

NILS HARTMANN

<https://nilshartmann.net>

Die Qual
der Wahl

Die passende

Frontend-
Technologie

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

Frontend-Entwicklung ...

Frontend-Entwicklung ...

NEXT.js



PREACT



Nuxt

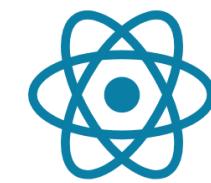
Astro

qwik

</> htmx



JS



React

Frontend-Entwicklung ...

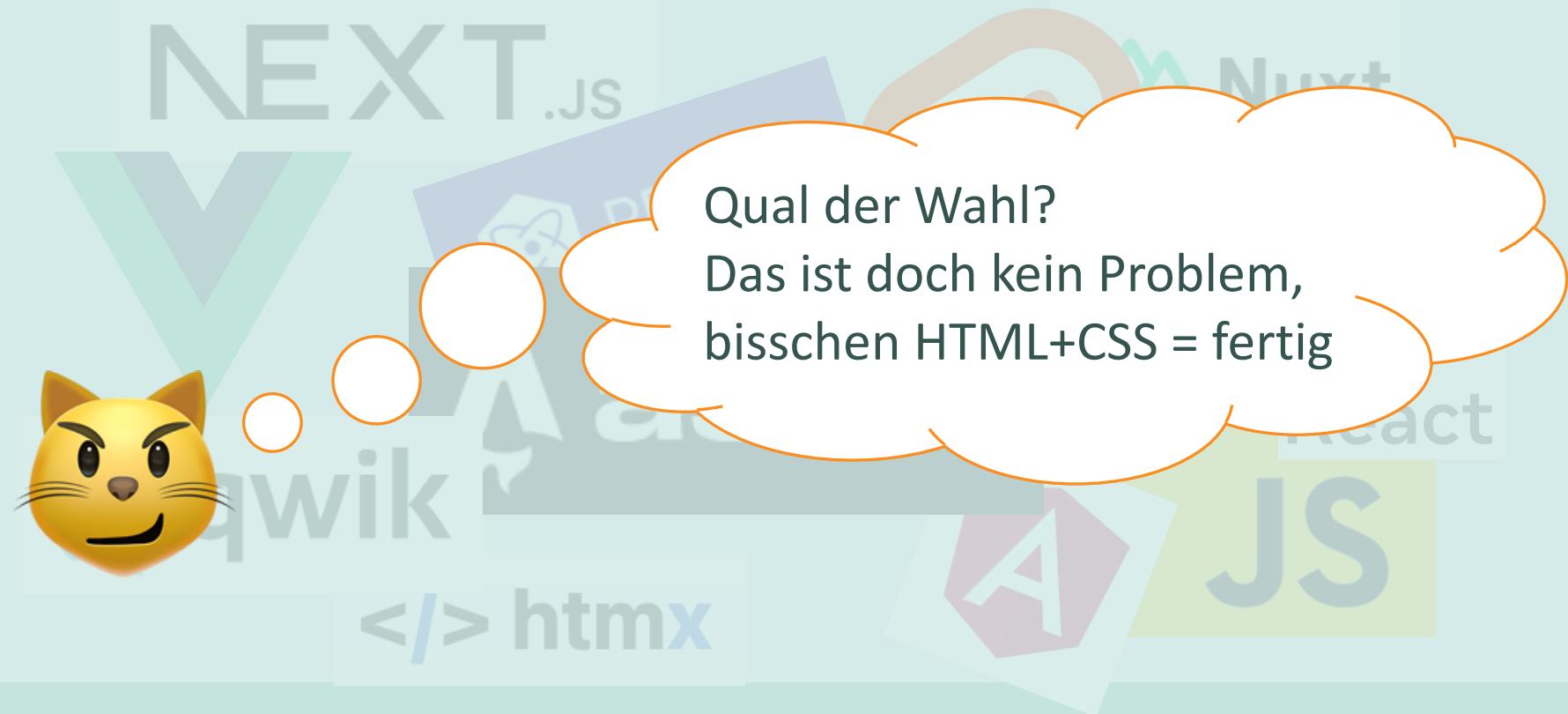


Was soll der ganze
JavaScript
Quatschkram?

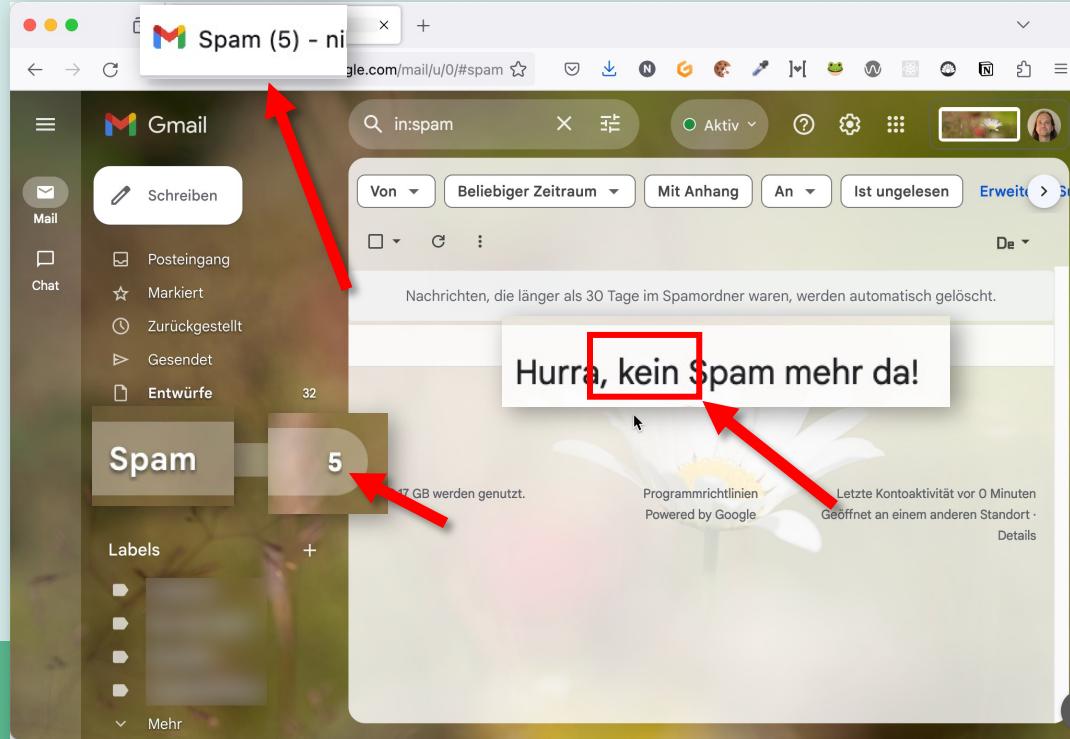
Frontend-Entwicklung ...



Qual der Wahl?
Das ist doch kein Problem,
bisschen HTML+CSS = fertig



Frontend-Entwicklung: Was soll schon schief gehen?



Frontend-Entwicklung: Was soll schon schief gehen?



The screenshot shows a code editor window for a file named `AppLink.tsx`. The code defines a component `AppLink` and an external link component `ExternalLink`, both using CSS-in-JS via `clsx`. The code editor has a sidebar titled "Highlight: All Problems" which shows "4 errors". Red arrows point from the sidebar to specific lines of code: one arrow points to the `type` parameter in the `AppLinkProps` type definition, another points to the `type` parameter in the `ExternalLinkProps` type definition, and two others point to the class names used in the `clsx` calls. The status bar at the bottom of the code editor says "No errors found by the IDE".

```
Project . . . datenschutzerklaerung . . . main . . . AppLink.tsx
datenschutzerklaerung
dienstleistungen
follow-me
graphql
impressum
kontakt
moderne-patttern-mit-read
page.tsx
talks-and-workshops
workshops
SectionTitle.tsx
Summary.tsx
assets
components
AppLink.tsx
Card.tsx
ContactCard.tsx
FaIcon.tsx
HomeButton.tsx
List.tsx
MarkdownContent.tsx
NavCard.tsx
PostList.tsx
Title.tsx
data
articles.ts
old-talks.ts
profile-links.ts
talks-and-workshops.ts
talks-data.ts
types.ts

export function AppLink({ type, ...props }: AppLinkProps) {
  const className = clsx("underline hover:font-bold", {
    "bg-highlight-light": type === "Highlight",
    "font-bold": type === "Bold",
  });
  // return <Link className={className} {...props} prefetch={false}></Link>;
  return <Link className={className} {...props} prefetch={false}></Link>;
}

type ExternalLinkProps = WithChildren<
  type?: "highlight" | "semihighlight";
  href: string;
>;
export function ExternalLink({ type, href, children }: ExternalLinkProps) {
  const className = clsx("underline hover:font-bold", {
    "bg-highlight-light": type === "Highlight",
    "decoration-2": type === "Semihighlight",
  });
  return (
    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
```

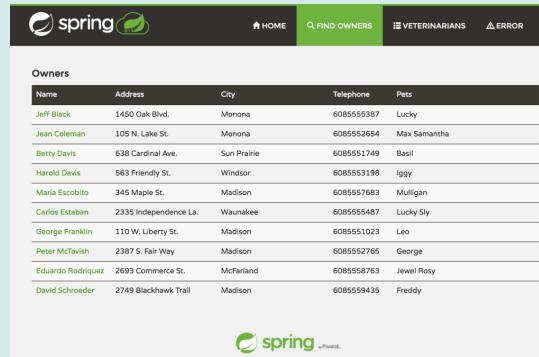
4 errors

Highlight: All Problems

No errors found by the IDE.



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 13 rows of data.

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.	Waunakee	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

WEBANWENDUNGEN

The ICE Portal website displays real-time train information for the ICE 109. Key details include:

- Nächster Halt in 6 min (Next stop in 6 minutes)
- Gleis 2 (Platform 2)
- Osnabrück Hbf (Arrival time: 12:35, Departure time: 12:46)
- 108 km/h (Current speed)
- Karte (Map) button
- Schnelles Internet (Fast Internet) button
- Aktuelle Verbindung (Current connection) section showing Hamburg-Altona to ICE 109 nach Basel SBB.
- Buttons for Ab (Departure), An (Arrival), Suchen (Search), Erweiterte Suche (Advanced Search), and Aktuelle Meldungen (Current News).
- Advertisement for hw plus: Jetzt Tickets für die Sasha-Show gewinnen!
- Logos for hhv Deutschlandticket, Mach mit!, and Schüler*innen.

The Microsoft Teams Chat interface shows a list of messages from various users, including Nils Hartmann, MS, and others, with timestamps ranging from 18.3. 10:38 to 18.3. 10:40.

The Spring owners database table lists the following data:

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	6085551749	Basil
Harold Davis	563 Friendly St.	Windsor	6085553198	Iggy
Maria Escobito	345 Maple St.	Madison	6085557683	Mulligan
Carlos Esteban	2335 Independence La.	Waunakee	6085555487	Lucky Sly
George Franklin	110 W. Liberty St.	Madison	6085551023	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

WEBANWENDUNGEN

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

- Booking.com**: A travel booking website showing flight information for the ICE 109 from Osnabrück Hbf to Gleis 2 at 12:46.
- ICE Portal**: A real-time train status board showing the next stop in 6 minutes, speed (108 km/h), and a link to a map.
- Hibernate**: A project management tool showing a list of tasks and a sidebar for navigating projects and filters.
- Microsoft Teams**: A communication platform displaying a chat history between users.
- GitHub**: A code repository showing a pull request titled "@MapId fields without @JoinColumn gets prefixed automatically" by user HHH-17920.
- Custom Event App**: A mobile-style app for managing event attendees, showing a list of people going or not going, and a feature to see what you have in common with others.

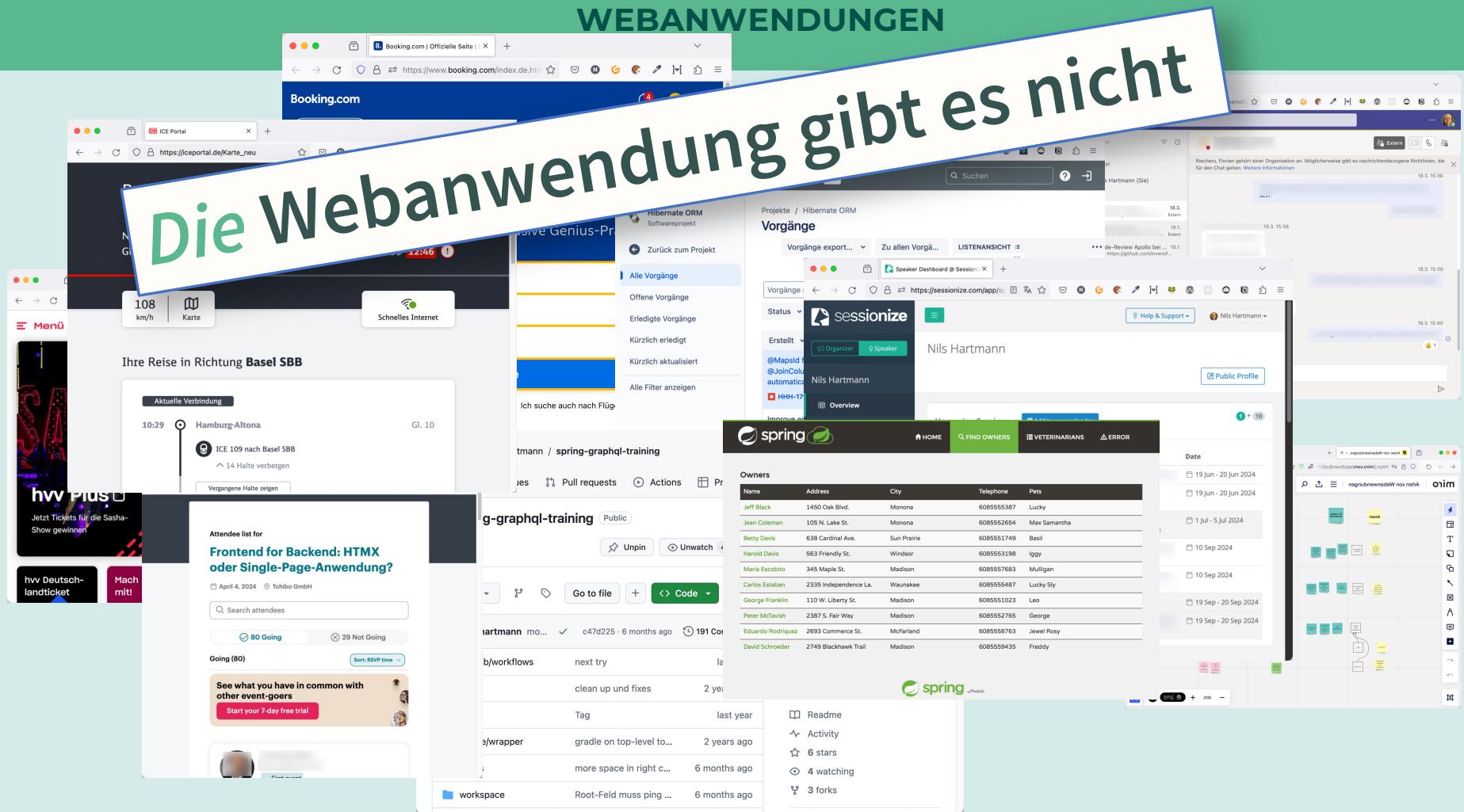
WEBANWENDUNGEN

The image displays a Mac desktop environment with several browser windows open, illustrating a multitasking session across different applications:

- Top Left:** A window for **Booking.com** showing travel information for the ICE 109 train.
- Middle Left:** A window for **ICE Portal** displaying real-time travel information for the ICE 109 train from Hamburg-Altona to Osnabrück Hbf.
- Center:** A window for **HIBERNATE** showing project management and task history.
- Bottom Center:** A GitHub repository for **g-graphql-training** with code snippets and commit history.
- Right Side:** A Microsoft Teams window showing a conversation and a calendar view.
- Bottom Right:** A window for **sessionize** showing a speaker profile for Nils Hartmann.
- Bottom Left:** A window for **spring** showing a list of owners with their details.

WEBANWENDUNGEN

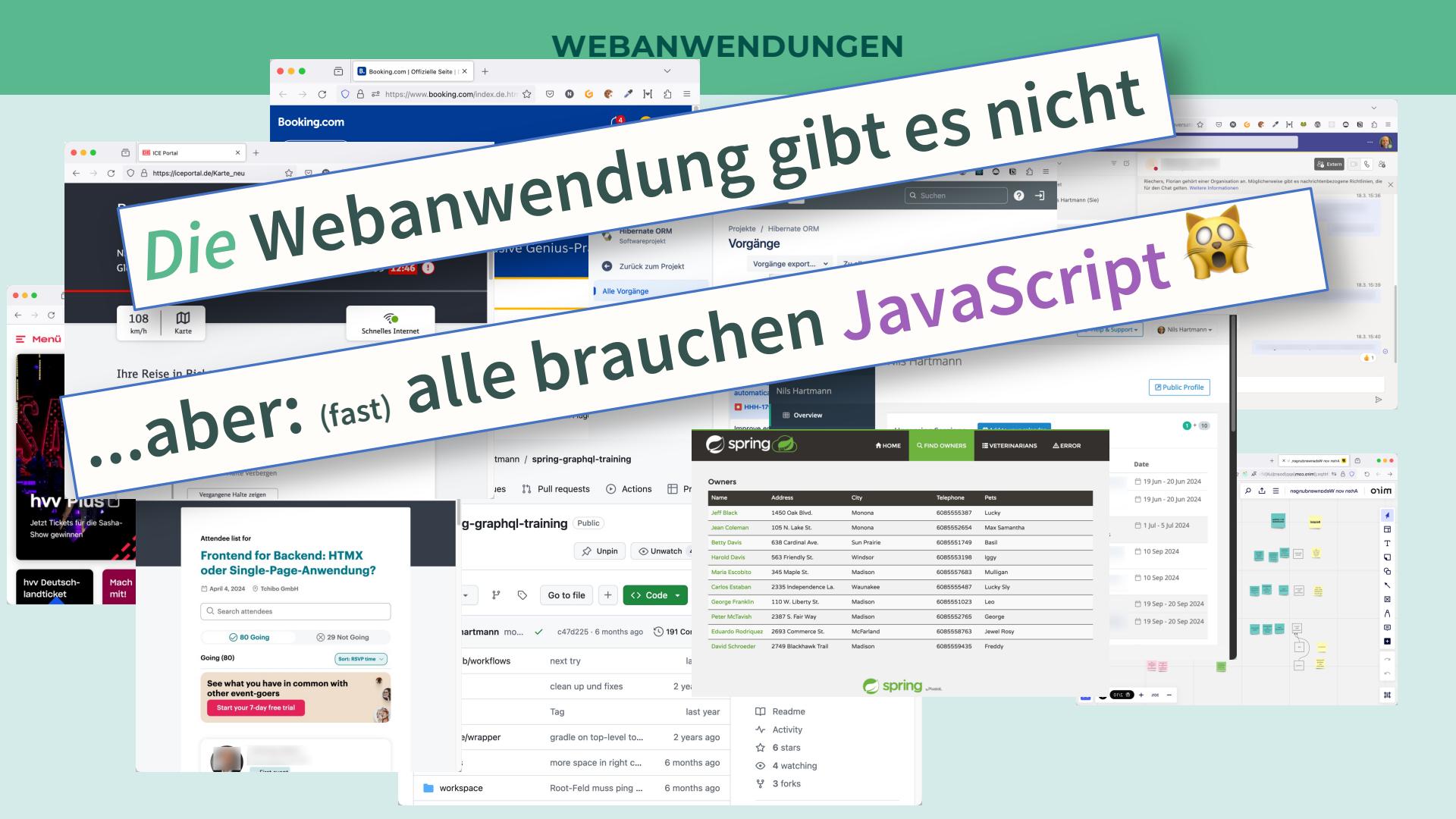
Die Webanwendung gibt es nicht



WEBANWENDUNGEN

Die Webanwendung gibt es nicht

...aber: (fast) alle brauchen JavaScript



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 gray 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. Underneath the price is a light gray rectangular button containing a circular arrow icon with a red border and a white arrow pointing clockwise. A black cursor arrow points to the center of this button. Below the button, a red arrow points to the right, followed by the text "Rückfahrt hinzufügen".

Ladeanimation

WEBANWENDUNGEN

Ein Beispiel...

Verbindung Abfahrten

Wir bringen dich hin

Dein Start *
St.Pauli

Dein Ziel *
Stellingen (Arenen)

Bitte gib ein gültiges
Datum ein

Di, 12.04.2024 ▼ < 10:43 Uhr >

Ab | An Suchen

Erweiterte Suche Aktuelle Meldungen

A screenshot of a travel search interface. It features two main input fields: 'Dein Start' (St.Pauli) and 'Dein Ziel' (Stellingen (Arenen)). Below these is a note about entering valid dates. A date picker shows 'Di, 12.04.2024'. At the bottom, there's a 'Suchen' (Search) button and links for 'Erweiterte Suche' (Advanced Search) and 'Aktuelle Meldungen' (Current News). A prominent blue button labeled 'Start und Ziel tauschen' (Swap Start and Destination) is positioned between the two main input fields.

Reisende, Fahrräder, BahnCards X

1 Person (27-64 Jahre) BahnCard 25, 1. Klasse

+ Reisende, Fahrräder, Hunde hinzufügen

Erklärung der Reisendentypen ▾

1 Reisender insgesamt Zurücksetzen Übernehmen

A screenshot of a passenger selection dialog. It shows '1 Reisender insgesamt'. There are dropdown menus for 'Person (27-64 Jahre)' and 'BahnCard 25, 1. Klasse'. A large red '+' icon with a cursor over it is centered above a link 'Reisende, Fahrräder, Hunde hinzufügen'. Below the '+' icon is a link 'Erklärung der Reisendentypen'. At the bottom right are 'Zurücksetzen' and 'Übernehmen' buttons.

Interaktive Formulare

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 section titled "Privatsphäre-Einstellungen" with a detailed explanation of data processing, a "Alle akzeptieren" button, and a "Einstellungen anpassen" link. There's also a note about third-party cookies and a link to the privacy policy.

Privatsphäre-Einstellungen

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.

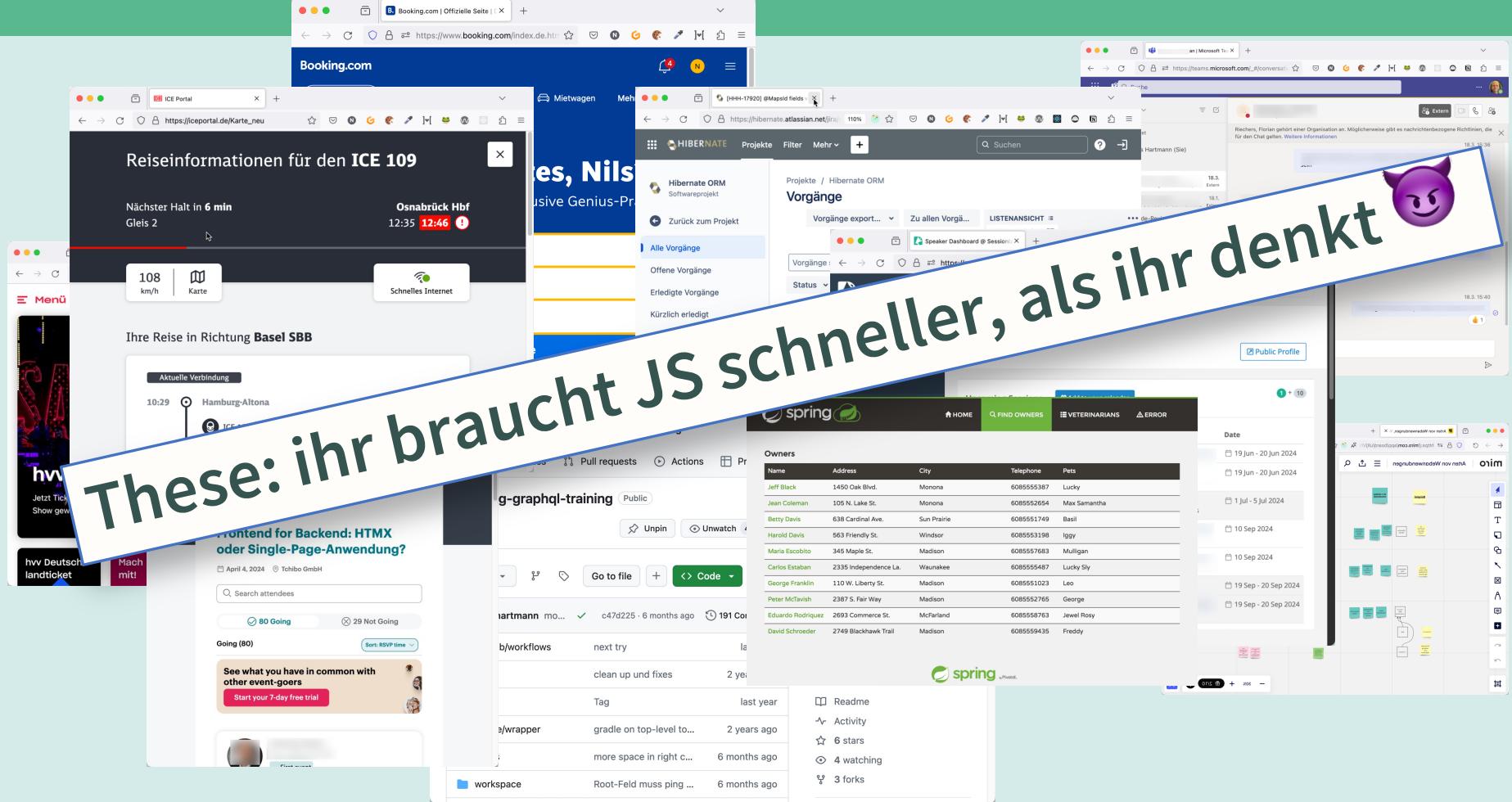
Mit Klick auf „Alle akzeptieren“ willst Du zugleich in die Übermittlung von Daten in Drittstaaten ein, die kein mit der EU vergleichbares Datenschutzniveau aufweisen. Sofern personenbezogene Daten dohrt übermittelt werden, besteht das Risiko, dass Behörden diese erfassen und analysieren ohne dass Du 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

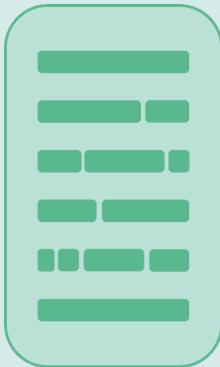


Es geht also "nur" darum, wie wir das JavaScript managen

Welche Konsequenz hat JavaScript **zur Laufzeit**

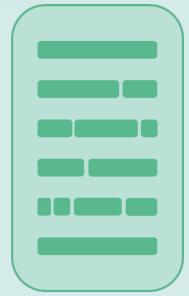
...und bereits während der **Entwicklung?**

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



HTML-Seite

Ja, das geht!



HTML-Seite

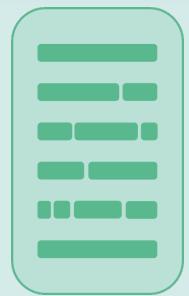
- Typische Vertreter dieser Architektur zum Beispiel Spring WebMVC
- Templatesprache (z.B. Thymeleaf)
- JavaScript Schnipsel ("vanilla" oder zum Beispiel jQuery)

Ja, das geht!

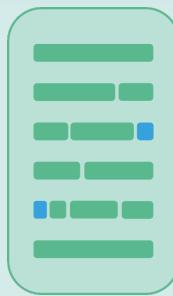


- Eigentlich optimal:
 - wir haben **JavaScript** nur da, wo wir es **wirklich** brauchen, für Interaktivität
 - alles andere kann statisches HTML und CSS sein ❤️

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

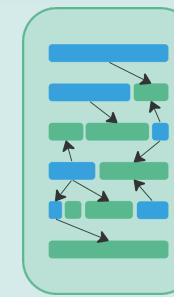
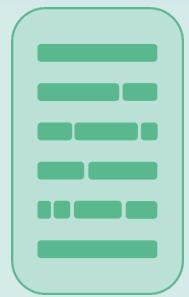


HTML-Seite



- Wir schreiben also noch etwas mehr JavaScript Schnipsel

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



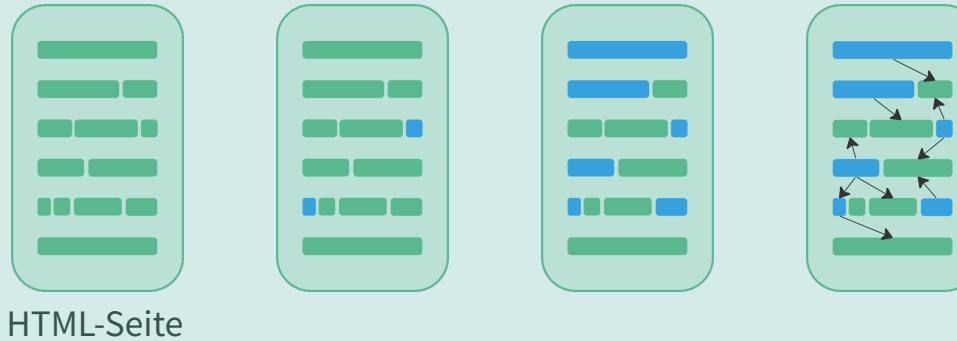
HTML-Seite

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

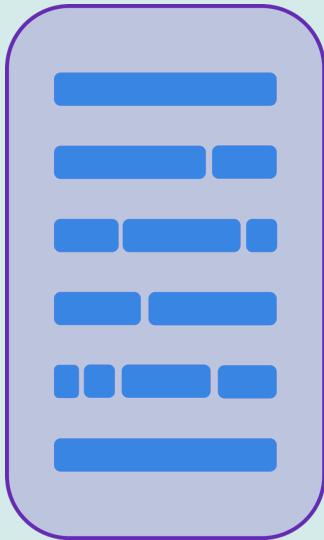
Dann besser alles in JavaScript? Single-Page-Anwendungen



JavaScript-Code

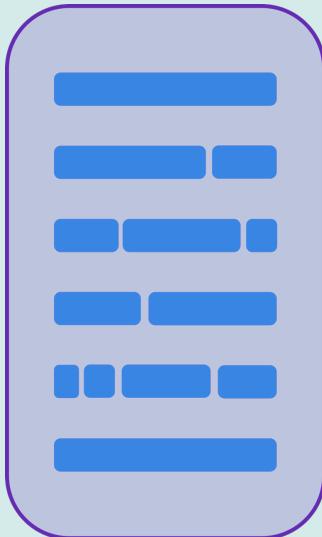
- ab ca. 2010
- Aus "Seiten" werden jetzt "Anwendungen"
- Klare Verantwortlichkeit: Server für Logik und Daten, Browser für UI
- Es gibt stabile und verbreitete Frameworks für jeden Geschmack

Single-Page-Anwendungen



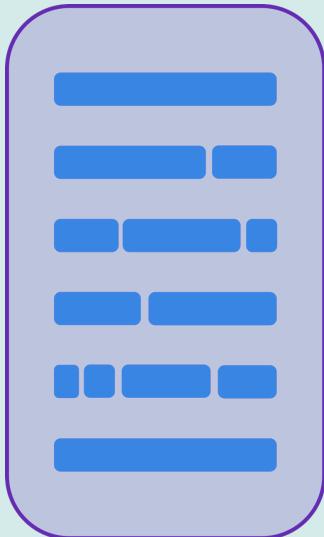
- Darstellung erfolgt vollständig mit JavaScript
- Statisches HTML spielt (fast) keine Rolle
- Die Anwendung kommuniziert mit dem Backend über API
- Ausgetauscht werden Daten, aber keine UI

Single-Page-Anwendungen



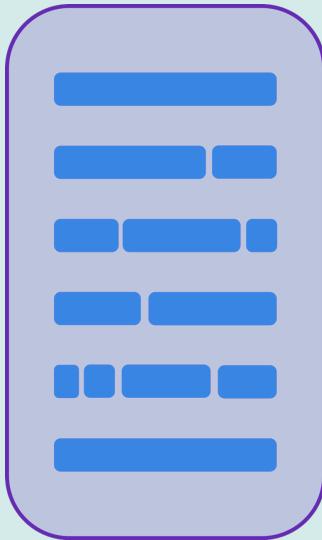
- Konsequenz #1: sehr viel JavaScript zur Entwicklungszeit

Single-Page-Anwendungen



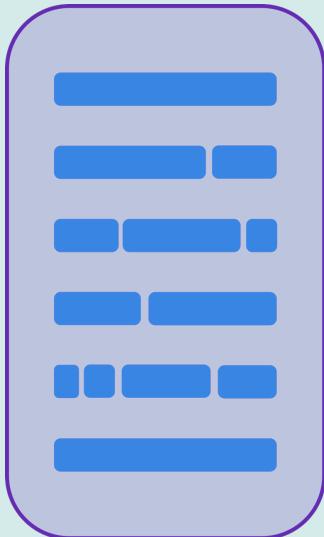
- Konsequenz #1: viel JavaScript zur Entwicklungszeit
- Konsequenz #2: viel JavaScript zur Laufzeit

Single-Page-Anwendungen



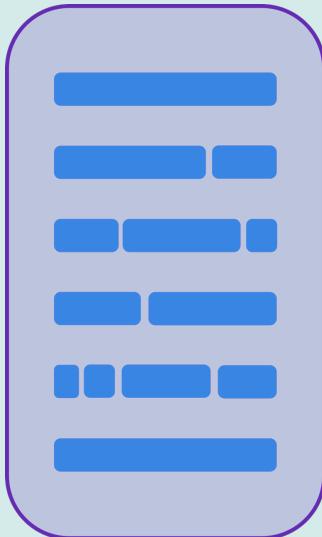
- Konsequenz #1: viel **JavaScript** zur **Entwicklungszeit**
- Wenn man JavaScript doof findet, falscher Ansatz
- Es gibt modernes Tooling für die Entwicklung
 - Wie aus der Enterprise Backend-Entwicklung mit Java gewohnt
 - IDEs und Debugger, Typsicherheit, Unit- und Integrationstest, ...

Single-Page-Anwendungen



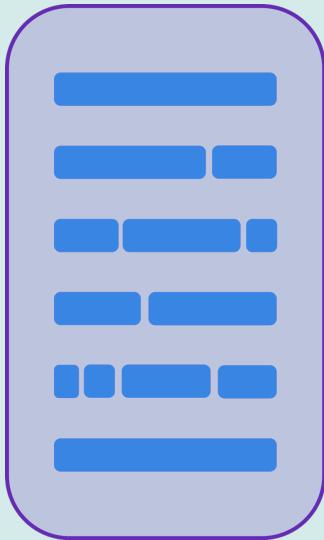
- **Konsequenz #2: viel JavaScript zur Laufzeit**
- Auch für **statische Inhalte**

Single-Page-Anwendungen



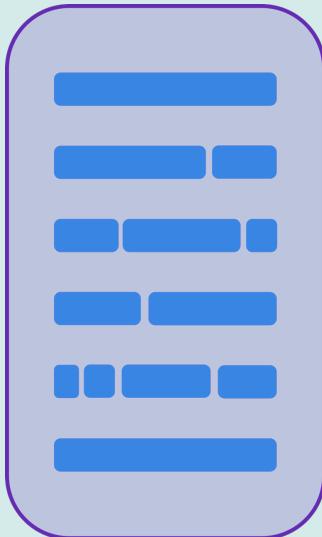
- JavaScript-Code:
 1. muss zum Browser gesendet werden
 2. muss vom Browser ausgeführt werden
 3. kann dann die darzustellenden Daten lesen
 4. kann dann erst die Daten anzeigen
 5. erst dann ist die Anwendung einsatzbereit
 6. Mit jedem Feature wird es mehr

Single-Page-Anwendungen

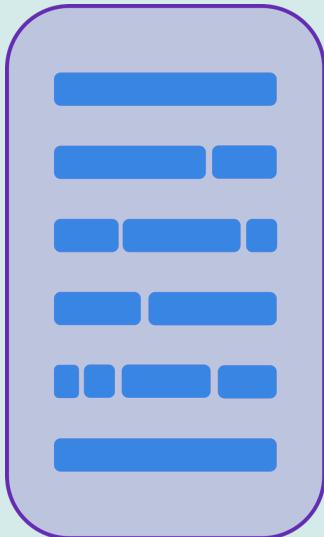


- Das hat Auswirkungen auf die Laufzeit-Performance
- Insbesondere beim Starten
- Ob das ein Problem für die eigene Anwendung ist, muss man von Fall zu Fall entscheiden
 - In-House- oder B2B-Anwendungen haben andere Anforderungen als ein Online-Shop

Fullstack-Anwendungen (JavaScript)

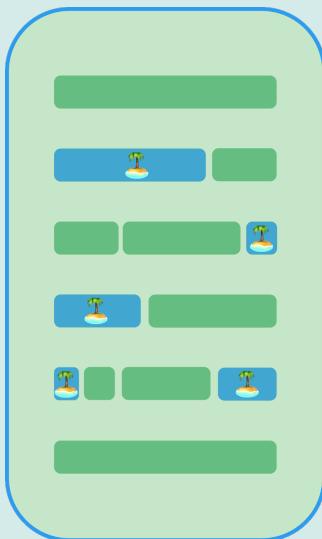


Fullstack-Anwendungen (JavaScript)



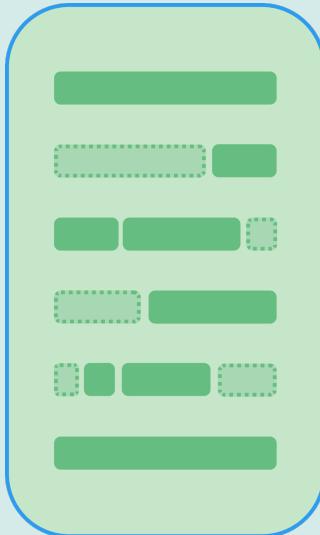
- Ebenfalls vollständig in **JavaScript** geschrieben

Fullstack-Anwendungen (JavaScript)



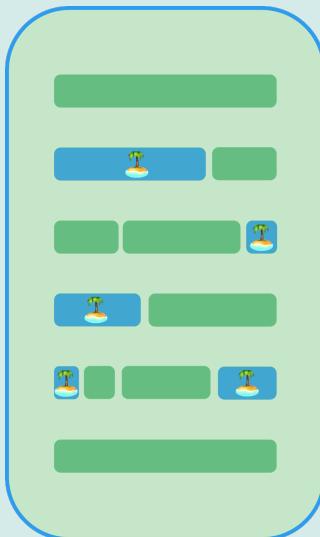
- Mehrere Ansätze und Frameworks
- Funktionalität und Herangehensweise unterschiedlich
 1. Next.js (React)
 2. SvelteKit (Svelte)
 3. Nuxt.js (Vue)
 4. Astro (eigenes Framework + Support für alle SPAs)
 5. Qwik (eigenes Framework)

Fullstack-Anwendungen (JavaScript)



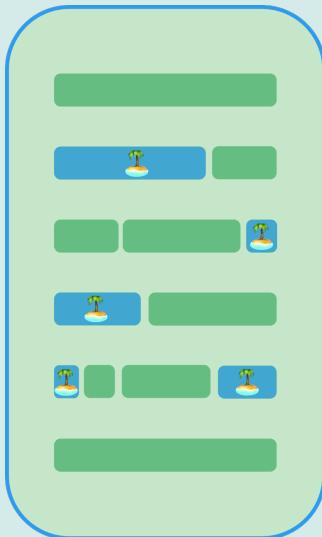
- Grundsätzliche Idee:
 1. **UI-Code** wird serverseitig vorgerendert
 2. **UI-Code** wird zum Browser gesendet und angezeigt

Fullstack-Anwendungen (JavaScript)



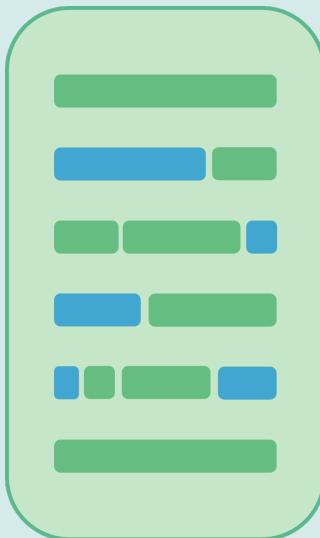
- Grundsätzliche Idee:
 1. **UI-Code** wird serverseitig vorgerendert
 2. **UI-Code** wird zum Browser gesendet und angezeigt
 3. Nur der JavaScript-Code ("Islands") **für Interaktionen** wird zum Browser geschickt

Fullstack-Anwendungen (JavaScript)



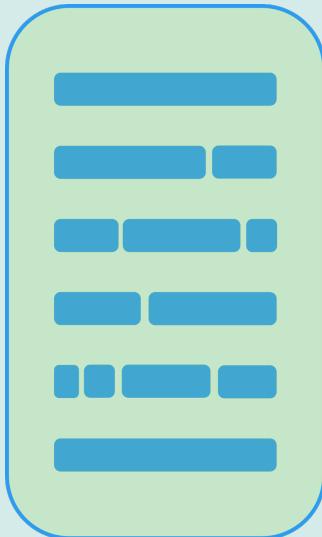
- Anwendung startet schneller:
 1. Browser bekommt **UI-Code** zur Darstellung
 2. Der **notwendige** JavaScript-Code wird nachgeladen
 3. Anwendung jetzt **interaktiv**

Fullstack-Anwendungen (JavaScript)



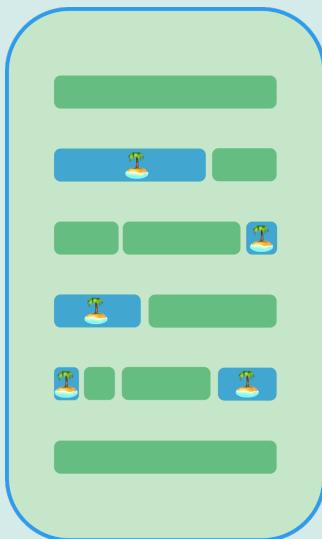
- Wir sind zurück zur **JavaScript-Schnipsel-Architektur**
 - aber: die Schnipsel werden automatisch vom Framework erzeugt
 - die Schnipsel existieren nur zur **Laufzeit**

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)



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

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")

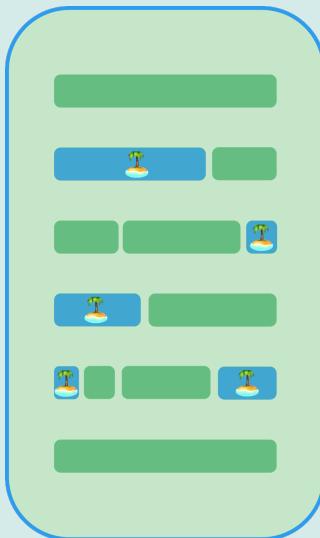
```
<html lang="en">
  <body>
    <div hx-get="/hello-world"
          hx-trigger="click"
          hx-target="#result">
      Get Greeting
    </div>

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

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

Demo: Hello World

HTMX: Grenzen



- HTMX macht bei "triggern" Server Requests und tauscht UI aus
- Man muss eine Art eigene DSL beherrschen
- Für alles andere braucht man... JavaScript!

Die
Ansätze
unter der
Lupe

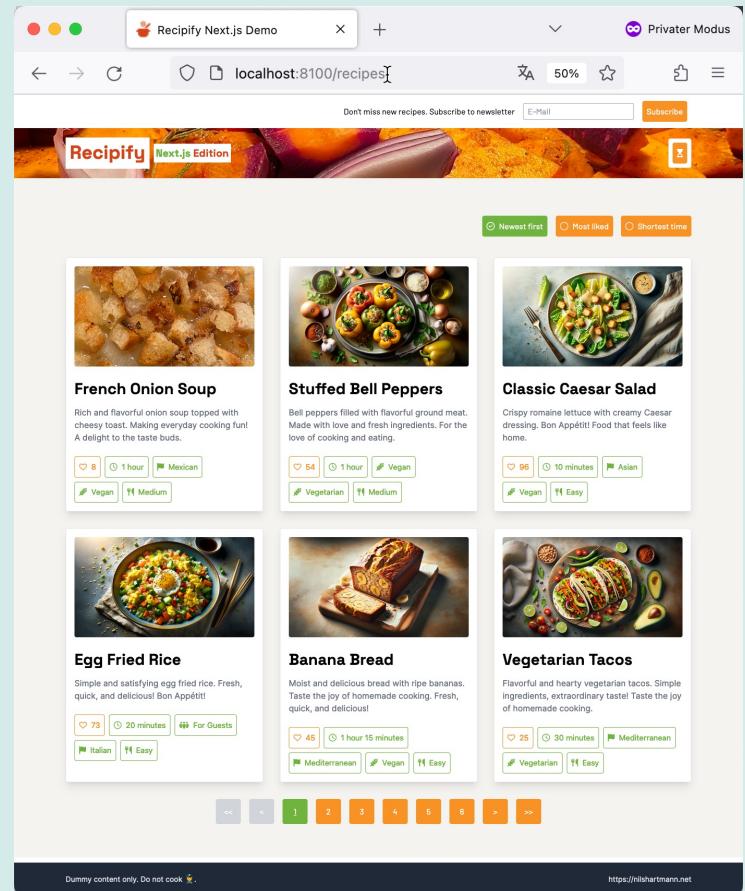
EIN PAAR BEISPIELE...

BEISPIEL: INITIALER SEITENAUFRUF

- Beispiel: initialer Seitenaufruf
- Demo: Next.js-Variante
- Demo: React

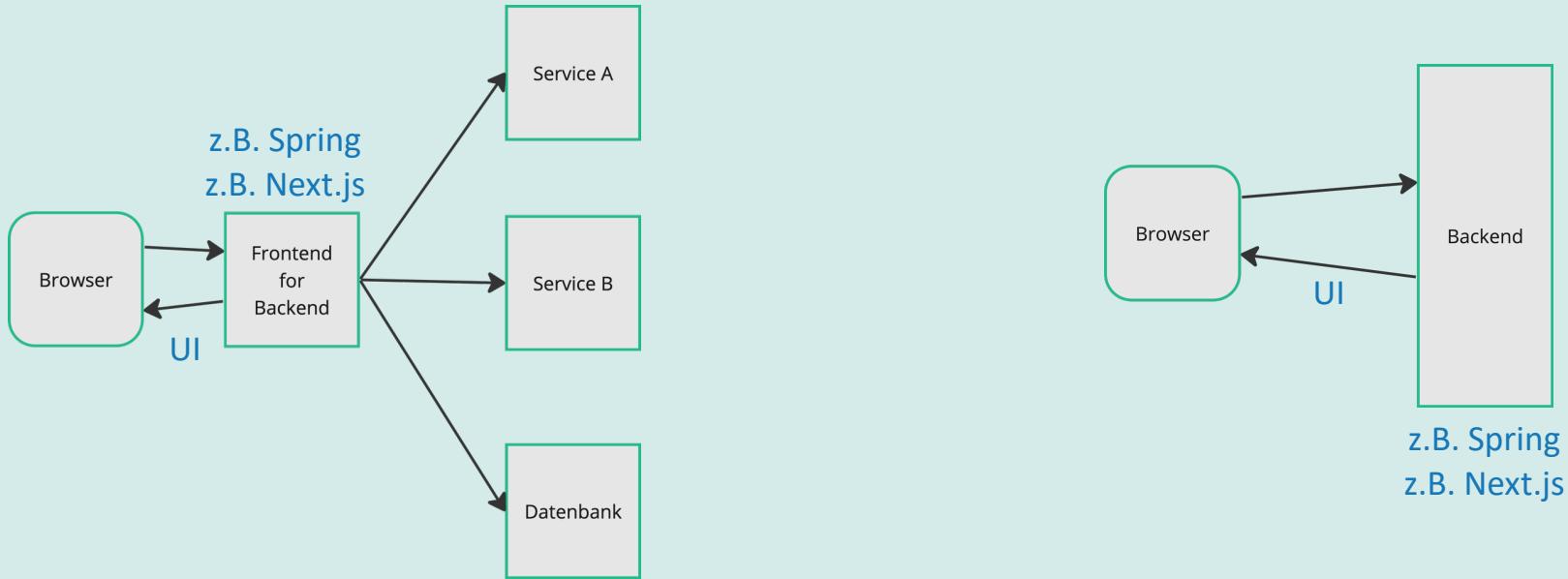
Laufzeit

- SPA: Seite leer (oder SSR)
- Fullstack: HTML
- HTMX: HTML



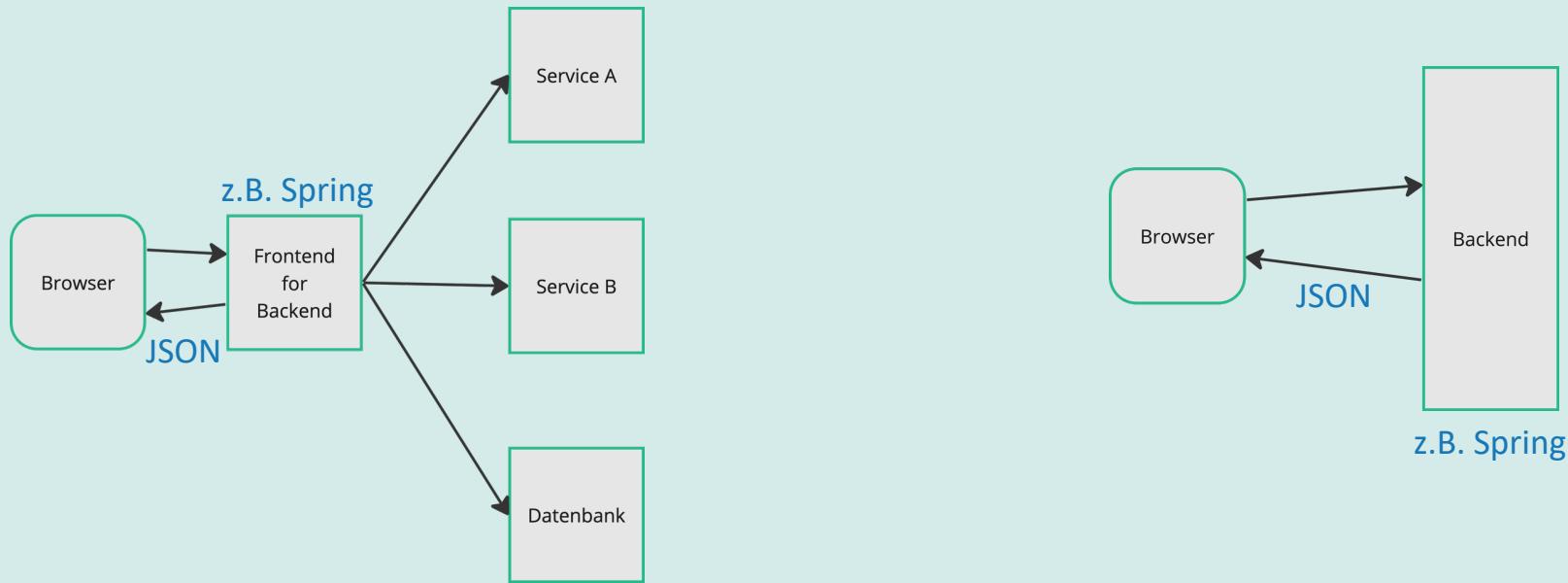
BEISPIEL: INITIALER SEITENAUFRUF

- **Datenzugriff**
- Fullstack / HTMX: In-Process, DB, API, ...
- Browser bekommt **UI**



BEISPIEL: INITIALER SEITENAUFRUF

- **Datenzugriff**
- SPA: HTTP / REST / GraphQL API
- Browser bekommt **Daten**

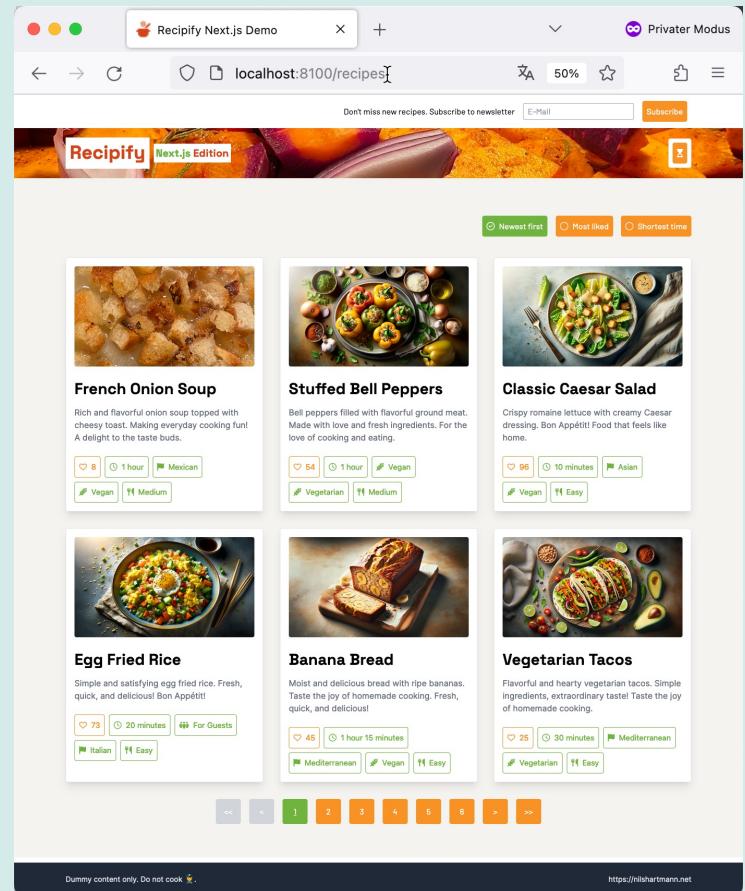


BEISPIEL: INITIALER SEITENAUFRUF

• Beispiel: initialer Seitenaufruf

Code

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

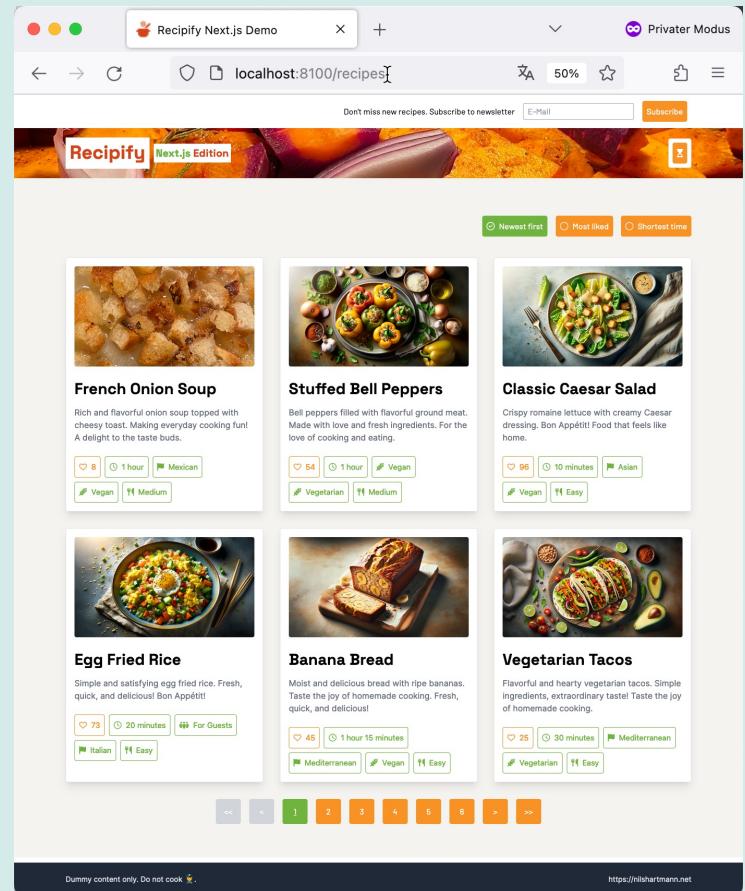


BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren
- Demo: Next.js-Variante
- Demo: Next.js Cache
- Demo: React SPA

Laufzeit

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



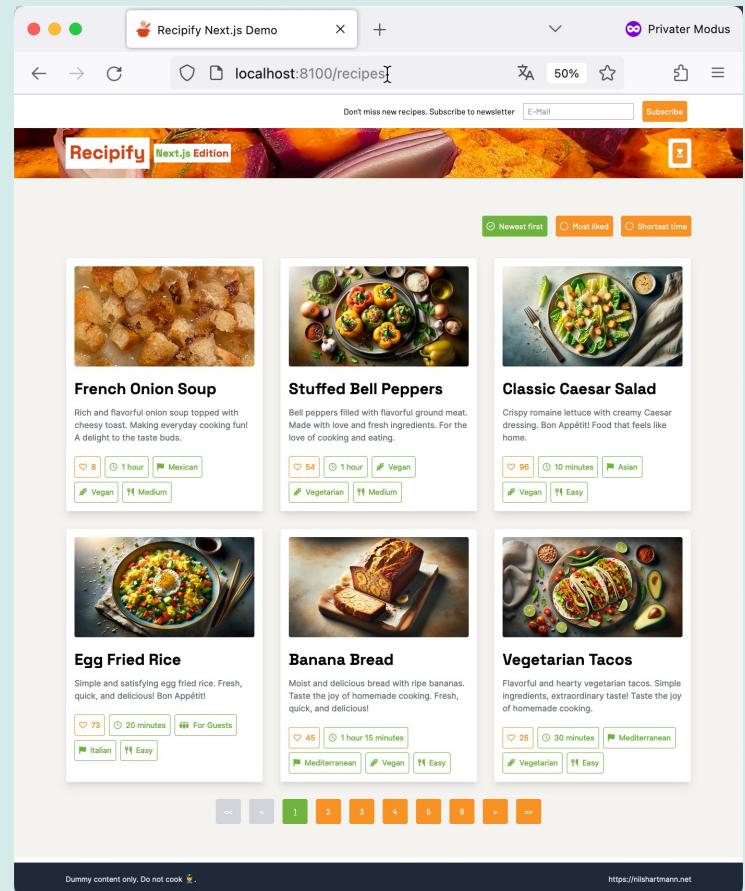
BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren

HTMX

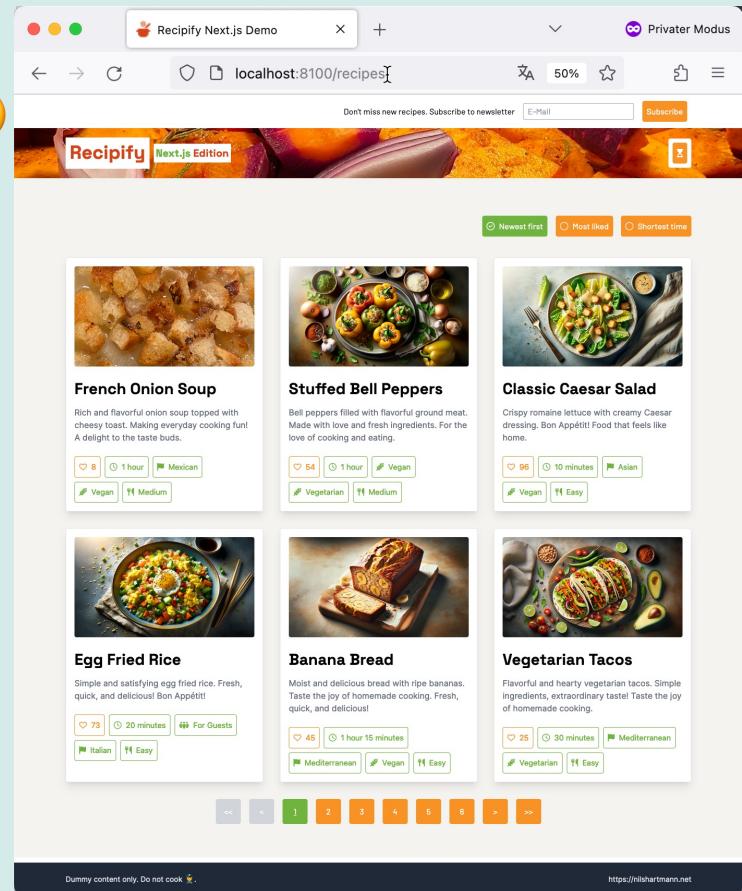
- Trigger auf (a)-Element
- Verhindert das "klassische" Navigieren
- Lädt neues HTML, ersetzt "main"

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



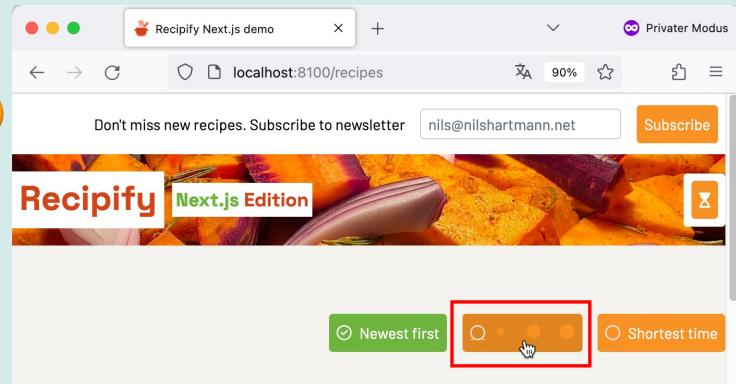
BEISPIEL: SEITE AKTUALISIEREN

- Beispiel: Seite aktualisieren
- 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>  
  );  
}
```

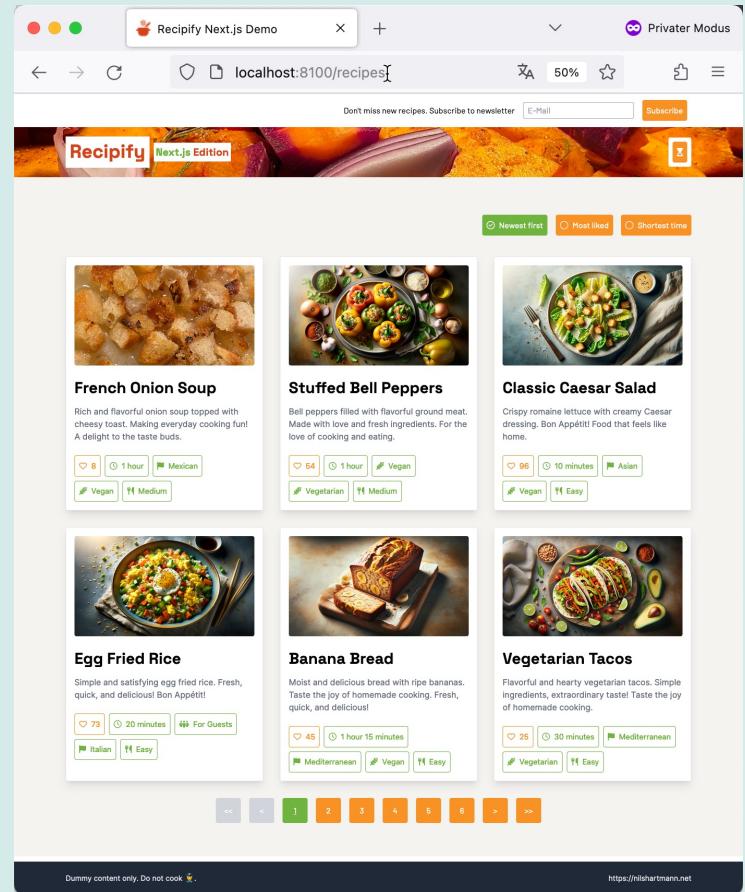
BEISPIEL: SEITE AKTUALISIEREN

- 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

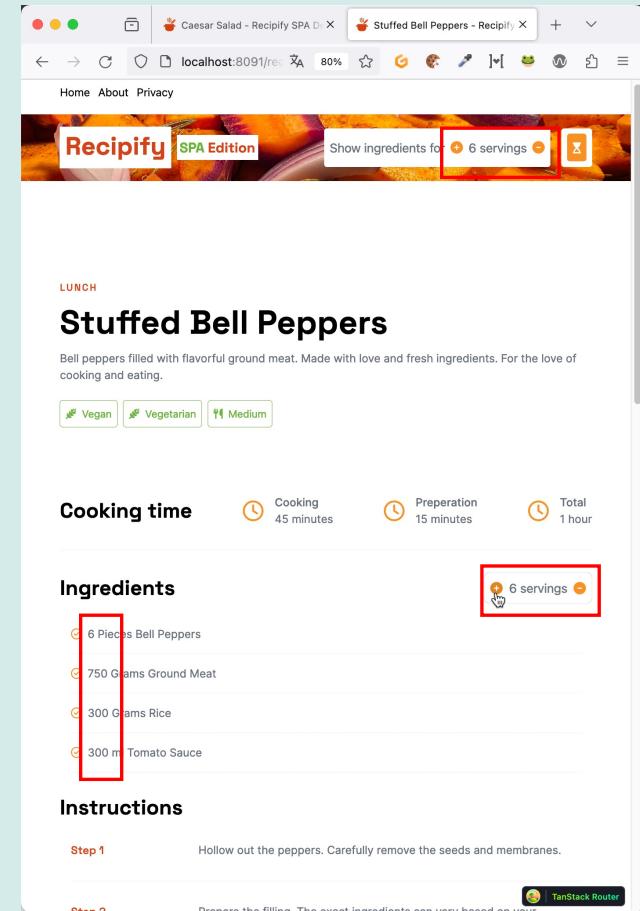
- Beispiel: Verlinkbare Seiten
- HTMX: **Zwei** Endpunkte erforderlich
 - Endpunkt **eins** für initiale, vollständige Seite (mit oder ohne Sortierung)
 - Endpunkt zwei für den Trigger zum Sortieren auf dem Link (nur Austausch des main-Elementes)

BEISPIEL: CLIENT-ZUSTAND

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

Laufzeit

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



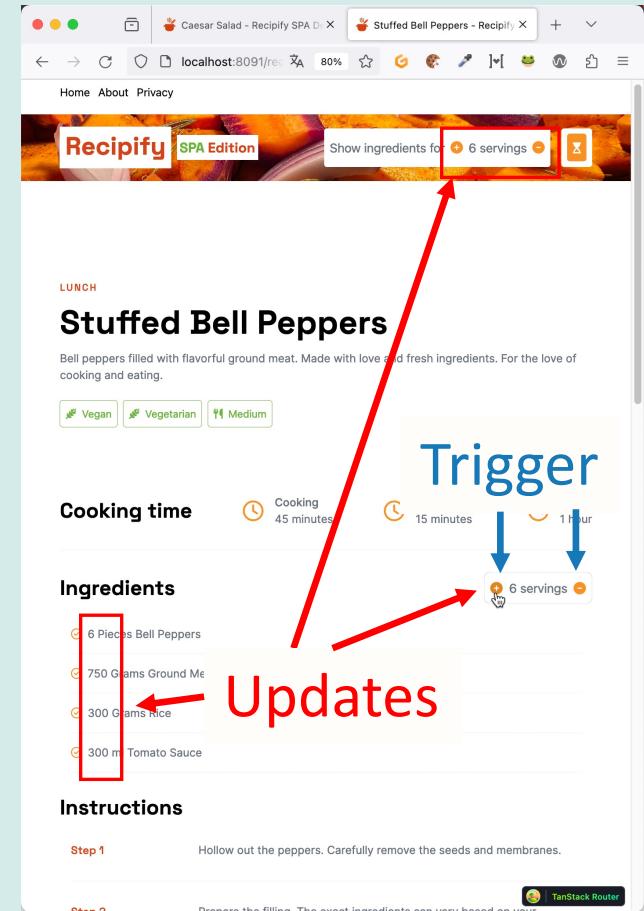
BEISPIEL: CLIENT-ZUSTAND

- 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

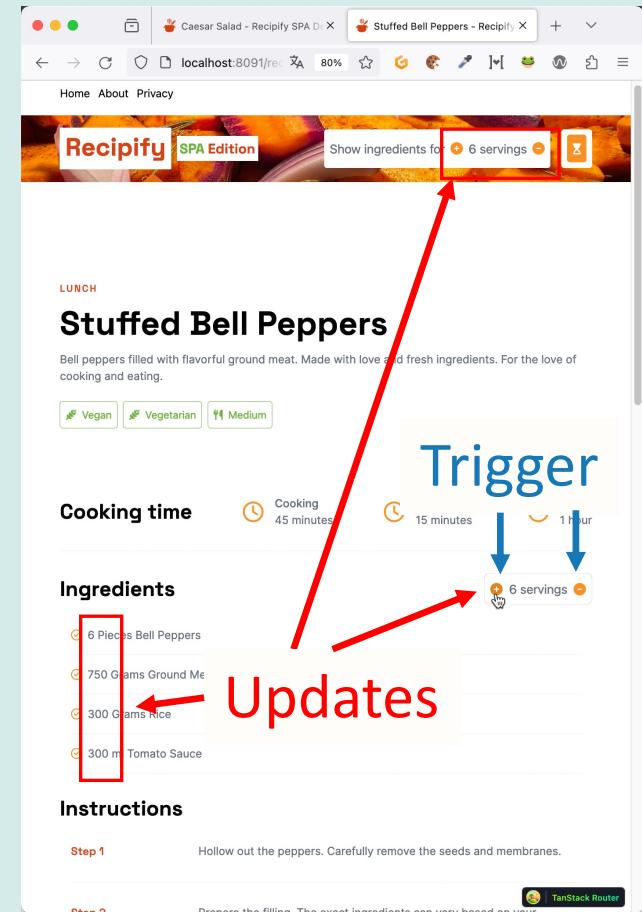
- HTMX Option 2: Serverseitig

- 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

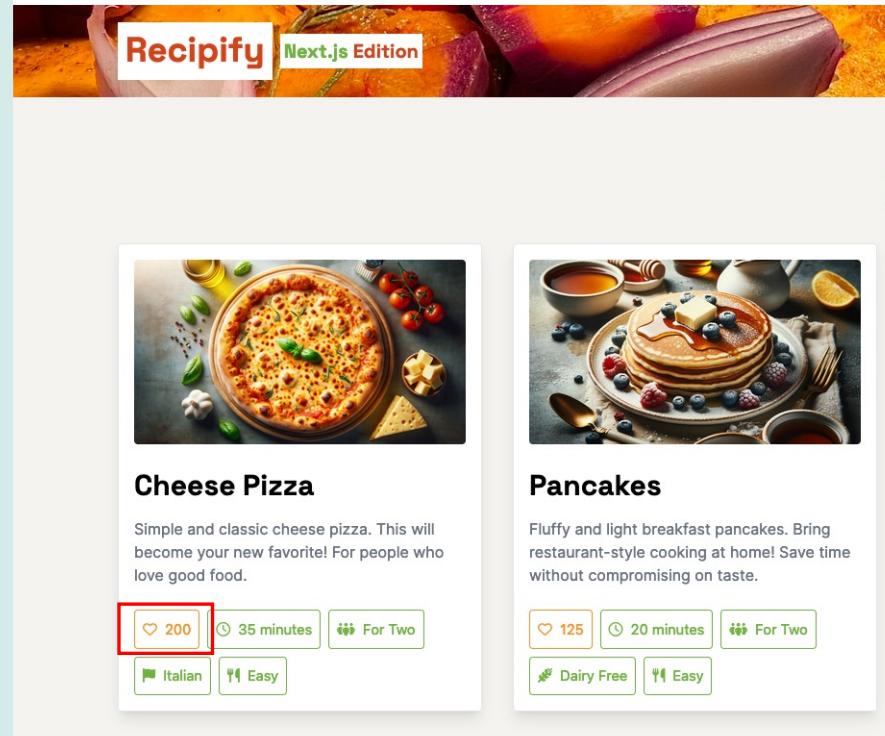


BEISPIEL: DATEN ÄNDERUNGEN

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

Laufzeit

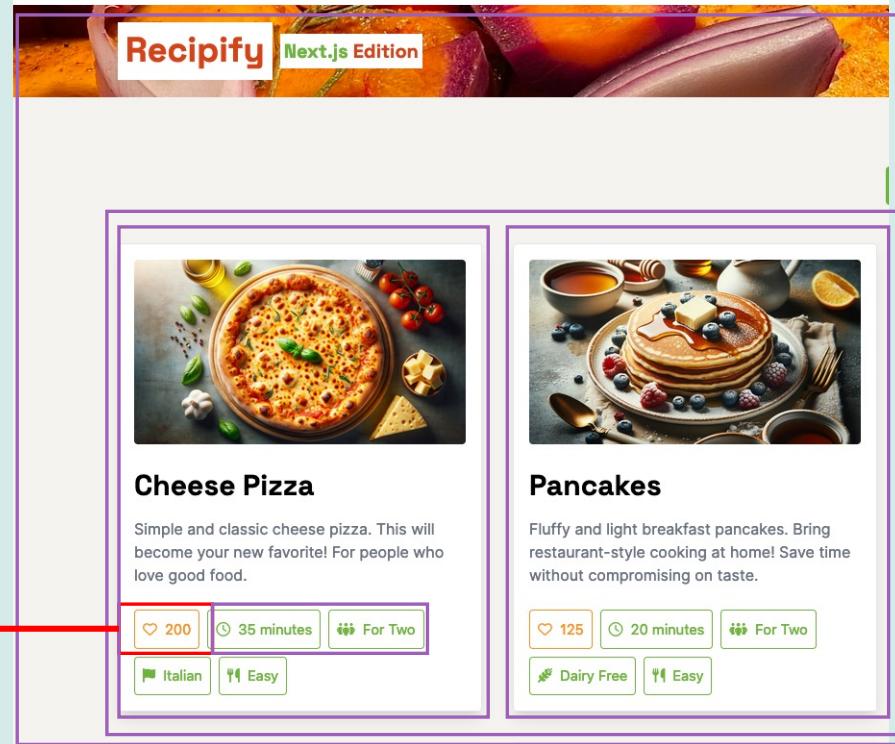
- SPA: HTTP Request, aktualisierter Like kommt zurück
- Fullstack: HTTP Request (ggf. proprietäres Protokoll, z.B. React Server Action)
- HTMX: Server Request (UI kommt zurück)



BEISPIEL: DATEN ÄNDERUNGEN

- Beispiel: Next.JS
- 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: Next.JS**

- 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?

- **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: Fullstack + HTMX vorsichtig vergleichbar
- SPA: tendenziell nur Daten

FAZIT

- Entwicklung: Fullstack + SPA vorsichtig vergleichbar
- HTMX fällt hier deutlich raus

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
 - ganz andere fachlicher Komplexität
 - viel höhere Ansprüche



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

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 😊)

Vielen Dank!

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

Fragen & Kontakt: nils@nilshartmann.net

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