

NILS HARTMANN

<https://nilshartmann.net>

Die Qual

der Wahl

Die passende

Frontend-  
Technologie

Slides: <https://react.schule/jax-2024>

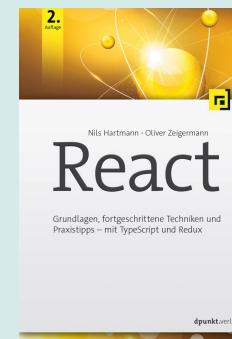
# NILS HARTMANN

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)

# BEISPIEL-CODE

Ihr findet das "Recipify"-Beispiel in unterschiedlichen Implementierungen (und leicht unterschiedlichem Feature-Set) hier:

HTMX und React SPA

<https://github.com/nilshartmann/jughh-spa-oder-htmx-backend>

Next.JS

<https://github.com/nilshartmann/nextjs-workshop>

# Frontend-Entwicklung ...

# Frontend-Entwicklung ...

NEXT.js



PREACT



Nuxt

Astro

qwik

</> htmx



React

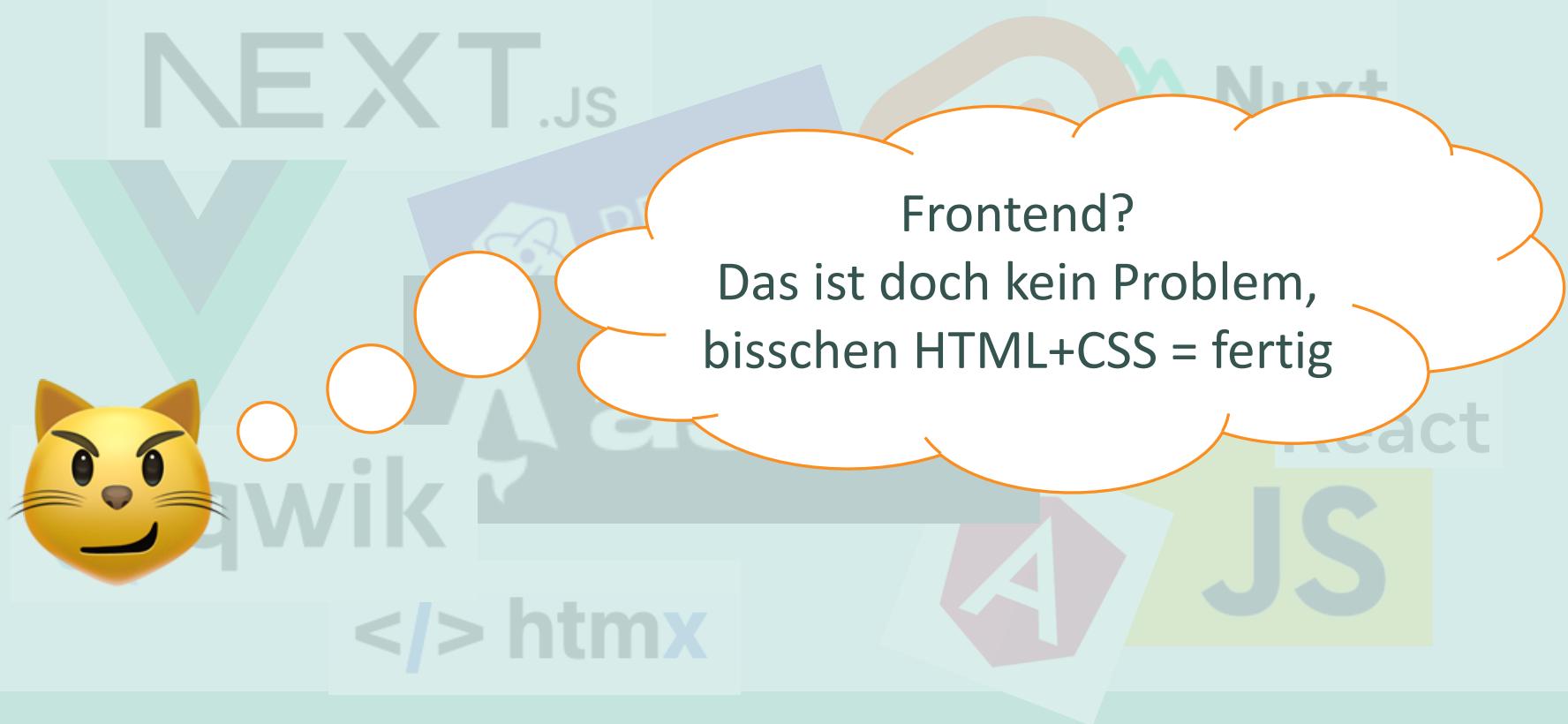
JS

# Frontend-Entwicklung ...

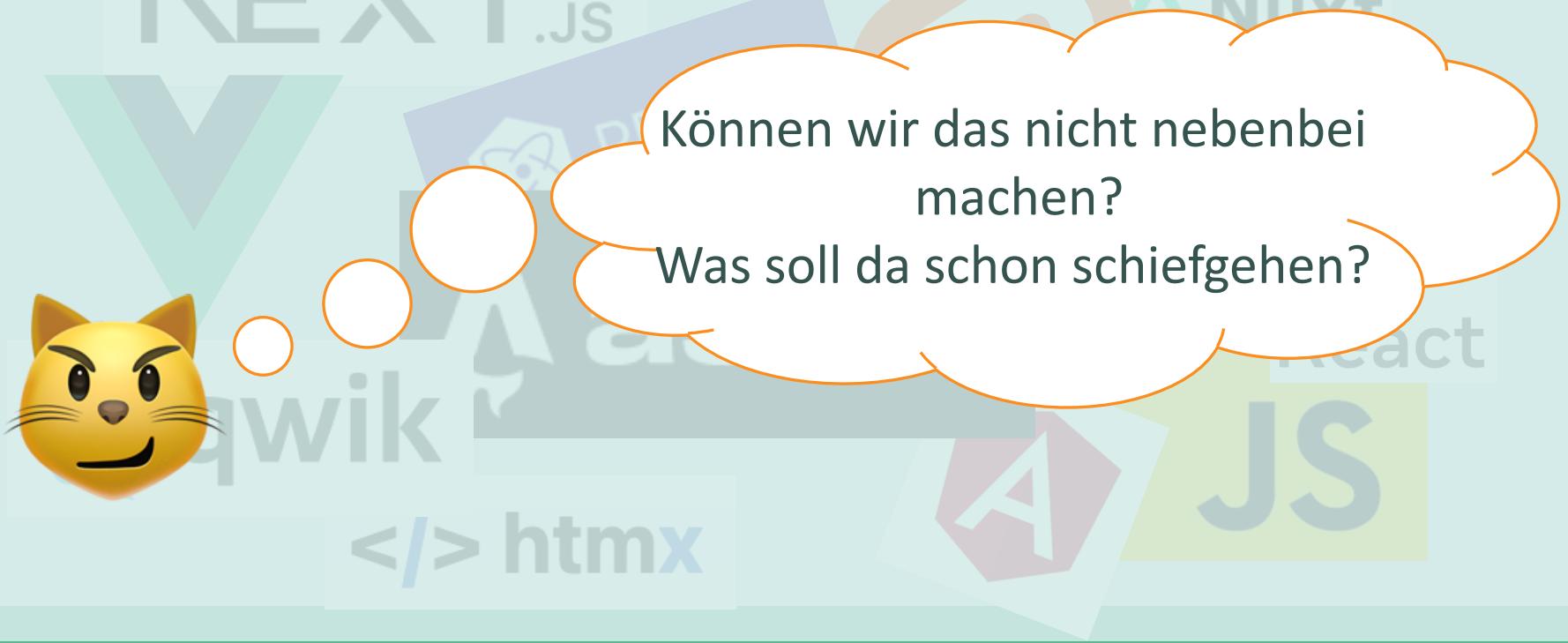


Was soll der ganze  
(JavaScript-)  
Quatsch?

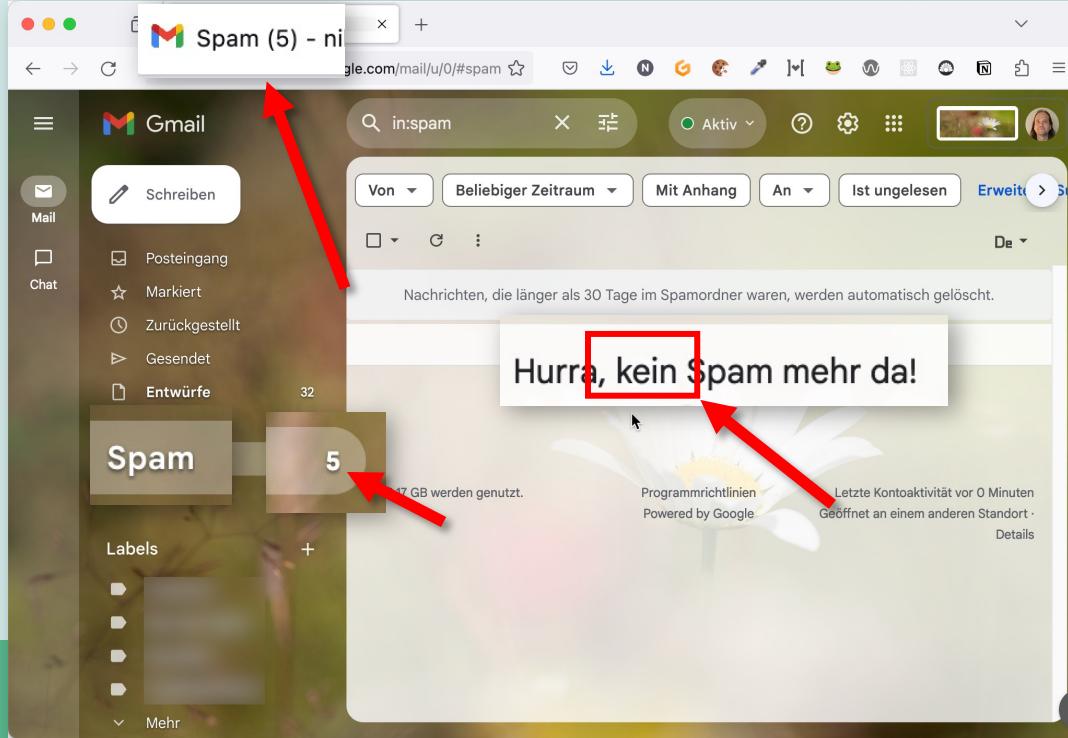
# Frontend-Entwicklung ...



# Frontend-Entwicklung ...



# Frontend-Entwicklung: Was soll schon schief gehen?



# Frontend-Entwicklung: Was soll schon schief gehen?



A screenshot of the WebStorm IDE interface. On the left, the project tree shows a folder structure for a Next.js application. In the center, the code editor displays `AppLink.tsx` with several lines of TypeScript code. A red arrow points from the text "No errors found by the IDE." at the bottom right of the editor area to the status bar at the bottom, which also says "No errors found by the IDE." To the right of the editor, a floating window titled "Highlight: All Problems" shows a count of "4 errors". Another red arrow points from the text "No errors found by the IDE." at the bottom right to this floating window. On the far right, there is another large yellow cat emoji with a surprised expression.

```
AppLink() : className: "bg-highlight-light"
  1 > import ...
  5
  6 type AppLinkProps = Parameters<typeof Link>[0] & {
  7   type?: "Highlight" | "Bold";
  8 };
  9
 10 export function AppLink({ type, ...props }: AppLinkProps) {
 11   const className = clsx("underline hover:font-bold", {
 12     "bg-highlight-light": type === "Highlight",
 13     "font-bold": type === "Bold",
 14   });
 15
 16   return (
 17     <Link className={className} {...props} prefetch={false}></Link>
 18     <Link className={className} {...props} prefetch={false}></Link>
 19   );
 20 }
 21
 22
 23 export function ExternalLink({ type, href, children }: ExternalLinkProps) {
 24   const className = clsx("underline hover:font-bold", {
 25     "bg-highlight-light": type === "Highlight",
 26     "decoration-2": type === "SemiHighlight",
 27   });
 28
 29   return (
 30     <a href={href} type={type}>{children}</a>
 31   );
 32 }
```

4 errors

Highlight: All Problems

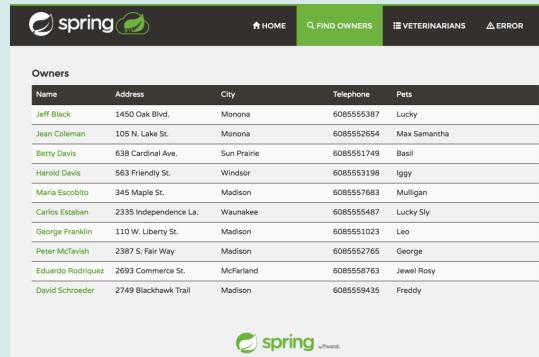
No errors found by the IDE.

No errors found by the IDE.

Share your feedback on WebStorm startup performance! Please answer a few questions. It will take about 2 minutes. // Respond // Don't show again (15 minutes ago)

12:32 @ () Language Services LF UTF-8 2 spaces

# WEBANWENDUNGEN



The screenshot displays a web application interface for managing pet owners. At the top, there is a navigation bar with the Spring logo, followed by links for HOME, FIND OWNERS (highlighted in green), VETERINARIANS, and ERROR.

The main content area is titled "Owners" and contains a table with the following data:

| Name              | Address               | City        | Telephone  | Pets         |
|-------------------|-----------------------|-------------|------------|--------------|
| Jeff Black        | 1450 Oak Blvd.        | Monona      | 6085555387 | Lucky        |
| Jean Coleman      | 105 N. Lake St.       | Monona      | 6085552654 | Mox Samantha |
| Betty Davis       | 638 Cardinal Ave.     | Sun Prairie | 6085511749 | Basil        |
| Harold Davis      | 563 Friendly St.      | Windsor     | 6085553198 | Iggy         |
| Maria Escobito    | 345 Maple St.         | Madison     | 608557683  | Mulligan     |
| Carlos Esteban    | 2335 Independence La. | Wauakee     | 6085555487 | Lucky Sly    |
| George Franklin   | 110 W. Liberty St.    | Madison     | 6085511023 | Leo          |
| Peter McTavish    | 2387 S. Fair Way      | Madison     | 6085552765 | George       |
| Eduardo Rodriguez | 2693 Commerce St.     | McFarland   | 6085558763 | Jewel Rosy   |
| David Schroeder   | 2749 Blackhawk Trail  | Madison     | 6085559435 | Freddy       |

At the bottom of the page, there is a footer with the Spring logo and the word "powered".

# WEBANWENDUNGEN

The screenshot shows a web browser window titled "ICE Portal" displaying travel information for ICE 109. At the top, it says "Reiseinformationen für den ICE 109". It indicates the next stop in 6 minutes at Gleis 2. The train is currently at Osnabrück Hbf, time 12:35, speed 108 km/h. A red bar highlights the speed. Below this, there's a map icon and a "Schnelles Internet" button. The main content area shows the route from Hamburg-Altona to Basel SBB, with stops like Hamburg-Altona, ICE 109 nach Basel SBB, and Gl. 10. Buttons for "Vergangene Halte zeigen", "Ab", "An", and "Suchen" are visible. The bottom of the page features promotional banners for "hw plus" and "hvv Deutschlandticket", along with social media links for "Mach mit!" and "Schüler\*innen".

The screenshot shows a web application for "spring". The header includes a logo, navigation links for "HOME", "FIND OWNERS", "VETERINARIANS", and "ERROR", and a search bar. The main content is a table titled "Owners" listing pet owners with their addresses, city, telephone number, and pets. The table has columns for Name, Address, City, Telephone, and Pets.

| Name              | Address               | City        | Telephone  | Pets         |
|-------------------|-----------------------|-------------|------------|--------------|
| Jeff Black        | 1450 Oak Blvd.        | Monona      | 6085555387 | Lucky        |
| Jean Coleman      | 105 N. Lake St.       | Monona      | 6085552654 | Max Samantha |
| Betty Davis       | 638 Cardinal Ave.     | Sun Prairie | 6085511749 | Basil        |
| Harold Davis      | 563 Friendly St.      | Windsor     | 6085553198 | Iggy         |
| Maria Escobito    | 345 Maple St.         | Madison     | 608557683  | Mulligan     |
| Carlos Esteban    | 2335 Independence La. | Wauakee     | 6085555487 | Lucky Sly    |
| George Franklin   | 110 W. Liberty St.    | Madison     | 6085511023 | Leo          |
| Peter McTavish    | 2387 S. Fair Way      | Madison     | 6085552765 | George       |
| Eduardo Rodriguez | 2693 Commerce St.     | McFarland   | 6085558763 | Jewel Rosy   |
| David Schroeder   | 2749 Blackhawk Trail  | Madison     | 6085559435 | Freddy       |

At the bottom, there's a "spring" logo and the word "invest".

# WEBANWENDUNGEN

The image displays four distinct web applications arranged horizontally:

- Booking.com**: A travel booking website showing flight information for flight 108 from Osnabrück Hbf to Nils? at 12:35. It also shows a Genius Prämien offer.
- ICE Portal**: A real-time train status application for the ICE 109. It shows the train is at Gleis 2, moving at 108 km/h, and the next stop is in 6 minutes at Osnabrück Hbf.
- hw plus!**: An event management application showing an attendee list for an event on April 4, 2024. It includes a search bar, filters for 'Going' and 'Not Going', and a section for finding common interests.
- spring**: A pet adoption platform listing owners and their pets. The table includes columns for Name, Address, City, Telephone, and Pets. Owners listed include Jeff Black, Jean Coleman, Betty Davis, Harold Davis, Maria Escobito, Carlos Esteban, George Franklin, Peter McTavish, Eduardo Rodriguez, and David Schroeder.

# WEBANWENDUNGEN

The image shows a Mac desktop with five browser tabs open:

- Booking.com**: A flight search results page for the ICE 109 from Osnabrück Hbf to Basel SBB.
- ICE Portal**: A travel information page for the ICE 109, showing the next stop at Gleis 2 in 6 minutes.
- HIBERNATE**: A Jira project management interface for the "Hibernate ORM" project.
- sessionize**: A speaker dashboard for Nils Hartmann.
- spring**: A GitHub repository for a "graphql-training" project.

The "spring" tab displays the following table of owners:

| Name              | Address               | City        | Telephone  | Pets         |
|-------------------|-----------------------|-------------|------------|--------------|
| Jeff Black        | 1450 Oak Blvd.        | Monona      | 608555387  | Lucky        |
| Jean Coleman      | 105 N. Lake St.       | Monona      | 6085552654 | Max Samantha |
| Betty Davis       | 638 Cardinal Ave.     | Sun Prairie | 608551749  | Basil        |
| Harold Davis      | 563 Friendly St.      | Windsor     | 608553198  | Iggy         |
| Maria Escobito    | 345 Maple St.         | Madison     | 608557683  | Mulligan     |
| Carlos Esteban    | 2335 Independence La. | Waunakee    | 608555487  | Lucky Sly    |
| George Franklin   | 110 W. Liberty St.    | Madison     | 608551023  | Leo          |
| Peter McTavish    | 2387 S. Fair Way      | Madison     | 608552765  | George       |
| Eduardo Rodriguez | 2693 Commerce St.     | McFarland   | 6085558763 | Jewel Rosy   |
| David Schroeder   | 2749 Blackhawk Trail  | Madison     | 6085559435 | Freddy       |

# WEBANWENDUNGEN

The image displays a collage of several web application interfaces:

- Booking.com**: A travel booking website showing flight information for the ICE 109 from Osnabrück Hbf to Basel SBB.
- ICE Portal**: A real-time train status board showing the next stop at Gleis 2 in 6 minutes, speed (108 km/h), and a link to 'Schnelles Internet'.
- Hibernate**: A project management tool showing a list of tasks and projects related to 'Hibernate ORM'.
- Microsoft Teams**: A communication platform showing a chat history between users.
- Sessionize**: An event management tool showing a speaker dashboard for Nils Hartmann.
- GitHub**: A code repository for a 'spring-graphql-training' project, listing pull requests and issues.
- Eventbrite**: A screenshot of an event page for a 'Frontend for Backend: HTMX oder Single-Page-Anwendung?' event.

# WEBANWENDUNGEN

The image displays a collage of several web application screenshots, illustrating various types of web-based tools and services:

- Booking.com**: A travel booking website showing flight information for the ICE 109.
- ICE Portal**: A real-time train status monitor showing the next stop in 6 minutes at Gleis 2.
- ChatGPT - DALLE**: An AI collaboration interface where ChatGPT generates a cooking recipe for a "Classic Beef Burger" and DALLE creates two images of the burger.
- HIBERNATE**: A project management tool for the Hibernate ORM software project.
- sessionize**: A speaker dashboard for Nils Hartmann.
- spring**: A developer-oriented website showing a table of owners with columns for Name, Address, City, Telephone, and Pets.
- Microsoft Teams**: A communication and collaboration platform showing a chat history and a grid of active conversations.

# WEBANWENDUNGEN

# Die Webanwendung gibt es nicht

The image is a collage of several screenshots from various web applications, demonstrating different types of web-based tools and interfaces:

- Booking.com**: A travel booking website showing flight and hotel search results.
- ICE Portal**: A travel-related portal showing flight status (108 km/h) and a map.
- ChatGPT - DALL-E**: A screenshot of a conversation with AI models, showing two generated images of a "Classic Beef Burger".
- Hibernate ORM**: A screenshot of a project management tool showing a list of tasks and their details.
- Vorgänge**: A screenshot of a task management interface with a sidebar menu and a list of tasks.
- sessionize**: A screenshot of a speaker dashboard for a session, showing details for Nils Hartmann.
- spring**: A screenshot of a web application for managing pet owners, displaying a table of owners with columns for Name, Address, City, Telephone, and Pets.
- Calendar**: A screenshot of a calendar application showing events for June and July 2024.
- Diagram Editor**: A screenshot of a software for creating and editing diagrams or flowcharts.

# WEBANWENDUNGEN

Die Webanwendung gibt es nicht  
...aber: (fast) alle brauchen JavaScript

The screenshot shows a Mac OS X desktop with a windowed browser. The browser has four tabs open:

- ICE Portal (https://iceportal.de/Karte\_neu)
- Booking.com (https://www.booking.com/index.de.htm)
- ChatGPT - DALL-E (https://chat.openai.com/g/1234567890123456789012345678901234567890)
- hvg plus (https://www.hvgplus.de/)

The main content area of the browser displays a large green banner with the text "Die Webanwendung gibt es nicht" and a purple banner below it with the text "...aber: (fast) alle brauchen JavaScript".

The screenshot shows a desktop environment with several windows open:

- A central window titled "Projekte / Hibernate ORM" showing a list of "Vorgänge" (Tasks) for the "Hibernate ORM Softwareprojekt". One task is highlighted with a red progress bar.
- To the right, a "Public Profile" for "Nils Hartmann" is displayed, showing details like address, phone number, and pets.
- Below the profile, a GitHub repository for "springboard" is shown, listing issues and pull requests. One issue is highlighted with a yellow background.
- On the far right, a "Calendar" window is visible.
- At the bottom, a "File Explorer" window shows a directory structure with files like "Readme", "Activity", "6 stars", "4 watching", and "3 forks".

# WEBANWENDUNGEN

Die Webanwendung gibt es nicht

...aber: (fast) alle brauchen JavaScript

...die Frage ist nicht: ob, sondern wo und wieviel



# WEBANWENDUNGEN

Die Webanwendung gibt es nicht

...aber: (fast) alle brauchen JavaScript

...die Frage ist nicht: ob, sondern wo und wieviel

...und wer es schreibt (oder erzeugt)

## Ein Beispiel...

The screenshot shows a user profile for @nilshartmann with a picture of a man with long hair. Below the profile is a post card with the text "Heute lernen wir JavaScript!" and a smiley face icon. At the bottom of the card are icons for a clipboard, a list, and a globe, followed by the numbers "CW" and "EN". To the right of these is a large red arrow pointing down to the number "472", which is likely the character count. A blue button at the bottom right says "Publish!". The entire interface is set against a light green background.

Interaktiver Zeichenzähler

### Ein Beispiel...

The screenshot shows a travel booking interface. At the top right, there is a red vertical ellipsis button. Below it, the price "ab **61,40 €**" is displayed in large bold text. A large grey button below the price contains a circular arrow icon with a red dot, and a cursor arrow points towards it. At the bottom, a red arrow points right next to the text "Rückfahrt hinzufügen".

Ladeanimation

## Ein Beispiel...

The screenshot shows a web browser window for the JAX 2024 conference website (<https://jax.de/mainz/>). The page features a large blue header with the JAX logo and navigation links. Below the header, there's a yellow banner with the text "BIS KONFERENZBEGINN : ✓ Kollegenrabatt ✓ 5-Tages-Special". The main content area has a blue background with abstract white lines and the word "jax" in large white letters. At the bottom of the page, a cookie consent banner is displayed. It includes a heading "Privatsphäre-Einstellungen", a detailed text about data processing, two buttons ("Alle akzeptieren" and "Einstellungen anpassen"), and a link to the Datenschutzerklärung.

Wir setzen Cookies und Technologien auf unserer Website ein und verarbeiten technische Informationen und personenbezogene Daten (z.B. IP-Adresse), um Inhalte und Anzeigen zu personalisieren, Medien von Drittanbietern einzubinden oder Zugriffe auf unsere Website zu analysieren. Wir teilen diese Daten mit Dritten, die wir in den Privatsphäre-Einstellungen benennen. Dort kannst Du auch einzelne oder alle Cookies ablehnen.

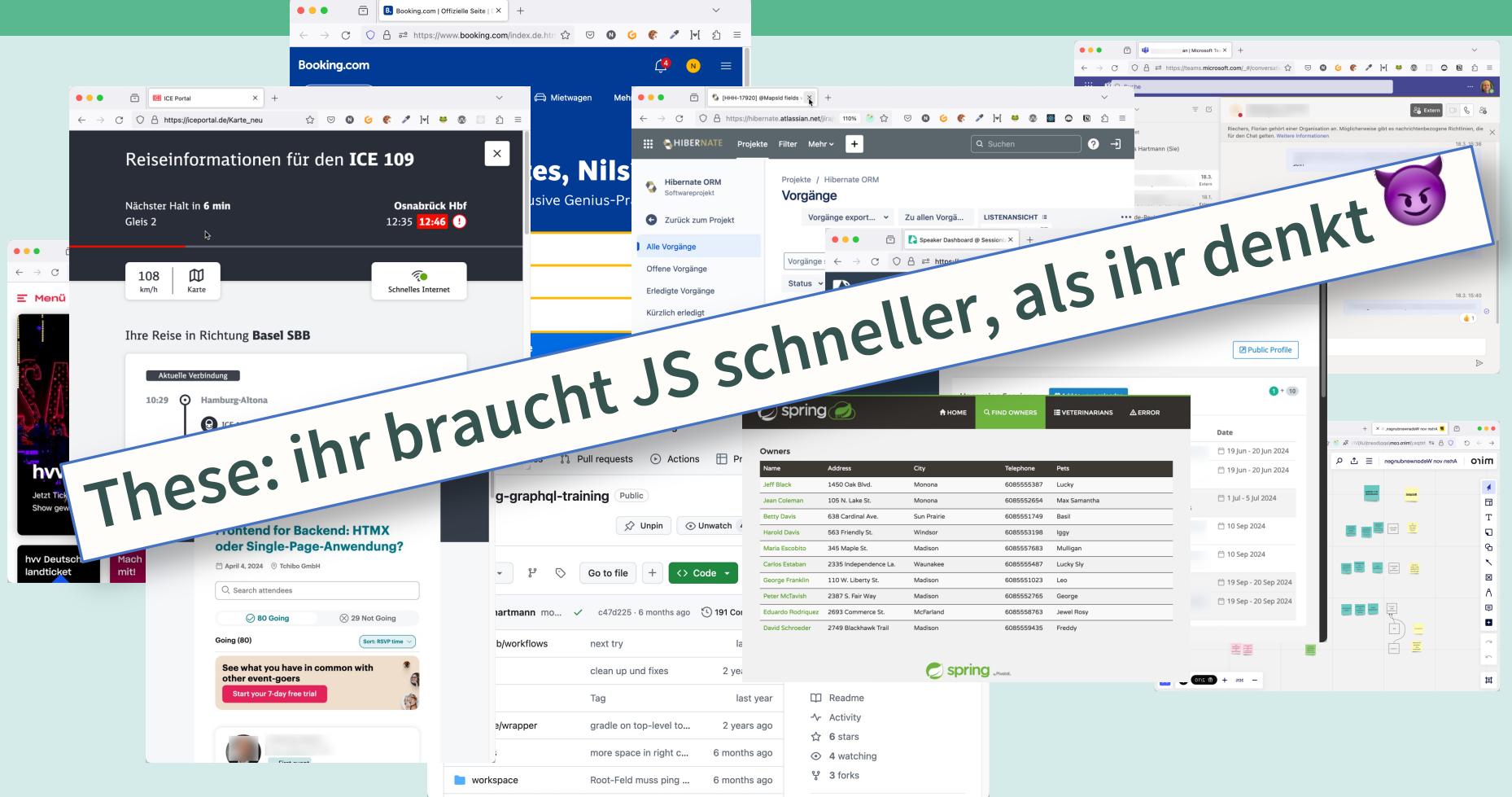
Mit Klick auf „Alle akzeptieren“ willst Du zugleich in die Übermittlung von Daten in Drittstaaten ein, die kein mit der EU vergleichbares Datenschutzniveau aufweisen. Sofern personenbezogene Daten dohrt übermittelt werden, besteht das Risiko, dass Behörden diese erfassen und analysieren ohne dass Du Deine Betroffenenrechte durchsetzen kannst. Unter „Einstellungen anpassen“ kannst Du einzelne oder alle optionalen Cookies ablehnen, wir übermitteln ggf. aber dennoch Daten in Drittstaaten. Wenn Du das völlig ausschließen willst, solltest Du diese Seite nicht nutzen.

Weitere Informationen zur Verwendung Deiner Daten findest du in unserer Datenschutzerklärung. Deine Einstellungen kannst Du dort jederzeit überprüfen und Deine Einwilligung mit Wirkung für die Zukunft widerrufen.

Datenschutzerklärung • Impressum • Deutsch

Cookie-Banner weglassen 😊

# WEBANWENDUNGEN



Es geht also "nur" darum, wie wir mit JavaScript umgehen:

Es geht also "nur" darum, wie wir mit JavaScript umgehen:

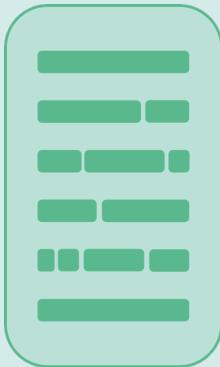
Welche Konsequenz hat JavaScript **zur Laufzeit**

Es geht also "nur" darum, wie wir mit JavaScript umgehen:

Welche Konsequenz hat JavaScript **zur Laufzeit**

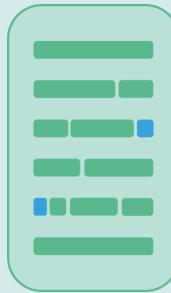
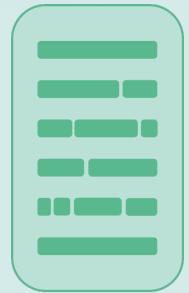
...und bereits während der **Entwicklung?**

**"Können wir nicht hier und da, ad-hoc JavaScript hinzufügen?"**



HTML-Seite

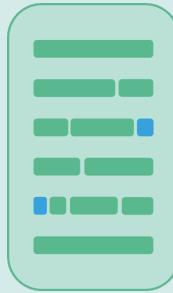
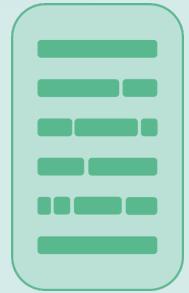
Ja, das geht!



HTML-Seite

- Typische Vertreter dieser Architektur zum Beispiel Spring WebMVC
- Templatesprache (z.B. Thymeleaf)
- JavaScript Schnipsel ("vanilla" oder zum Beispiel jQuery)

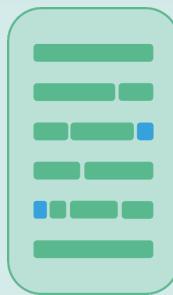
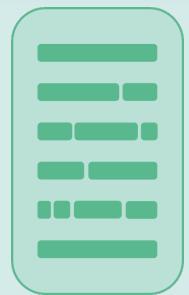
Ja, das geht!



HTML-Seite

- Eigentlich optimal:
  - wir haben **JavaScript** nur da, wo wir es **wirklich** brauchen, für Interaktivität
  - alles andere kann statisches HTML und CSS sein ❤️

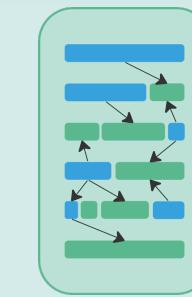
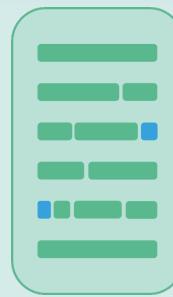
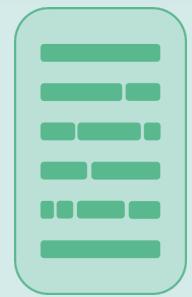
**"...hier und da müsste auch noch schnell was interaktives hin..."**



HTML-Seite

- Wir schreiben also noch etwas mehr JavaScript Schnipsel

**"...und hier... und hier ... und hier... und ..." au weia!**



HTML-Seite

## Das Problem von Schnipsel-Architektur



- Bunter Strauß an Server- und Client-Technologien (Backend-Sprache, Template-Sprache, JavaScript)
- Verantwortlichkeit willkürlich auf Frontend und Backend aufgeteilt

### Das Problem von Schnipsel-Architektur



- Das Problem "schleicht sich ein"
- Es gilt das "Gesetz des Umschlagens von Quantität in Qualität" (F. Engels):
- Plötzlich hat unser Code nicht "ein paar Probleme" sondern ist kaputt

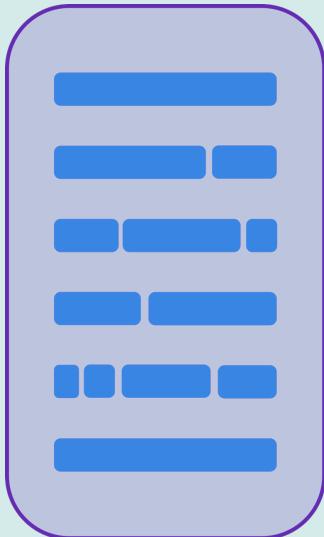
### *Dann besser alles in JavaScript? Single-Page-Anwendungen*



JavaScript-Code

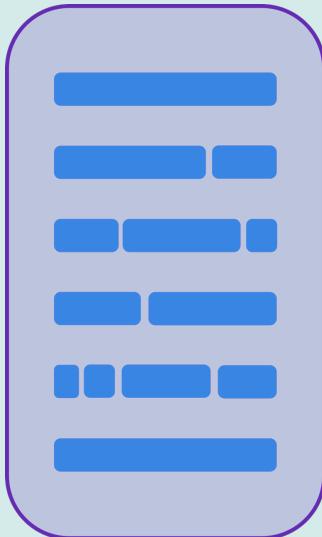
- ab ca. 2010
- Aus "Seiten" werden jetzt "Anwendungen"
- Klare Verantwortlichkeit: Server für Logik und Daten, Browser für UI
- Es gibt stabile und verbreitete Frameworks für jeden Geschmack

### Single-Page-Anwendungen



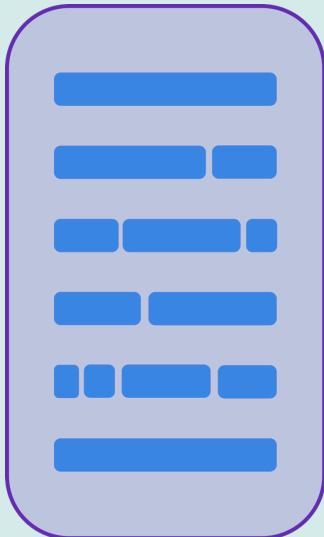
- Darstellung erfolgt vollständig mit JavaScript
- Statisches HTML spielt (fast) keine Rolle
- Die Anwendung kommuniziert mit dem Backend über API
- Ausgetauscht werden Daten, aber keine UI
- Vertreter: Angular, React, Svelte, Vue

### Single-Page-Anwendungen



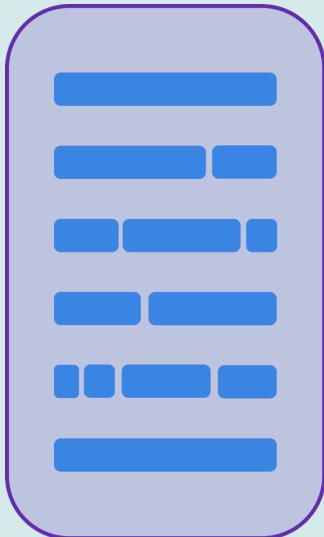
- Konsequenz #1: (viel) JavaScript zur Entwicklungszeit

### Single-Page-Anwendungen



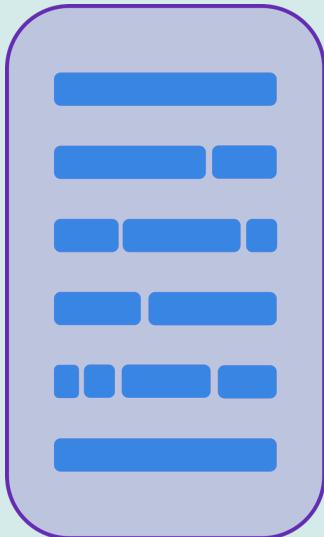
- Konsequenz #1: (viel) JavaScript zur Entwicklungszeit
- Konsequenz #2: (viel) JavaScript zur Laufzeit

### Single-Page-Anwendungen



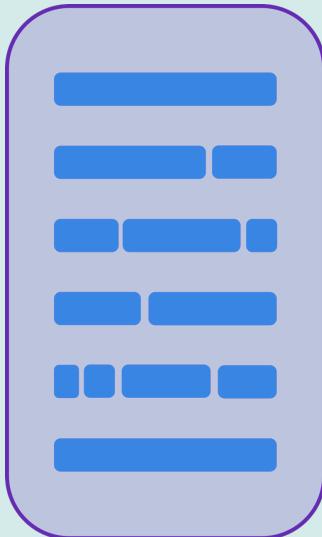
- Konsequenz #1: viel **JavaScript** zur **Entwicklungszeit**

### Single-Page-Anwendungen



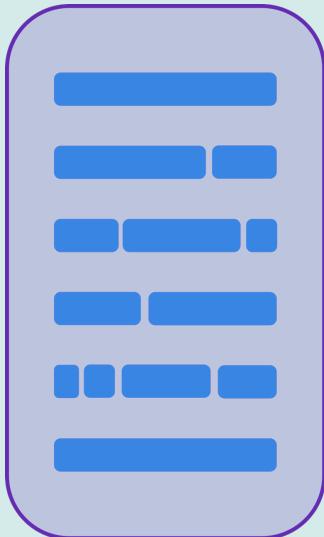
- Konsequenz #1: viel **JavaScript** zur **Entwicklungszeit**
- Es gibt modernes Tooling für die Entwicklung
  - Wie aus der Enterprise Backend-Entwicklung mit Java gewohnt

### Single-Page-Anwendungen



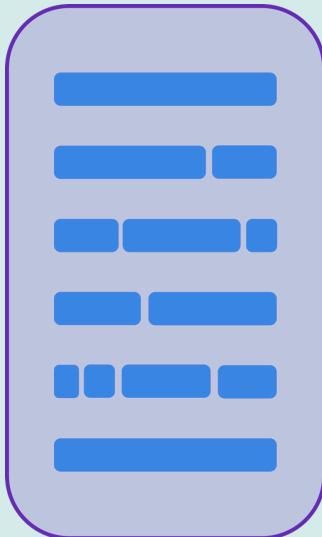
- Konsequenz #1: viel **JavaScript** zur **Entwicklungszeit**
- Es gibt modernes Tooling für die Entwicklung
  - Wie aus der Enterprise Backend-Entwicklung mit Java gewohnt
  - IDEs und Debugger, Typsicherheit, Unit- und Integrationstest, ...

### Single-Page-Anwendungen



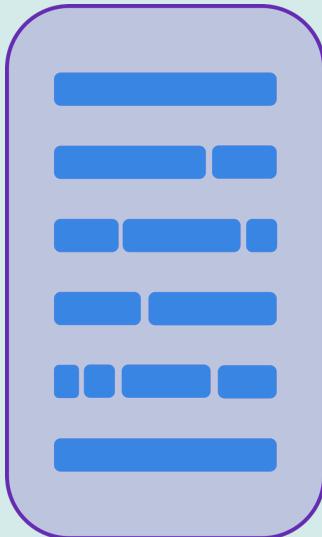
- Konsequenz #1: viel **JavaScript** zur **Entwicklungszeit**
- Es gibt modernes Tooling für die Entwicklung
  - Wie aus der Enterprise Backend-Entwicklung mit Java gewohnt
  - IDEs und Debugger, Typsicherheit, Unit- und Integrationstest, ...
- Aber: wenn man JavaScript doof findet, falscher Ansatz

### Single-Page-Anwendungen



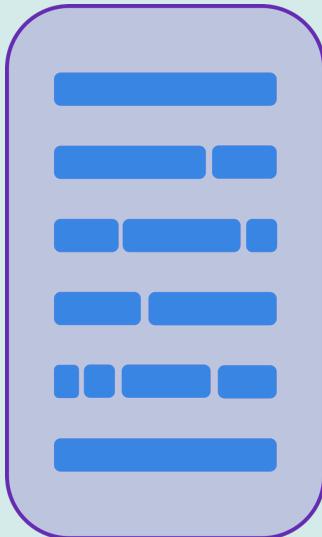
- Konsequenz #2: viel JavaScript **zur Laufzeit**
- Auch für **statische Inhalte**

### Single-Page-Anwendungen



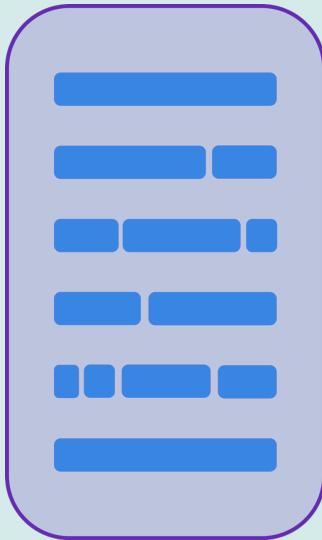
- JavaScript-Code:
  1. muss zum Browser gesendet werden
  2. muss vom Browser ausgeführt werden
  3. kann dann die darzustellenden Daten lesen
  4. kann dann erst die Daten anzeigen
  5. erst dann ist die Anwendung einsatzbereit
  6. Mit jedem Feature wird es mehr

### Single-Page-Anwendungen



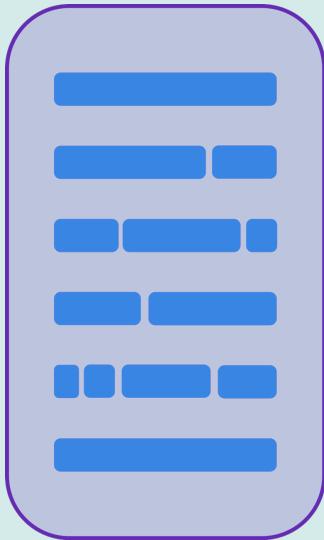
- Das hat Auswirkungen auf die **Laufzeit**-Performance
- Insbesondere beim Starten

### Single-Page-Anwendungen

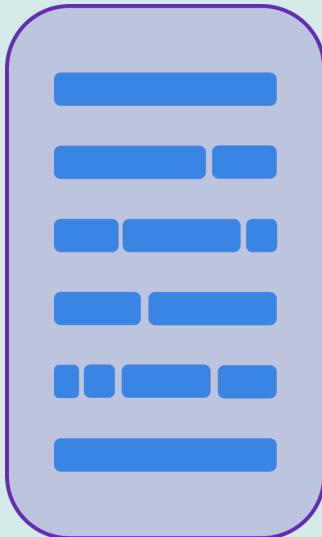


- Das hat Auswirkungen auf die **Laufzeit**-Performance
- Insbesondere beim Starten
- Ob das ein Problem für die eigene Anwendung ist, muss man von Fall zu Fall entscheiden
  - In-House- oder B2B-Anwendungen haben andere Anforderungen als ein Online-Shop

## Fullstack-Anwendungen (JavaScript)

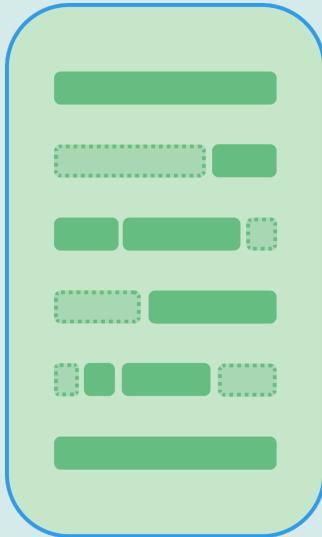


## Fullstack-Anwendungen (JavaScript)



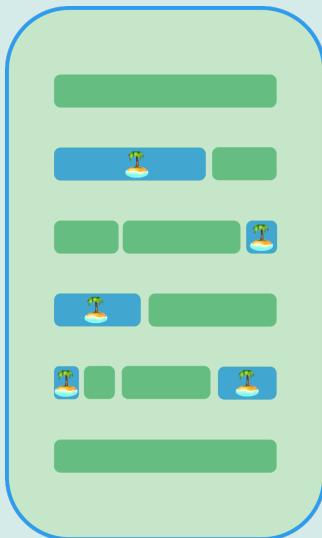
- Ebenfalls vollständig in **JavaScript** geschrieben

## Fullstack-Anwendungen (JavaScript)



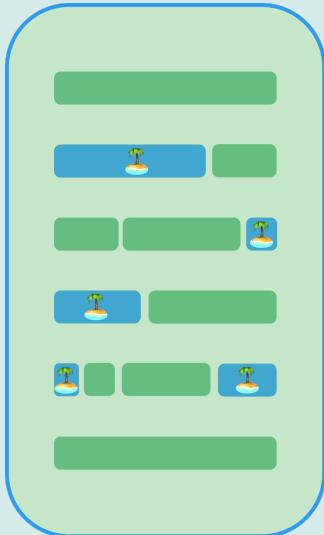
- Grundsätzliche Idee:
  1. **UI-Code** wird serverseitig vorgerendert
  2. **UI-Code** wird zum Browser gesendet und angezeigt

## Fullstack-Anwendungen (JavaScript)



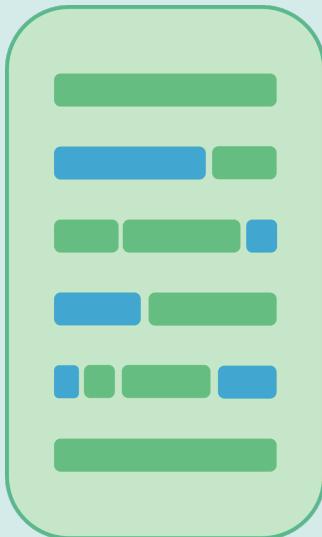
- Grundsätzliche Idee:
  1. **UI-Code** wird serverseitig vorgerendert
  2. **UI-Code** wird zum Browser gesendet und angezeigt
  3. Nur der JavaScript-Code ("Islands") **für Interaktionen** wird zum Browser geschickt

## Fullstack-Anwendungen (JavaScript)



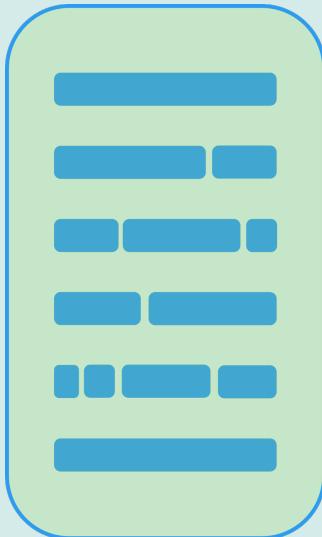
- Anwendung startet schneller:
  1. Browser bekommt **UI-Code** zur Darstellung
  2. Der **notwendige** JavaScript-Code wird nachgeladen
  3. Anwendung jetzt **interaktiv**

## Fullstack-Anwendungen (JavaScript)



- Wir sind zurück zur **JavaScript-Schnipsel-Architektur**
  - aber: die Schnipsel werden **automatisch** vom Framework erzeugt
  - die Schnipsel existieren nur zur **Laufzeit**

## Fullstack-Anwendungen (JavaScript)



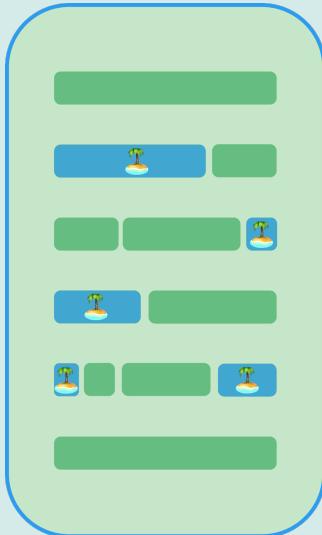
- Wir sind zurück zur **JavaScript-Schnipsel**-Architektur
  - aber: die Schnipsel werden **automatisch** vom Framework erzeugt
  - die Schnipsel existieren nur zur **Laufzeit**
  - In der **Entwicklung** ist "unser" Code aus "einem Guss"
  - aber weiterhin in **JavaScript**

## Fullstack-Anwendungen (JavaScript)

- Beispiel: Next.js
- Likes-Komponente ist interaktive Client-Komponente ("Island")

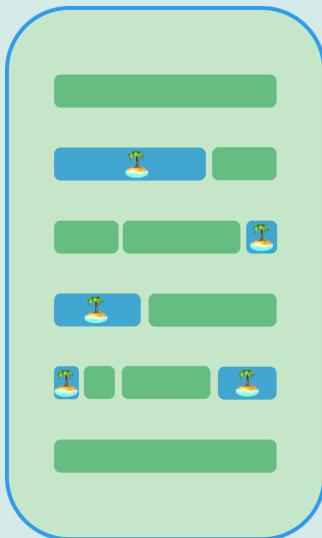
```
function RecipeCard({ recipe }: { recipe: RecipeDto }) {  
  return (  
    <section>  
      <H1>{recipe.title}</H1>  
      <RecipeCategories>  
        <Likes />  
        <CookingTime />  
      </RecipeCategories>  
    </section>  
  );  
}
```

## Fullstack-Anwendungen (JavaScript)



- Bekannte Vertreter:
  1. Next.js (React)
  2. SvelteKit (Svelte)
  3. Nuxt.js (Vue)
  4. Astro (eigenes Framework + Support für alle SPAs)
  5. Qwik (eigenes Framework)

## Fullstack-Anwendungen (JavaScript)



- Bekannte Vertreter:
  1. Next.js (React)
  2. SvelteKit (Svelte)
  3. Nuxt.js (Vue)
  4. Astro (eigenes Framework + Support für alle SPAs)
  5. Qwik (eigenes Framework)
- Funktionalität und Herangehensweise unterschiedlich

## HTMX

### introduction

htmx gives you access to [AJAX](#), [CSS Transitions](#), [WebSockets](#) and [Server Sent Events](#) directly in HTML, using [attributes](#), so you can build [modern user interfaces](#) with the [simplicity](#) and [power](#) of [hypertext](#)

htmx is small ([~14k min.gz'd](#)), [dependency-free](#), [extendable](#), IE11 compatible & has [reduced](#) code base sizes by [67%](#) when compared with react

<https://htmx.org/>

## HTMX - Grundlagen

- HTML-Elemente werden mit HTMX-Attributen ergänzt
- Damit wird beschrieben, welche Server Requests bei einem "Trigger" ausgeführt werden sollen
- HTMX kümmert sich um die Ausführung des Requests und die Verarbeitung der Antwort
- Der Server muss HTML-Schnipsel liefern ("Hypermedia")

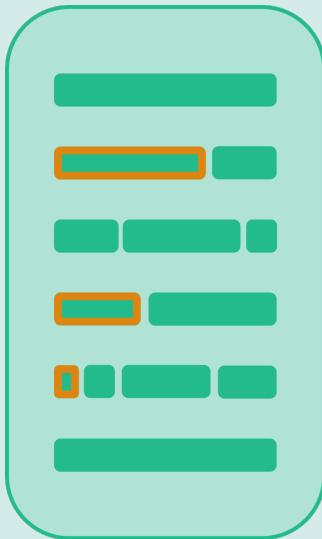
```
<html lang="en">
  <body>
    <div hx-get="/hello-world"
          hx-trigger="click"
          hx-target="#result">
      Get Greeting
    </div>

    <div id="result"></div>

    <script
      type="text/javascript"
      th:src="@{/htmx/htmx-1.9.10.min.js}"
    ></script>
  </body>
</html>
```

Demo: [localhost:8080/hello](http://localhost:8080/hello)

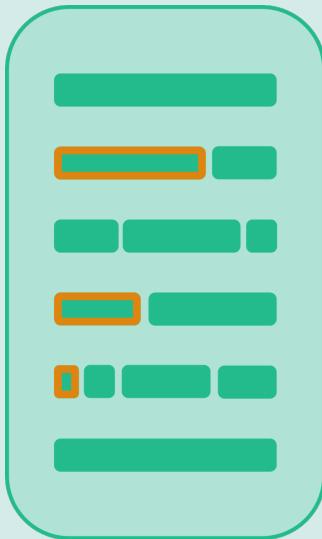
## HTMX



- Bei "Triggern" **Server Requests** für neue **UI**
- Man muss eine Art eigene DSL beherrschen

```
<input  
    type="search"  
    hx-get="/search"  
    hx-trigger="input[target.value.length > 2] changed delay:200ms"  
    hx-target="#searchResult"  
    hx-swap="outerHTML"  
    hx-indicator="#searchIndicator"  
/>
```

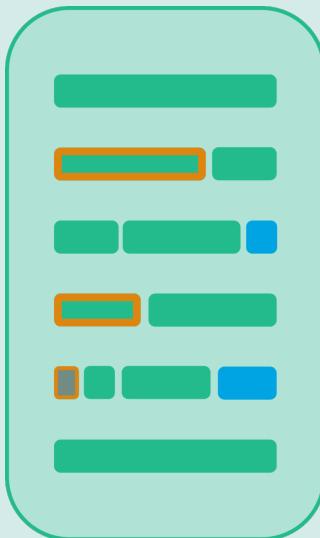
## HTMX



- Bei "Triggern" **Server Requests** für neue **UI**
- Man muss eine Art eigene DSL beherrschen
- ...und eine Template-Sprache im Backend

```
<a  
    th:href="@{/search(page=${nextPage},search=${search})}"  
    th:if="${hasMore == true}"  
    th:hx-get="@{/search(page=${nextPage})}"  
    th:classappend="|  
        ${!active && !disabled ? 'text-gray hover-underline' : ''}  
        ${active ? 'text-blue underline' : ''}  
        ${disabled ? 'text-gray' : ''}  
    |"  
    hx-target="#searchResult" hx-select="#searchResult > *"  
>Find more... </a>
```

## HTMX: Grenzen



- Für alles andere braucht man... JavaScript!

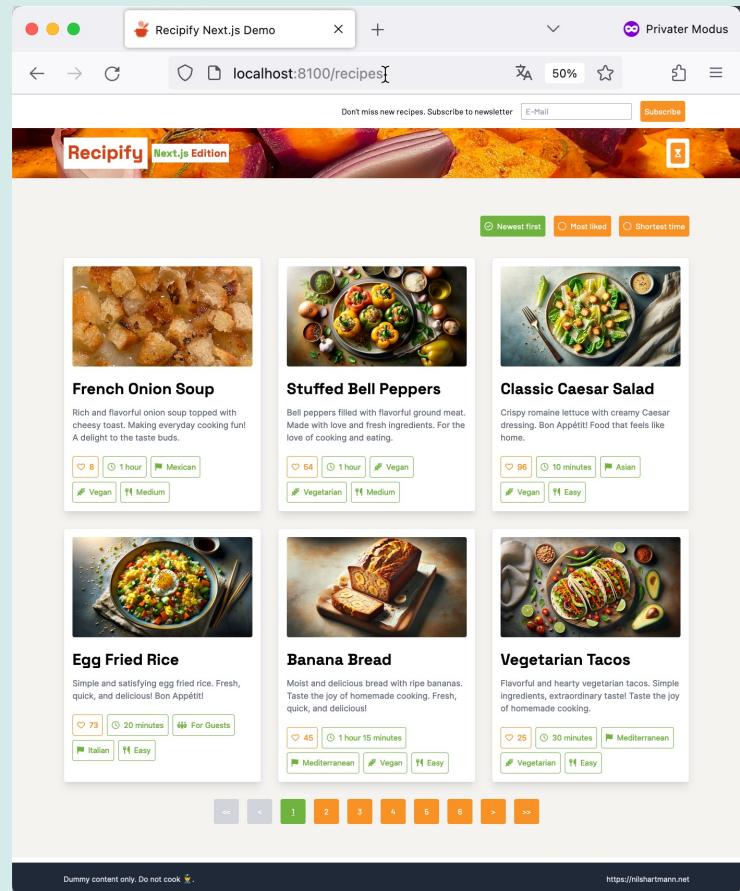
```
<input  
    type="search"  
    th:value="${search}"  
    th:hx-get="@{/search}"  
    hx-trigger="input[target.value.length > 2] changed delay:200ms"  
    hx-target="#searchResult"  
    hx-swap="outerHTML"  
    hx-indicator="#searchIndicator"  
    hx-on:input="handleInputChange(event)"  
/>  
  
<script>  
    function updateErrorMessage(e) {...}  
</script>
```

Die  
**Ansätze**  
unter der  
**Lupe**

EIN PAAR BEISPIELE...

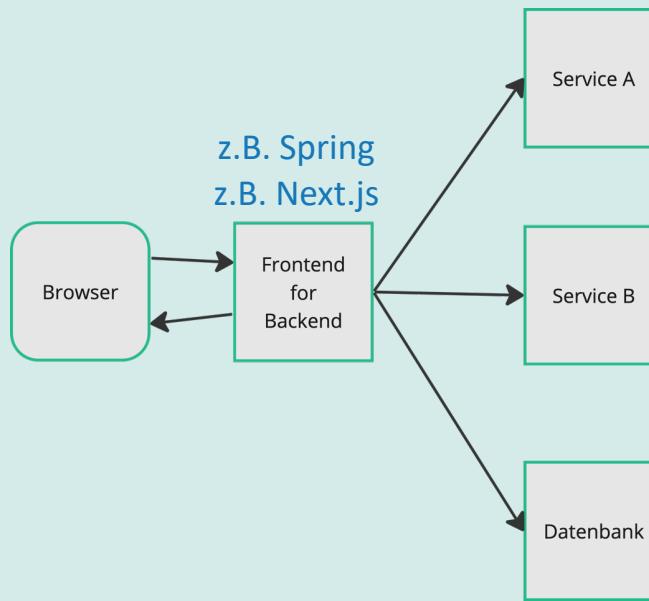
# BEISPIEL: INITIALER SEITENAUFRUF

- Beispiel: initialer Seitenaufruf
- Demo: localhost:8100



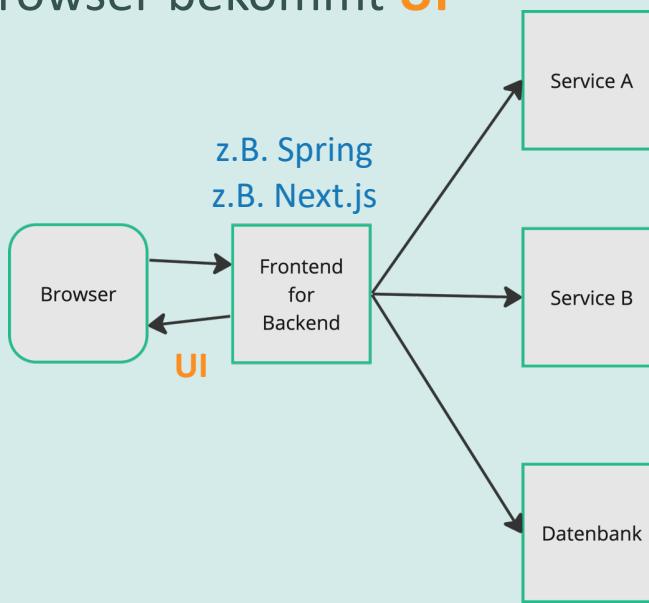
# ERMITTlung von DATEN

- Ermittlung von Daten
- Fullstack / HTMX: In-Process, DB, API, ...



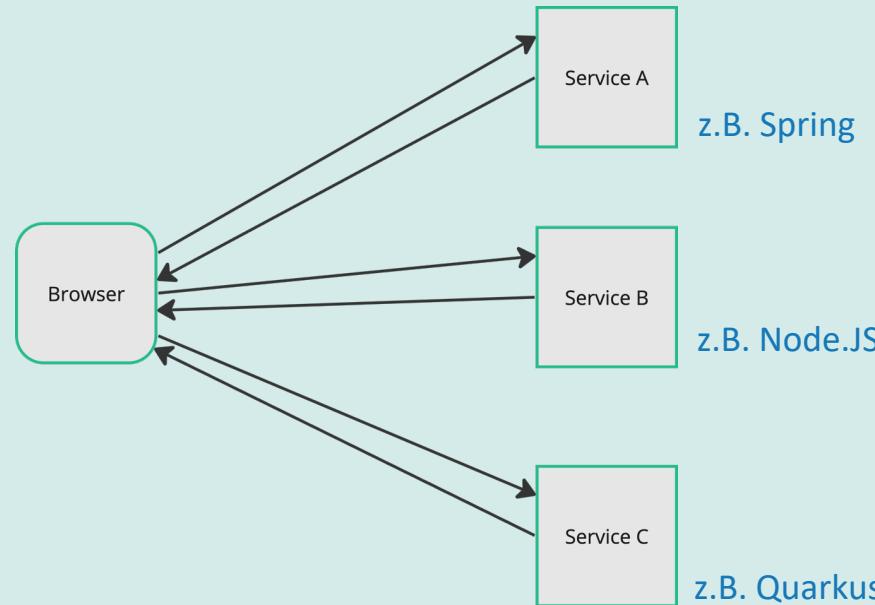
## BEISPIEL: INITIALER SEITENAUFRUF

- Ermittlung von Daten
- Fullstack / HTMX: In-Process, DB, API, ...
- Backend muss UI rendern (bei Fullstack ist das Implementierungsdetail)
- Browser bekommt UI



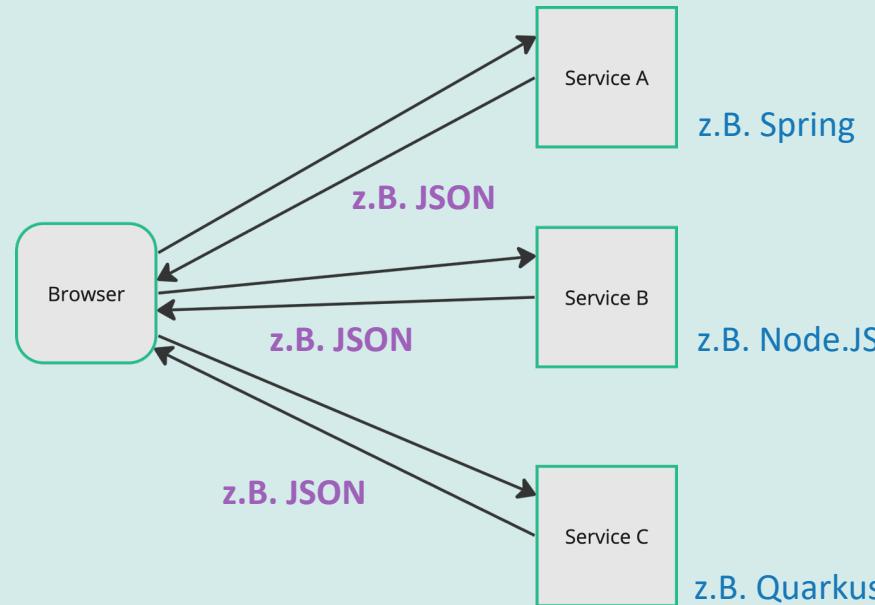
## BEISPIEL: INITIALER SEITENAUFRUF

- Ermittlung von Daten
- SPA: HTTP / REST / GraphQL API



## BEISPIEL: INITIALER SEITENAUFRUF

- Ermittlung von Daten
- SPA: HTTP / REST / GraphQL API
- Frontend-Anwendung bekommt **Daten**

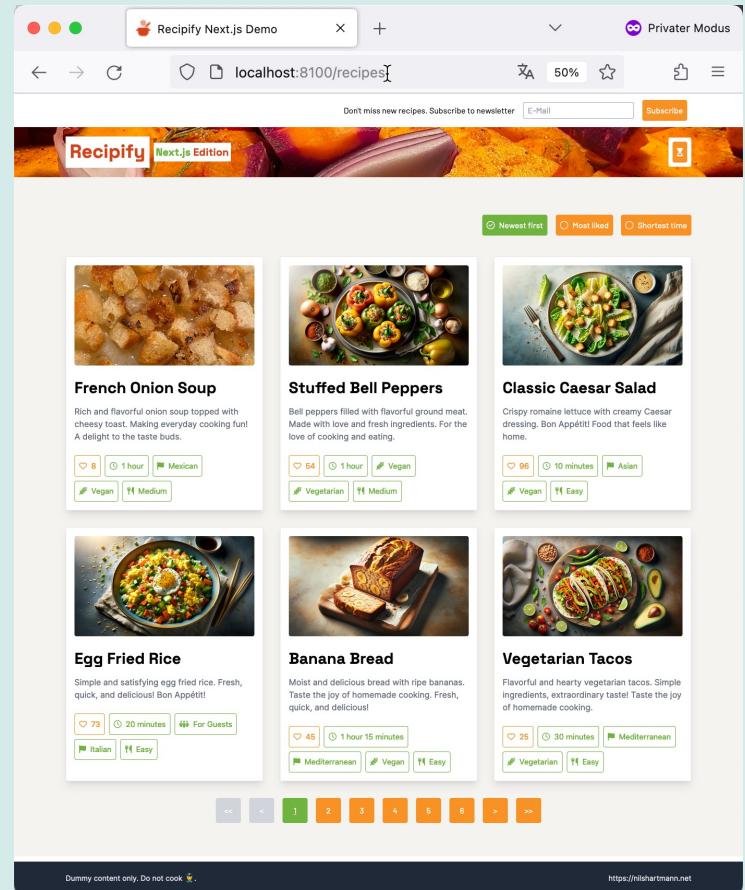


# BEISPIEL: INITIALER SEITENAUFRUF

## • Beispiel: initialer Seitenaufruf

### Code

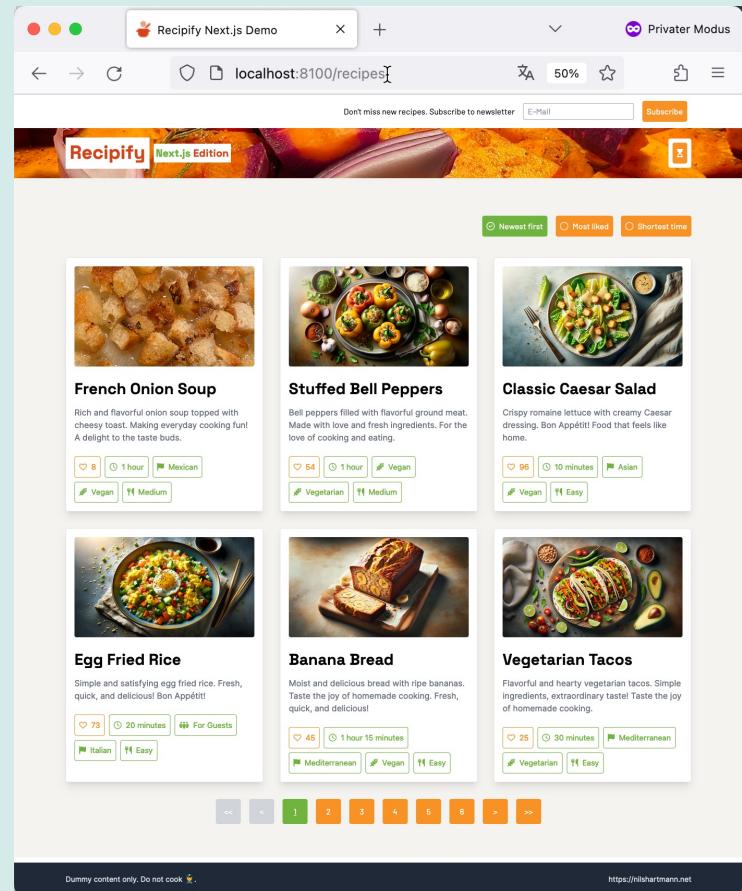
- SPA und Fullstack: JavaScript
- HTMX: Template-Sprache im Backend
  - Beispiel: Spring WebMVC + Thymeleaf



# BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren (Sortierung)

Demo: Next.js

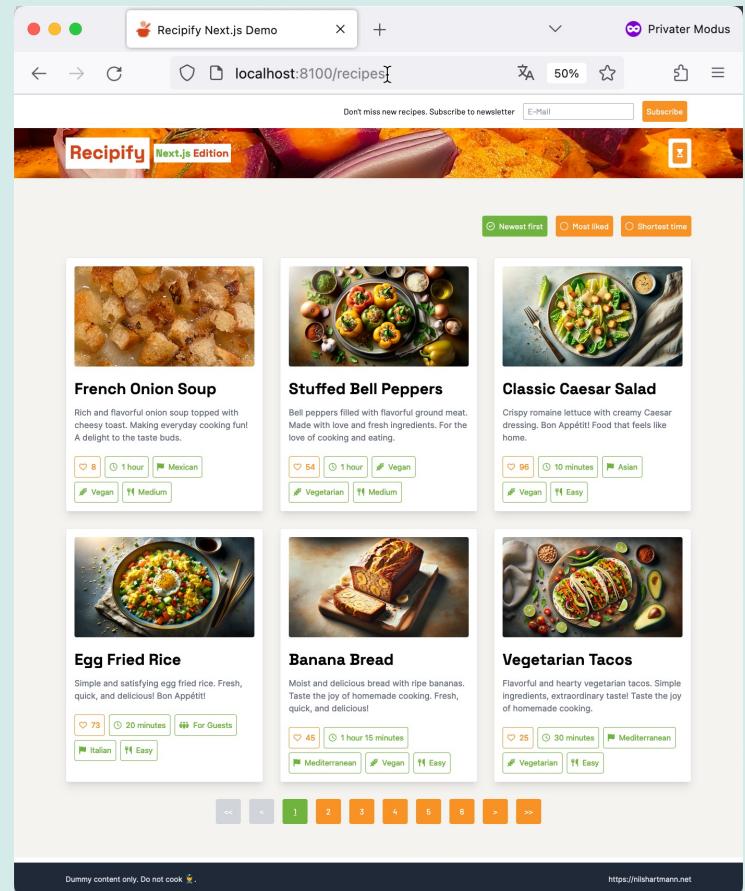


# BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren (Sortierung)

Zur Laufzeit

- SPA: Daten
- Fullstack: HTML / UI für Ausschnitt
- HTMX: HTML für Ausschnitt



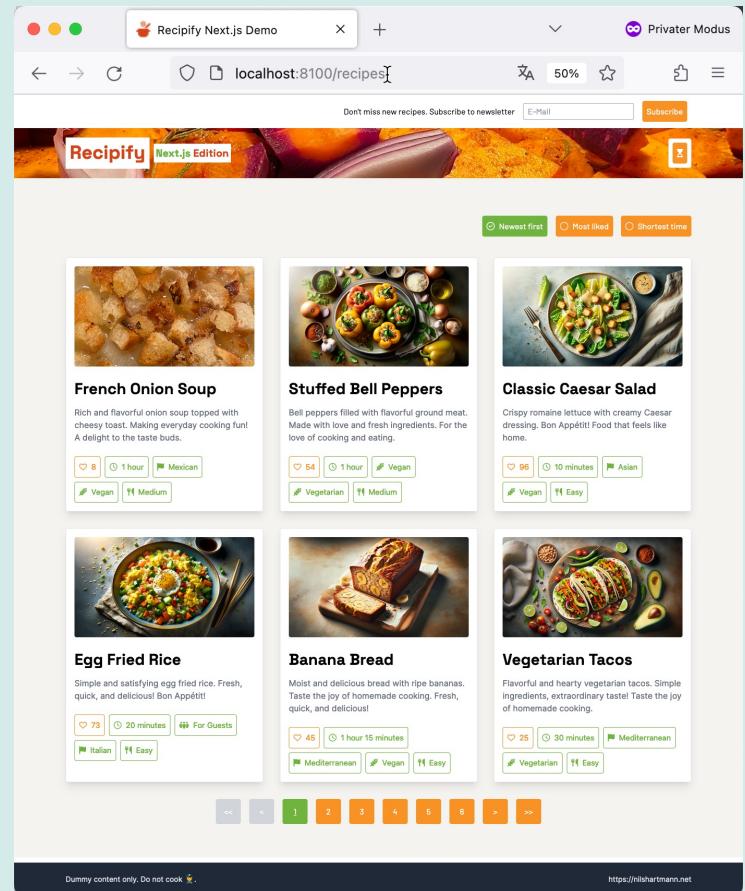
## BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren (Sortierung)

### HTMX

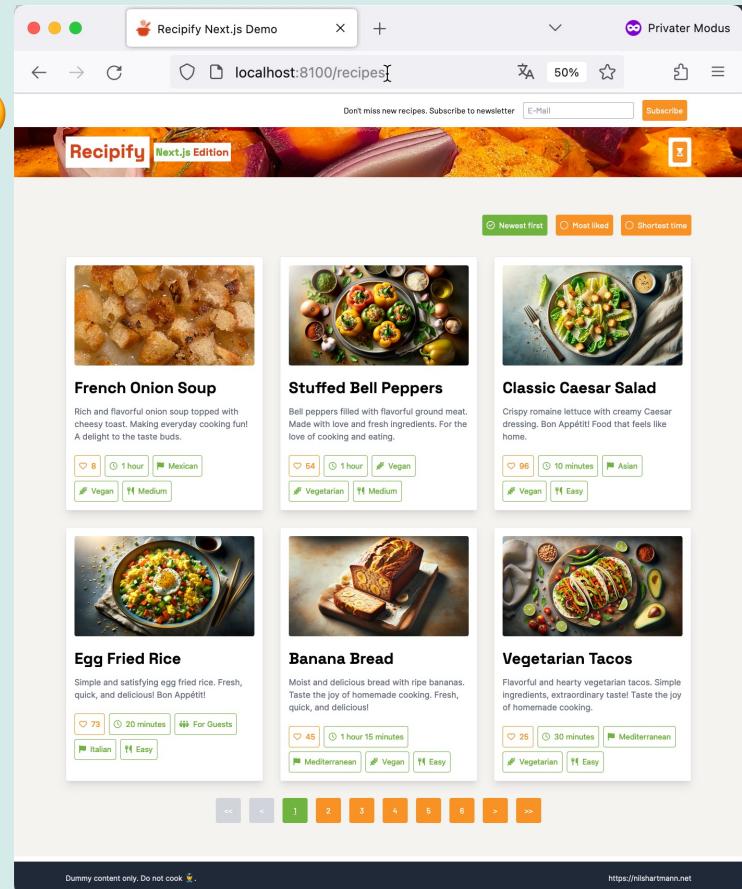
- Trigger auf (a)-Element
- Verhindert das "klassische" Navigieren
- Lädt neues HTML, ersetzt "main"

```
<a  
    th:hx-get="/recipes(orderBy=time)"  
    hx-swap="outerHTML"  
    hx-target="main"  
>  
    Time  
</a>
```



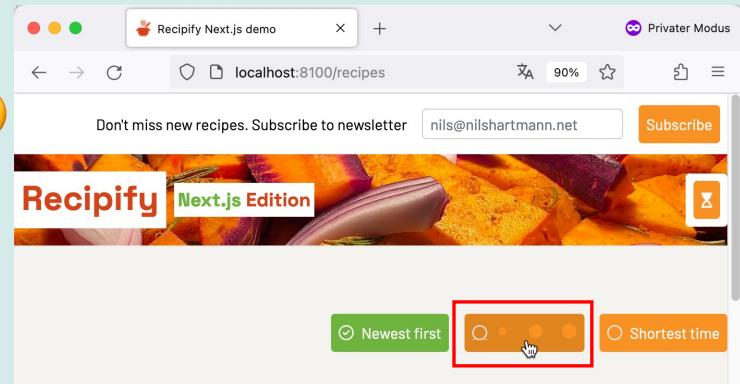
## BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren (Sortierung)
- Warum eigentlich kein Full-Page-Reload? 🤔
- Demo: React SPA Newsletter



## BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren
- Warum eigentlich kein Full-Page-Reload? 🤔
- Demo: NextJS Loading Indikator



## BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Loading Indikator
- Next.JS/React: Transition

```
export function OrderButton({ orderBy }) {  
  const [pending, startTransition] = useTransition();  
  const router = useRouter();  
  const href = "/recipes?orderBy=" + orderBy;  
  
  const handleClick = (e) => {  
    startTransition(() => {  
      router.push(href);  
    });  
  };  
  
  return (  
    pending ? <LoadingIndicator /> :  
    <Link href={href} onClick={handleClick}>  
      {orderBy}  
    </Link>  
  );  
}
```

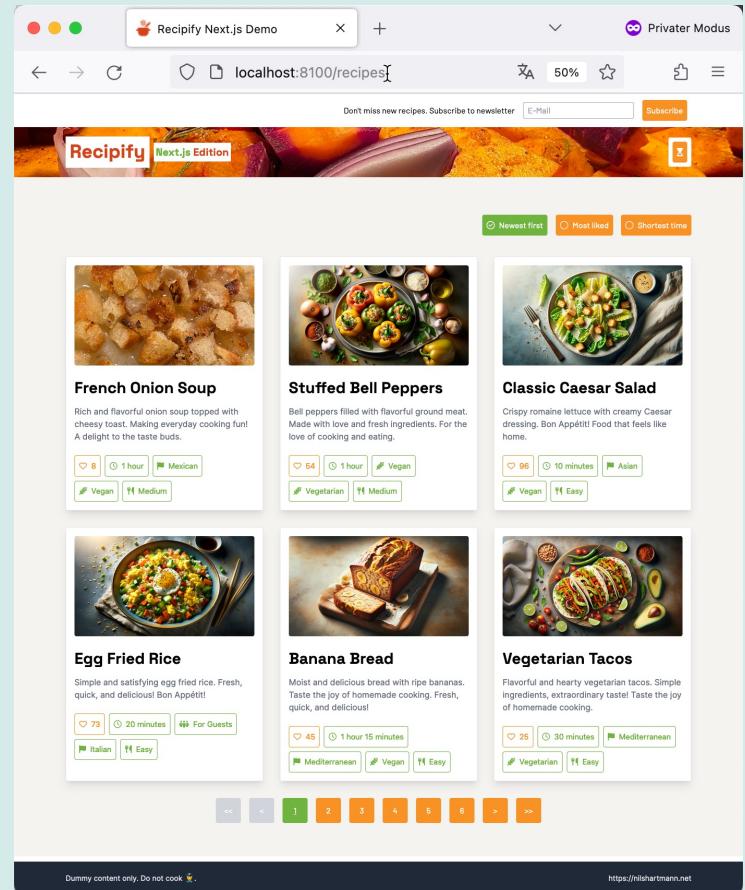
## BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Loading Indikator
- HTMX: CSS-Klasse wird während Request an markiertes Element gehängt
- Flexibilität?

```
<a  
    th:href="@{/recipes(orderBy=time)}"  
    hx-swap="outerHTML"  
    hx-target="#main"  
    hx-indicator="#searchIndicator"  
>  
    Time  
</a>  
  
<div id="searchIndicator" />
```

# BEISPIEL: VERLINKBARE SEITEN

- Beispiel: Verlinkbare Seiten
- Sortierung in der URL
- Demo: React SPA



## BEISPIEL: VERLINKBARE SEITEN

- **Beispiel: Verlinkbare Seiten**
- SPA (und Fullstack) Code arbeitet mit URL
- Identischer Code für initialen Seitenaufbau und für Neusortierung nach Klick

```
export default function RecipeListPage() {  
  const { orderBy } = recipeListRoute.useSearch();  
  const recipes = use GetAllRecipesQuery(0, orderBy);  
  
  return (  
    <main>  
      <RecipeListNavBar />  
      <RecipeList recipes={recipes} />  
    </main>  
  );  
}
```

## BEISPIEL: VERLINKBARE SEITEN

- Beispiel: Verlinkbare Seiten
- HTMX: Trigger definieren und URL umschreiben, um "SPA Verhalten" zu bekommen

```
<a  
    th:href="@{/recipes(orderBy=time)}"  
    th:hx-push-url="@{/recipes(orderBy=time)}"  
    hx-swap="outerHTML"  
    hx-target="main"  
>  
    Time  
</a>
```

## BEISPIEL: VERLINKBARE SEITEN

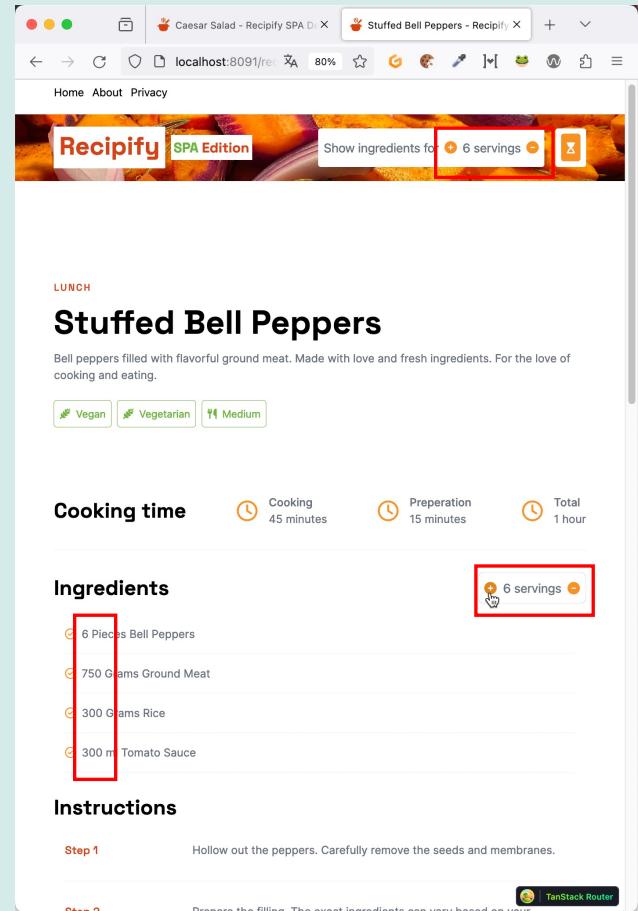
- Beispiel: Verlinkbare Seiten
- HTMX: **Zwei** Endpunkte erforderlich
  - Endpunkt **eins** für initiale, vollständige Seite (mit oder ohne Sortierung)

## BEISPIEL: VERLINKBARE SEITEN

- Beispiel: Verlinkbare Seiten
- HTMX: **Zwei** Endpunkte erforderlich
  - Endpunkt **eins** für initiale, vollständige Seite (mit oder ohne Sortierung)
  - Endpunkt **zwei** für den Trigger zum Sortieren auf dem Link (nur Austausch des main-Elementes)

# BEISPIEL: CLIENT-ZUSTAND

- Beispiel: Client-Zustand
- Demo: React SPA "Servings"

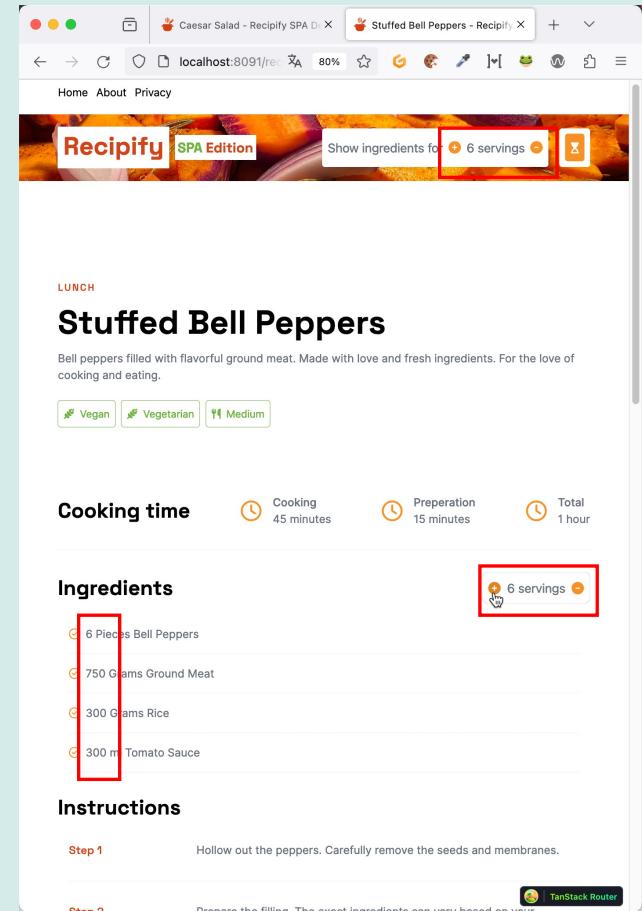


## BEISPIEL: CLIENT-ZUSTAND

- **Beispiel: Client-Zustand**

### Zur Laufzeit

- SPA: Nur neu rendern auf dem Client
- Fullstack: Nur Client und/oder Server
- HTMX: Nur Client und/oder Server



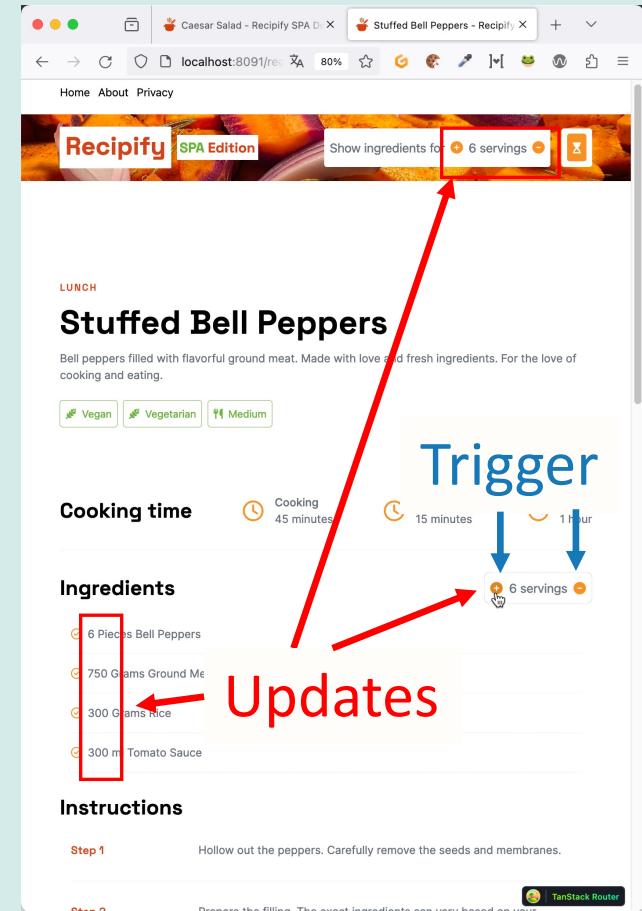
## BEISPIEL: CLIENT-ZUSTAND

- Klassisches Beispiel für globalen, nur Client-seitigen State
- Daten werden global im Client gehalten
- Jede Komponente darf die Daten ändern und lesen
  - vergleichbar mit UI-Service-Layer-Zugriff
- Komponenten können wiederverwendet werden

```
export default function ServingsWidget() {  
  const servingsStore = useServingsStore();  
  return (  
    <div>  
      <i  
        onClick={() => servingsStore.increaseServings()}  
      >  
        {servingsStore.servings} servings  
      <i  
        className={servingsStore.servings < 2 ? "Disabled": ""}  
        onClick={() => servingsStore.decreaseServings()}  
      >  
    </div>  
  );  
}
```

## BEISPIEL: CLIENT ZUSTAND

- HTMX Option 1:
- Entweder: Client-seitig per JavaScript
  - Ohne HTMX
  - JavaScript-Schnipsel einstreuen
  - Code-Probleme wie vor zehn Jahren



## BEISPIEL: CLIENT ZUSTAND

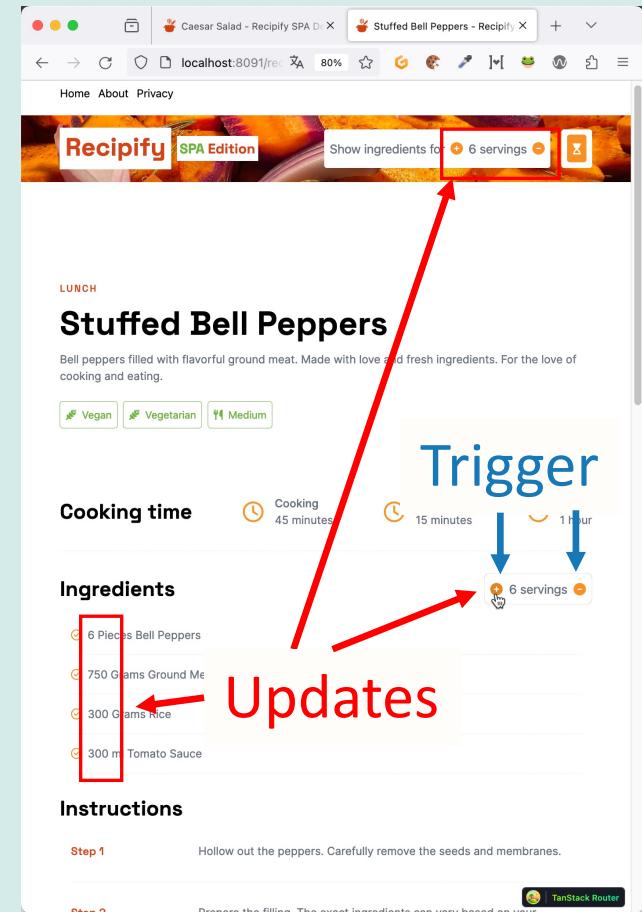
- HTMX Option 2: Serverseitig

- Klick auf +/-Button ist ein "Trigger"

- Lädt neue UI

- Problem:

- Aktualisierung an mehreren Stellen
- Das ginge mit "Out-of-Band"
- Sogar über "Seiten"-Wechsel hinaus
- Wo merken wir uns den Zähler?
- Wird schnell unübersichtlich



## BEISPIEL: DATEN ÄNDERUNGEN

- Beispiel: Daten-Änderungen
- Demo: Next.JS Like Button

The screenshot shows a cooking app interface with a header "Recipify | Next.js Edition" over a background image of various vegetables. Below are two recipe cards:

**Cheese Pizza**  
Simple and classic cheese pizza. This will become your new favorite! For people who love good food.

**Pancakes**  
Fluffy and light breakfast pancakes. Bring restaurant-style cooking at home! Save time without compromising on taste.

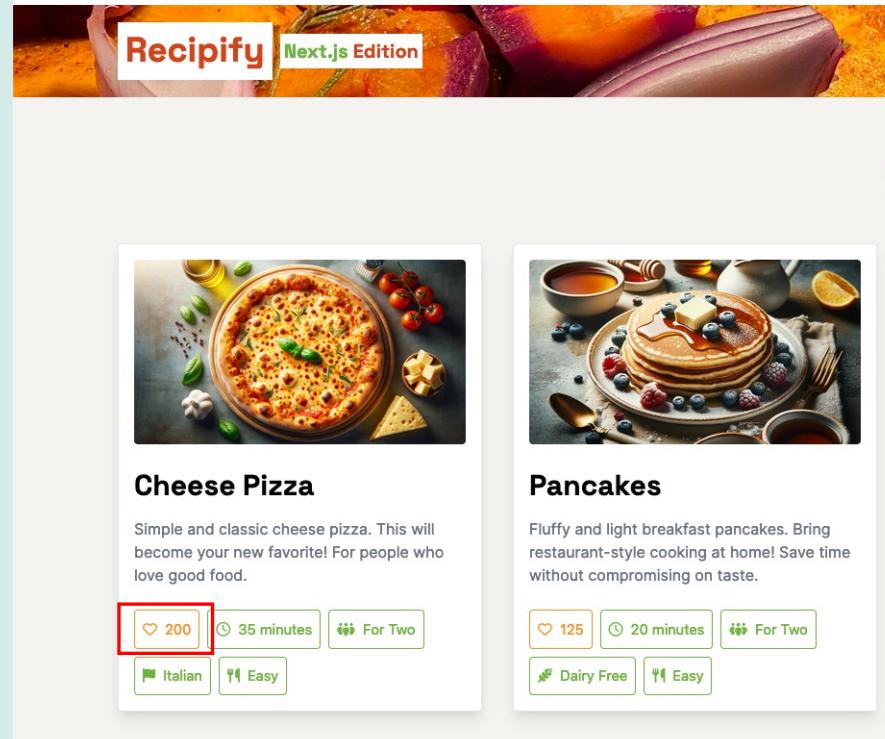
For the Cheese Pizza card, the "Like" button (heart icon with "200") is highlighted with a red border. Both cards include cooking time (35 minutes and 20 minutes), serving size (For Two and For Two), and category tags (Italian/Easy, Dairy Free/Easy).

## BEISPIEL: DATEN ÄNDERUNGEN

- Beispiel: Daten-Änderungen

### Zur Laufzeit

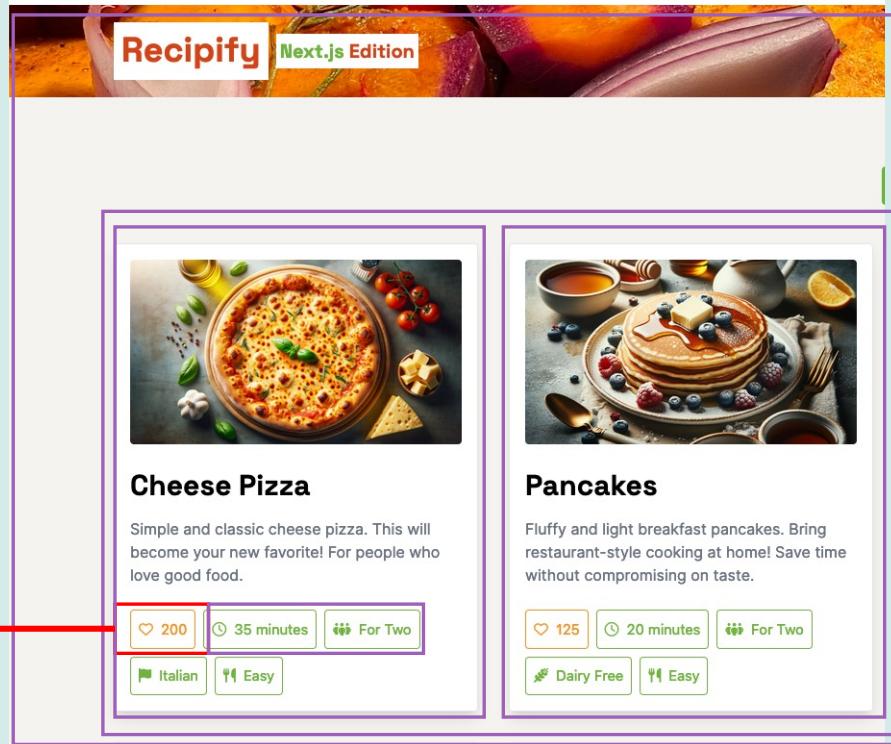
- SPA: HTTP Request, aktualisierter Like kommt zurück
- Fullstack: HTTP Request (ggf. proprietäres Protokoll, z.B. React Server Action)
- HTMX: Server Request (UI kommt zurück)



## BEISPIEL: DATEN ÄNDERUNGEN

- Beispiel: Daten-Änderungen
- Like-Button ist "Client-Komponente", da sie interaktiv ist (anklickbar)
- Kann aber transparent mit Server-Komponenten gemischt werden

Client Komponente



Server Komponenten

## BEISPIEL: DATEN ÄNDERUNGEN

- **Beispiel: Daten-Änderungen**
- Like-Button ist "Client-Komponente", da sie interaktiv ist (anklickbar)
- Kann aber transparent mit Server-Komponenten gemischt werden
- Nur der JS-Code der Client-Komponente kommt in den Browser!

Client Komponente      Server Komponenten

```
function RecipeCard({ recipe }: { recipe: RecipeDto }) {  
  return (  
    <section>  
      <H1>{recipe.title}</H1>  
      <RecipeCategories>  
        <Likes />  
        <CookingTime />  
      </RecipeCategories>  
    </section>  
  );  
}
```

# Auswahl- Kriterien

## AUSWAHLKRITERIEN

- Wie interaktiv ist meine Anwendung eigentlich?
  - Tipp: nicht unterschätzen!

## AUSWAHLKRITERIEN

- Habe ich mit (viel) JavaScript zur Laufzeit ein Problem?
  - Das ist nicht für jede Anwendung der Fall!

## AUSWAHLKRITERIEN

- Habe ich mit (viel) JavaScript in der Entwicklung ein Problem?
  - Dann eher kein JavaScript-basierter Ansatz

## AUSWAHLKRITERIEN

- Wie gefällt mir die Entwicklung

- Sprache
- Tooling, Entwicklungsumgebung etc.
- Bei HTMX beachten: wie sieht die Template-Sprache auch? Gefällt mir das?

## AUSWAHLKRITERIEN

- **Wie gefällt mir die Entwicklung**

- Sprache
- Tooling, Entwicklungsumgebung etc.
- Bei HTMX beachten: wie sieht die Template-Sprache auch? Gefällt mir das?

- **Ist Laufzeit nicht das wichtigere Kriterium?**

- Ja aber!
- Wenn wir kein Bock haben, die Anwendung zu bauen, wird das ja auch nichts
- Hier muss man vielleicht eine Balance finden

- **Komponentenmodell**

- Anwendungen sind komplex
- Mit Komponenten kann Komplexität aufgeteilt werden (wie im Backend)
- Unterstützt meine gewünschte Technologie das?
  - SPA und Fullstack (JavaScript): Ja
  - HTMX: kommt auf das Backend an

# Die Qual der Wahl

# Fazit

## FAZIT

- Laufzeit-Sicht: Fullstack + HTMX vorsichtig vergleichbar

## FAZIT

- **Entwicklung: Fullstack + SPA vorsichtig vergleichbar**
- HTMX fällt hier deutlich raus
- Insbesondere React-Ansätze (alles JavaScript, selbst "Templates") unterscheidet sich deutlich von HTMX (Backend-Sprache, Template-Sprache, JavaScript)

## FAZIT

- Meine persönliche Präferenz
- Wenn es geht, SPA machen: guter Kompromiss sowohl zur Laufzeit + Entwicklung

## FAZIT

- **Meine persönliche Präferenz**
- Wenn es geht, SPA machen: guter Kompromiss sowohl zur Laufzeit + Entwicklung
- Dann Fullstack, weil der Ansatz konsequente Weiterentwicklung ist

## FAZIT

- Meine persönliche Präferenz
- Wenn es geht, SPA machen: guter Kompromiss sowohl zur Laufzeit + Entwicklung
- Dann Fullstack, weil der Ansatz konsequente Weiterentwicklung ist

## FAZIT

- HTMX wirkt auf mich wie Technik von vor zehn Jahren
  - wir bauen aber heute ganz andere Anwendungen als damals:
    - ganz andere fachlicher Komplexität
    - viel höhere Ansprüche



Quelle: "Modern frontends with htmx", Wim Deblauwe, 2023 (Hervorhebungen von mir)

## FAZIT

*Wenn ihr nur eins aus diesem Vortrag mitnehmen wollt:*

*Wenn ihr nur eins aus diesem Vortrag mitnehmen wollt:*

**Frontend-Entwicklung bitte, bitte ernstnehmen!**

**Das ist nicht "weniger" als Backend-Entwicklung**

*(Wenn ihr kein Bock auf Frontend habt, dann lasst das doch einfach die Kolleg:innen machen 😊)*

# Frontend-Entwicklung ...



Frontend?

Vielleicht doch nicht so trivial

# Vielen Dank!

Slides: <https://react.schule/jax-2024>

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

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