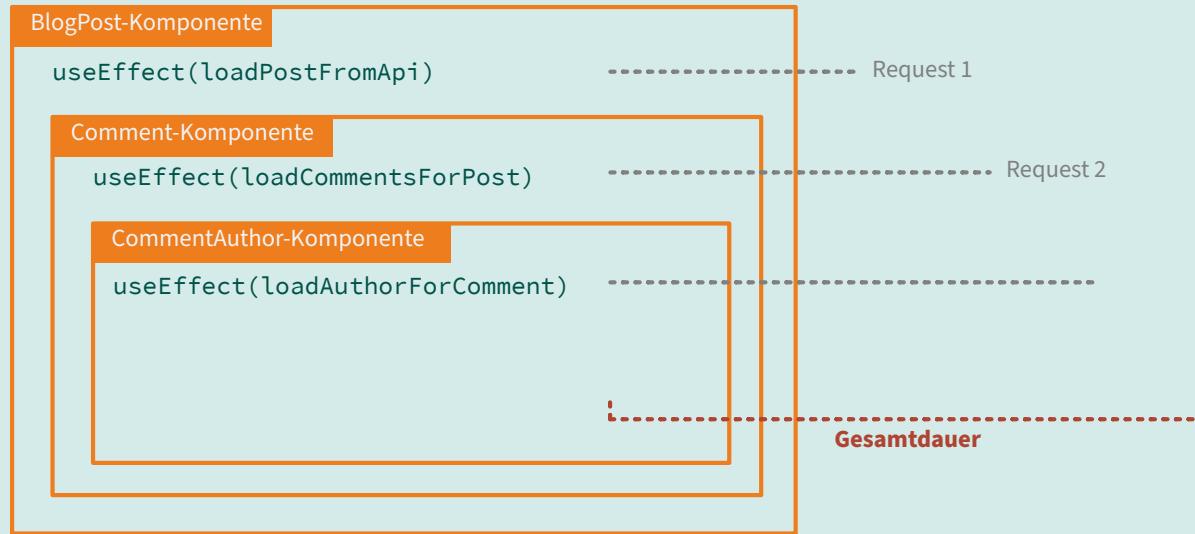
- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten



DATEN LADEN

Laden von Daten auf dem Client

- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten



😓 Wasserfall...

Data Fetching in Server Components

- Komponenten, die Daten laden, können das direkt *auf dem Server* tun

Data Fetching in Server Components

- Komponenten, die Daten laden, können das direkt *auf dem Server* tun
- Kann Latenz sparen und bessere Performance bringen
- "No Client-Server Waterfalls"

Data Fetching in Server Components

- Komponenten, die Daten laden, können das direkt *auf dem Server* tun
- Kann Latenz sparen und bessere Performance bringen
- "No Client-Server Waterfalls"

👉 Server Components können **asynchron** sein 😱

SERVER COMPONENTS

Beispiel: Eine asynchrone Server Komponente

```
async function fetchPost(postId) { ... }

async function fetchComments(postId) { ... }
```

SERVER COMPONENTS

Beispiel: Eine asynchrone Server Komponente

```
async function fetchPost(postId) { ... }

async function fetchComments(postId) { ... }

export async function PostPage(props) {
  const post = await fetchPost(props.params.postId);
  const commentsPromise = fetchComments(props.params.postId);

  return (
    <>
      <Post post={post.data} />
      <Comments comments={commentsPromise} />
    </>
  );
}
```

Beispiel: Eine asynchrone Server Komponente

- Server Komponenten können *async functions* sein und auf Promises warten
 - HTTP-Requests, DB-Queries, Zugriff auf das Filesystem
 - (Alles was "echte" Backend-Services auch können)

```
export async function Comments(props) {  
  const comments = await fetchPost(props.commentsPromise);  
  
  return comments.map( comment => ( ... ) );  
}
```

Beispiel: Eine asynchrone Server Komponente

- Server Komponenten können *async functions* sein und auf Promises warten
 - HTTP-Requests, DB-Queries, Zugriff auf das Filesystem
 - (Alles was "echte" Backend-Services auch können)

```
export async function Comments(props) {  
  const comments = await fetchPost(props.commentsPromise);  
  
  return comments.map(comment => ( ... ));  
}
```

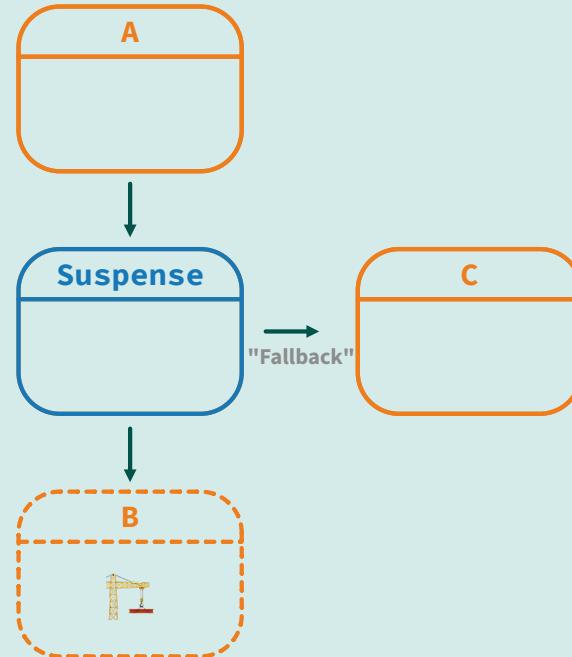


Was machen wir, bis das Promise augelöst wurde (z.B. mit den Daten)?

SUSPENSE

Suspense: Unterbricht das Rendern, solange "etwas" fehlt

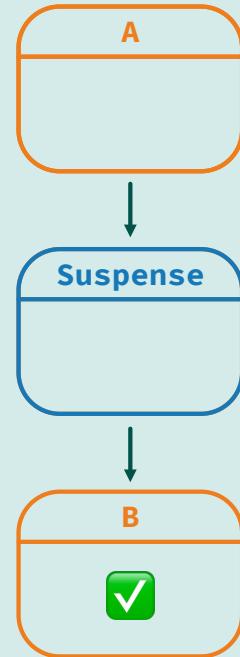
- Asynchron geladenes JS, asynchron geladene Daten, Promises, ...



SUSPENSE

Suspense: Unterbricht das Rendern, solange "etwas" fehlt

- Asynchron geladenes JS, asynchron geladene Daten, Promises, ...



SUSPENSE

Beispiel: Data Fetching mit Suspense

```
export async function Comments(props) {  
  const comments = await fetchPost(props.commentsPromise);  
  
  return comments.map( comment => ( ... ) );  
}
```

"Suspense for Data Loading"

- Wartet auf Daten, Server-Call, ...

SUSPENSE

Beispiel: Data Fetching mit Suspense

```
export async function Comments(props) {  
  const comments = await fetchPost(props.commentsPromise);  
  
  return comments.map( comment => ( ... ) );  
}  
  
export function PostPage(props) {  
  const commentsPromise = fetchComments(props.params.postId);  
  
  return (  
    <>  
      <Post post={post.data} />  
      <Suspense fallback={<LoadingIndicator />}>  
        <Comments />  
      </Suspense>  
    </>  
  );  
}
```

Suspense-Komponente

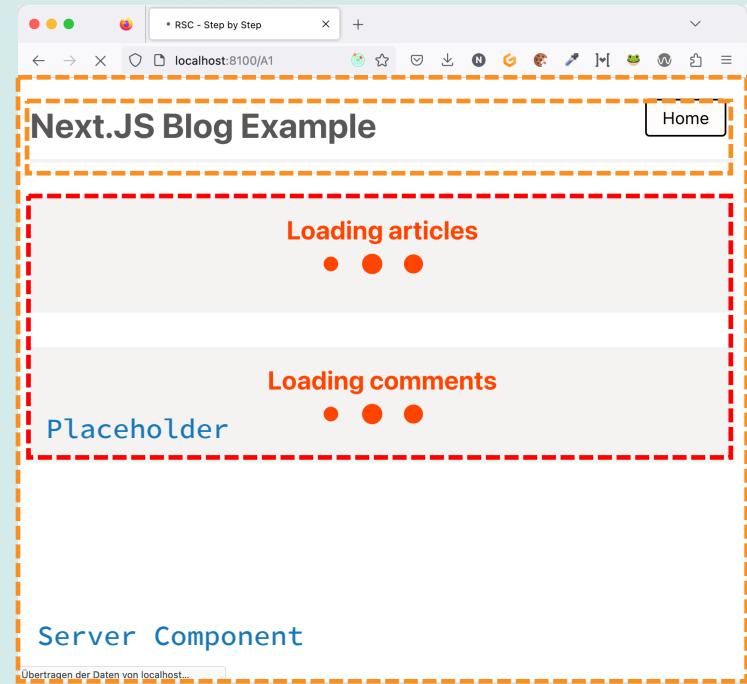
Streaming

- Teile "außerhalb" von Suspense werden auf den Client sofort übertragen
- Innerhalb der Suspense-Komponente kommt Platzhalter
- Sind die Daten da, wird nur der Bereich auf den Client geschickt und dort aktualisiert



Demo

- [post/postId/page.tsx](#)



Aufteilung in Server-Client: Konsequenzen

```
import { Article, OrderBy } from "@app/articles";

type ArticleListProps = {
  articles: Article[];
  | onToggleOrder(): void;
};

export default function ArticleList({
  articles,
  onToggleOrder,
}: ArticleListProps) {
  return (
    <div>
      <h1>Articles</h1>
      <ul>
        {articles.map((a) => (
          <li key={a.id}>{a.title}</li>
        ))}
      </ul>
      <button onClick={onToggleOrder}>Toggle Order</button>
    </div>
  );
}
```

CAN YOU SPOT THE PROBLEM?



```
<button onClick={onToggleOrder}>Toggle Order</button>
```

- **error** Error: Event handlers cannot be passed to Client Component props.

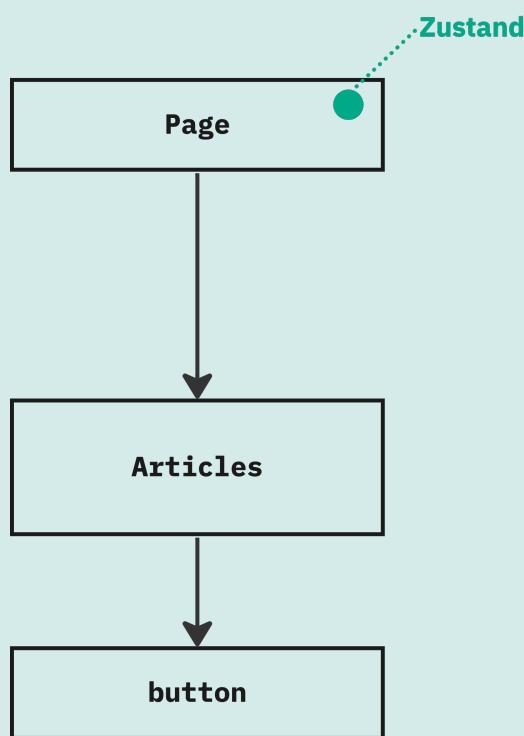
```
<button onClick={function} children=...>  
    ^^^^^^^^^^
```

If you need interactivity, consider converting part of this to a Client Component.

at stringify (<anonymous>)

CAN YOU SPOT THE PROBLEM?

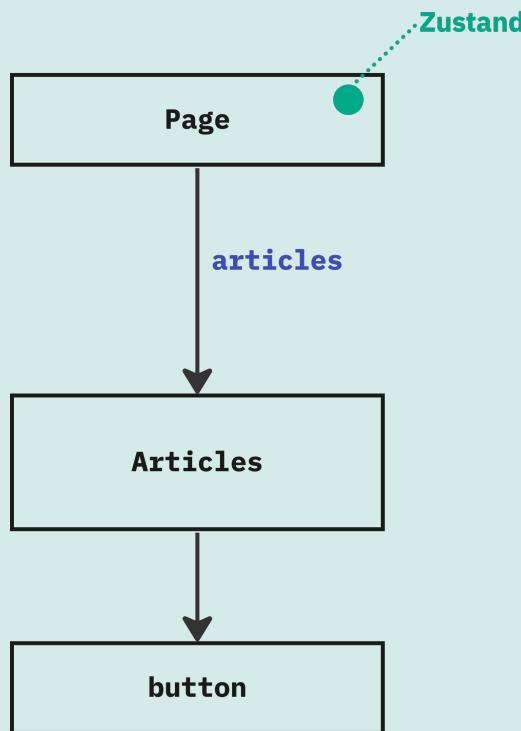
EINE REACT ANWENDUNG IM BROWSER



- State befindet sich oben

Eine "normale" React-Anwendung...

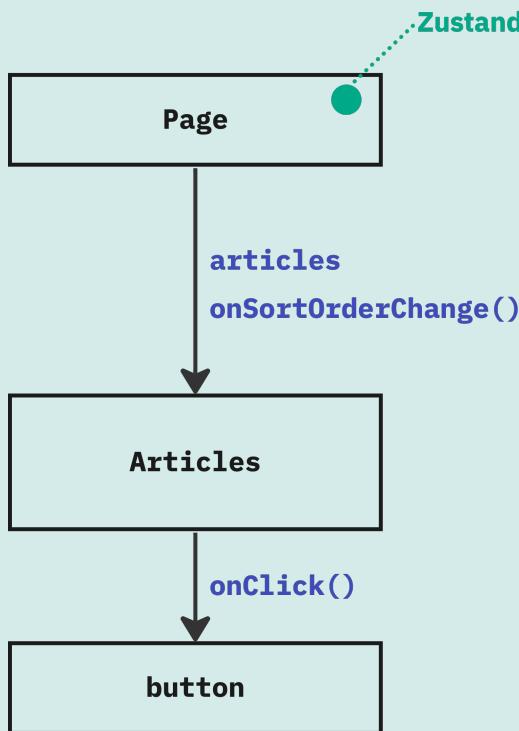
EINE REACT ANWENDUNG IM BROWSER



- State befindet sich oben
- Daten werden runtergereicht ("props")

Eine "normale" React-Anwendung...

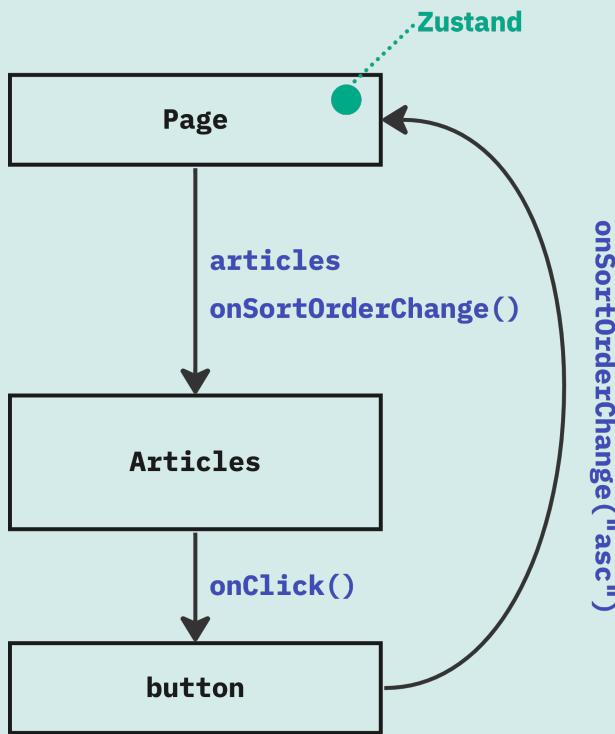
EINE REACT ANWENDUNG IM BROWSER



- State befindet sich oben
- Daten werden runtergereicht ("props")
- Callbacks werden runtergereicht

Eine "normale" React-Anwendung...

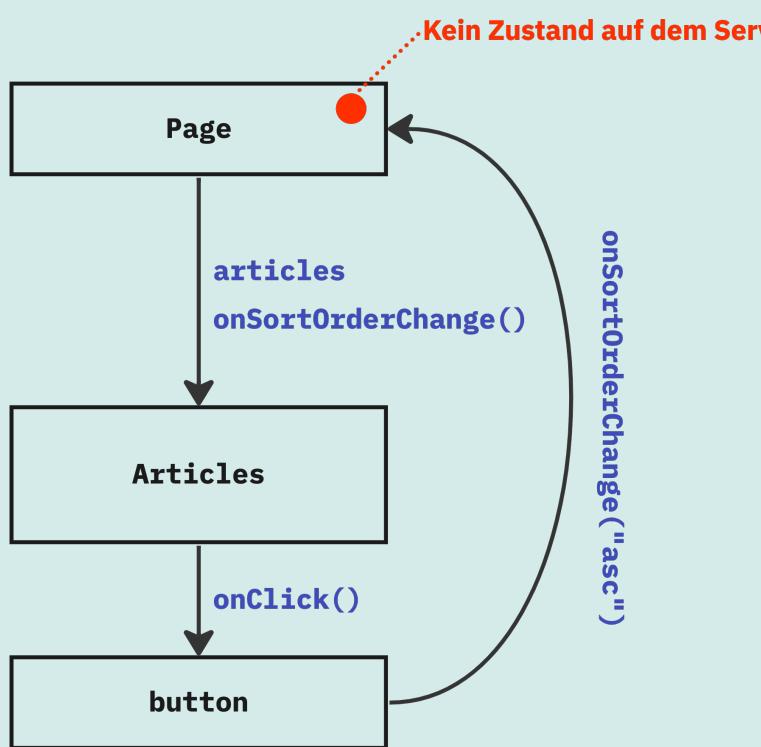
EINE REACT ANWENDUNG IM BROWSER



- State befindet sich oben
- Daten werden runtergereicht ("props")
- Callbacks werden runtergereicht
- Über Callbacks kann State-Veränderung ausgelöst werden

Eine "normale" React-Anwendung...

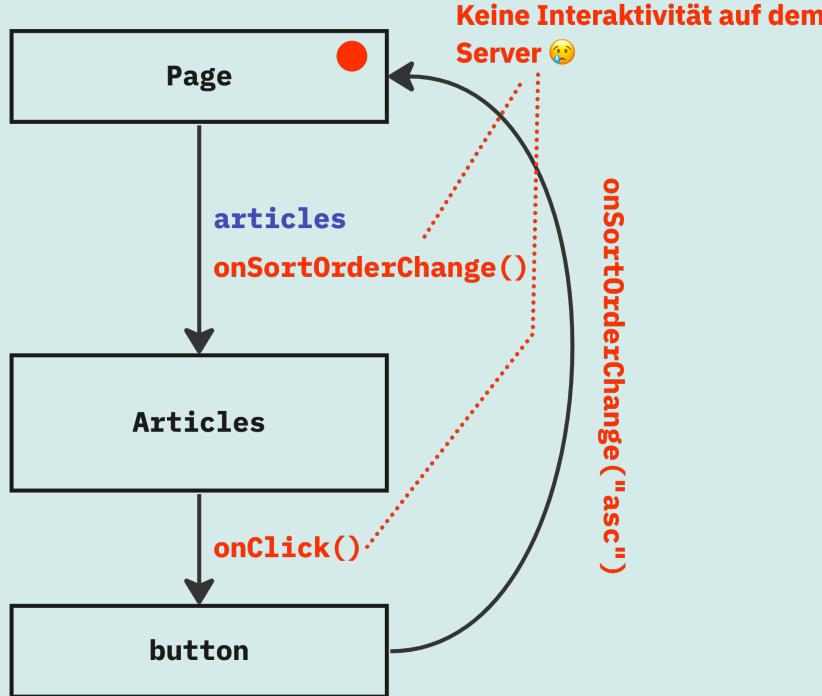
...UND AUF DEM SERVER



- Auf dem Server gibt es keinen State!

Mit Next.js sind wir aber auf dem Server (by Default)

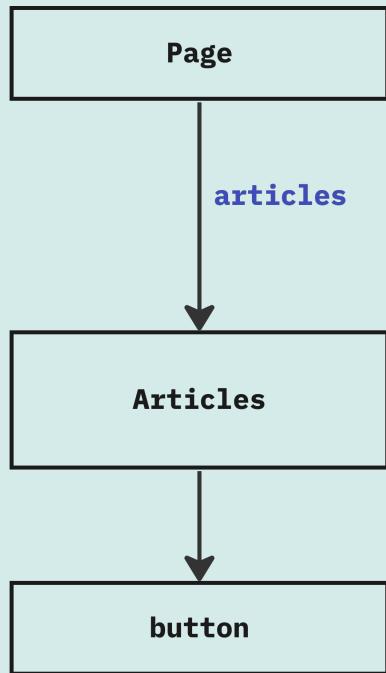
...UND AUF DEM SERVER



- Auf dem Server gibt es keinen State!
- ...und keine Interaktion

Mit Next.js sind wir aber auf dem Server (by Default)

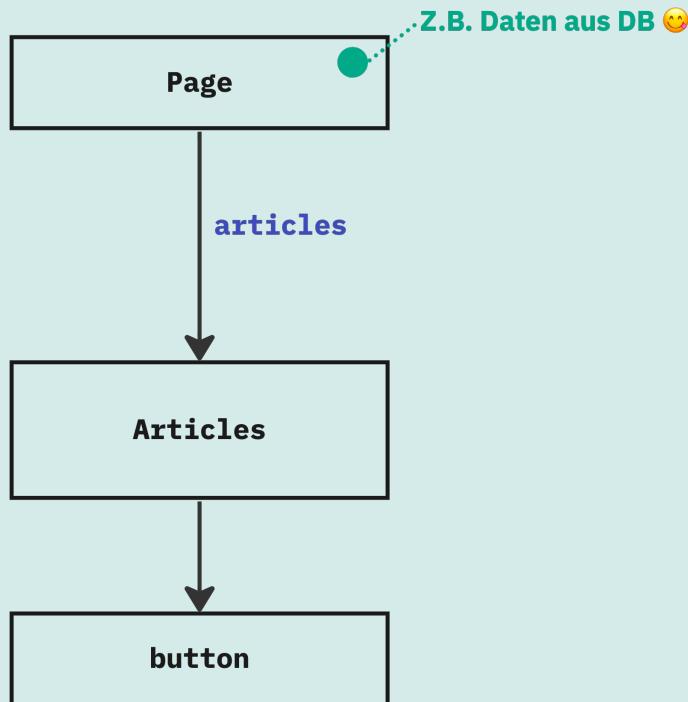
...UND AUF DEM SERVER



- Auf dem Server gibt es keinen State!
- ...und keine Interaktion
- Wir haben nur statischen Content

Mit Next.js sind wir aber auf dem Server (by Default)

...UND AUF DEM SERVER

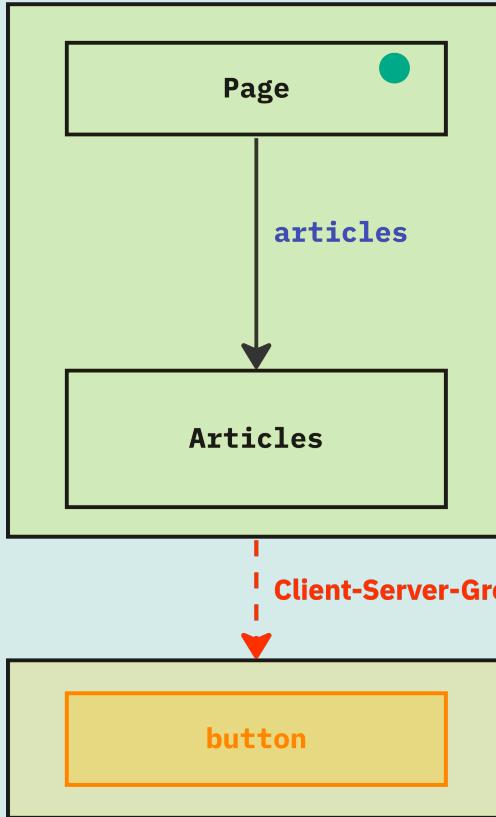


- Auf dem Server gibt es keinen State!
- ...und keine Interaktion
- Wir haben nur statischen Content
- Wir haben aber **Daten**
z.B. aus DB, Microservice, Filesystem...

Mit Next.js sind wir aber auf dem Server (by Default)

...UND AUF DEM SERVER

Server

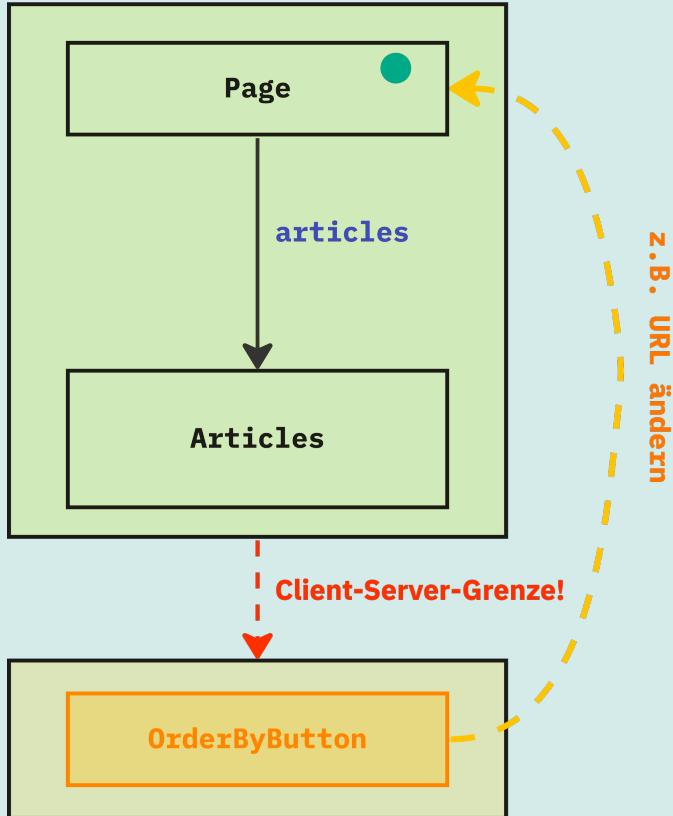


- Properties müssen Client-Server-Grenze überwinden
- Müssen serialisierbare Daten sein
- **Keine (Callback-)Funktionen!**

Client

...UND AUF DEM SERVER

Server

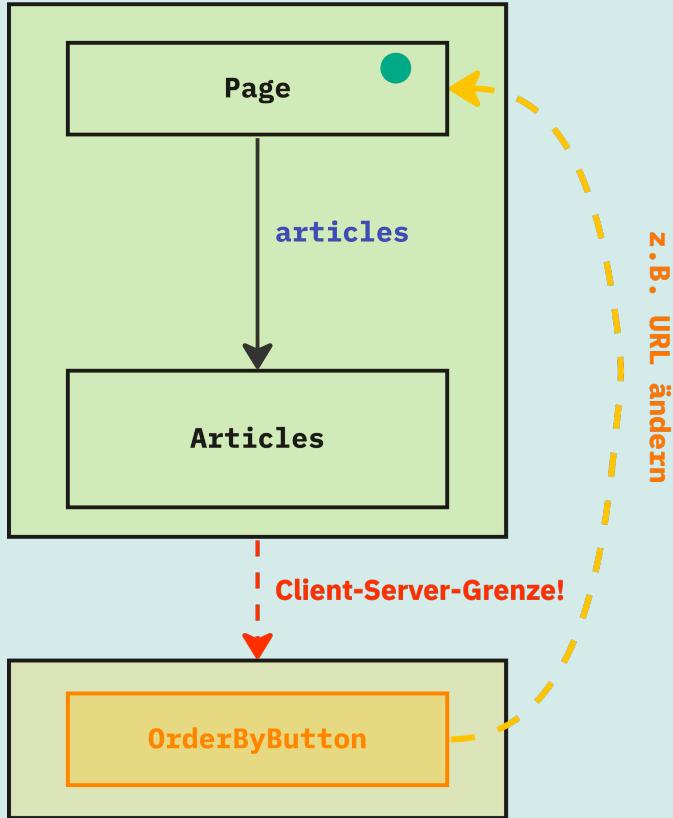


- Properties müssen Client-Server-Grenze überwinden
- Müssen serialisierbare Daten sein
- Keine (Callback-)Funktionen!
- Zur Kommunikation müssen **Server-Requests** durchgeführt werden
 - z.B. URL ändern

Client

...UND AUF DEM SERVER

Server



- Properties müssen Client-Server-Grenze überwinden
- Müssen serialisierbare Daten sein
- Keine (Callback-)Funktionen!
- Zur Kommunikation müssen Server-Requests durchgeführt werden
 - z.B. URL ändern
- **Server-Komponente hat Zugriff auf Request Informationen**
 - URL mit Search Params
 - Cookies
 - Headers

Client

SERVER COMPONENTS

Mutations

- **PostList** ist nicht als Komponente auf dem Client vorhanden

Blog Posts

[Order by date Desc](#) [Order by date Asc](#)

10.01.2021
Keep calm and learn React!

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

Read this Post

Latest comment:
Disrupt inspire and think tank, social entrepreneur but preliminary thinking think tank compelling.

Add your comment

Great Article!

Save

17.04.2020
Increasing React developer experience

Tweeting a baseball. Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

Read this Post

Latest comment:
Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

02.04.2020
Using Redux with care

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et a...

Read this Post

SERVER COMPONENTS

Mutations

- PostList ist nicht als Komponente auf dem Client vorhanden
- Die **Posts mit Kommentaren** (Daten) sind folglich ebenso nicht auf dem Client vorhanden

Blog Posts

[Order by date Desc](#) [Order by date Asc](#)

10.01.2021

Keep calm and learn React!

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

[Read this Post](#)

Latest comment:
Disrupt inspire and think tank, social entrepreneur but preliminary thinking think tank compelling.

Add your comment

[Save](#)

17.04.2020

Increasing React developer experience

Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

[Read this Post](#)

Latest comment:
Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

02.04.2020

Using Redux with care

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et a...

[Read this Post](#)

SERVER COMPONENTS

Konsequenzen

- PostList ist nicht als Komponente auf dem Client vorhanden
- Die Posts mit Kommentaren (Daten) sind folglich ebenso nicht auf dem Client vorhanden
- Nach dem Hinzufügen eines Kommentars ([CommentEditor-Komponente](#)) haben wir keinen State zum Verändern 😢

Blog Posts

10.01.2021

Keep calm and learn React!

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

[Read this Post](#)

Latest comment:

Disrupt inspire and think tank, social entrepreneur but preliminary thinking think tank compelling.

Add your comment

Great Article!

Save

17.04.2020

Increasing React developer experience

Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

[Read this Post](#)

Latest comment:

Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

02.04.2020

Using Redux with care

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et a...

[Read this Post](#)

SERVER COMPONENTS

Konsequenzen

- PostList ist nicht als Komponente auf dem Client vorhanden
- Die Posts mit Kommentaren (Daten) sind folglich ebenso nicht auf dem Client vorhanden
- Nach dem Hinzufügen eines Kommentars (CommentEditor-Komponente) haben wir keinen State zum Verändern 😢
- Wir brauchen **aktualisierte UI vom Server**

Blog Posts

Order by date Desc Order by date Asc

10.01.2021 **Keep calm and learn React!**

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

Latest comment:
Great Article!

Add your comment

Save

17.04.2020 **Increasing React developer experience**

Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

Latest comment:
Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

02.04.2020 **Using Redux with care**

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et a...

Read this Post

Read this Post

Read this Post

SERVER COMPONENTS

Konsequenzen

- PostList ist nicht als Komponente auf dem Client vorhanden
- Die Posts mit Kommentaren (Daten) sind folglich ebenso nicht auf dem Client vorhanden
- Nach dem Hinzufügen eines Kommentars (CommentEditor-Komponente) haben wir keinen State zum Verändern 😢
- Wir brauchen **aktualisierte UI vom Server**
- **Server Actions!**

Blog Posts

Order by date Desc Order by date Asc

10.01.2021 **Keep calm and learn React!**

Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...

Latest comment:
Great Article!

Add your comment

Save

Read this Post

17.04.2020 **Increasing React developer experience**

Tweeting a baseball.Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then reg...

Latest comment:
Yolo ipsum dolor sit amet, consectetur adipiscing elit. Curabitur mattis odio at erat viverra lobortis.

Read this Post

02.04.2020 **Using Redux with care**

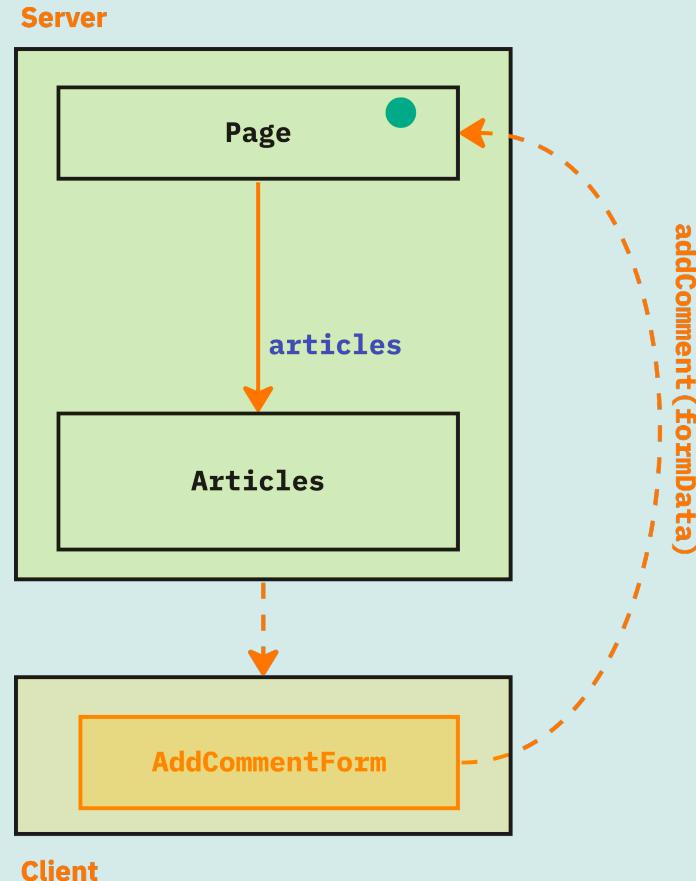
Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et a...

Read this Post

MUTATIONS

Server Actions

- Das ist eine Art Remote Funktion, die UI (statt Daten) zurückliefert
- In React und Next.js noch Beta!



MUTATIONS

Server Actions

```
export function CommentEditor( {postId} ) {  
  
  async function addComment() {  
    "use server";  
    const comment = form.get("comment");  
  
    const response = await fetch(...);  
  
    if (response.status === 201) revalidatePath("/");  
  
    return { error: await response.json() };  
  }  
  
  return <form action={addComment}>  
    <input name="comment" />  
    <button type="submit">Add</button>  
  </form>;  
}
```

Server-Action

- Eigener Endpunkt auf dem Server
- Aufruf für zu Server-Call

SERVER COMPONENTS

Demo: UI aktualisieren

The screenshot shows a web browser window with the title "React Training Blog" at "localhost:4001". The main content area is titled "Blog Posts" and displays two posts:

- Post 1:** Date 10.01.2021, Title "Keep calm and learn React!", Content "Pommy ipsum air one's dirty linen fork out plum pudding the lakes, naff gallivanting around ridicule knows bugger all about nowt. Jolly full English b...", Buttons: "Read this Post", "CommentEditor (client)" with placeholder "Add your comment", and a text input field containing "Nice article!" with a "Save" button.
- Post 2:** Date 17.04.2020, Title "Increasing React developer experience", Content "Tweeting a baseball. Sit on human they not getting up ever make it to the carpet before i vomit mmmmmm for cats are cute dismember a mouse and then req...", Buttons: "Read this Post".

To the right of the posts is a sidebar titled "Tags" with a list of tags: Marzipan, URL, Bootstrap, WebDev (selected), DX, Tutorial, Routing, Best Practice, CSS, JavaScript, Context, State, React, and Redux.

Gesendet (HTTP POST) werden Daten, gelesen wird UI



Demo

- Kommentar hinzufügen -> Netzwerk-Tab (JS & XHR)

Go full-stack with a framework

...aber

welches? 

Fullstack-Frameworks

- **Remix & Next.js**
 - für "vollständige" React-Anwendungen
 - Remix (noch) kein Support für RSC
- **Gatsby etc.**
 - eher für statische Websites

Next.js und Remix

- Bringen kompletten Stack mit
 - Server
 - Build-System
- Serverseitiger Router über Filesystem / Konventionen
- Unterstützung für SSR und Streaming

 Kein Client-seitiges Routing

 Kein Migrationspfad (von "klassischem" React)

Next.js

- Entspricht der "Fullstack Architektur-Vision" des React Teams
- Unterstützt React Server Components
- Aggressives Caching auf allen Ebenen
- Statische vs. dynamische Komponenten
 - Statische Komponenten werden im Build vorgerendert
- Statische vs. dynamische Komponenten
 - Statische Komponenten werden im Build vorgerendert
 - Nicht immer offensichtlich, was ist was

Remix

- Weiterentwicklung des React Routers
- Einige Features stehen über den React Router auch Client-seitig zur Verfügung
- Noch keine RSC
- Daten laden und schreiben nur in Routen
 - etwas weniger flexibel als RSC
- "Nur" SSR
- Verbereitung nicht so hoch wie Next (?)

Go full-stack
with a framework

...oder

doch nicht?! 😭

SERVER COMPONENTS

Müssen wir jetzt alle serverseitiges React machen?

Ginge es nicht auch ohne Framework?

Müssen wir jetzt alle serverseitiges React machen?

Ginge es nicht auch ohne Framework?

DEEP DIVE

Can I use React without a framework?

^ Hide Details

You can definitely use React without a framework—that's how you'd [use React for a part of your page](#). However, if you're building a new app or a site fully with React, we recommend using a framework.

Client-seitiges React (SPA) wird bleiben

- "Nur" Tooling-Frage offen
 - Status von create-react-app unklar
 - vite bietet mittlerweile eigenes React Template
 - Mit Next.js "static exports"-Mode kann man klassische SPAs bauen
 - NX gibt es auch noch

"Moderne" React-Architekturen lösen Probleme

- Data Fetching Libs vereinfachen Arbeit mit Daten
- Suspense wird auf dem Client funktionieren
- **use-Hook** in Client-Komponenten statt async-Komponenten (Server)

Fazit

React - Neue Ära?

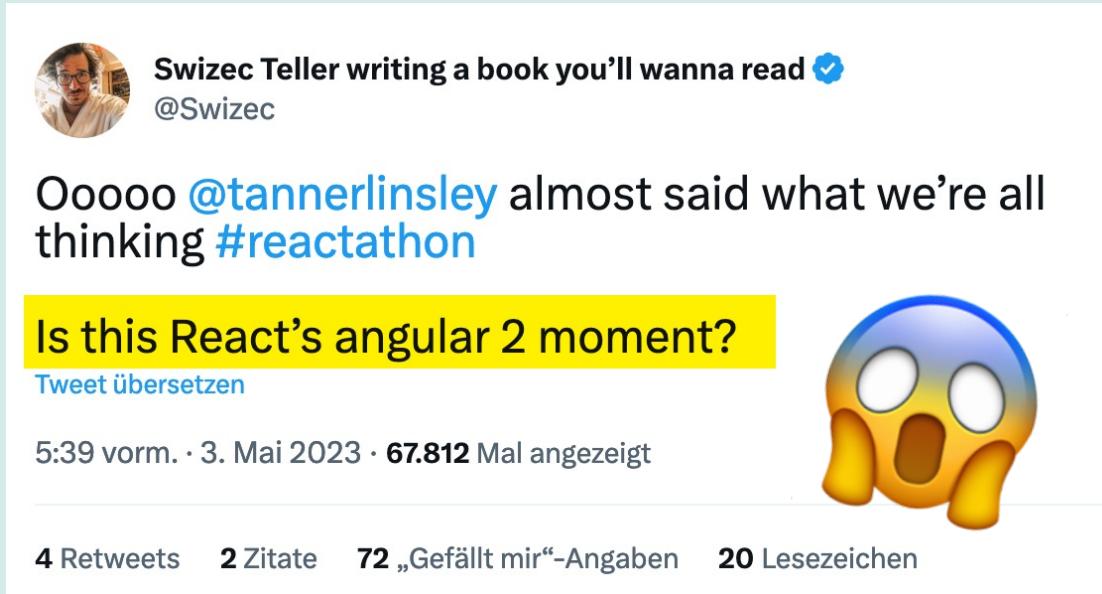
FAZIT

Neue Ära? Ja, aber!

FAZIT

Neue Ära? Ja, aber!

- Im Gegensatz zu Hooks gibt es viel mehr Kritik, Widerspruch und Unverständnis



Swizec Teller writing a book you'll wanna read ✅
@Swizec

Ooooo @tannerlinsley almost said what we're all thinking #reactathon

Is this React's angular 2 moment?

Tweet übersetzen

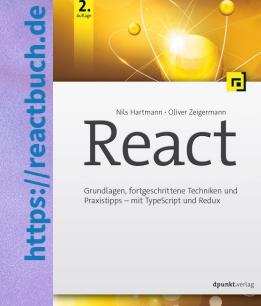
5:39 vorm. · 3. Mai 2023 · 67.812 Mal angezeigt

4 Retweets 2 Zitate 72 „Gefällt mir“-Angaben 20 Lesezeichen

<https://twitter.com/Swizec/status/1653605092371873792?s=20>

Neue Ära? Ja, aber!

- Im Gegensatz zu Hooks gibt es viel mehr Kritik, Widerspruch und Unverständnis
- Ob sich die "Fullstack Architektur-Vision" durchsetzt, werden wir sehen
- Die Kommunikation könnte besser laufen



vielen Dank!

Slides: <https://react.schule/ejs23>

Code: <https://github.com/nilshartmann/fullstack-react-playground>

Fragen & Kontakt: nils@nilshartmann.net

Twitter: [@nilshartmann](https://twitter.com/nilshartmann)