

NILS HARTMANN
<https://nilshartmann.net>



Nächste Generation Webframeworks

Alles
Fullstack
oder was?

Slides: <https://react.schule/fullstack-oder-was>

NILS HARTMANN

nils@nilshartmann.net

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



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

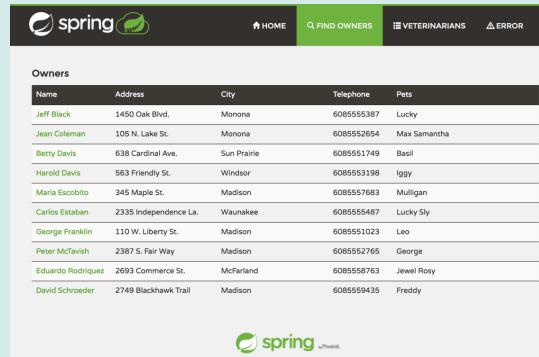


<https://reactbuch.de>

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

Webanwendungen...

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 columns: Name, Address, City, Telephone, and Pets. The table lists 15 entries, each with a small thumbnail image of a pet next to its name.

Name	Address	City	Telephone	Pets
Jeff Black	1450 Oak Blvd.	Monona	6085555387	Lucky
Jean Coleman	105 N. Lake St.	Monona	6085552654	Mae 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 the ICE Portal website interface. At the top, it displays "Reiseinformationen für den ICE 109". Below this, it says "Nächster Halt in 6 min" and "Gleis 2". The train's current speed is 108 km/h, and the destination is Osnabrück Hbf at 12:35, with a red digital clock showing 12:46. A warning icon is present. On the left, there's a sidebar with "Menü" and a "hw plus" advertisement for Sasha Show tickets. The main content area shows the route from Hamburg-Altona to Basel SBB, listing stops like Hamburg-Altona, ICE 109 nach Basel SBB, and Gl. 10. It includes buttons for "Vergangene Halte zeigen", "Ab", "An", "Suchen", "Erweiterte Suche", and "Aktuelle Meldungen". At the bottom, there are links for "hvv Deutschlandticket", "Mach mit!", "Schüler*innen", and "Schnelles Internet".

The screenshot shows the Spring website with a navigation bar featuring "HOME", "FIND OWNERS" (which is highlighted in green), "VETERINARIANS", and "ERROR". The main content area is titled "Owners" and displays a table of pet owners with their details:

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

At the bottom, there's a logo for "spring" and "powered by JBoss Seam".

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 six browser tabs open, illustrating various web applications:

- Booking.com**: A travel search engine showing flight information for flight 108 from Osnabrück Hbf to Basel SBB.
- ICE Portal**: A real-time train status board for ICE 109, showing the next stop at Gleis 2 in 6 minutes.
- HIBERNATE**: A Jira project management interface for the "Hibernate ORM" software project.
- sessionize**: A speaker dashboard for Nils Hartmann, showing session details and a public profile.
- spring-graphql-training**: A GitHub repository for a GraphQL training project, listing pull requests and issues.
- Attendee list for Frontend for Backend: HTMX oder Single-Page-Anwendung?**: An event attendee list from Ichiba GmbH, showing RSVP status for April 4, 2024.

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 with Nils Hartmann and other team members.
- 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 mobile-style interface showing travel details, including "Nächster Halt in 6 min" (Next stop in 6 min) and "Gleis 2" (Platform 2).
- ChatGPT - DALLE**: A collaboration between AI chat and image generation, showing a request for a cooking recipe website and two generated images of a beef burger.
- HIBERNATE**: A project management tool showing a list of projects and tasks, with a focus on "Hibernate ORM".
- sessionize**: A platform for organizing events, showing a speaker dashboard for Nils Hartmann.
- Microsoft Teams**: A communication and collaboration tool showing a chat history and file attachments.
- Spring**: A developer-oriented platform showing a list of owners with details like name, address, city, phone, and pets.

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 interface for the Hibernate ORM software.
- Vorgänge**: A screenshot of a task or event management system.
- sessionize**: A screenshot of a speaker dashboard for sessionize.com, showing profile information for Nils Hartmann.
- spring**: Two screenshots from the Spring framework's developer portal, one showing owners and another showing a code repository.
- Google Sheets**: A screenshot of a Google Sheets spreadsheet with multiple tabs and data.

WEBANWENDUNGEN

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

The screenshot shows a Mac OS X desktop with a windowed browser. The top tab bar has three tabs: "ICE Portal", "Booking.com | Offizielle Seite", and "ChatGPT - DALL-E". Below the tabs, several browser windows are open:

- ICE Portal**: Shows a speed limit of 108 km/h.
- Booking.com**: The main page.
- ChatGPT - DALL-E**: A tab showing two images of a beef burger.
- hvg plus**: A ticketing service for the Sasha Show.

The screenshot displays several web application interfaces:

- Hibernate ORM**: A software project management tool showing tasks like "Serve Genius Project" and "Zurück zum Projekt".
- ChatGPT - DALL-E**: A tab showing two images of a beef burger.
- Spring Boot**: A Java framework interface showing a list of owners with columns for Name, Address, City, Telephone, and Pets.
- ChatGPT-openai.com**: A tab showing a message from DALL-E about creating a cooking recipe website.
- ICE Portal**: A tab showing a speed limit of 108 km/h.
- Booking.com**: The main page.
- hvg plus**: A ticketing service for the Sasha Show.

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)

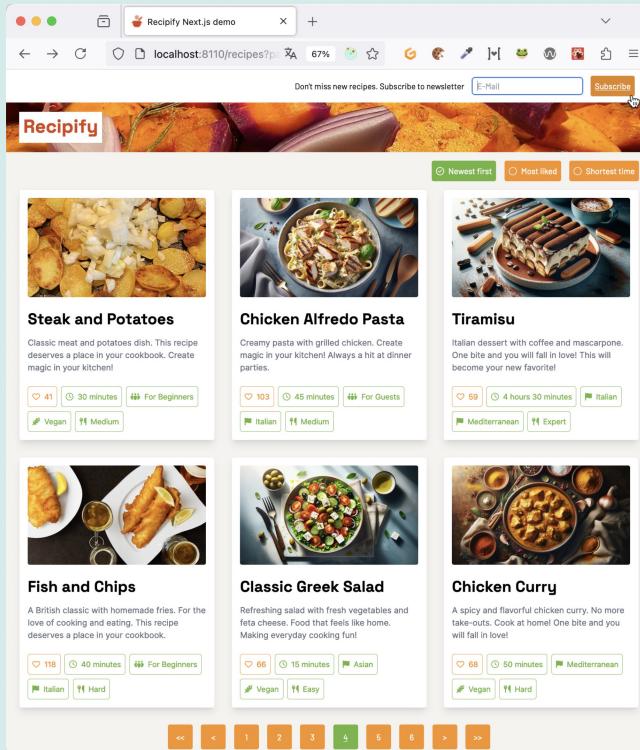
WEBANWENDUNGEN

These: ihr braucht eher JS, als ihr denkt

The image is a collage of several web browser screenshots, each displaying a different web application or service. A large, semi-transparent watermark with the text "These: ihr braucht eher JS, als ihr denkt" is overlaid diagonally across the center of the image.

- Booking.com:** Shows a flight search result for the ICE 109 from Osnabrück Hbf to Berlin Hbf at 12:46.
- ICE Portal:** Displays real-time train status information for the ICE 109, showing a speed of 108 km/h and the next stop at Gleis 2.
- Hibernate:** A project management tool showing tasks for the "Hibernate ORM Softwareprojekt".
- Microsoft Teams:** A messaging interface showing a conversation between users.
- GitHub:** A repository named "g-graphql-training" showing pull requests and code snippets.
- Spring Boot Application:** A simple application titled "spring" with endpoints for "HOME", "FIND OWNERS", "VETERINARIANS", and "ERROR". It lists owners with their addresses, cities, phones, and pets.
- Other Applications:** Includes a travel ticket search interface, a social media-like platform with posts and profiles, and a task management tool with a board view.

BEISPIEL-ANWENDUNG



- <http://localhost:8110>

BEISPIEL-CODE

Ihr findet das "Recipify"-Beispiel hier:

Next.JS

<https://github.com/nilshartmann/alles-fullstack>

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?**

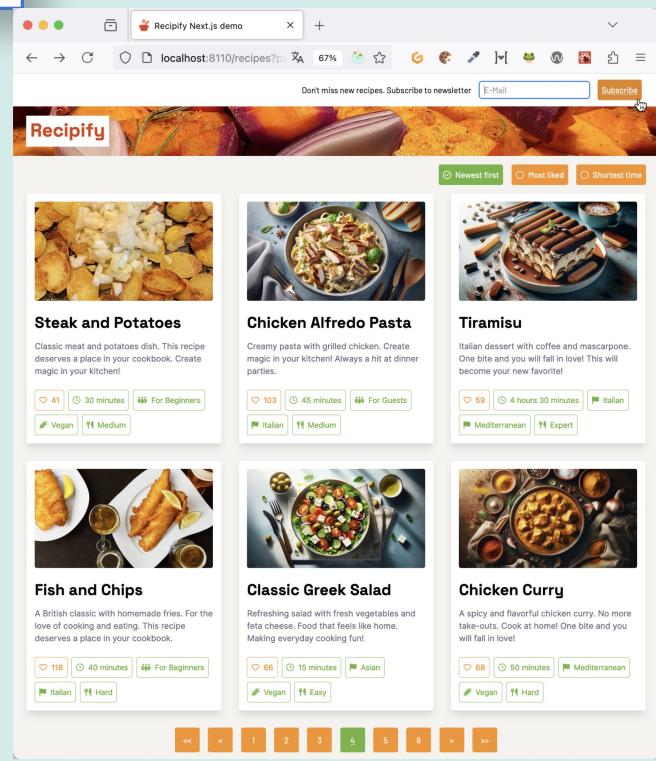
Klassische serverseitige Anwendung...

Klassische serverseitige Anwendung...

- Typische Vertreter zum Beispiel Spring MVC, PHP, .NET, Node.JS
- Oft Model-View-Controller (MVC) Pattern
- Templatesprache (z.B. Thymeleaf, Handlebars)

Unser Server erzeugt eine HTML-Seite...

```
@Controller  
public class RecipeController {  
  
    private final RecipeRepository repository;  
  
    RecipeController(RecipeRepository repository) {  
        this.repository = repository;  
    }  
  
    @GetMapping("/")  
    String getRecipesList(Pageable pageable, Model model) {  
  
        var recipes = this.repository.findAll(pageable);  
  
        // Model mit allen Informationen erzeugen, die  
        // das Template zum rendern braucht  
        model.addAttribute("recipes", recipes);  
        model.addAttribute("pageable", pageable);  
  
        // Name des Templates, das den HTML-Code für die  
        // Seite erzeugt  
        return "index";  
    }  
}
```



WEBANWENDUNGEN

...die ist aber komplett statisch

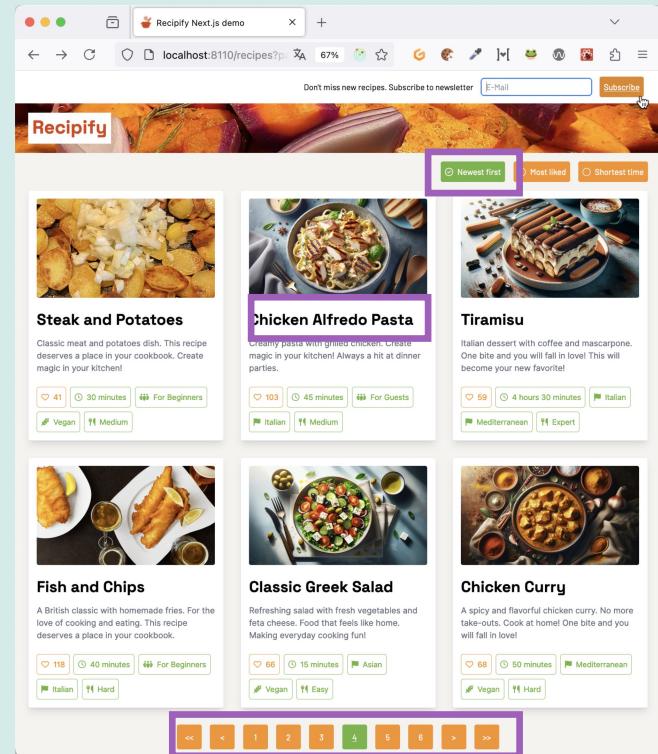
The screenshot shows a web browser window displaying a static website for a recipe platform. The header features the word "Recipify" in a white, rounded font. Below the header is a navigation bar with links like "Home", "About", "Contact", and "Logout". A search bar is present, along with a "Cart" icon showing a count of 3. The main content area displays a grid of six recipe cards:

- Steak and Potatoes**: A dish of meat and potatoes. Details: 41 likes, 30 minutes, For Beginners, Vegan, Medium.
- Chicken Alfredo Pasta**: A bowl of pasta with chicken. Details: 103 likes, 45 minutes, For Guests, Italian, Medium.
- Tiramisu**: A dessert with layers of cake and coffee. Details: 59 likes, 30 minutes, Italian, Expert.
- Fish and Chips**: A plate of fish and chips. Details: 118 likes, 40 minutes, For Beginners, Italian, Hard.
- Classic Greek Salad**: A bowl of salad with olives and feta cheese. Details: 66 likes, 15 minutes, Asian, Vegan, Easy.
- Chicken Curry**: A bowl of spicy chicken curry. Details: 68 likes, 50 minutes, Mediterranean, Vegan, Hard.

At the bottom of the page is a navigation bar with icons for back, forward, and search, followed by a series of numbered buttons from 1 to 6.

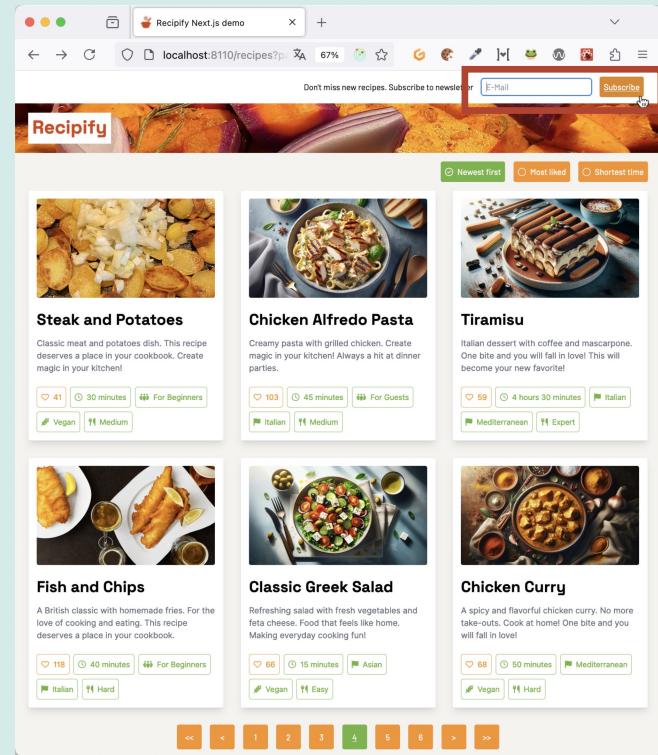
...die ist aber komplett statisch

- Interaktion per Link



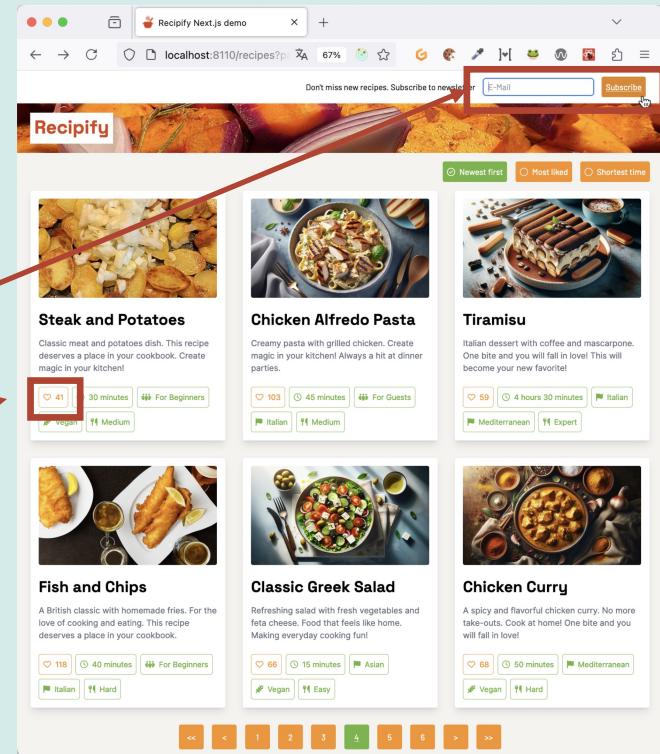
...die ist aber komplett statisch

- Interaktion per **Link**
- oder **Formular**



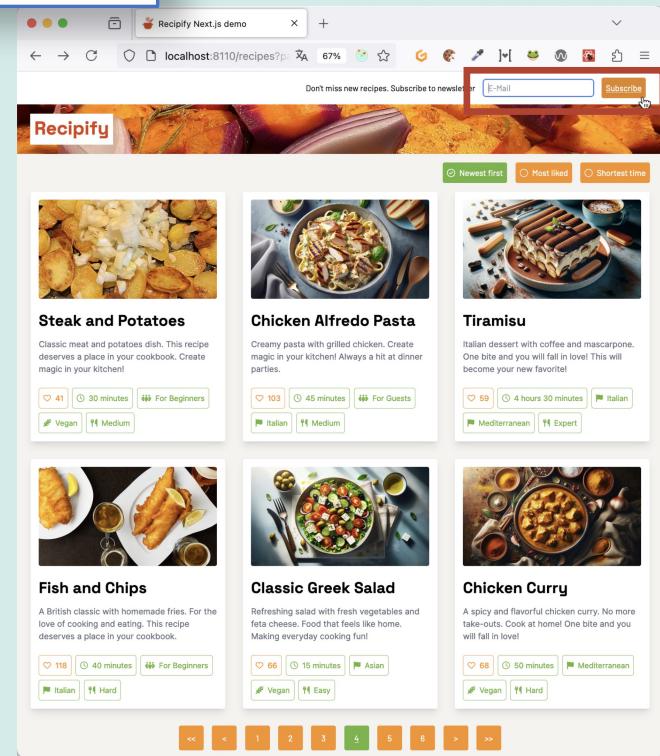
...die ist aber komplett statisch

- Interaktion per **Link**
- oder **Formular**
- in jedem Fall wird eine **komplett neue** Seite abgefragt



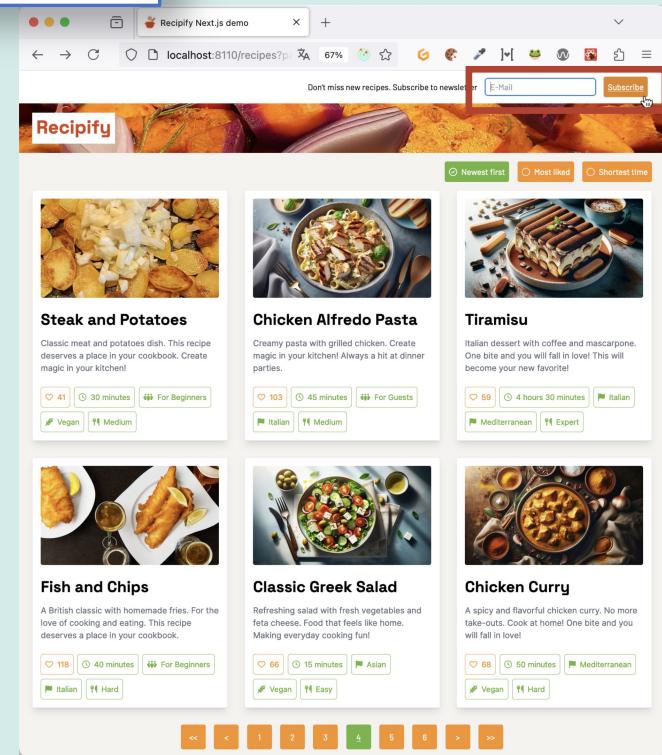
Bei jeder Interaktion: ein Server-Roundtrip...

- ist das ein Problem? 🤔
- was hat das für Konsequenzen? 🤔



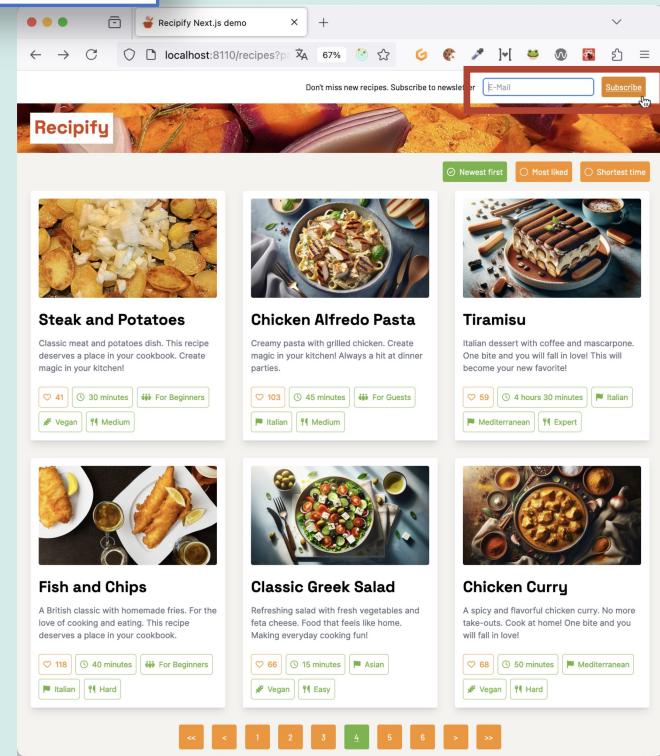
Bei jeder Interaktion: ein Server-Roundtrip...

- Menge an Daten, Latenz
- Zustand in Eingabefeldern geht verloren
- Animationen etc. laufen nicht weiter



Bei jeder Interaktion: ein Server-Roundtrip...

- Menge an Daten, Latenz
 - Zustand in Eingabefeldern geht verloren
 - Animationen etc. laufen nicht weiter
-
-  Newsletter
 -  Like-Button
 -  Timer



Weitere Interaktionen...

- Feedback direkt beim Tippen
- z.B. Zeichen-Zähler

Your opinion?

Your name:

Nils

Your rating:



Your comment:

Lovely!

7/500 characters

Submit Rating

Weitere Interaktionen...

- Feedback direkt beim Tippen
- z.B. Zeichen-Zähler
- Doppeltes Submit verhindern

Your opinion?

Your name:

Nils

Your rating:



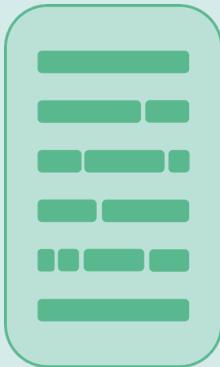
Your comment:

Lecker...!

10/500 characters

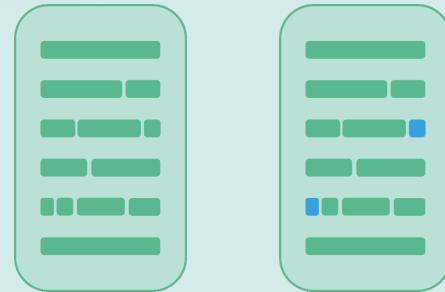
Submit Rating

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



HTML-Seite

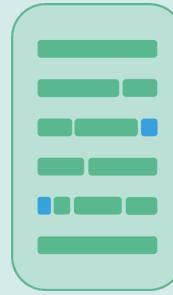
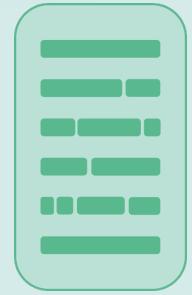
Ja, das geht!



HTML-Seite

- JavaScript Schnipsel einstreun ("vanilla JS" oder zum Beispiel jQuery)

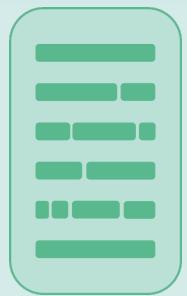
Ja, das geht!



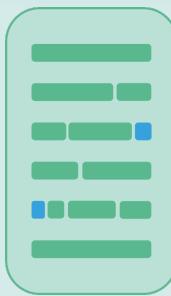
HTML-Seite

- 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..."

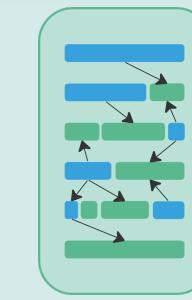
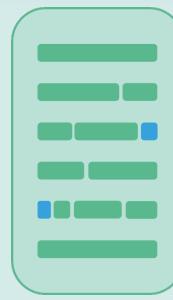
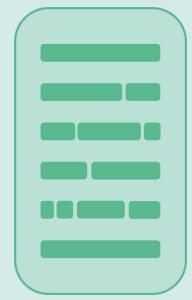


HTML-Seite



- Wir schreiben also noch etwas mehr JavaScript Schnipsel

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



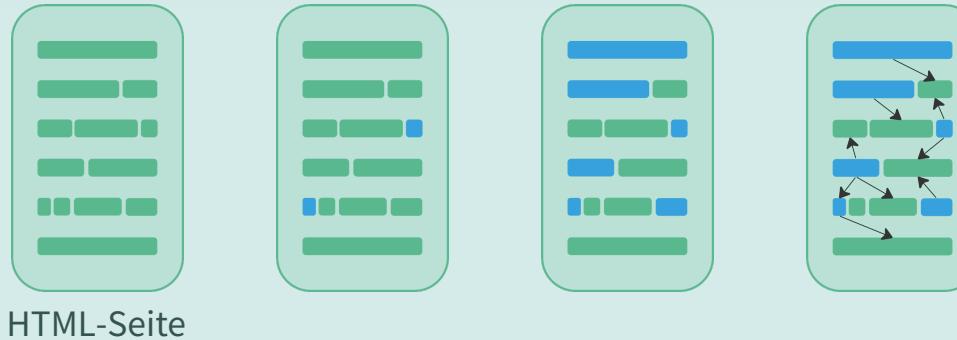
HTML-Seite

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



- Die Probleme "schleichen sich ein"
- Spätestens wenn es um Interaktionen geht, die nicht nur lokal sind
- Plötzlich hat unser Code nicht "ein paar Probleme" sondern ist kaputt

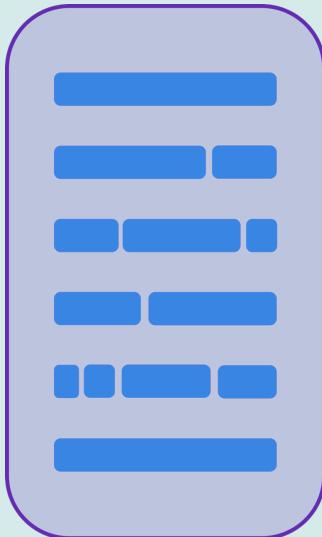
Dann besser alles in JavaScript? Single-Page-Anwendungen



JavaScript-Code

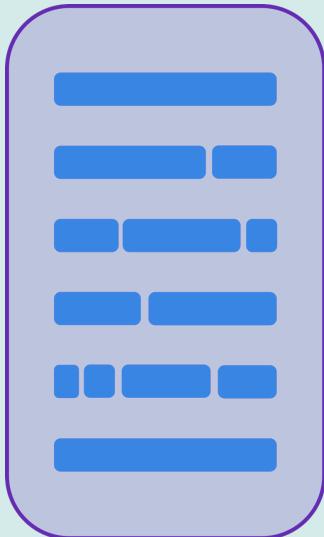
- ab ca. 2010
- Aus "Seiten" werden jetzt "Anwendungen"
- Klare Verantwortlichkeit: Server für Logik und Daten, Browser für UI
- Es gibt stabile und verbreitete Frameworks für jeden Geschmack

Single-Page-Anwendungen



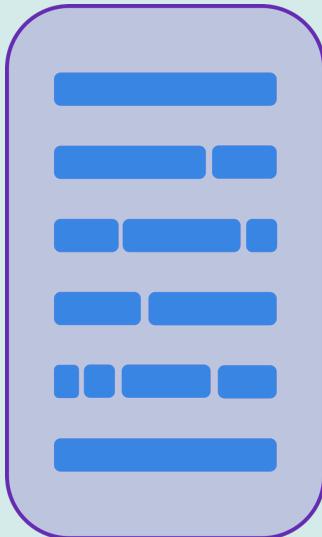
- Darstellung erfolgt vollständig mit JavaScript
- Statisches HTML spielt (fast) keine Rolle
- Die Anwendung kommuniziert mit dem Backend über API
- Ausgetauscht werden Daten, aber keine UI
- Vertreter: Angular, React, Svelte, Vue

Single-Page-Anwendungen



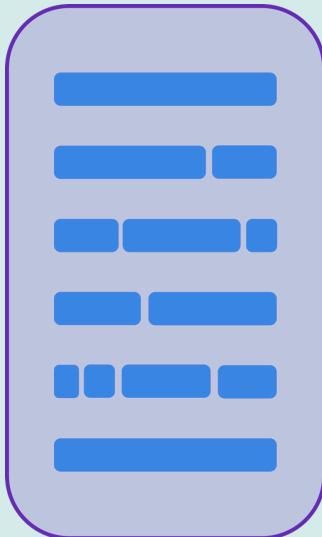
- Konsequenz #1: (viel) JavaScript zur **Entwicklungszeit**

Single-Page-Anwendungen



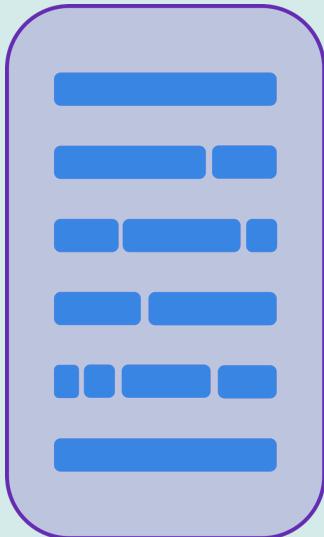
- Konsequenz #1: (viel) JavaScript zur Entwicklungszeit
- Konsequenz #2: (viel) JavaScript zur Laufzeit

Single-Page-Anwendungen



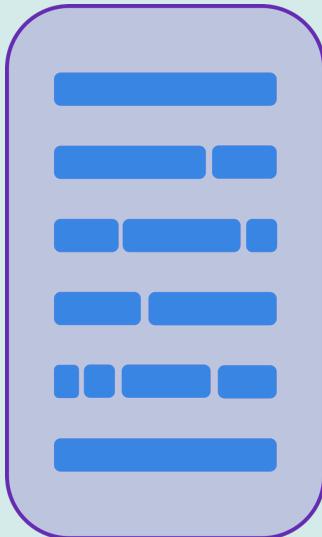
- Konsequenz #2: viel JavaScript **zur Laufzeit**
- Auch für **statische Inhalte**

Single-Page-Anwendungen



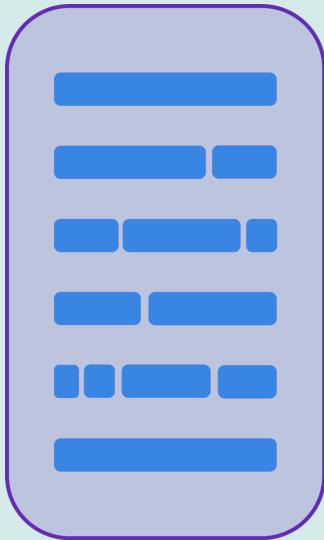
- JavaScript-Code:
 1. muss zum Browser gesendet werden
 2. muss vom Browser ausgeführt werden
 3. kann dann die darzustellenden Daten lesen
 4. kann dann erst die Daten anzeigen
 5. erst dann ist die Anwendung einsatzbereit
 6. Mit jedem Feature wird es mehr

Single-Page-Anwendungen



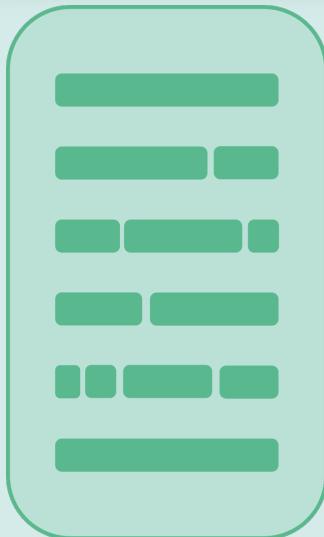
- Das hat Auswirkungen auf die **Laufzeit**-Performance
- Insbesondere beim Starten

Single-Page-Anwendungen

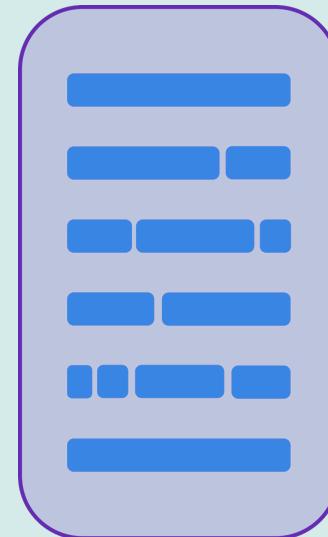


- Das hat Auswirkungen auf die **Laufzeit**-Performance
- Insbesondere beim Starten
- Ob das ein Problem für die eigene Anwendung ist, muss man von Fall zu Fall entscheiden
 - In-House- oder B2B-Anwendungen haben andere Anforderungen als ein Online-Shop

"Gibt es denn nichts dazwischen"?



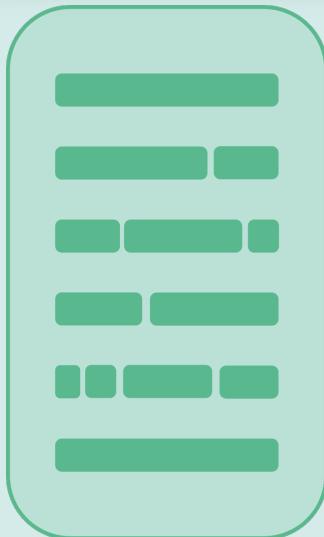
Nur Server (+JS)



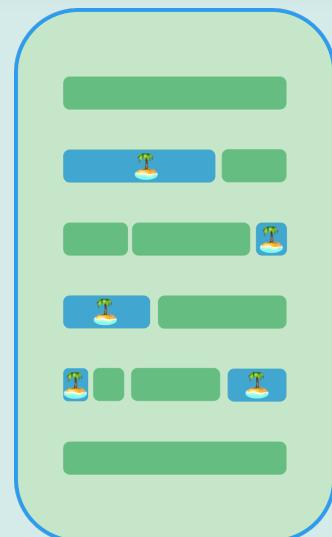
Nur Client (SPA)

WEBANWENDUNGEN

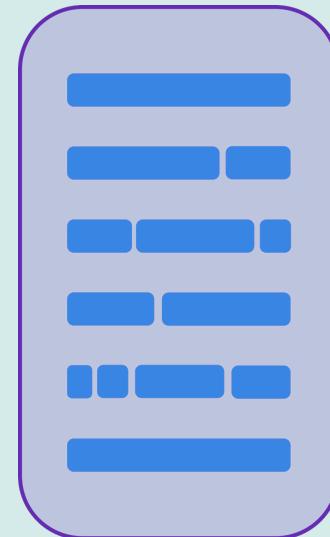
"Doch – Fullstack-Anwendungen 😊"



Nur Server (+JS)



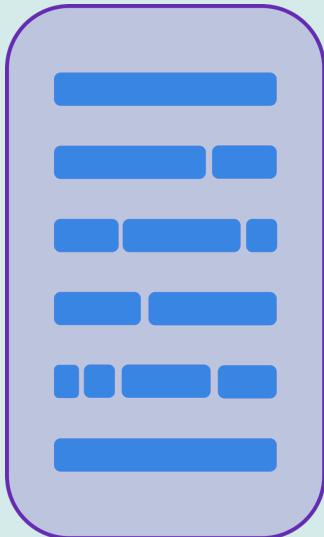
JS Fullstack Anwendung



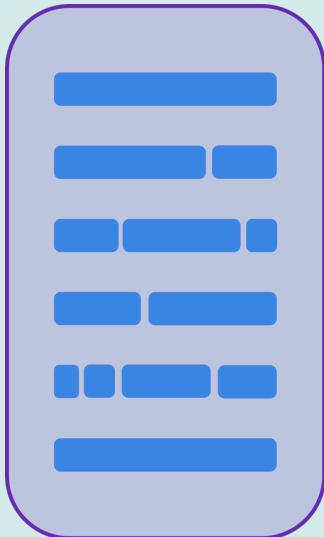
Nur Client (SPA)

FULLSTACK-ANWENDUNGEN (JAVASCRIPT)

Fullstack-Anwendungen (JavaScript)

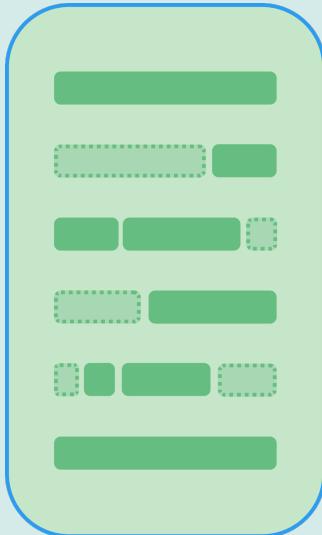


Fullstack-Anwendungen (JavaScript)



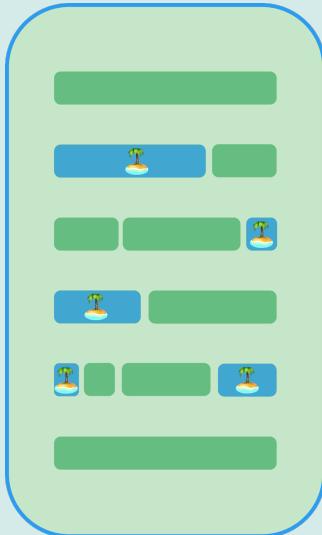
- Ebenfalls vollständig in **JavaScript** geschrieben

Fullstack-Anwendungen (JavaScript)



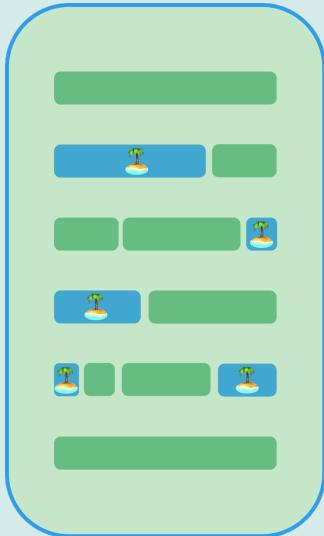
- Grundsätzliche Idee:
 1. **UI-Code** wird serverseitig vorgerendert
 2. **UI-Code** wird zum Browser gesendet und angezeigt

Fullstack-Anwendungen (JavaScript)



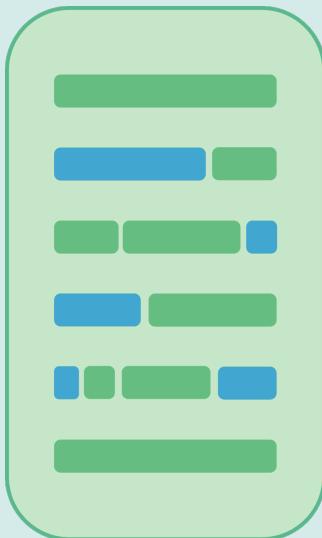
- Grundsätzliche Idee:
 1. **UI-Code** wird serverseitig vorgerendert
 2. **UI-Code** wird zum Browser gesendet und angezeigt
 3. Nur der JavaScript-Code ("Islands") **für Interaktionen** wird zum Browser geschickt

Fullstack-Anwendungen (JavaScript)



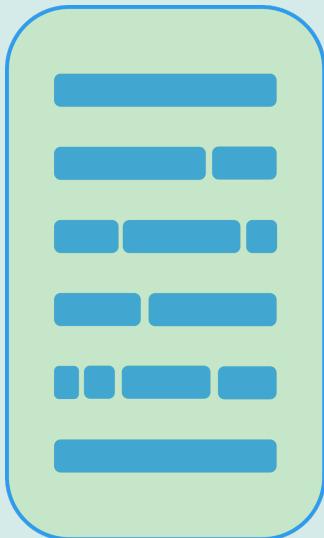
- Anwendung startet schneller:
 1. Browser bekommt **UI-Code** zur Darstellung
 2. Der **notwendige JavaScript-Code** wird nachgeladen
 3. Anwendung jetzt **interaktiv**

Fullstack-Anwendungen (JavaScript)



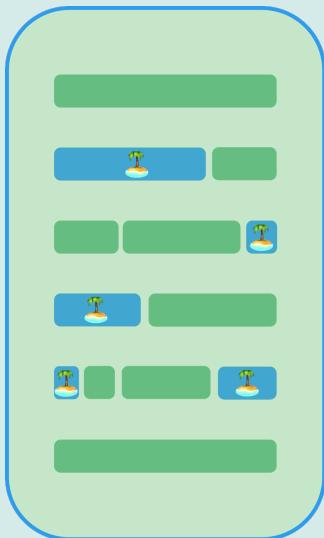
- Wir sind zurück zur **JavaScript-Schnipsel**-Architektur
 - aber: die Schnipsel werden **automatisch** vom Framework erzeugt
 - die Schnipsel existieren nur zur **Laufzeit**

Fullstack-Anwendungen (JavaScript)



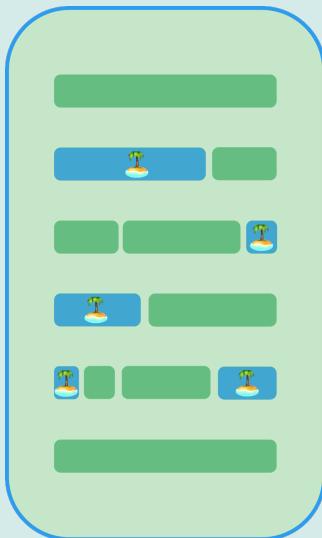
- Wir sind zurück zur **JavaScript-Schnipsel**-Architektur
 - aber: die Schnipsel werden **automatisch** vom Framework erzeugt
 - die Schnipsel existieren nur zur **Laufzeit**
 - In der **Entwicklung** ist "unser" Code aus "einem Guss"
 - aber weiterhin in **JavaScript**

Fullstack-Anwendungen (JavaScript)



- 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)



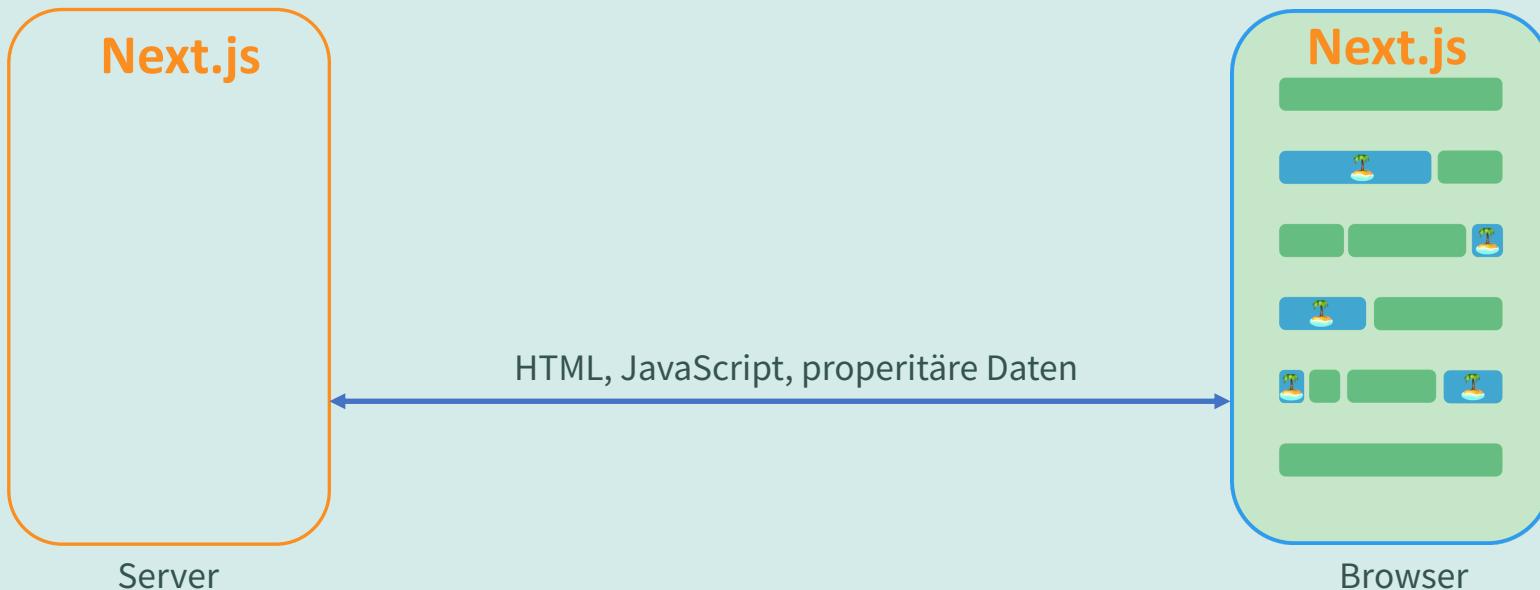
- Bekannte Vertreter:
 1. Next.js (React)
 2. SvelteKit (Svelte)
 3. Nuxt.js (Vue)
 4. Astro (eigenes Framework + Support für alle SPAs)
 5. Qwik (eigenes Framework)
- Funktionalität und Herangehensweise unterschiedlich

Beispiel: Next.js

EIN PAAR BEISPIELE...

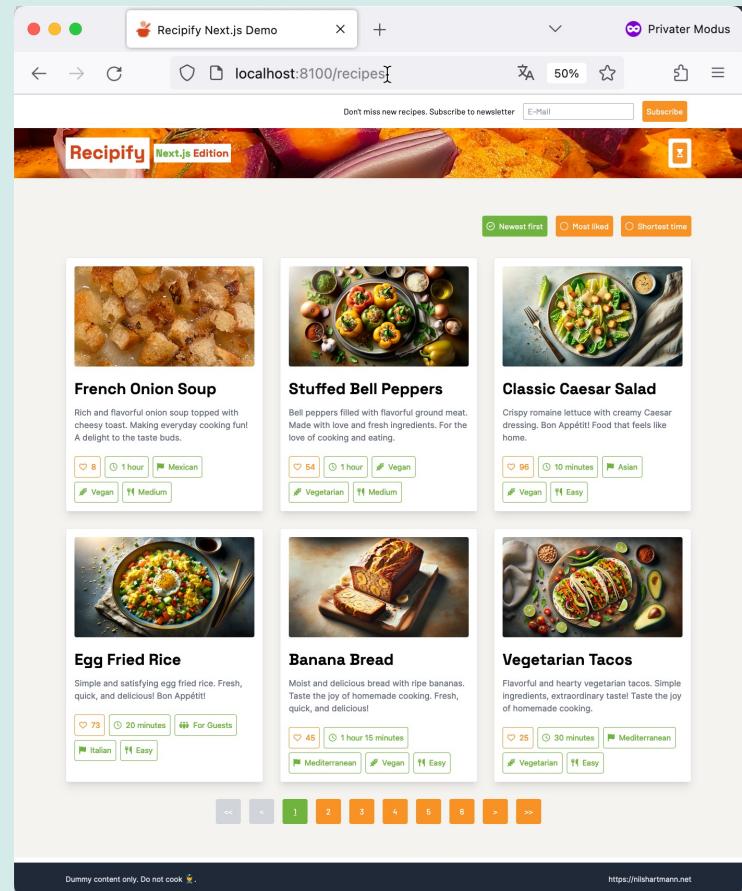
Next.js

- Basiert auf React
- Läuft im Client und im Server



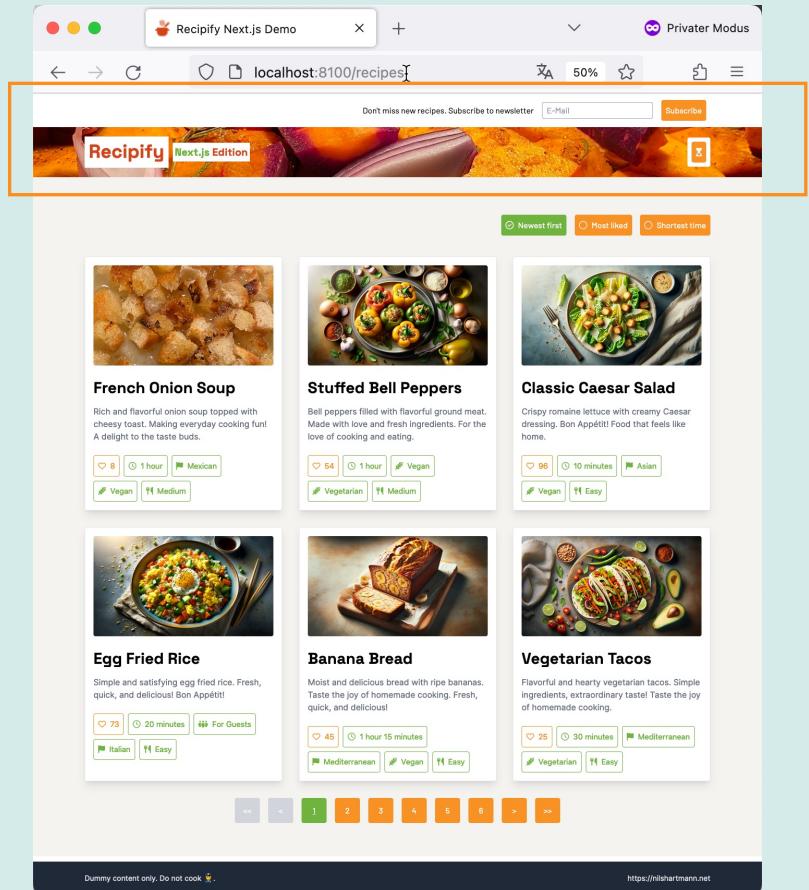
BEISPIEL: INITIALER SEITENAUFRUF

- **Beispiel: Seitenaufrufe**
- Demo: localhost:8100
- React-Komponente statt Controller
 - Aufgaben sind vergleichbar
- Seite kommt als HTML zurück
 - 🧑‍🍳 Vergleichbar mit klassisch serverseitig
 - 🧑‍🍳 Wo wird die Komponente ausgeführt?
 - 🧑‍🍳 Seitenwechsel mit PaginationBar
 - 🧑‍🍳 Open in new Tab
 - 🧑‍🍳 Was passiert ohne JavaScript?



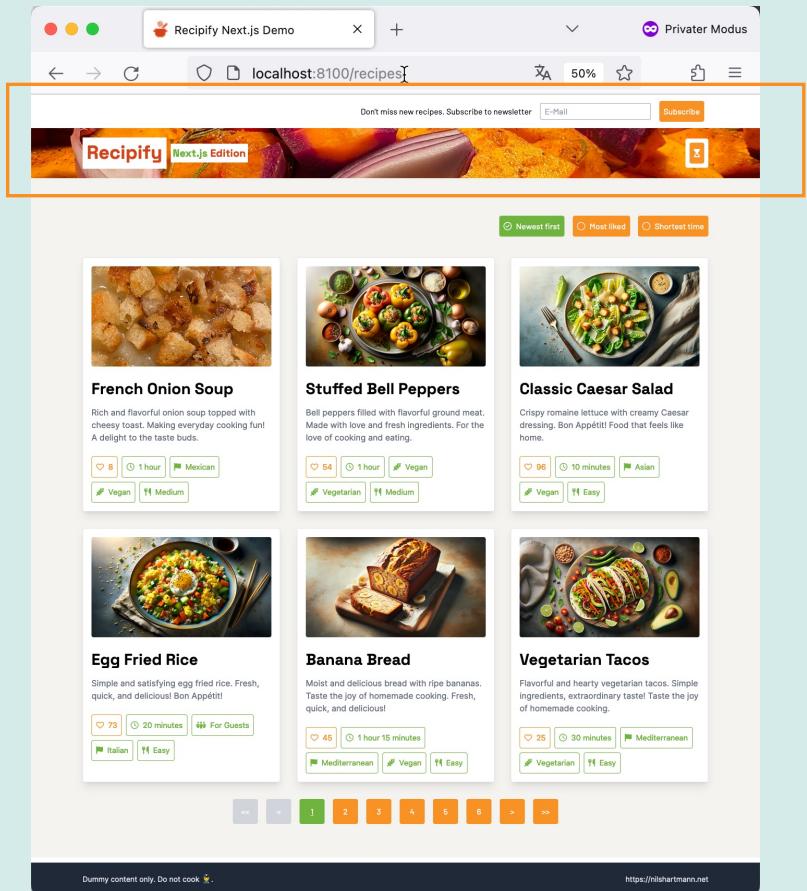
BEISPIEL: INITIALER SEITENAUFRUF

- **Beispiel: Layout**
- Wiederverwendbare Rahmen
 - 🧑 Timer-Komponente im Header
 - 🧑 Was passiert beim Seitenwechsel
 - 🧑 Wie würden wir das "klassisch" machen?



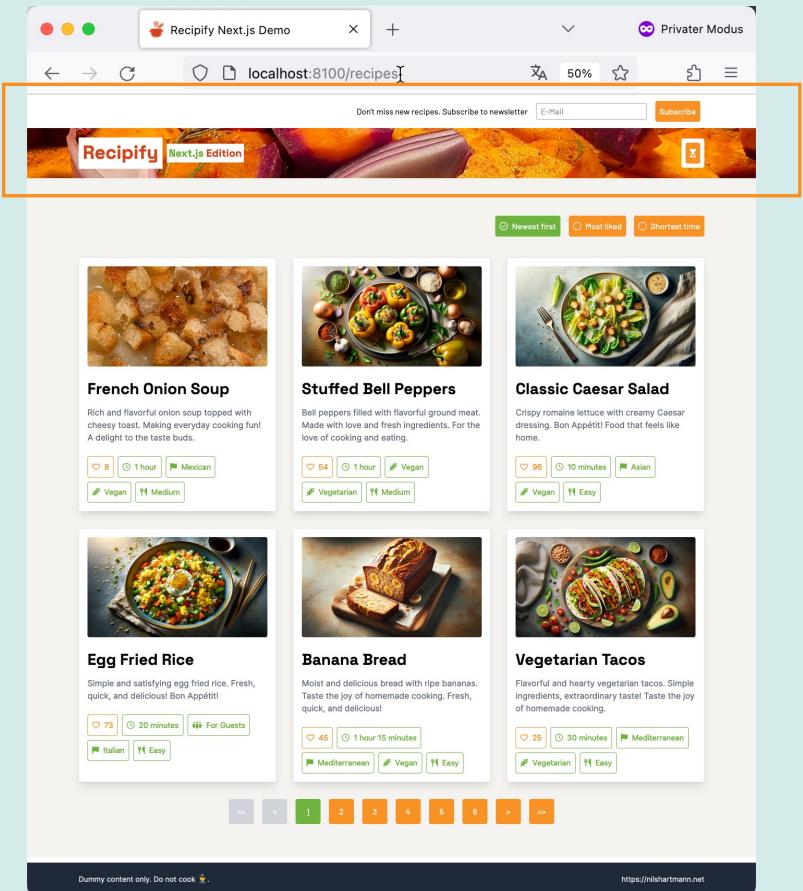
BEISPIEL: INITIALER SEITENAUFRUF

- Beispiel: Interaktion
- Zustand bleibt erhalten
 - LikeButton
 - Server-Action (impliziter Endpunkt)
 - Was passiert beim Submit mit dem Timer?
 - Was passiert ohne JS?



BEISPIEL: INITIALER SEITENAUFRUF

- **Beispiel: Interaktion #2**
- Portionszähler
 - Reine Client-Information, Berechnung soll nur auf dem Client stattfinden
 - Wieder ähnlich wie bei klassischem Ansatz:
 - kein fetch etc.
 - Templates aus einem Guß
 - aber: Logik wird jetzt auch im Client ausgeführt



FAZIT

- Alles Fullstack oder was?

FAZIT

- Alles Fullstack oder was?
- Nö.

FAZIT

- Alles Fullstack oder was?
- Nö.
- Aber interessante Alternative

FAZIT

- Alles Fullstack oder was?
- Nö.
- Aber interessante Alternative
- Wie immer: es kommt auf die Anforderung drauf an 😎

Vielen Dank!

Slides: <https://react.schule/fullstack-oder-was>

Fragen & Kontakt: nils@nilshartmann.net

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