

**NILS HARTMANN**

<https://nilshartmann.net>

Slides: <https://react.schule/bedcon2023>

# Fullstack React

am Beispiel Next.js

# NILS HARTMANN

nils@nilshartmann.net

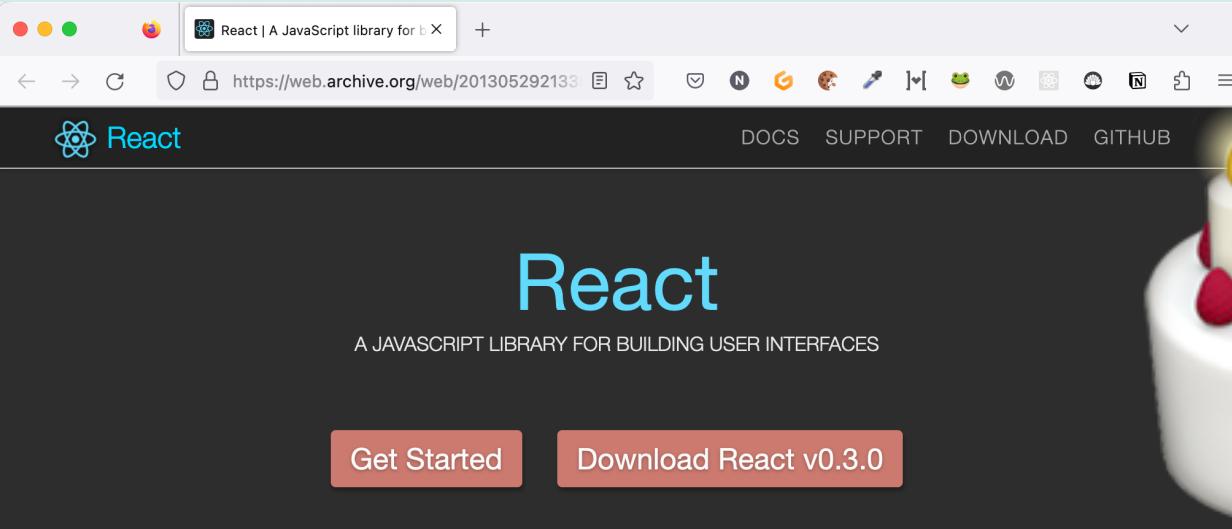
**Freiberuflicher Entwickler, Architekt, Trainer aus Hamburg**  
**Java, Spring, GraphQL, React, TypeScript**



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

<https://reactbuch.de>

**HTTPS://NILSHARTMANN.NET**



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

The screenshot shows the React homepage from May 29, 2013. The page features a large central title "React" with the subtitle "A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES". Below the title are two prominent red buttons: "Get Started" and "Download React v0.3.0". To the right of the main content area, there is a large, stylized image of a white birthday cake with four lit candles and red frosting. The date "29.05.2013" is displayed next to the cake. The browser's address bar shows the URL "https://web.archive.org/web/201305292133/".

VOR ZEHN JAHREN...



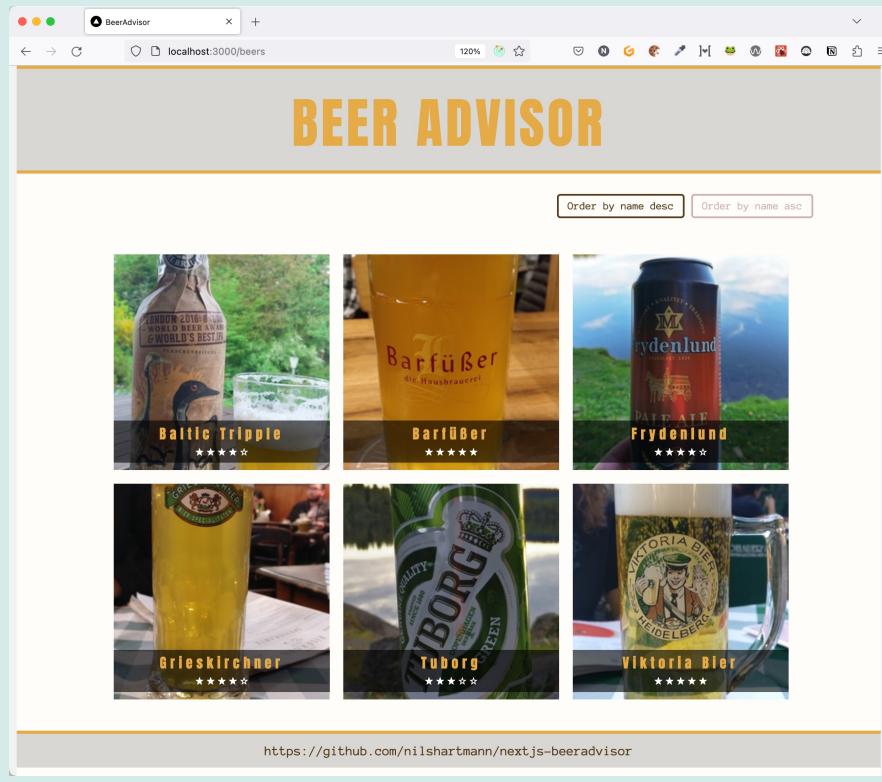
<https://web.archive.org/web/20130529213355/http://facebook.github.io/react/>

**SEIT ZEHN MAL...**

# 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/>



Beispiel-Code: <https://github.com/nilshartmann/nextjs-beeradvisor>

## EIN BEISPIEL...

# EIN BEISPIEL

## Was macht die Beispiel-Anwendung aus?

- Viel statischer Content 😊

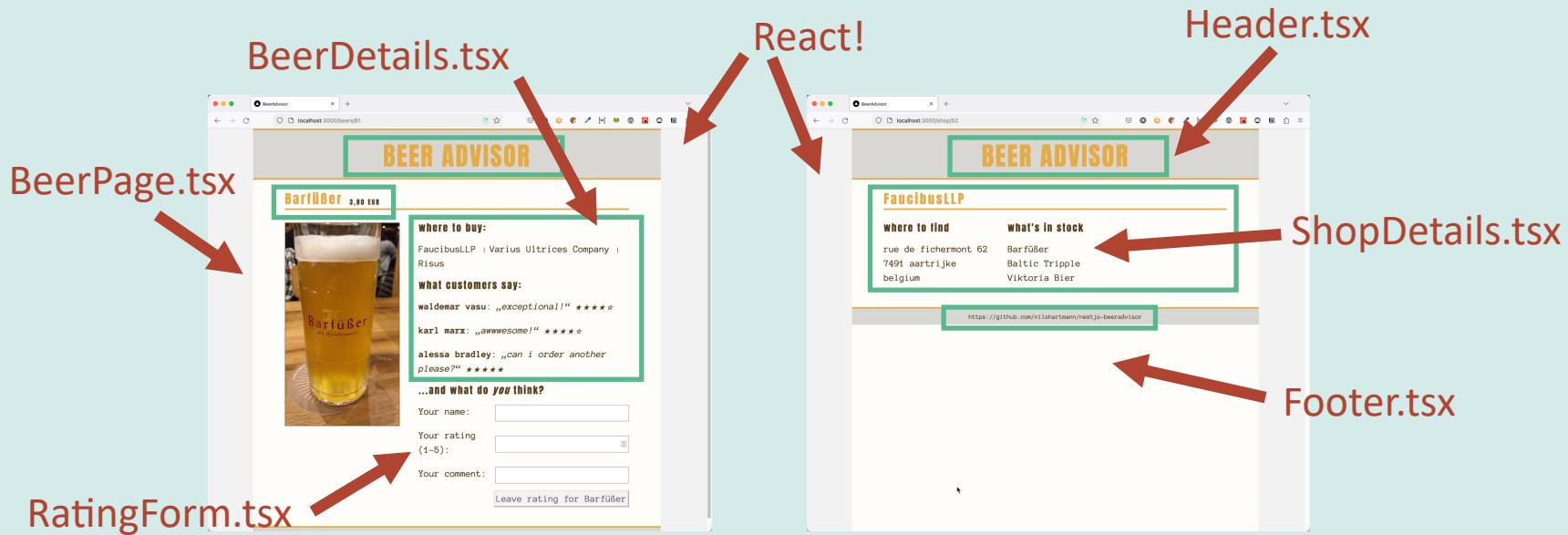
The screenshot shows a web browser window titled "Beer Advisor" at the URL "localhost:3000/beer/81". The main heading is "BEER ADVISOR". A green box highlights the first beer entry: "Barfüßer 3,80 EUR". Below it is a photograph of a glass of light beer with a white head, labeled "Barfüßer". To the right of the image, the name "Barfüßer" and its price "3,80 EUR" are displayed. A section titled "Where to buy:" lists "FaucibusLLP | Varius Ultrices Company | Risus". Another section titled "what customers say:" contains reviews from "waldemar vasu", "karl marx", and "alessa bradley", each accompanied by a five-star rating icon. At the bottom, there's a form for users to leave their own rating and comment: "...and what do **you** think?", "Your name:", "Your rating (1-5):", and "Your comment:". A button at the bottom right says "Leave rating for Barfüßer".

The screenshot shows a web browser window titled "Beer Advisor" at the URL "localhost:3000/beer/82". The main heading is "BEER ADVISOR". A green box highlights the second beer entry: "FaucibusLLP". Below it is a photograph of a glass of beer. To the right of the image, the name "FaucibusLLP" and its location "rue de fichermon 62 7491 aartrijke belgium" are displayed. A section titled "what's in stock" lists "Barfüßer", "Baltic Triple", and "Viktoria Bier". At the bottom, there's a link "https://github.com/n13hartmann/nextjs-beeradvisor".

# EIN BEISPIEL

## Was macht die Beispiel-Anwendung aus?

- Viel statischer Content 😊
- Viel JavaScript 😱



# EIN BEISPIEL

## Was macht die Beispiel-Anwendung aus?

- Viel statischer Content 😊
- Viel JavaScript 😱
- ...gleichzeitig wenig Interaktion 😞

The screenshot shows a web browser window titled "BEER ADVISOR". The main content is for "Barfüßer 3,80 EUR". It includes a photo of a beer glass, a section for "Where to buy:", and a "what customers say:" section with reviews from Waldemar Vasu, Karl Marx, and Alessa Bradley. At the bottom, there's a form for users to leave a rating and comment, with a red border around it.

The screenshot shows a web browser window titled "BEER ADVISOR". The main content is for "FaucibusLLP". It includes a "where to find" section with address and city information, and a "what's in stock" section with beer names like Barfüßer, Baltic Triple, and Viktoria Bier. A URL "https://github.com/nishartmann/reactjs-beeradvisor" is visible at the bottom of the page.

## EIN BEISPIEL

### Anforderung

👉 Die Seiten sollen möglichst schnell für den Benutzer **sichtbar** und **bedienbar** sein

## EIN BEISPIEL

### Mögliche Probleme

- (Viel) JavaScript-Code, der...

## EIN BEISPIEL

### Mögliche Probleme

- (Viel) JavaScript-Code, der...
  - ... vom Browser geladen werden muss

### Mögliche Probleme

- (Viel) JavaScript-Code, der...
  - ... vom Browser geladen werden muss
  - ... interpretiert und ausgeführt werden muss

### Mögliche Probleme

- (Viel) JavaScript-Code, der...
  - ... vom Browser geladen werden muss
  - ... interpretiert und ausgeführt werden muss
- ...und mit jeder neuen Komponente mehr wird

# "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:**

- Platzhalter für "langsame" Teile einer Seite
- Mit Streaming können diese Teile einer Seite "nachgeliefert" werden, sobald sie gerendert sind

## React empfiehlt "Fullstack-Framework"

- **Server Components** erfordern Rendern auf dem Server oder im Build
- Dazu braucht man ein "**Fullstack-Framework**"

## React empfiehlt "Fullstack-Framework"

- **Server Components** erfordern Rendern auf dem Server oder im Build
- Dazu braucht man ein "**Fullstack-Framework**"
- "**Framework**" ist verharmlosend, weil es sich in der Regel um einen kompletten Stack samt Build-Tools und Laufzeitumgebung handelt

## React empfiehlt "Fullstack-Framework"

- **Server Components** erfordern Rendern auf dem Server oder im Build
- Dazu braucht man ein "**Fullstack-Framework**"
- "**Framework**" ist verharmlosend, weil es sich in der Regel um einen kompletten Stack samt Build-Tools und Laufzeitumgebung handelt
- Deswegen werden solche Frameworks auch als "**Meta-Frameworks**" bezeichnet (=> Sammlung von Frameworks)

## React empfiehlt "Fullstack-Framework"

- **Next.js** entspricht den Vorstellungen des React-Team
  - Mit dem **App-Router**, stabil ab Next.js 13.4
  - Einige React-Entwickler sind zu Vercel in das Next.js-Team gewechselt

### React empfiehlt "Fullstack-Framework"

- **Next.js** entspricht den Vorstellungen des React-Team
  - Mit dem **App-Router**, stabil ab Next.js 13.4
  - Einige React-Entwickler sind zu Vercel in das Next.js-Team gewechselt
- **Remix** ist aus dem React Router Projekt entstanden
  - Viele ähnliche Konzepte ("React Router mit Server")

**Zero-Bundle-Size**

**Server**

**Components**

## SERVER COMPONENTS

**Idee:** Komponenten werden nicht im Client ausgeführt

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

## SERVER COMPONENTS

### Arten von Komponenten

# ARTEN VON KOMPONENTEN

## Client-Komponenten (wie bisher)

- Werden auf dem Client gerendert

## BEER ADVISOR

**Barfüßer** 3,80 EUR



**where to buy:**

FaucibusLLP | Varius Ultrices Company | Risus

**what customers say:**

waldemar vasu: „exceptional!“ ★★★★☆

karl marx: „awwwesome!“ ★★★★☆

alessa bradley: „can i order another please?“ ★★★★☆

**...and what do you think?**

Your name:

Your rating (1-5):

Your comment:

Leave rating for Barfüßer

# ARTEN VON KOMPONENTEN

## Client-Komponenten (wie bisher)

- Werden auf dem Client gerendert
- oder auf dem Server 😊

### BEER ADVISOR

**Barfüßer** 3,80 EUR



**where to buy:**  
FaucibusLLP | Varius Ultrices Company | Risus

**what customers say:**  
waldemar vasu: „exceptional!“ ★★★★☆  
karl marx: „awwwesome!“ ★★★★☆  
alessa bradley: „can i order another please?“ ★★★★☆

**...and what do you think?**

Your name:

Your rating (1-5):

Your comment:

Leave rating for Barfüßer

# ARTEN VON KOMPONENTEN

## Client-Komponenten (wie bisher)

- Werden auf dem Client gerendert
  - oder auf dem Server 😊
- 
- JavaScript-Code immer zum Client gesendet
  - Können deshalb interaktiv sein

**BEER ADVISOR**

**Barfüßer** 3,80 EUR

**where to buy:**  
FaucibusLLP | Varius Ultrices Company | Risus

**what customers say:**  
waldemar vasu: „exceptional!“ ★★★★☆  
karl marx: „awwwesome!“ ★★★★☆  
alessa bradley: „can i order another please?“ ★★★★☆

**...and what do you think?**

Your name:

Your rating (1-5):

Your comment:

Leave rating for Barfüßer



## ARTEN VON KOMPONENTEN

### Neu: Server-Komponenten

- werden auf dem Server gerendert

### Neu: Server-Komponenten

- werden auf dem Server gerendert
- oder im Build 😊

### Neu: Server-Komponenten

- werden auf dem Server gerendert
- oder im Build 😊
  
- liefern UI (!) zum React-Client zurück (kein JavaScript-Code)
- API: "normale" React-Komponenten (JS/TS, JSX, ...)

### Neu: Server-Komponenten

- werden auf dem Server gerendert
- oder im Build 😊
  
- liefern UI (!) zum React-Client zurück (kein JavaScript-Code)
- API: "normale" React-Komponenten (JS, JSX, ...)

# ARTEN VON KOMPONENTEN

Komponenten können gemischt werden

**BEER ADVISOR**

**Barfüßer 3,80 EUR**



where to buy:  
Faucibus LLP | Varius Ultrices Company | Risus

what customers say:  
Waldemar: „exceptional!“ ★★★★☆  
I mark: „awwwesome!“ ★★★★☆  
alessa bradley: „can i order another please?“ ★★★★☆

...and what do you think?

Your name:

Your rating (1-5):

Your comment:

Leave rating for Barfüßer

**Server Components**

**Client Component**

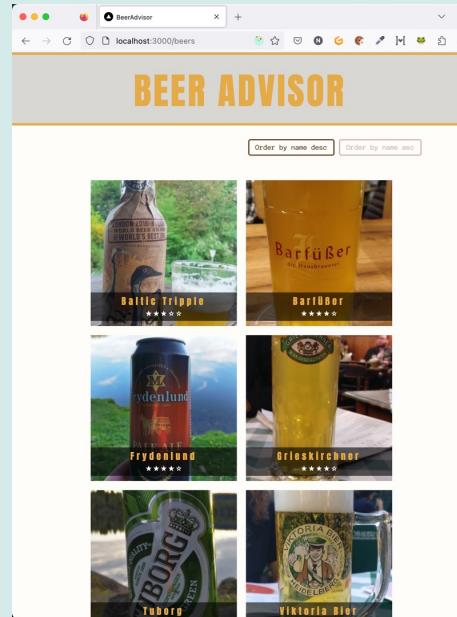
# RSC am Beispiel Next.js

## React empfiehlt "Fullstack-Framework"

- **Next.js** entspricht den Vorstellungen des React-Team
  - Mit dem **App-Router**, stabil ab Next.js 13.4

## Schritt-für-Schritt

- Beispiel-Code: <https://github.com/nilshartmann/nextjs-step-by-step>



## Schritt 1: Eine Server Komponente



Demo

- Landing-Page mit Link auf /beers
- Children in Layout
- console.log in Page-Komponente

# Data Fetching

## SERVER COMPONENTS

### Idee

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

## SERVER COMPONENTS

### Idee

- Komponenten, die Daten laden, können das direkt *in der Komponente* tun
- Server Components können die Server-Infrastruktur nutzen (DB, Filesystem)

## SERVER COMPONENTS

### Idee

- Komponenten, die Daten laden, können das direkt *in der Komponente* tun
- Server Components können die Server-Infrastruktur nutzen (DB, Filesystem)

👉 Server-Komponenten können dazu asynchron sein

## Schritt 2: Eine asynchrone Server-Komponente



Demo

- BeerListPage anlegen
- DB-Zugriff mit loadBeers
  - loadBeers zeigen
- BeerImageList verwenden, um Beers anzuzeigen
- 🔎 **statische Komponenten bislang! (Build!)**

## Schritt 3: Eine asynchrone Server-Komponente, die träge ist



Demo

- beers/[beerId] Beer-Page mit DB (loadBeer)
- `type BeerPageProps = { params: { beerId: string } };`
- Fertige Komponente aus beer-details-page-fragment.tsx kopieren
- Aufruf künstlich verzögern (sleep in loadBeer)
- loading.tsx
- prefetch auf der /beers-Seite

## Schritt 3: Eine asynchrone Server-Komponente, die zwei Daten Quellen benötigt

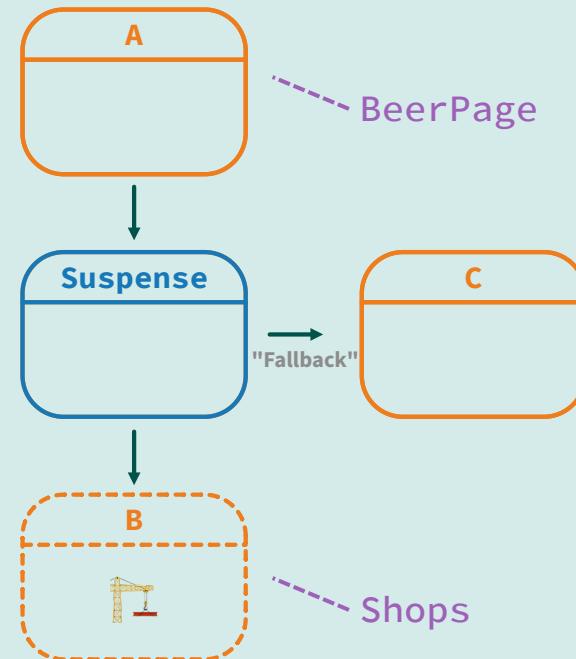


Demo

- beers/[beerId] Beer-Page wieder schnell machen (slow entfernen)
- beers/[beerId] Beer-Page shops erweitern (fertiges fetch in db-queries)
- Zeigen: Promise an Unterkomponente (Shops)
  - -> Parallel fetching!
- Aufruf künstlich verzögern (slow=2400)
- 😞 Jetzt wartet die ganze Seite auf die Shops...
- -> Suspense vorstellen (Slides)

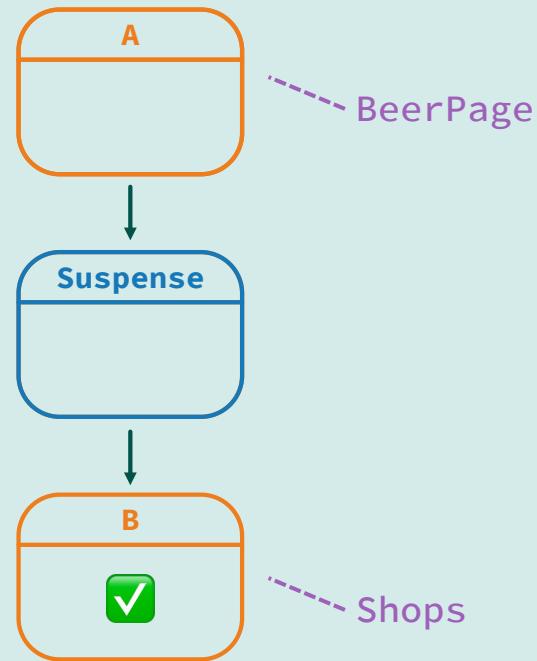
## SUSPENSE

**Suspense:** Unterbricht das Rendern, solange "etwas" fehlt



## SUSPENSE

**Suspense:** Unterbricht das Rendern, solange "etwas" fehlt



## Schritt 4: Suspense



Demo

- Aufruf von `loadShops`  
verlangsamen (`[beerId]/page.tsx`)
- Suspense in `BeerDetails`  
einführen

# Aufteilung in Server-Client: Konsequenzen

```
type BeerListProps = {
  beers: SingleBeer[];
  onToggleOrder(): void;
};

export default function BeerList({ beers, onToggleOrder }: BeerListProps) {
  return (
    <div>
      <h1>Beers</h1>

      <ul>
        {beers.map((b) => (
          <li key={b.id}>{b.name}</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.

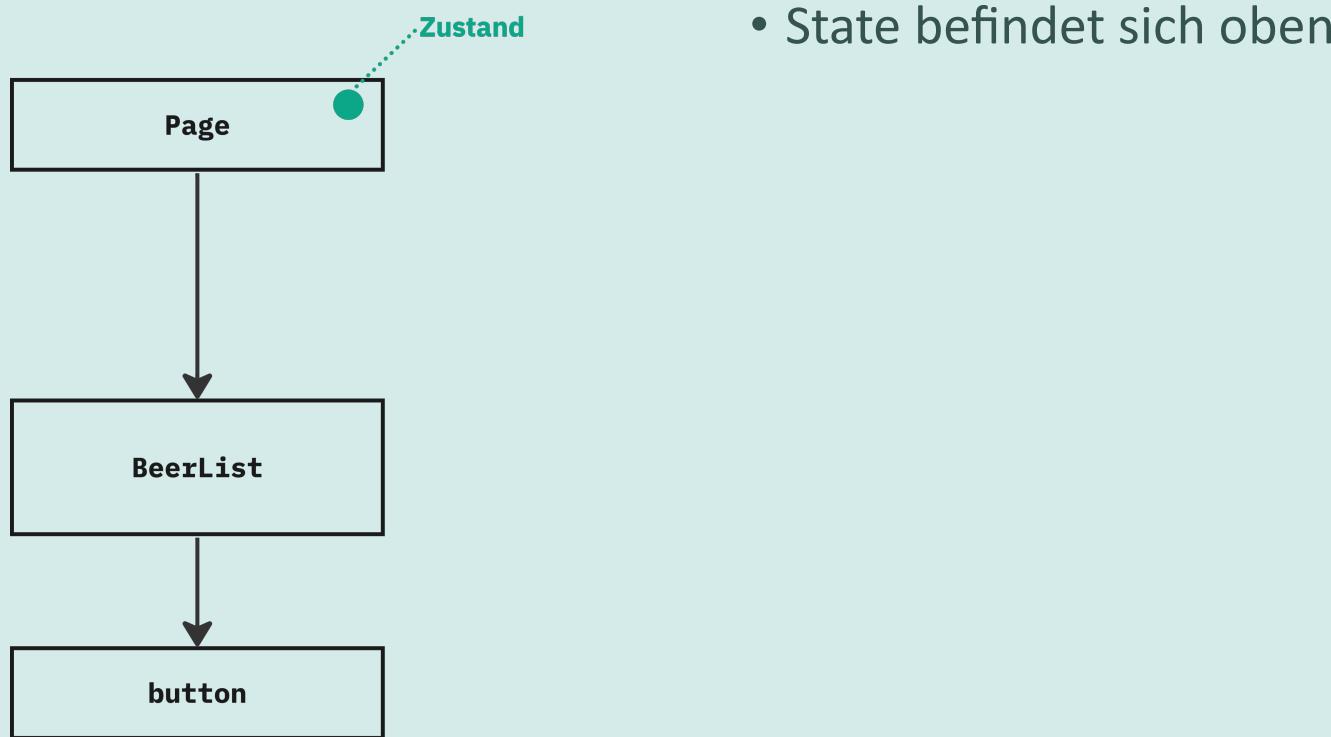
```
<button onClick={function} children=...>  
    ^^^^^^^^^^
```

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

at stringify (<anonymous>)

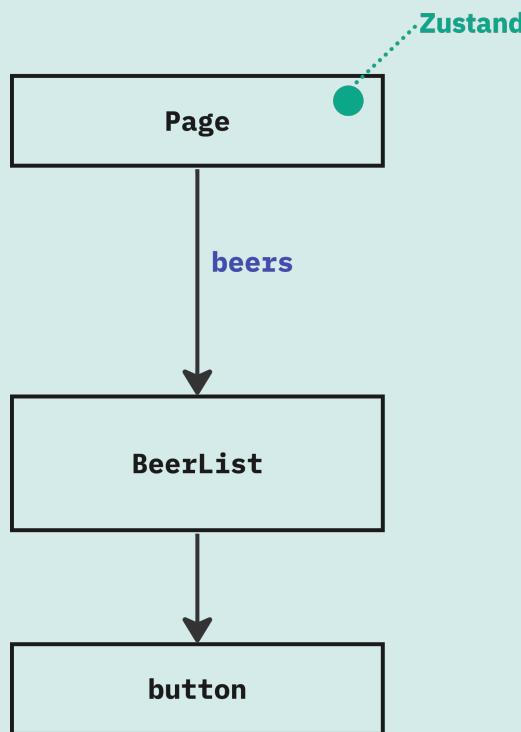
CAN YOU SPOT THE PROBLEM?

# EINE REACT ANWENDUNG IM BROWSER



Eine "normale" React-Anwendung...

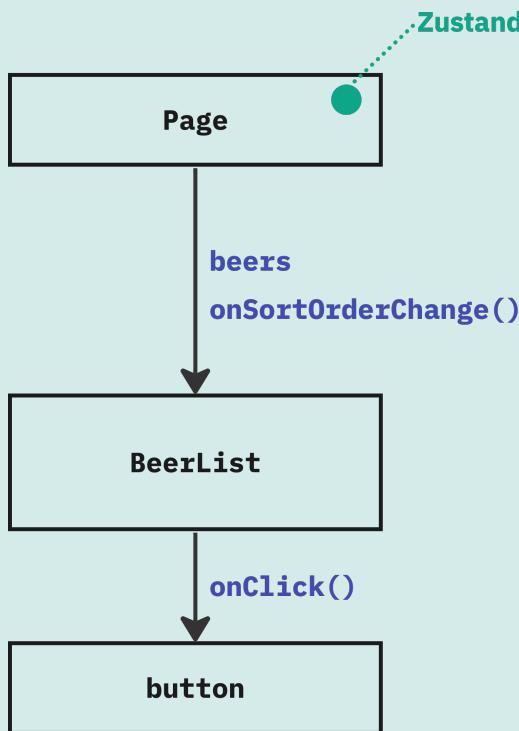
# EINE REACT ANWENDUNG IM BROWSER



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

Eine "normale" React-Anwendung...

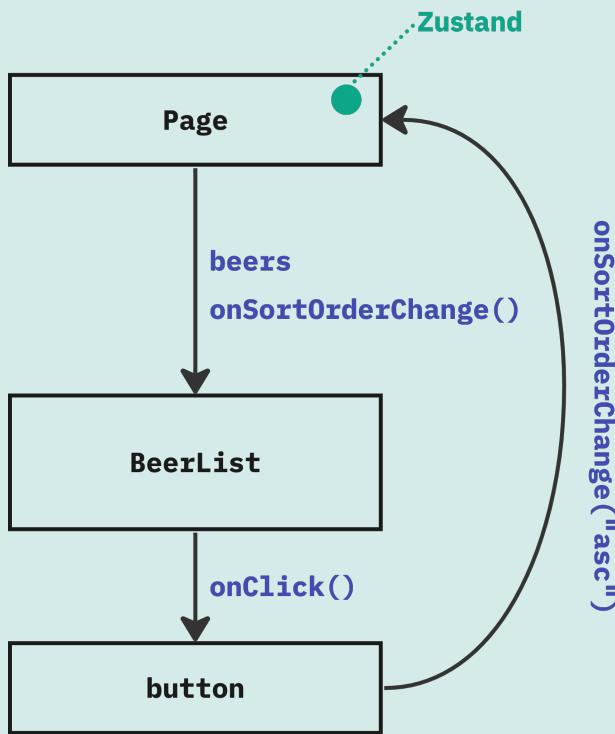
# EINE REACT ANWENDUNG IM BROWSER



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

Eine "normale" React-Anwendung...

# EINE REACT ANWENDUNG IM BROWSER

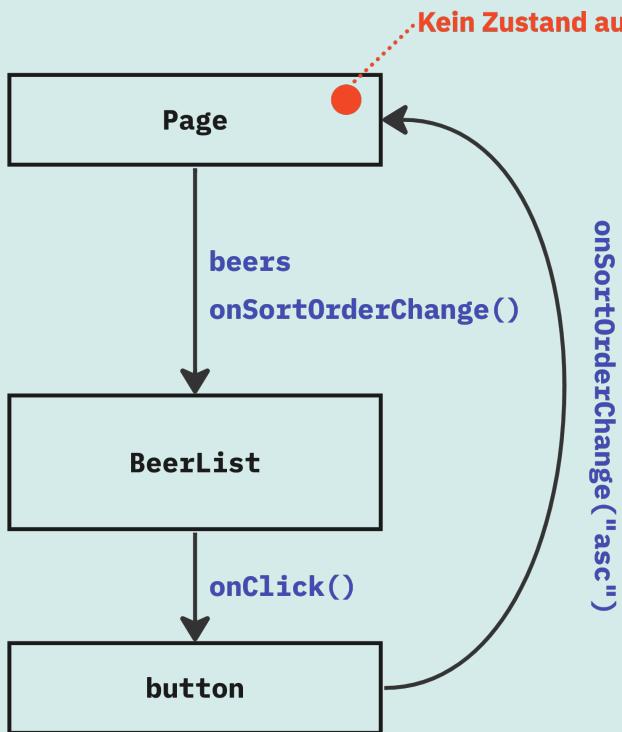


- State befindet sich oben
- Daten werden runtergereicht ("props")
- Callbacks werden runtergereicht
- Über Callbacks kann State-Veränderung ausgelöst werden

Eine "normale" React-Anwendung...

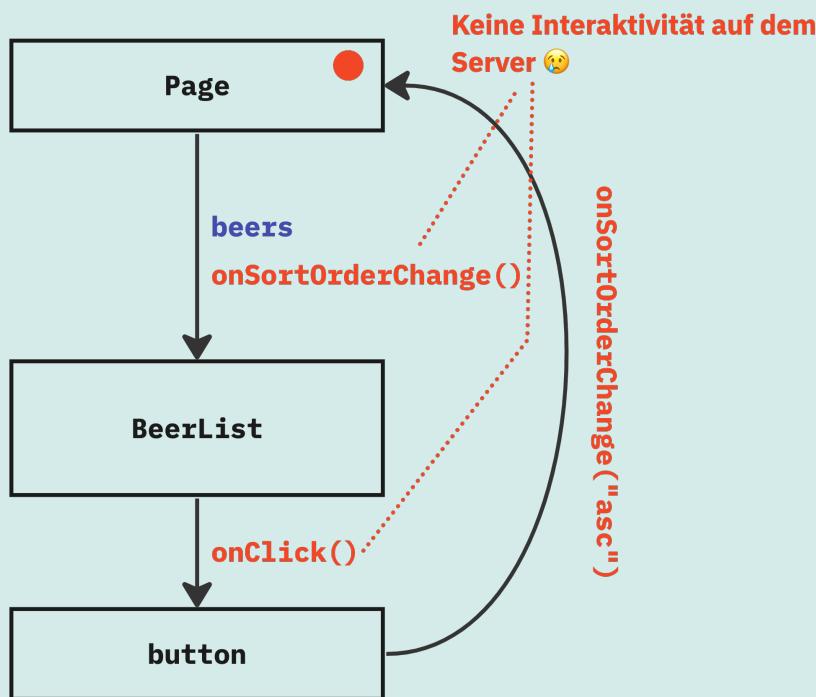
## ...UND AUF DEM SERVER

- Auf dem Server gibt es keinen State!



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

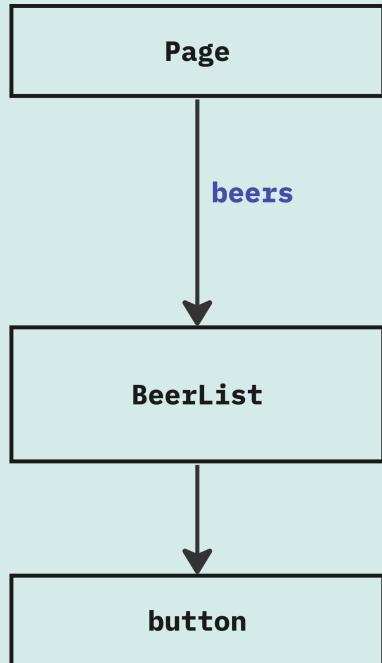
## ...UND AUF DEM SERVER



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

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

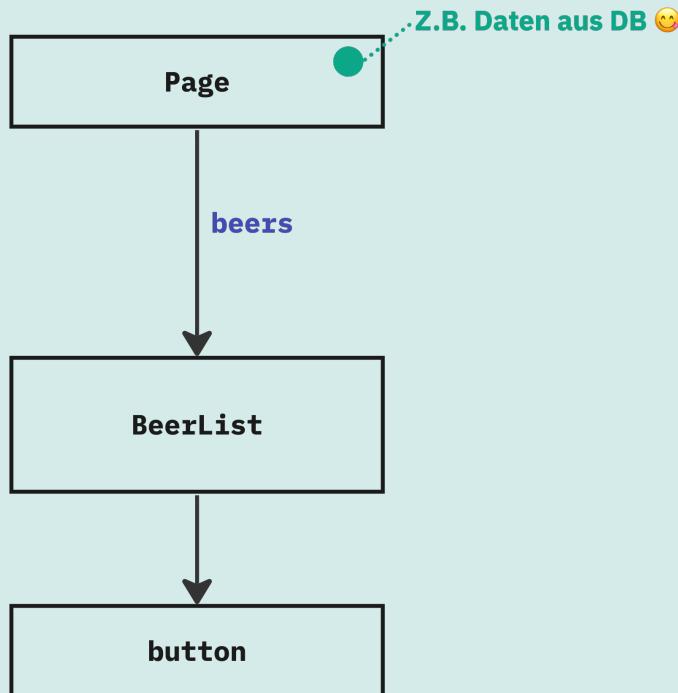
## ...UND AUF DEM SERVER



- 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

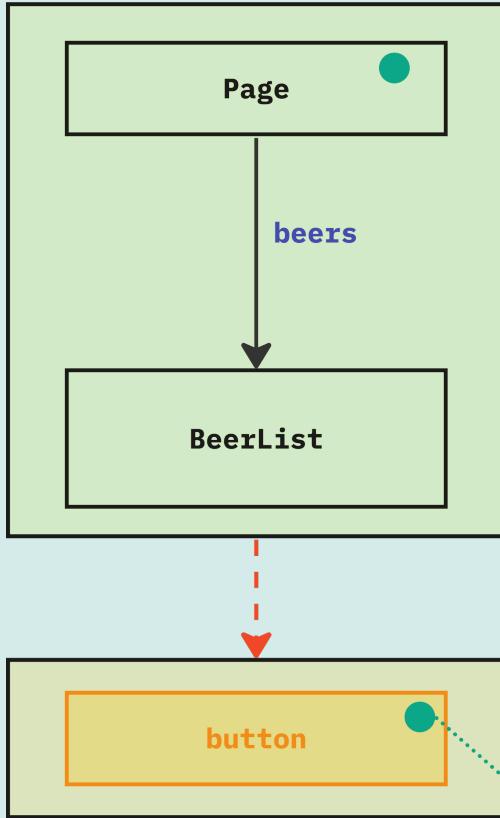


- Auf dem Server gibt es keinen State!
- ...und keine Interaktion
- Wir haben nur statischen Content
- Wir haben aber **Daten**  
z.B. aus DB, Microservice, Filesystem...

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

## ...UND AUF DEM SERVER

Server



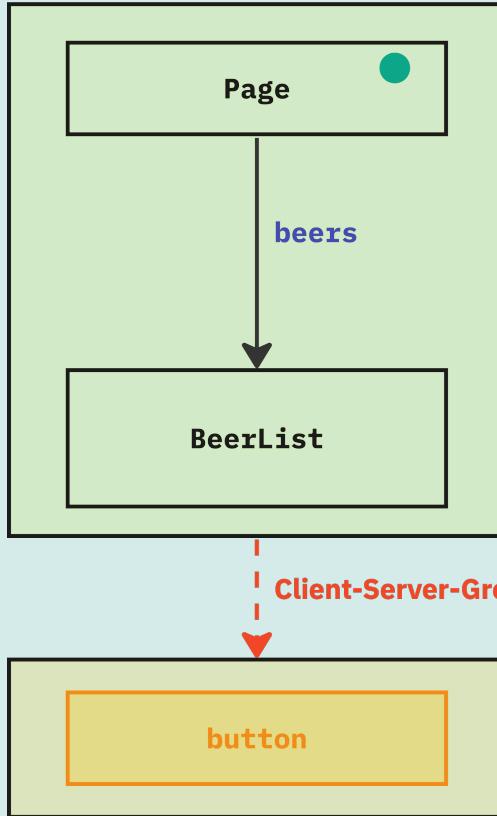
- Bestimmte Teile müssen auf den Client
  - Event-Handler
  - Lokaler State
  - Effekte

Client

Interaktives muss auf den Client 😎

## ...UND AUF DEM SERVER

Server

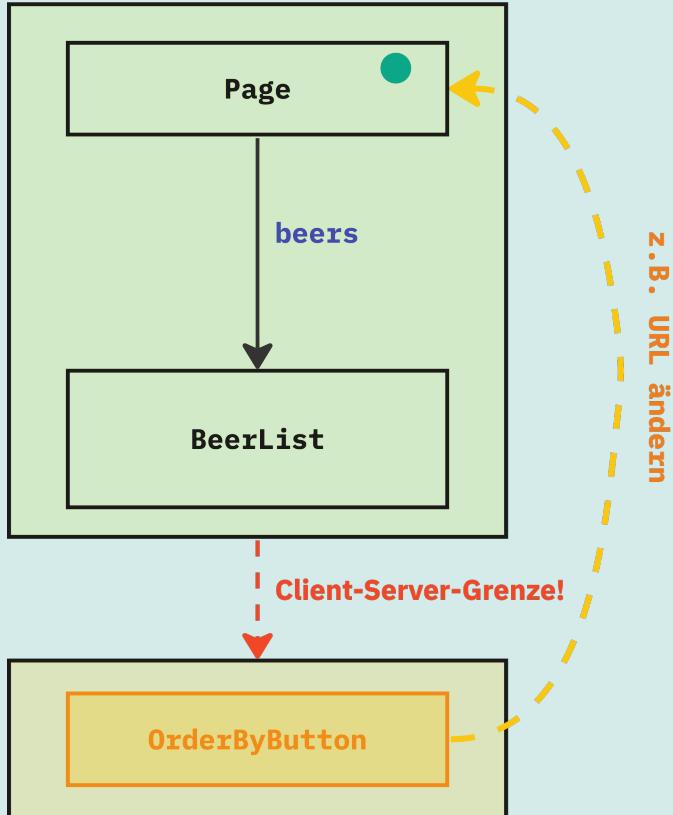


- Properties müssen hier Client-Server-Grenze überwinden
- Müssen serialisierbare Daten sein
- **Keine (Callback-)Funktionen!**

Client

## ...UND AUF DEM SERVER

Server

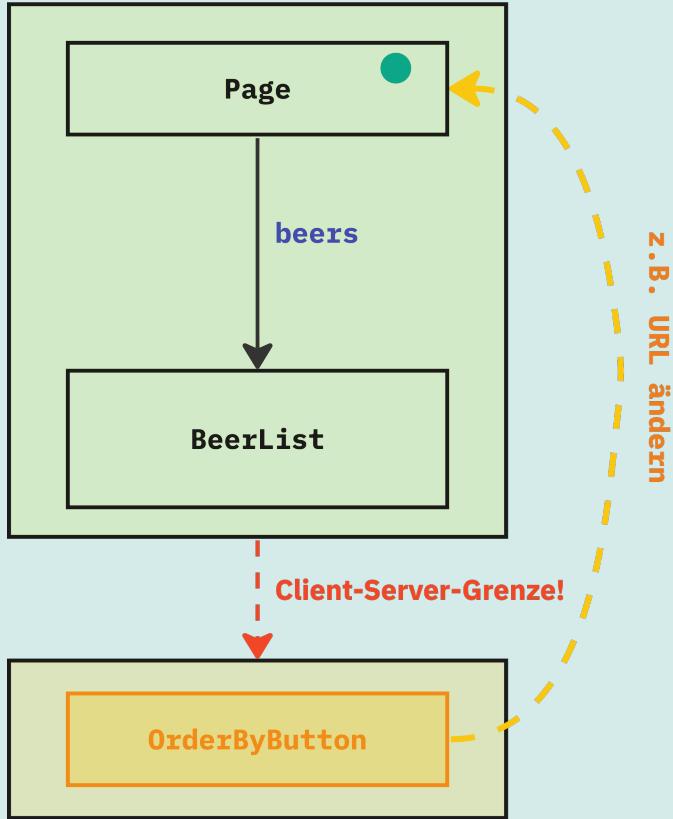


- Properties müssen Client-Server-Grenze überwinden
- Müssen serialisierbare Daten sein
- Keine (Callback-)Funktionen!
- Zur Kommunikation müssen **Server-Requests** durchgeführt werden
  - z.B. URL ändern

Client

## ...UND AUF DEM SERVER

Server

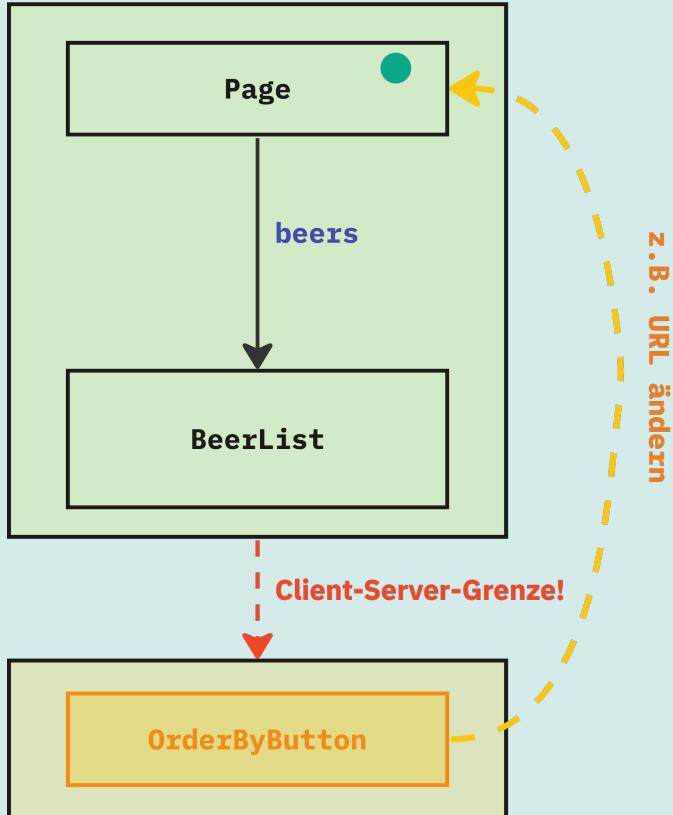


- Properties müssen Client-Server-Grenze überwinden
- Müssen serialisierbare Daten sein
- Keine (Callback-)Funktionen!
- Zur Kommunikation müssen Server-Requests durchgeführt werden
  - z.B. URL ändern
- **Server-Komponente hat Zugriff auf Request Informationen**
  - URL mit Search Params
  - Cookies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

Client

## ...UND AUF DEM SERVER

Server



### • Konsequenz

- Code wird durch URL-Handling komplexer?
- Wo ziehen wir Server/Client-Grenze?
  - Button? Ganzes Formular?
  - Hier werden sich Patterns entwickeln
- Ganze Seite (oder Teile) werden neu generiert
- Fertiges UI kommt dafür vom Server
  - Das kann mehr Daten als bei (REST-)API-Call bedeuten!

Client

## Schritt 5: Eine Client-Komponente



### Demo

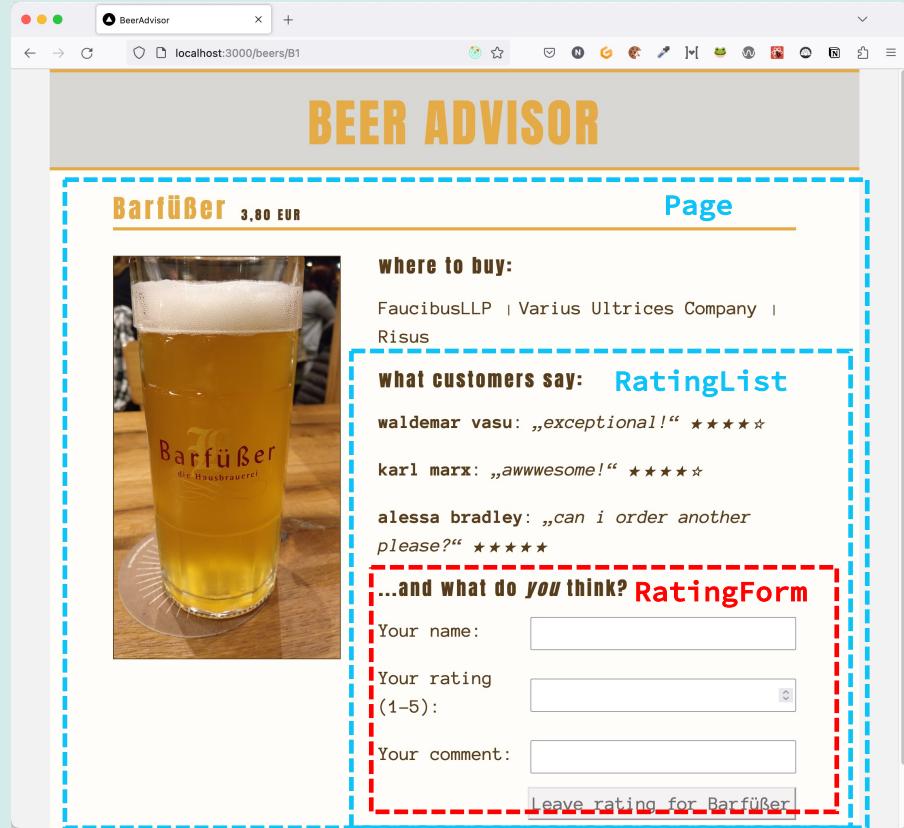
- OrderByButton Komponente bauen
  - OrderBy-Type als Property ("name\_asc" , "name\_desc")
  - Client-Komponente: Hooks möglich, useBeerAdvisorSearchParams
    - URL als "globaler Zustand"
- In BeerListPage einbauen
- In BeerListPage abhängig von SearchParams sortieren
  - An dieser Stelle Server Komponente, d.h. Hook ist hier nicht verwendbar
- ```
type BeerListPageProps = {
  searchParams?: { [key: string]: string };
};

const orderBy: OrderBy = (searchParams?.order_by || "name_asc") as OrderBy;
```

# MUTATIONS

## Verändern von Daten: Hinzufügen einer Bewertung

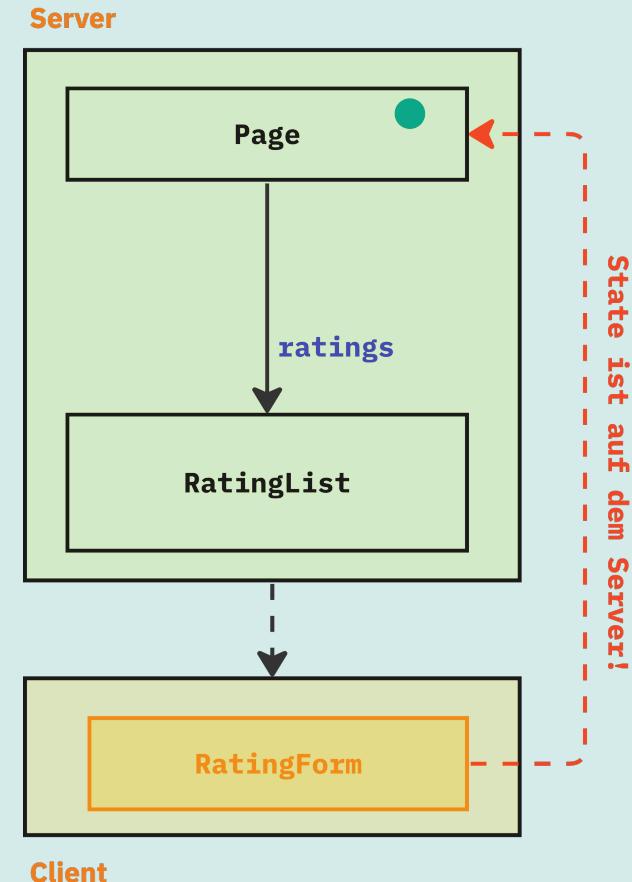
Server-Komponente  
Client-Komponente



## MUTATIONS

### Verändern von Daten

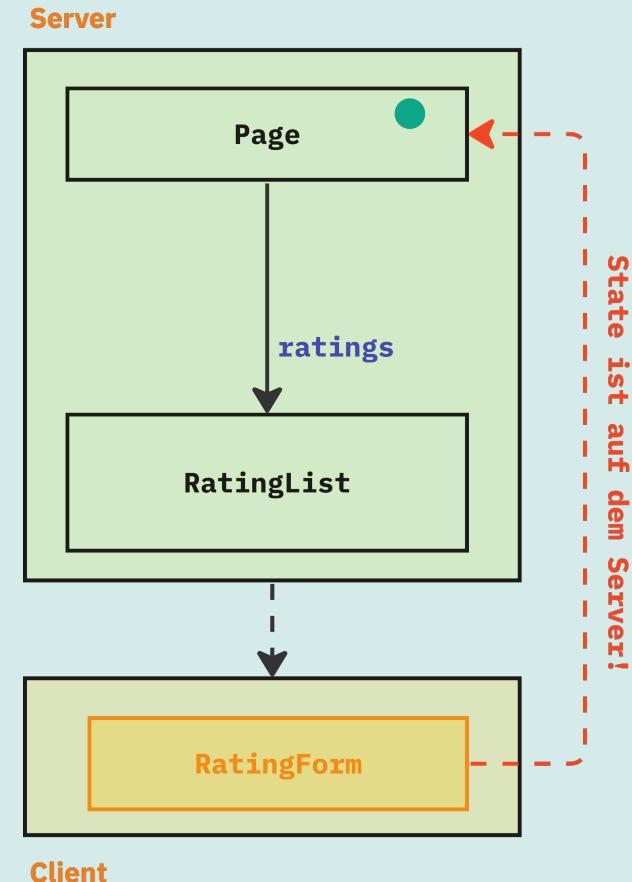
- Nach dem Verändern von Daten muss die UI aktualisiert werden
- Mangels State auf dem Client geht das aber nicht wie bislang



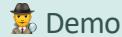
## MUTATIONS

### Verändern von Daten

- Nach dem Verändern von Daten muss die UI aktualisiert werden
- Mangels State auf dem Client geht das aber nicht wie bislang
- Der **Server** muss nach Datenänderungen **aktualisierte UI** liefern



## Schritt 6a: Exkurs API Endpunkte mit Next.js



Demo

- **API Route Handler**
  - Beispiel: GET Endpunkt implementieren !!
  - api/beers/route.ts
  - GET mit simplem NextResponse.json

## Schritt 6: Ein Formular

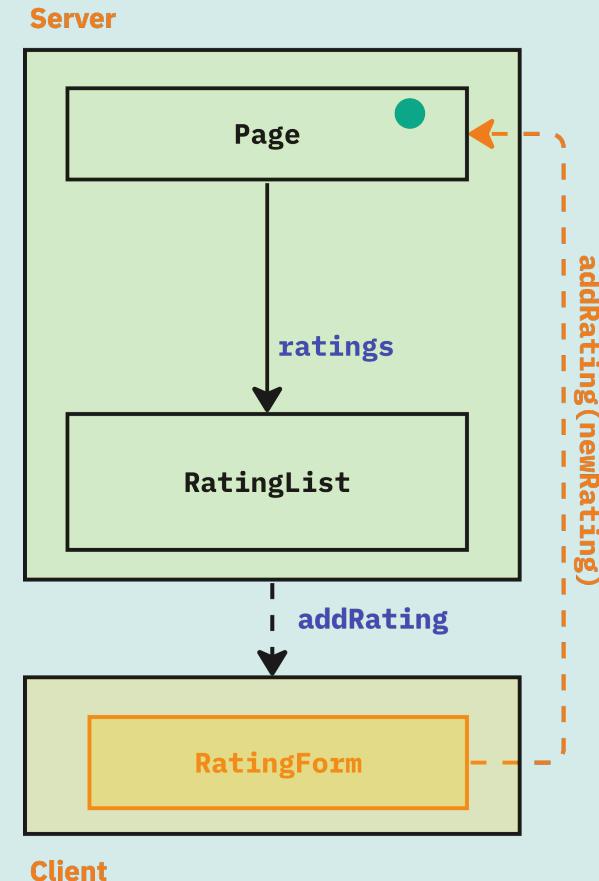


- API Route Handler
  - Fertigen POST Handler kopieren! (api/beers/[beerId])
- In AddRatingForm onSave implementieren
- mit fetch:
  - POST `/api/beers/\${beerId}/rating`
  - Content-Type-Header
  - Body: AddRatingRequestBody)
- Was passiert mit der Darstellung => nix
  - Warum?
- router.refresh()

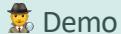
## MUTATIONS

### Alternative: Server Actions

- **Experimentelles** React-Feature
- Das ist eine Art Remote Funktion, die aus einer Server- oder Client-Komponente aufgerufen werden kann

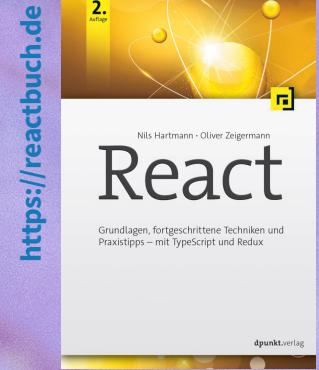


## Ausblick: Server Action



- Fertige Action in src-Verzeichnis kopieren (aus 99\_fertig)
  - Zeigen
- AddRatingForm anpassen
  - useTransition
  - wenn response.status === "created" dann ist alles gut, State zurücksetzen
  - Aktualisierung der UI: revalidatePath hinzufügen

**NILS HARTMANN**  
<https://nilshartmann.net>



# vielen Dank!

Slides: <https://react.schule/bedcon2023>

Source-Code: <https://github.com/nilshartmann/nextjs-beeradvisor>

Fragen & Kontakt: [nils@nilshartmann.net](mailto:nils@nilshartmann.net)

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