

NILS HARTMANN

<https://nilshartmann.net>

Serverside Rendering 2.0?

React Server Components

Slides: <https://react.schule/codetalks-react>

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)

A screenshot of a web browser displaying a blog application titled "React Training Blog". The URL in the address bar 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 a grid of tags:

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

<https://github.com/nilshartmann/server-components-blogexample>

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 right, there is a red arrow pointing to the word "React!" above a "Create new Post" button. On the left, another red arrow points to the word "MomentJS!" above the first blog post. The first post, dated 10.01.2021, has the title "Keep calm and learn React!" and contains placeholder text about a 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... Below the post is a "Read this Post" button. A "Latest comment:" section follows, containing a single line of placeholder text. The second post, dated 17.04.2020, has the title "Increasing React developer experience" and contains placeholder text about tweeting a baseball. Below the post is a "Read this Post" button. A "Latest comment:" section follows, containing a single line of placeholder text. To the right of the posts is a sidebar titled "Tags" with a red arrow pointing to it labeled "tag-cloud.js". The sidebar lists various tags in a grid: WebDev, State, JavaScript, Tutorial, Context, DX, Marzipan, React, URL, Redux, Bootstrap, Best Practice, Routing, and CSS.

React!

Tags

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

tag-cloud.js

Marked!

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)

[Create new Post](#)

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.

Tags

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

EIN BEISPIEL

Herausforderung

👉 Für Besucher des Blogs sollen die Artikel schnell zur Verfügung stehen!

Der Klassiker:

Serverseitiges Rendern

Serverseitiges Rendern (SSR)

1. Bei SSR wird die Anwendung auf dem Server ausgeführt

Serverseitiges Rendern (SSR)

1. Bei SSR wird die Anwendung auf dem Server ausgeführt
2. Der Server schickt **fertiges HTML** zum Client
 - Gut: Client braucht HTML nur anzuzeigen (schnell!)
 - Gut: Suchmaschinen können HTML indizieren

Serverseitiges Rendern (SSR)

1. Bei SSR wird die Anwendung auf dem Server ausgeführt
2. Der Server schickt **fertiges HTML** zum Client
 - Gut: Client braucht HTML nur anzuzeigen (schnell!)
 - Gut: Suchmaschinen können HTML indizieren
3. Ebenfalls wird der **komplette Anwendungscode** zum Client geschickt
 - 😢 Auch für "statische" Komponenten
 - 😢 Bandbreite! Performance!

Zero-Bundle-Size

Server

Components

"-experimental-

```
"dependencies": {  
  "react": "0.0.0-experimental-7ec4c5597",  
  "react-dom": "0.0.0-experimental-7ec4c5597",  
  "react-fetch": "0.0.0-experimental-7ec4c5597",  
  "react-fs": "0.0.0-experimental-7ec4c5597",  
  "react-pg": "0.0.0-experimental-7ec4c5597",  
  "react-server-dom-webpack": "0.0.0-experimental-7ec4c5597",
```



CURRENT STATE

SERVER COMPONENTS

Idee: Komponenten werden nur auf dem Server ausgeführt

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

👉 "Zero-Bundle-Size"

SERVER COMPONENTS

Drei Arten von Komponenten

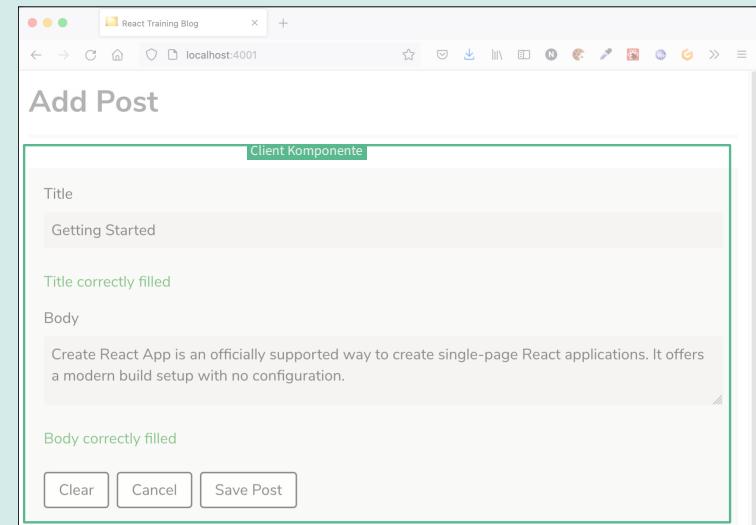
DREI ARTEN VON KOMPONENTEN

Drei Arten von Komponenten

DREI ARTEN VON KOMPONENTEN

Wie bisher: Client-Komponenten

- werden *nur* auf dem Client ausgeführt
- JavaScript-Code wird zum Client gesendet



DREI ARTEN VON KOMPONENTEN

Neu: Server-Komponenten

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

DREI ARTEN VON KOMPONENTEN

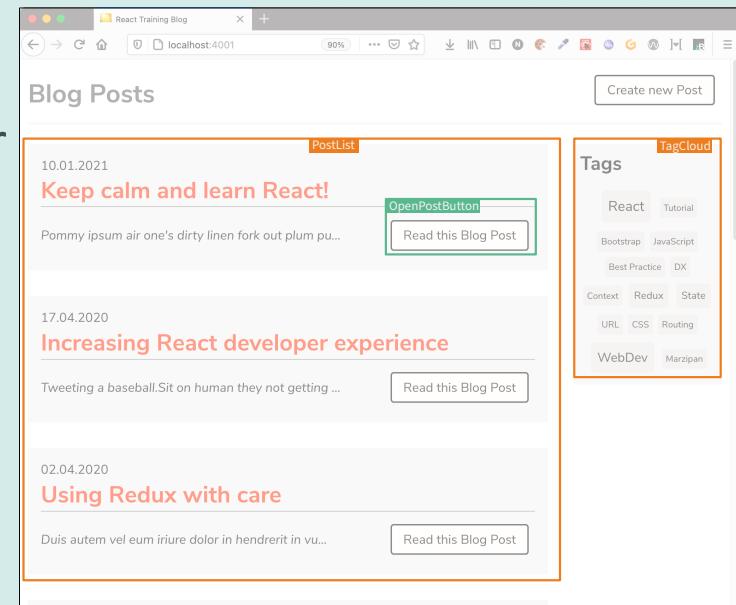
Neu: Server-Komponenten

- werden *nur* auf dem Server ausgeführt
- liefern UI (!) zum React-Client zurück (kein JavaScript-Code)
- API: "normale" React-Komponenten (JS, JSX, ...)
- Restriktionen: kein useState, useEffect, Browser APIs
- aber: können Server Umgebung und Ressourcen nutzen (!)
 - Datenbanken
 - Filesystem

DREI ARTEN VON KOMPONENTEN

Weiterhin ein Komponenten-Baum

- Ein Teil der Komponenten kommt jetzt vom Server...
- **Server Komponenten sind nicht auf dem Client vorhanden!**
- Der Server rendert die Komponenten, bis er auf eine Client-Komponente trifft



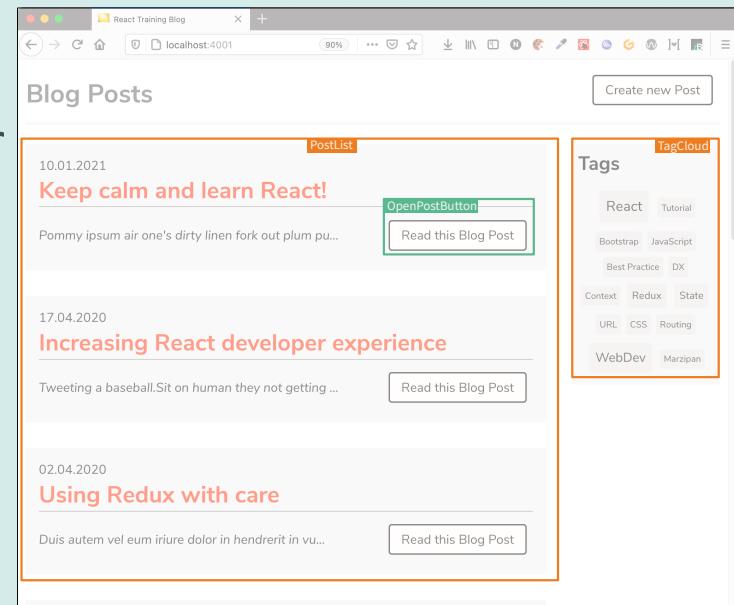
DREI ARTEN VON KOMPONENTEN

Weiterhin ein Komponenten-Baum

- Ein Teil der Komponenten kommt jetzt vom Server...
- **Server Komponenten sind nicht auf dem Client vorhanden!**
- Der Server rendert die Komponenten, bis er auf eine Client-Komponente trifft

Demo

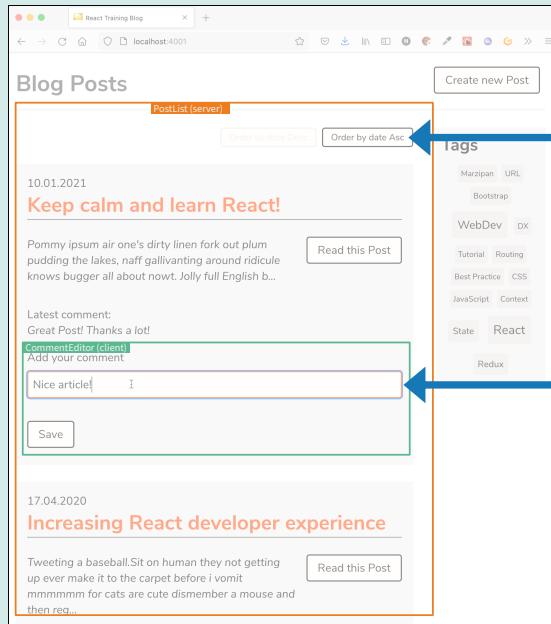
- PostListPage im Code:
- Server-Komponenten "PostList" und "TagCloud" gibt es als Komponenten, aber nicht auf dem Client (-> React Dev Tools)
- Netzwerktab:
 - react?location
 - Client Komponenten wie gewohnt



DREI ARTEN VON KOMPONENTEN

Weiterhin ein Komponenten-Baum

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



Button löst Server Request aus, rendert PostList neu

Client-Komponente mit (use)State



Demo

- PostPreview: CommentEditor hinzufügen
- Kommentar eingeben
- Sortierung ändern

DREI ARTEN VON KOMPONENTEN

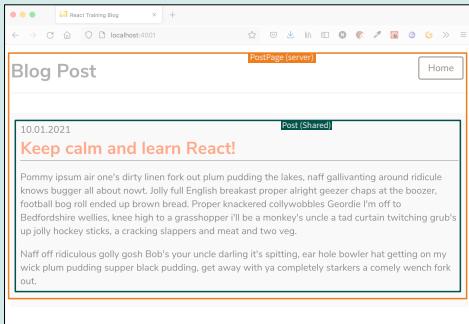
Neu: Shared Komponenten

- werden auf dem Server und dem Client ausgeführt
 - es gelten also die Restriktionen von Server und Client-Komponenten
 - können von Server- und Client-Komponenten verwendet werden
-
- der entsprechende JavaScript-Code wird erst auf den Client übertragen, wenn er wirklich benötigt wird

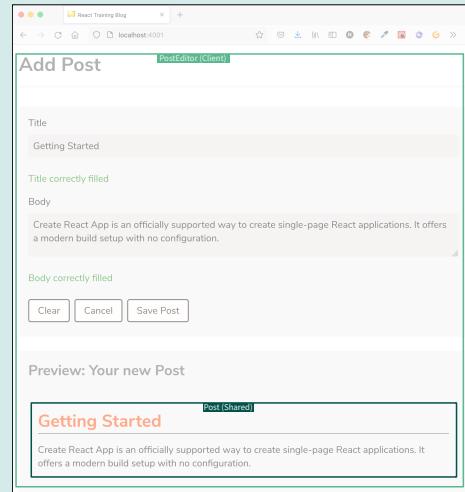
DREI ARTEN VON KOMPONENTEN

Shared Components

- 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)

SERVER COMPONENTS

Konsequenzen

- **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

Konsequenzen

- 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

Demo: UI aktualisieren

The screenshot shows a web application interface. On the left, there's a list of blog posts with titles like "Keep calm and learn React!" and "Increasing React developer experience". Each post has a "Read this Post" button. Below the posts is a "CommentEditor (client)" section with a text input containing "Nice article!" and a "Save" button. On the right, there's a sidebar titled "Tags" with categories like Marzipan, URL, Bootstrap, WebDev, 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)

Data Fetching

DATEN LADEN

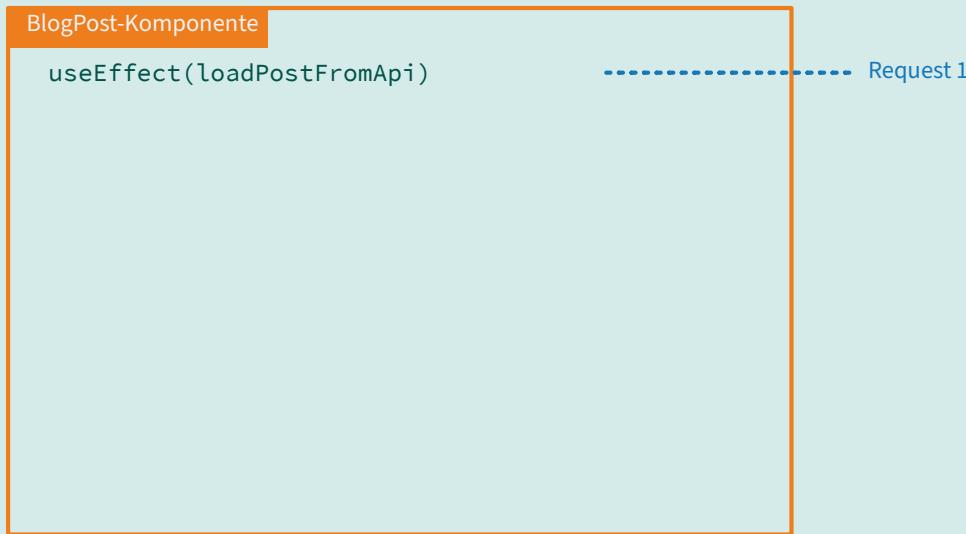
Mögliches Problem: Laden von Daten auf dem Client

- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten

DATEN LADEN

Laden von Daten auf dem Client

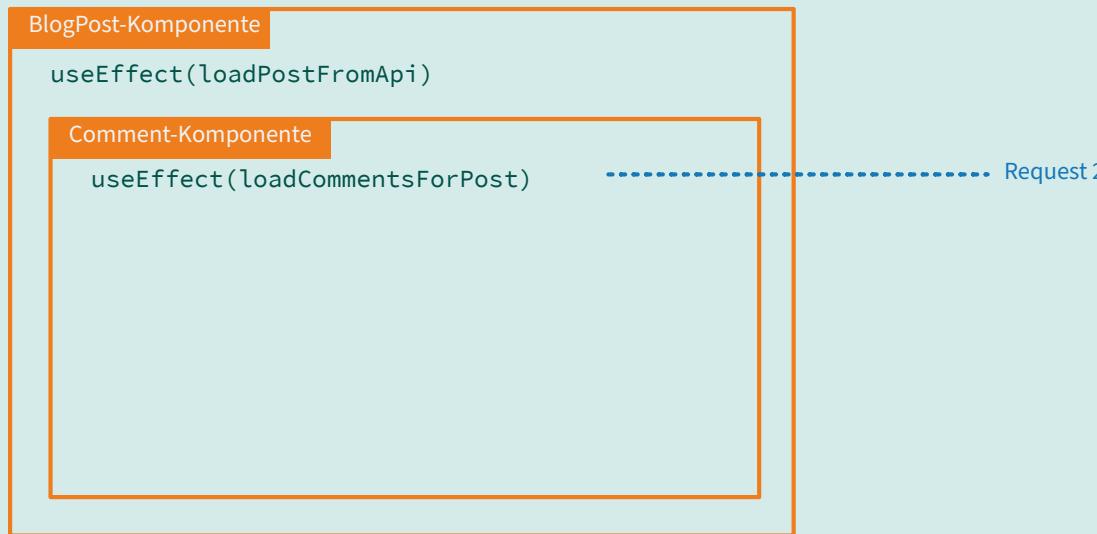
- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten



DATEN LADEN

Laden von Daten auf dem Client

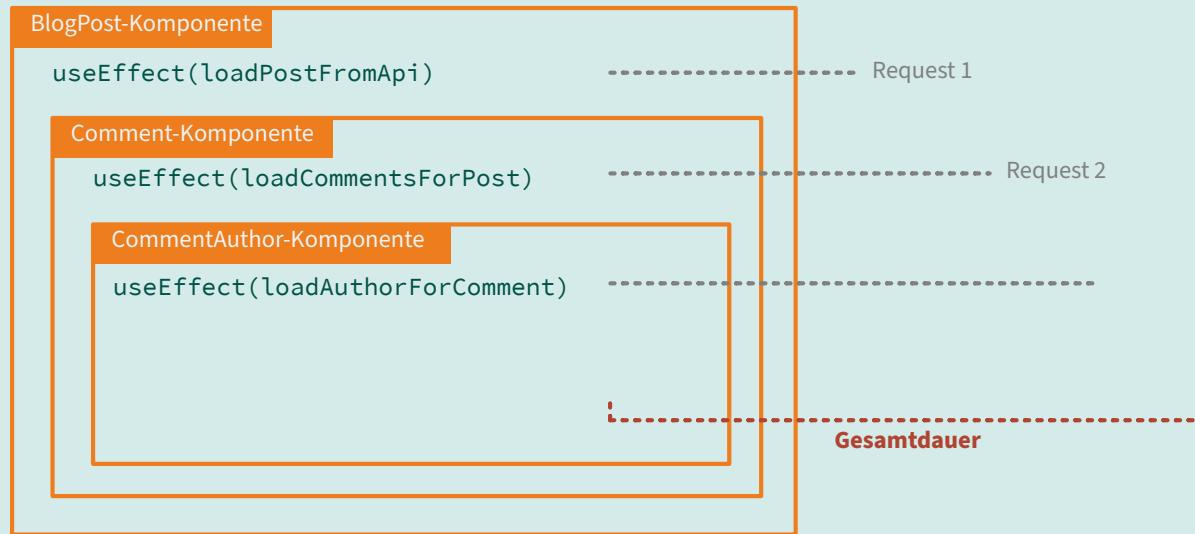
- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten



DATEN LADEN

Laden von Daten auf dem Client

- Eine Komponente lädt ihre Daten, Unterkomponenten müssen warten



😅 Wasserfall...

SERVER COMPONENTS

Idee

- Komponenten, die Daten laden, können das direkt *auf dem Server* tun
- Kann Latenz sparen und bessere Performance bringen

👉 "No *Client-Server* Waterfalls"

SERVER COMPONENTS

Beispiel: Eine Server Komponente

SUSPENSE

Beispiel: Eine Server Komponente

experimental!

```
import { db } from "./db.server";

export default function PostComments({ post }) {
  const comments = db.query("select id, comment from comments where post_id = $1", [post.id]);

  return (
    <div className="Container">
      <h1>Comments</h1>
      {comments.rows.map((comment) => (
        <p key={comment.id}>{comment.comment}</p>
      ))}
    </div>
  );
}
```

Beispiel: Eine Server Komponente

experimental!

```
import { db } from "./db.server";

export default function PostComments({ post }) {
  const comments = db.query("select id, comment from comments where post_id = $1", [post.id]);

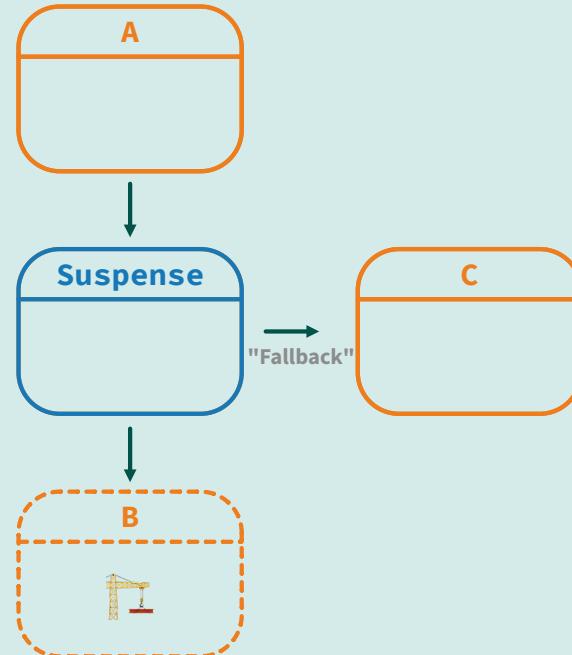
  return (
    <div className="Container">
      <h1>Comments</h1>
      {comments.rows.map((comment) => (
        <p key={comment.id}>{comment.comment}</p>
      ))}
    </div>
  );
}
```

- Server Komponenten können direkt DB-Queries ausführen, auf das Filesystem zugreifen etc.
 - (Alles was "echte" Backend-Services auch können)
- Client Komponenten können hier zum Beispiel fetch-Requests ausführen
- *Was machen wir, bis die Daten vorhanden sind, während der Query läuft?*

SUSPENSE

Suspense: Unterbricht das Rendern, solange "etwas" fehlt

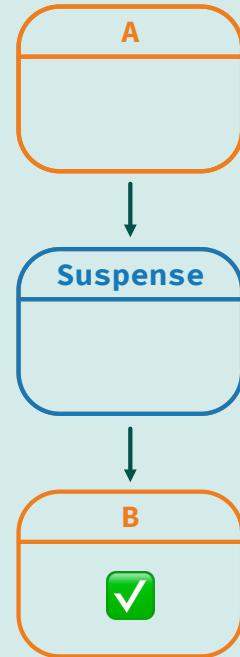
- Asynchron geladenes JS, asynchron geladene Daten, Promises, ...



SUSPENSE

Suspense: Unterbricht das Rendern, solange "etwas" fehlt

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



Suspense: Funktioniert aktuell (React 18) für **Code Splitting**

- Code Splitting in Server-Komponenten eingebaut
- **In der Zukunft** auch zum **Laden von beliebigen Daten** (Client und Server)
 - "That will likely come after the 18.0 release, but we're hoping that to have something during the next 18.x minor releases." (<https://github.com/reactwg/react-18/discussions/47#discussioncomment-847004>)

SUSPENSE

Beispiel: Daten laden auf dem Server

```
import {db} from "./db.server";  
  
function PostComments({post}) {  
  const comments = db.query(...); -----  
  return ...; // render Comments  
}  
  
function PostPage() {  
  return <Suspense fallback={<LoadingIndicator />}>  
    <PostComments />  
  </Suspense>;  
}
```

"Suspense for Data Loading"

- Zugriff auf "etwas", das Daten lädt und Aufruf blockiert bis Daten da sind

SUSPENSE

Beispiel: Daten laden auf dem Server

```
import {db} from "./db.server";

function PostComments({post}) {
  const comments = db.query(...);

  return ...; // render Comments
}

function PostPage() {
  return <Suspense fallback={<LoadingIndicator />}>
    <PostComments />
  </Suspense>;
}
```

Suspense-Komponente

- "Sollbruchstelle", wenn unterhalb in der Anwendung auf "etwas" gewartet wird, wird `fallback` angezeigt

SUSPENSE

Beispiel: Suspense



Demo (falls noch Zeit ist)

- Delay für PostList und TagCloud aktivieren (delay.server.js)
 - Daten bleiben gecached (Home => Post => Home)
 - Suspense in PostListPage verschieben
-
- Delay für Post aktivieren
 - Einzelnen Post aufrufen
 - Suspense in Post-Komponente deaktivieren

Server Components

SERVER COMPONENTS

Aktueller Stand

- Nicht in React 18 enthalten

Server Components is Still in Development

Server Components is an upcoming feature that allows developers to build apps that span the server and client, combining the rich interactivity of client-side apps with the improved performance of traditional server rendering. Server Components is not inherently coupled to Concurrent React, but it's designed to work best with concurrent features like Suspense and streaming server rendering.

<https://reactjs.org/blog/2022/03/29/react-v18.html>

Aktueller Stand

- Nicht in React 18 enthalten
- Beta-Support u.a. in Next.js
- Vieles noch offen, u.a. Support für die üblichen Bundler

Server Components is Still in Development

Server Components is an upcoming feature that allows developers to build apps that span the server and client, combining the rich interactivity of client-side apps with the improved performance of traditional server rendering. Server Components is not inherently coupled to Concurrent React, but it's designed to work best with concurrent features like Suspense and streaming server rendering.

Server Components is still experimental, but we expect to release an initial version in a minor 18.x release. In the meantime, we're working with frameworks like Next.js, Hydrogen, and Remix to advance the proposal and get it ready for broad adoption.

<https://reactjs.org/blog/2022/03/29/react-v18.html>

NILS HARTMANN
<https://nilshartmann.net>



vielen Dank!

Slides: <https://react.schule/codetalks-react>

Fragen & Kontakt: nils@nilshartmann.net

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