

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**



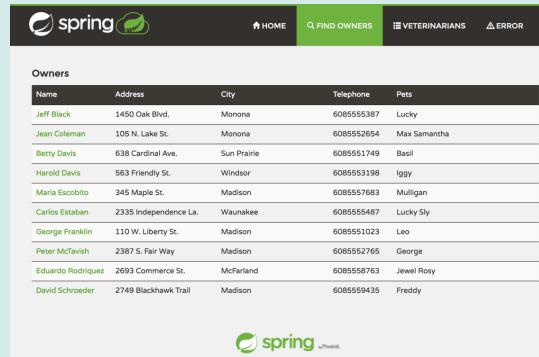
<https://graphql.schule/video-kurs>

<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 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.	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

# 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 Basel SBB. It includes details like speed (108 km/h), arrival time (12:46), and a promotional offer for Genius Prämien.
- ICE Portal**: A real-time train information system for the ICE 109. It shows the next stop at Gleis 2 in Osnabrück Hbf, with an arrival time of 12:35 and a red "12:46" button.
- hw plus**: An event management platform for the Sasha-Show. It lists attendees, including Jeff Black, Betty Davis, and Harold Davis, along with their addresses and phone numbers.
- Spring**: A database application showing a list of pet owners and their pets. The table includes 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	608551749	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	608551023	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 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, showing a list of recent activities.
- sessionize**: A speaker dashboard for Nils Hartmann, showing session details and a public profile.
- spring**: A GitHub repository for "g-graphql-training" by Nils Hartmann, displaying code, issues, and pull requests.

# 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**: 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 showing a list of tasks and projects, including "Hibernate ORM Softwareprojekt".
- sessionize**: A platform for organizing events, showing a speaker dashboard for Nils Hartmann.
- Microsoft Teams**: A communication and collaboration tool showing a chat history between users.
- Spring**: A developer-oriented platform showing a list of owners with columns for Name, Address, City, Telephone, and Pets.

# WEBANWENDUNGEN

# Die Webanwendung gibt es nicht

The image is a collage of several screenshots from various web applications, demonstrating a wide range of software tools and services:

- ICE Portal:** A travel-related portal showing a map and flight information. It includes a speedometer icon indicating 108 km/h and a button labeled "Karte".
- Booking.com:** A screenshot of the Booking.com homepage.
- ChatGPT - DALL-E:** A conversation with AI models. One message shows two images of a beef burger, with the caption: "Here are two teaser images for a cooking recipe website featuring a "Classic Beef Burger". These images are designed to appeal to a young, modern urban audience."
- Hibernate ORM:** A screenshot of a project management interface for the Hibernate ORM software.
- sessionize:** A user profile page for Nils Hartmann on sessionize.com, showing basic details like name and a public profile link.
- Spring:** A screenshot of a Spring framework application showing a table of owners with columns for Name, Address, City, Telephone, and Pets.
- Google Sheets:** A view of a Google Sheets document with multiple sheets visible on the left.
- GitHub:** A screenshot of a GitHub repository page for "Hibernate ORM" showing commit history and pull requests.
- LinkedIn:** A screenshot of a LinkedIn inbox showing several messages and notifications.

# WEBANWENDUNGEN

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

Booking.com

ICE Portal

ChatGPT - DALL-E

hvv plus

DALL-E

Spring

GitHub

Microsoft To Do

# 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)

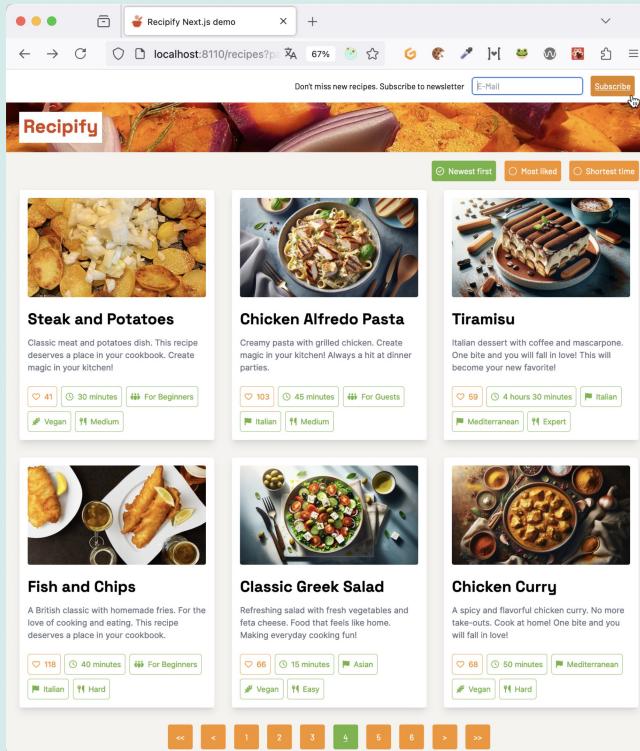
# WEBANWENDUNGEN

These: ihr braucht eher JS, als ihr denkt

The image is a collage of several web browser screenshots demonstrating various web applications:

- Booking.com**: A travel booking site showing flight information for an ICE 109 train from Osnabrück Hbf to Gleis 2.
- ICE Portal**: A real-time train status board showing the next stop in 6 minutes at Gleis 2.
- Hibernate**: A project management tool showing tasks and progress for a "Hibernate ORM Softwareprojekt".
- Microsoft Teams**: A communication platform showing a team channel and a message from Florian Hartmann.
- GitHub**: A screenshot of a GitHub 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 displays a table of owners with columns for Name, Address, City, Telephone, and Pets.
- Eventbrite**: A screenshot of an event page for "Frontend for Backend: HTMX oder Single-Page-Anwendung?".

# 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?**

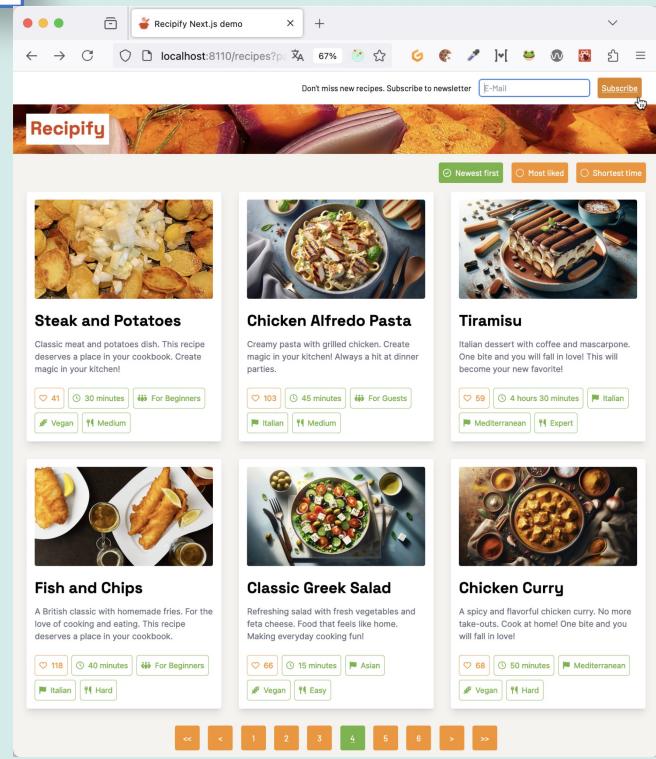
*Klassische serverseitige Anwendung...*

### ***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:

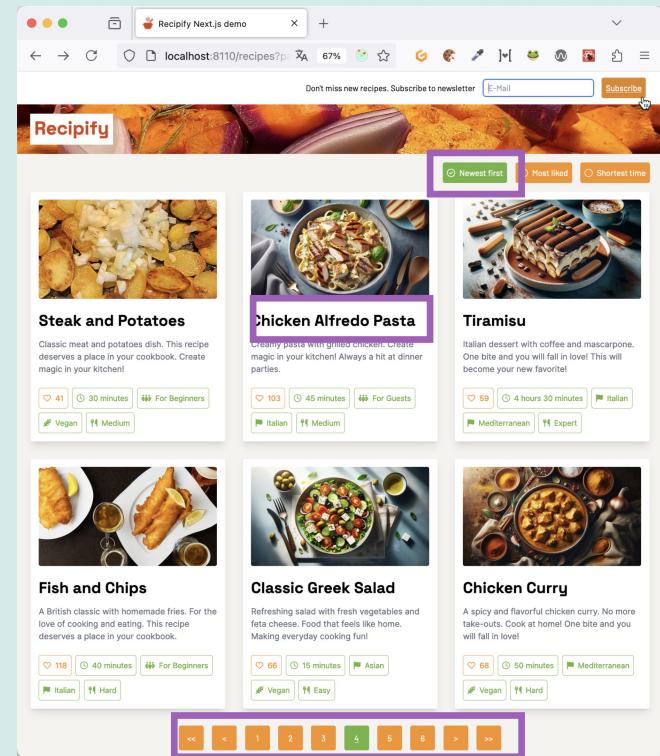
- Steak and Potatoes**: A dish of meat and potatoes. Details: 41 likes, 30 minutes, For Beginners, Vegan, Medium.
- Chicken Alfredo Pasta**: A bowl of pasta with chicken. Details: 103 likes, 45 minutes, For Guests, Italian, Medium.
- Tiramisu**: A dessert with layers of cake and coffee. Details: 59 likes, 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 salad with various toppings. Details: 66 likes, 15 minutes, Asian, Vegan, Easy.
- Chicken Curry**: A bowl of 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.

# WEBANWENDUNGEN

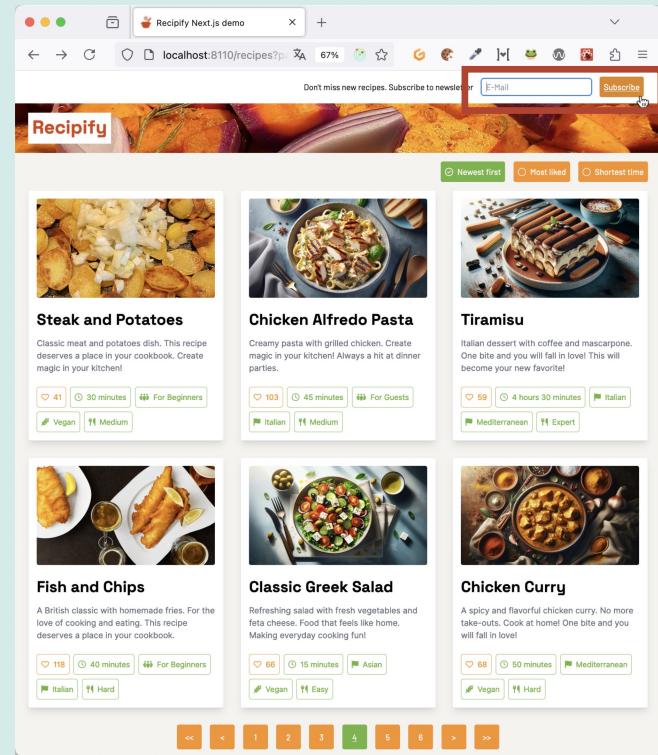
*...die ist aber komplett statisch*

- Interaktion per Link



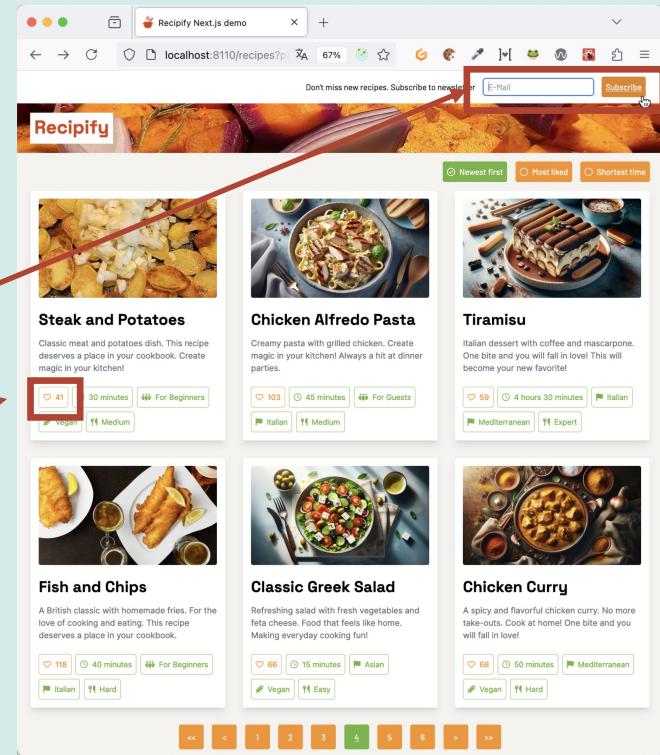
*...die ist aber komplett statisch*

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



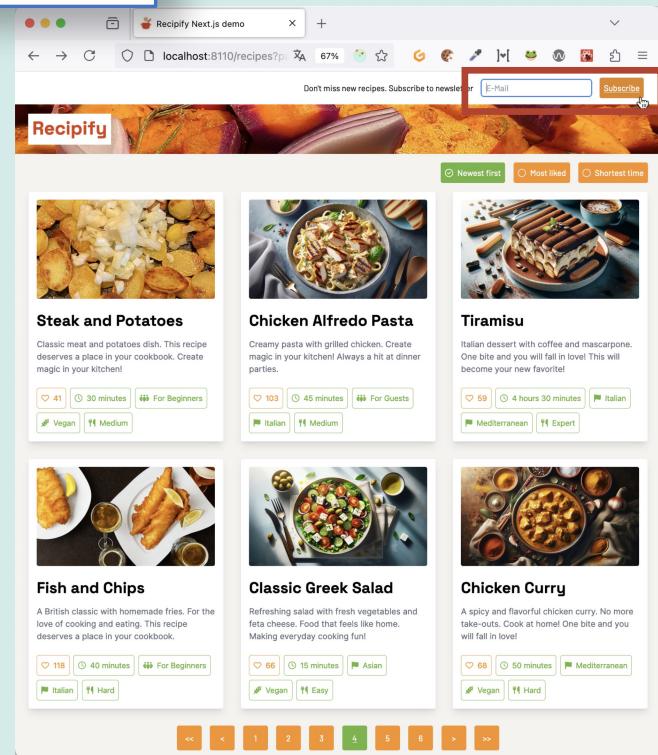
*...die ist aber komplett statisch*

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



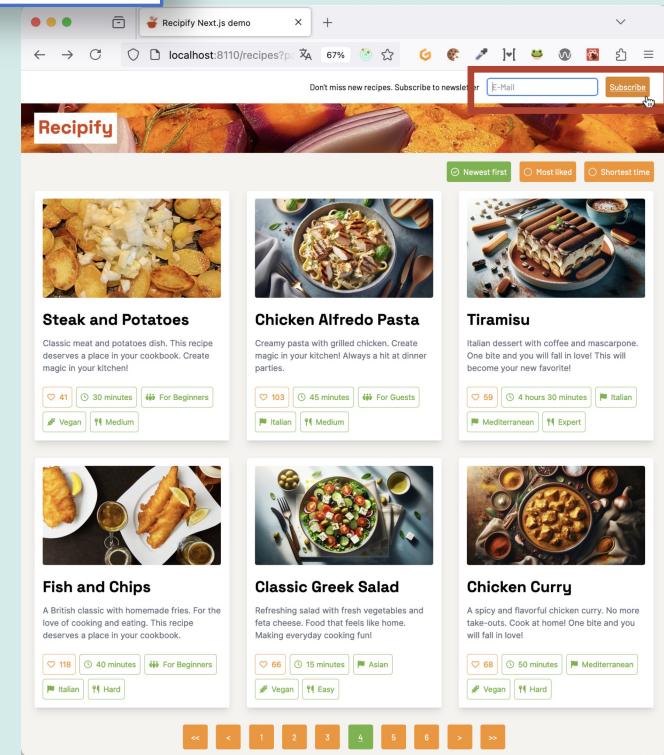
**Bei jeder Interaktion: ein Server-Roundtrip...**

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



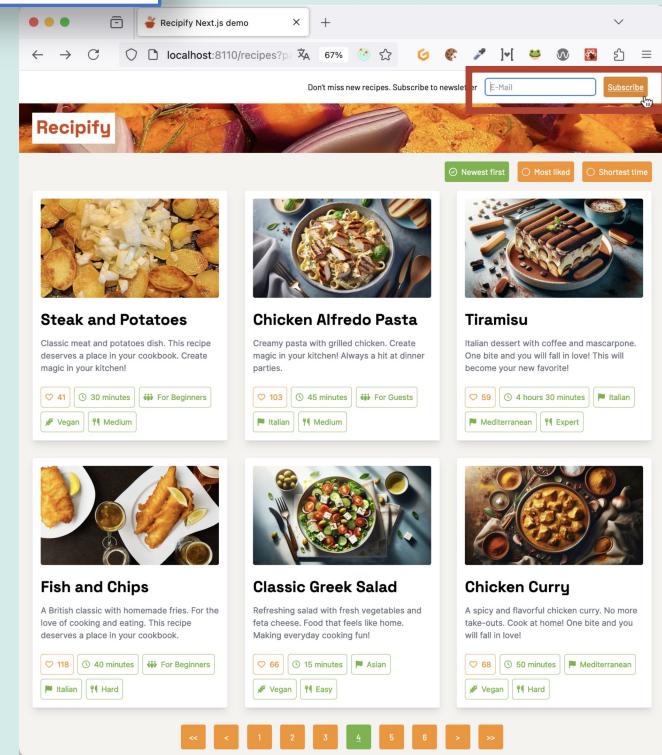
## Bei jeder Interaktion: ein Server-Roundtrip...

- 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:



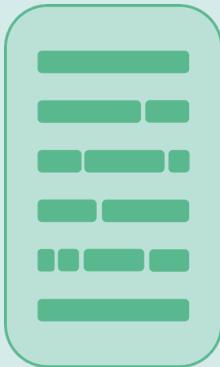
Your comment:

Lecker...!

10/500 characters

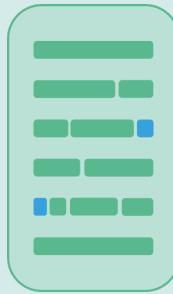
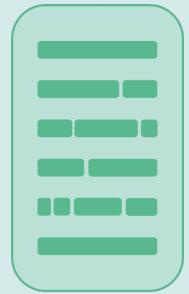
Submit Rating

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



HTML-Seite

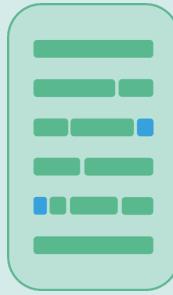
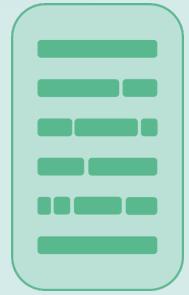
Ja, das geht!



HTML-Seite

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

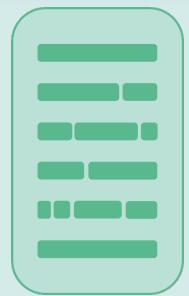
Ja, das geht!



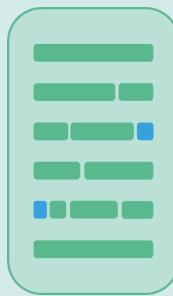
HTML-Seite

- 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..."**

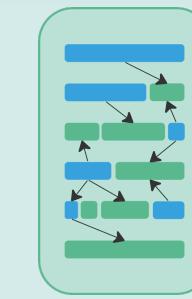
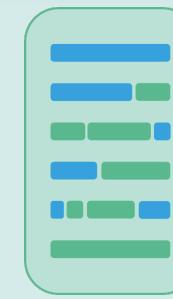
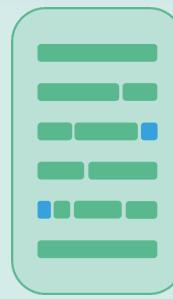
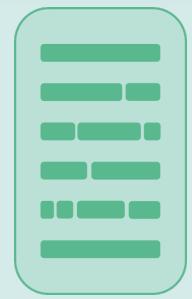


HTML-Seite



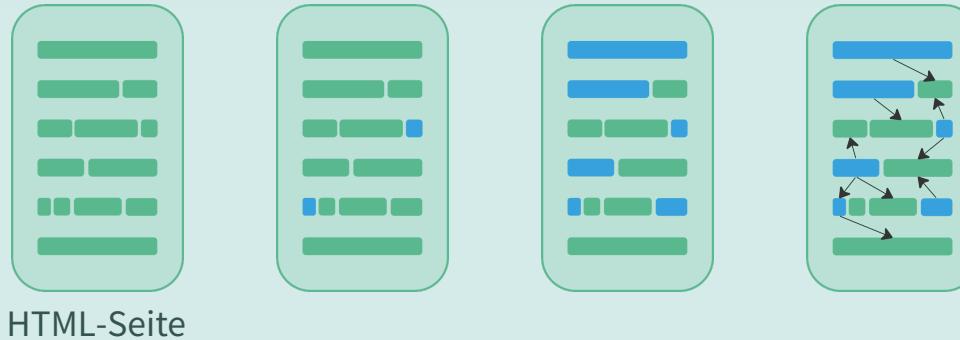
- Wir schreiben also noch etwas mehr JavaScript Schnipsel

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



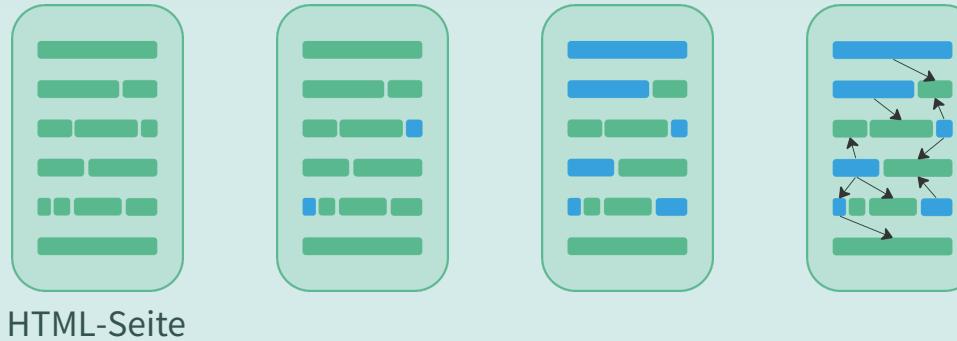
HTML-Seite

### 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

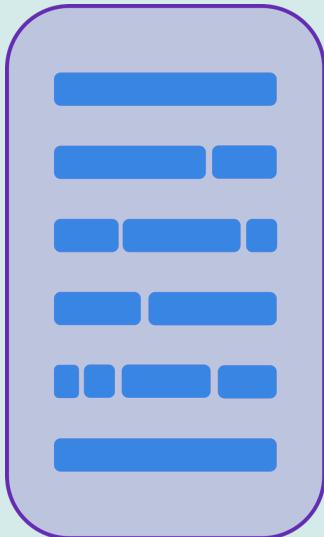
### *Dann besser alles in JavaScript? Single-Page-Anwendungen*



JavaScript-Code

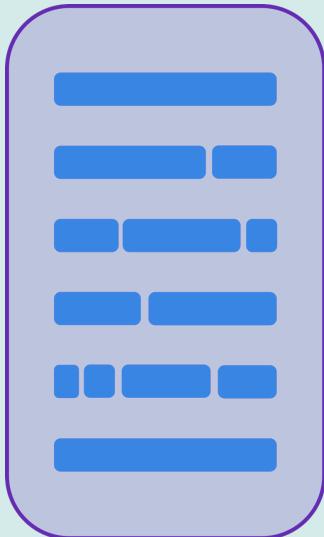
- ab ca. 2010
- Aus "Seiten" werden jetzt "Anwendungen"
- Klare Verantwortlichkeit: Server für Logik und Daten, Browser für UI
- Es gibt stabile und verbreitete Frameworks für jeden Geschmack

### Single-Page-Anwendungen



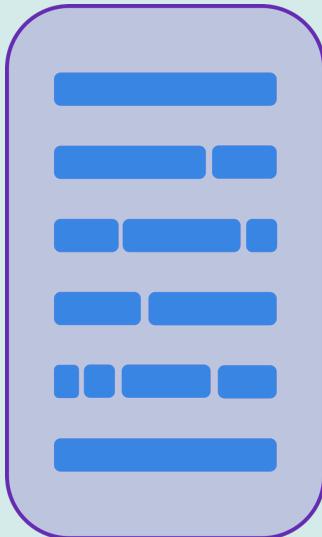
- Darstellung erfolgt vollständig mit JavaScript
- Statisches HTML spielt (fast) keine Rolle
- Die Anwendung kommuniziert mit dem Backend über API
- Ausgetauscht werden Daten, aber keine UI
- Vertreter: Angular, React, Svelte, Vue

### Single-Page-Anwendungen



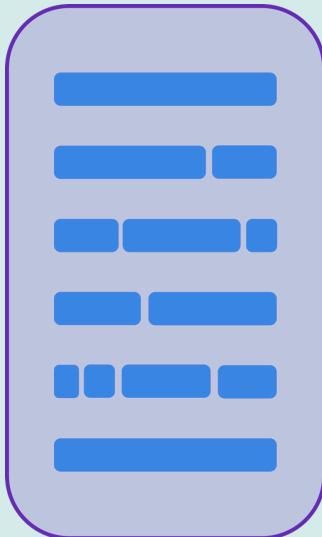
- Konsequenz #1: (viel) JavaScript zur **Entwicklungszeit**

### Single-Page-Anwendungen



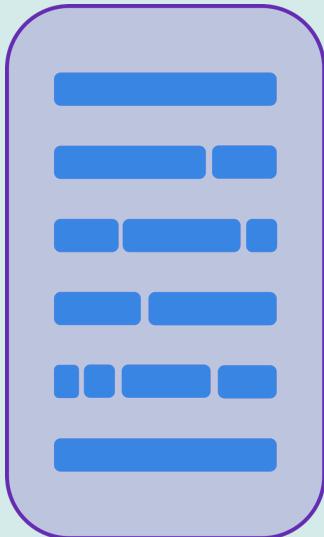
- Konsequenz #1: (viel) JavaScript zur Entwicklungszeit
- Konsequenz #2: (viel) JavaScript zur Laufzeit

### Single-Page-Anwendungen



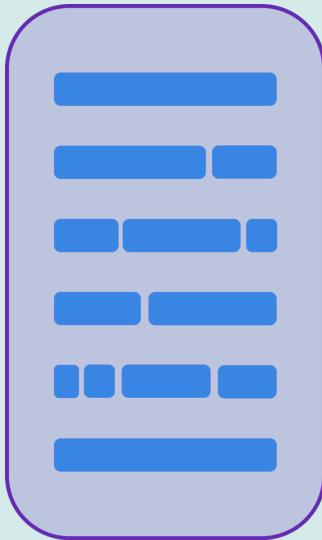
- Konsequenz #2: viel JavaScript **zur Laufzeit**
- Auch für **statische Inhalte**

### Single-Page-Anwendungen



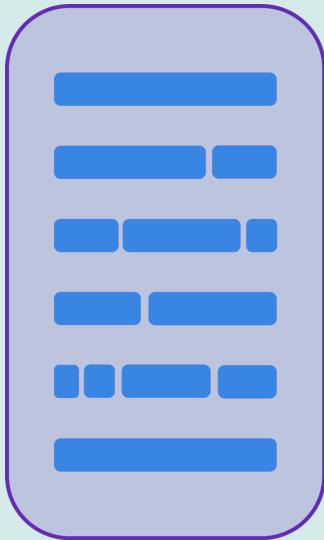
- JavaScript-Code:
  1. muss zum Browser gesendet werden
  2. muss vom Browser ausgeführt werden
  3. kann dann die darzustellenden Daten lesen
  4. kann dann erst die Daten anzeigen
  5. erst dann ist die Anwendung einsatzbereit
  6. Mit jedem Feature wird es mehr

### Single-Page-Anwendungen



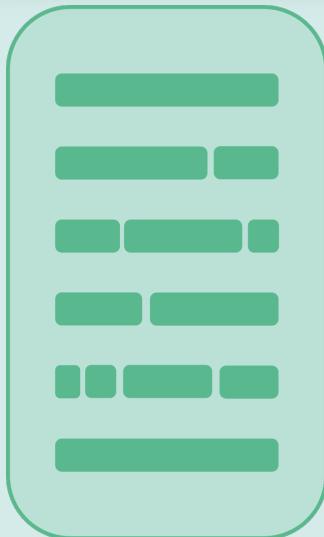
- Das hat Auswirkungen auf die **Laufzeit**-Performance
- Insbesondere beim Starten

### Single-Page-Anwendungen

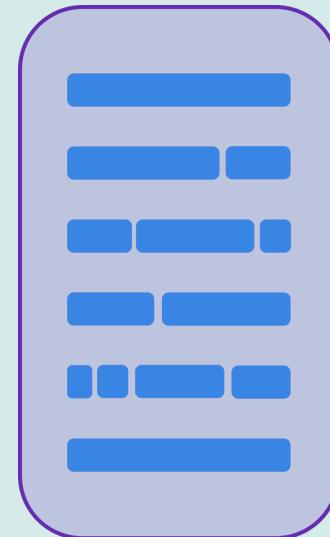


- Das hat Auswirkungen auf die **Laufzeit**-Performance
- Insbesondere beim Starten
- Ob das ein Problem für die eigene Anwendung ist, muss man von Fall zu Fall entscheiden
  - In-House- oder B2B-Anwendungen haben andere Anforderungen als ein Online-Shop

**"Gibt es denn nichts dazwischen"?**



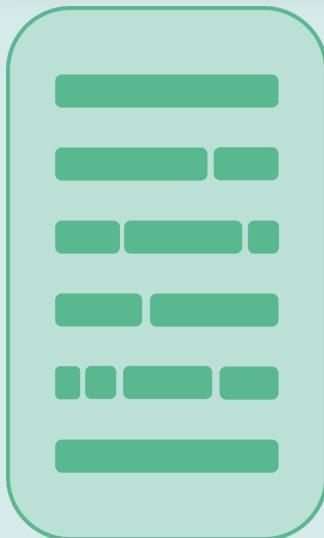
Nur Server (+JS)



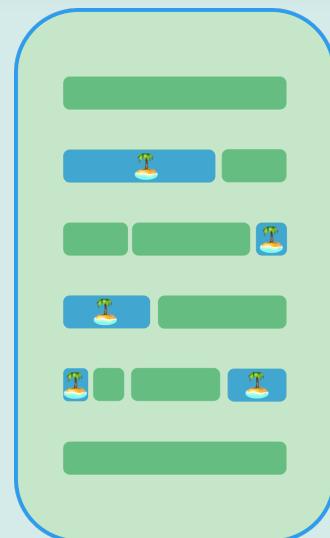
Nur Client (SPA)

# WEBANWENDUNGEN

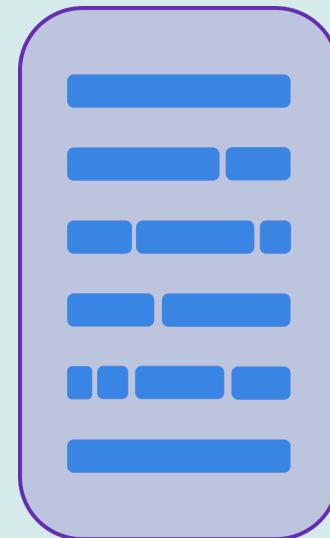
"Doch – Fullstack-Anwendungen 😊"



Nur Server (+JS)



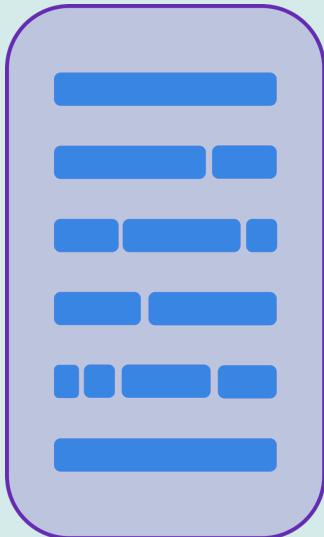
JS Fullstack Anwendung



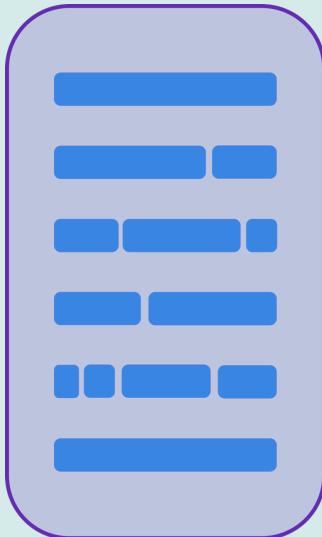
Nur Client (SPA)

# FULLSTACK-ANWENDUNGEN (JAVASCRIPT)

## Fullstack-Anwendungen (JavaScript)

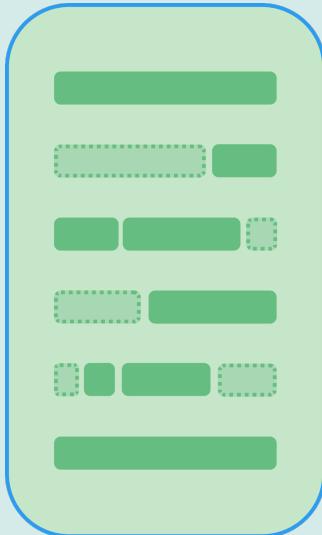


## Fullstack-Anwendungen (JavaScript)



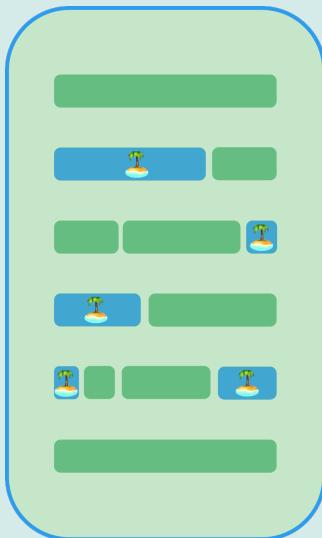
- Ebenfalls vollständig in **JavaScript** geschrieben

## Fullstack-Anwendungen (JavaScript)



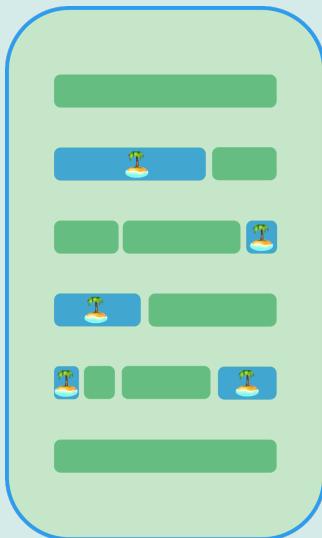
- Grundsätzliche Idee:
  1. **UI-Code** wird serverseitig vorgerendert
  2. **UI-Code** wird zum Browser gesendet und angezeigt

## Fullstack-Anwendungen (JavaScript)



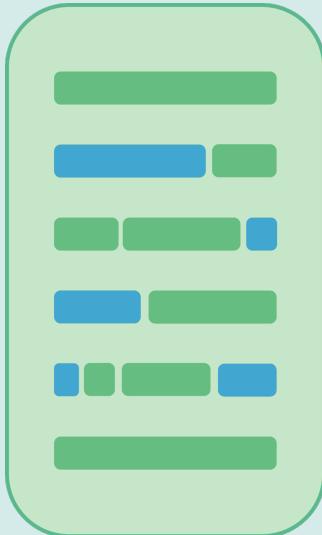
- Grundsätzliche Idee:
  1. **UI-Code** wird serverseitig vorgerendert
  2. **UI-Code** wird zum Browser gesendet und angezeigt
  3. Nur der JavaScript-Code ("Islands") **für Interaktionen** wird zum Browser geschickt

## Fullstack-Anwendungen (JavaScript)



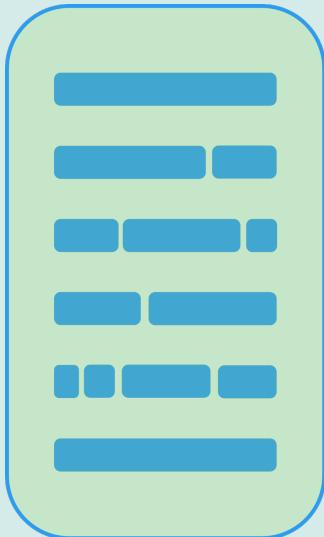
- Anwendung startet schneller:
  1. Browser bekommt **UI-Code** zur Darstellung
  2. Der **notwendige JavaScript-Code** wird nachgeladen
  3. Anwendung jetzt **interaktiv**

## Fullstack-Anwendungen (JavaScript)



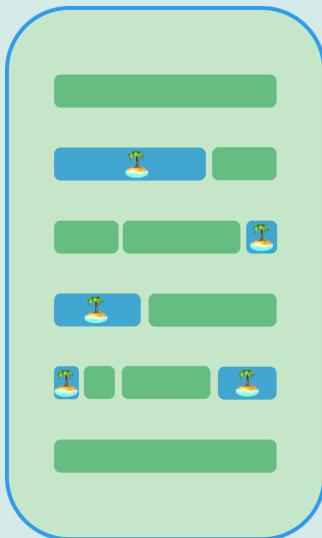
- Wir sind zurück zur **JavaScript-Schnipsel**-Architektur
  - aber: die Schnipsel werden **automatisch** vom Framework erzeugt
  - die Schnipsel existieren nur zur **Laufzeit**

## Fullstack-Anwendungen (JavaScript)



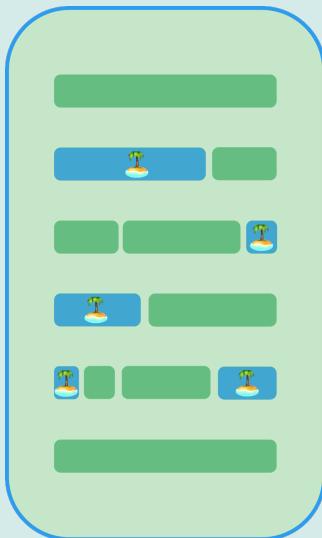
- Wir sind zurück zur **JavaScript-Schnipsel-Architektur**
  - aber: die Schnipsel werden **automatisch** vom Framework erzeugt
  - die Schnipsel existieren nur zur **Laufzeit**
  - In der **Entwicklung** ist "unser" Code aus "einem Guss"
  - aber weiterhin in **JavaScript**

## 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)



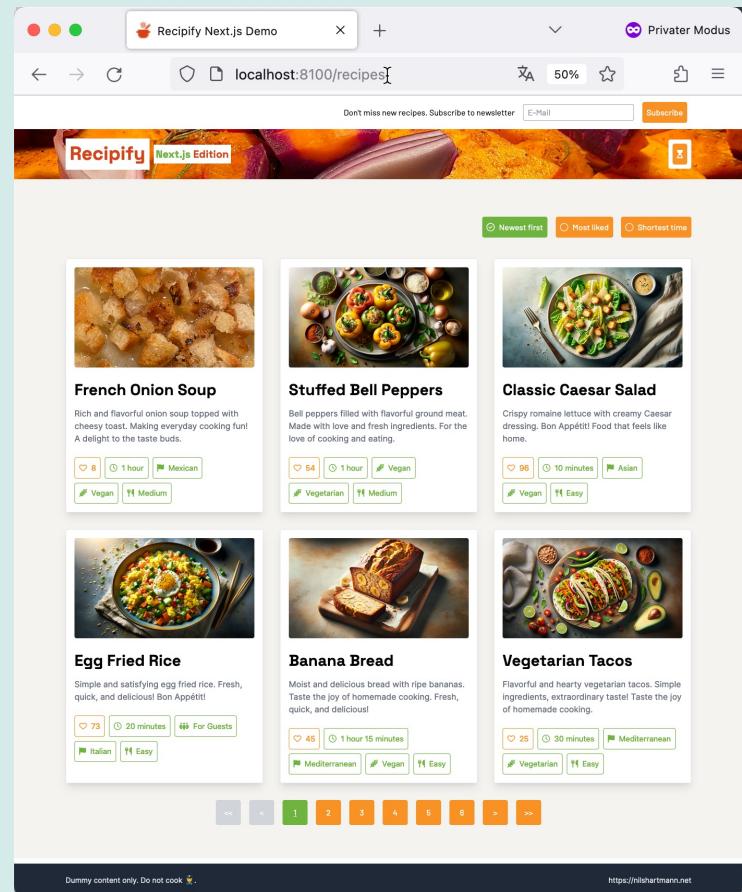
- Bekannte Vertreter:
  1. Next.js (React)
  2. SvelteKit (Svelte)
  3. Nuxt.js (Vue)
  4. Astro (eigenes Framework + Support für alle SPAs)
  5. Qwik (eigenes Framework)
- Funktionalität und Herangehensweise unterschiedlich

# Beispiel: Next.js

EIN PAAR BEISPIELE...

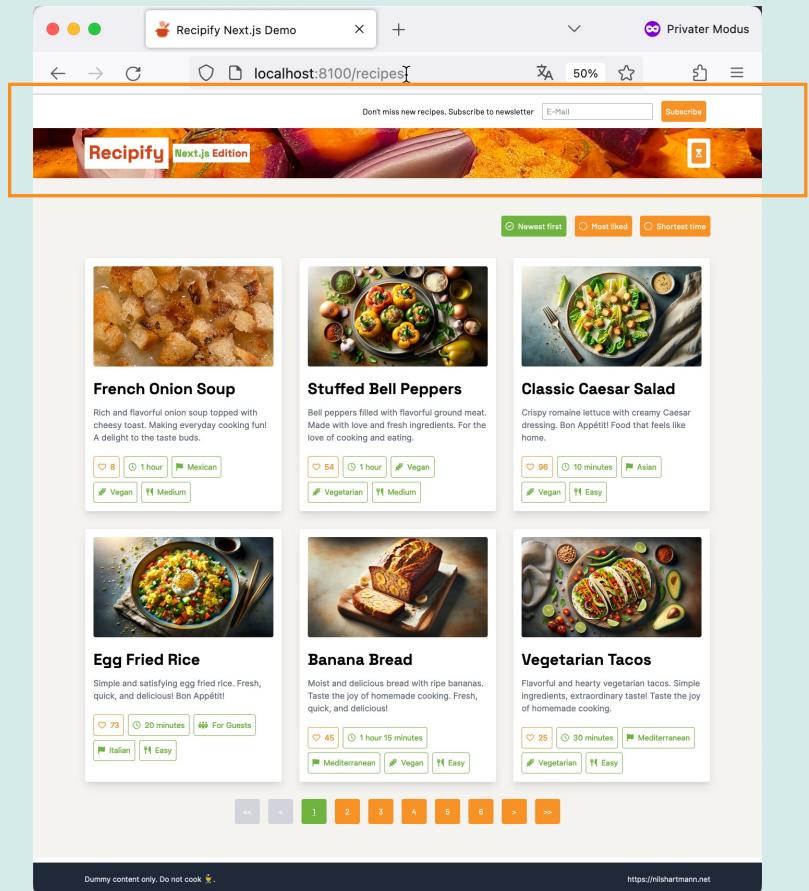
## BEISPIEL: INITIALER SEITENAUFRUF

- **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?



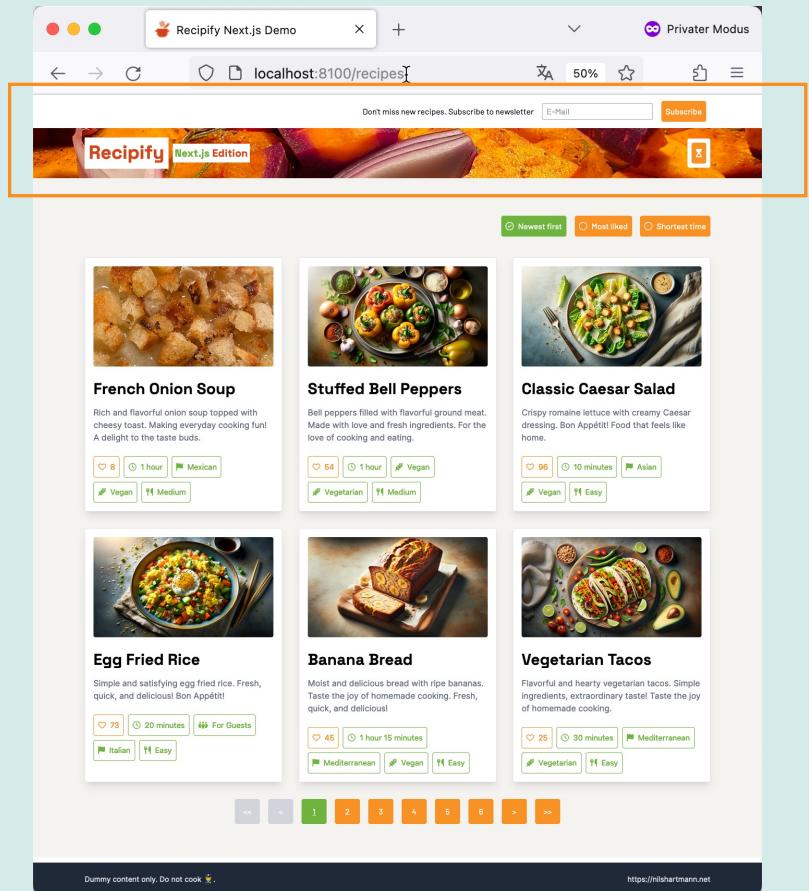
# BEISPIEL: INITIALER SEITENAUFRUF

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



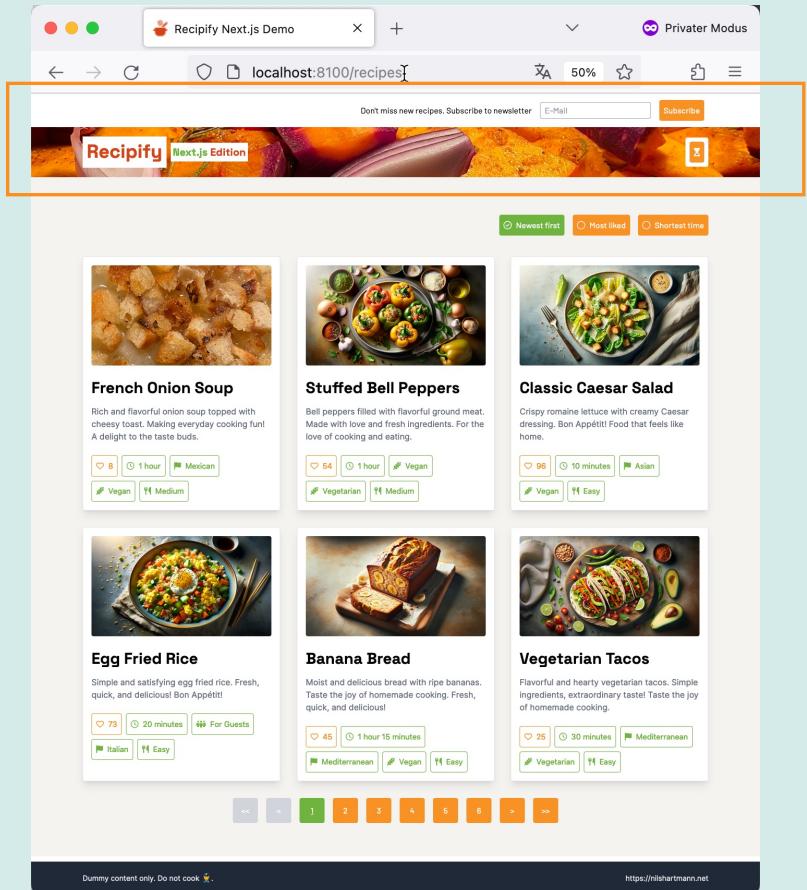
# BEISPIEL: INITIALER SEITENAUFRUF

- **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](mailto:nils@nilshartmann.net)

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