



NILS HARTMANN

<https://nilshartmann.net>

Das Backend im Frontend

Next.js

für Java-Entwickler:innen

NILS HARTMANN

nils@nilshartmann.net

Freiberuflicher Software-Entwickler, –Architekt, Coach, Trainer

Java, Spring, GraphQL, React, TypeScript



Video-Kurs

Tutorial ►

Modern React: von Null auf Full-
Stack



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

<https://reactbuch.de>

HTTPS://NILSHARTMANN.NET

EIN BEISPIEL...

<http://localhost:8110>

The screenshot shows a web browser window with the title "Recipify Next.js demo". The URL in the address bar is "localhost:8110/recipes". The page features a header with a "Recipify" logo, a newsletter subscription form, and a "Subscribe" button. Below the header is a banner with a photo of a dish and a call-to-action button. The main content area displays three recipe cards:

- French Onion Soup**: A bowl of soup with croutons and herbs. Description: "Rich and flavorful onion soup topped with cheesy toast. Making everyday cooking fun! A delight to the taste buds." Icons: 5 hearts (orange), 1 hour (blue), Mexican (green). Tags: Vegan (green), Medium (green).
- Stuffed Bell Peppers**: A plate of bell peppers filled with meat and vegetables. Description: "Bell peppers filled with flavorful ground meat. Made with love and fresh ingredients. For the love of cooking and eating." Icons: 65 hearts (orange), 1 hour (blue), Vegan (green). Tags: Vegetarian (green), Medium (green).
- Classic Caesar Salad**: A bowl of salad with dressing and croutons. Description: "Crispy romaine lettuce with creamy Caesar dressing. Bon Appétit! Food that feels like home." Icons: 92 hearts (orange), 10 minutes (blue), Asian (green). Tags: Vegan (green), Easy (green).

At the bottom of the page, there is a navigation bar with buttons for page navigation: <<, <, 1 (green), 2, 3, 4, 5, 6, >, >>. The footer contains the text "Dummy content only. Do not cook 😊." and the URL "https://nilshartmann.net".

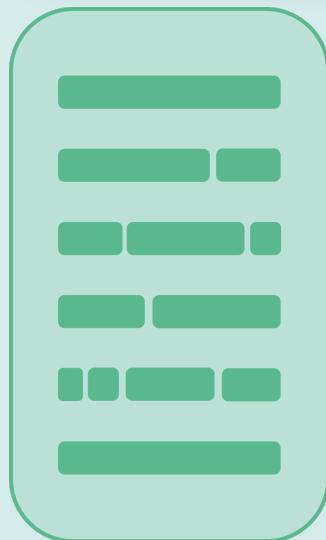
EIN BEISPIEL...

Ein Beispiel...

- Brauchen wir dafür JavaScript im Browser? React?

Architekturen für Webanwendungen

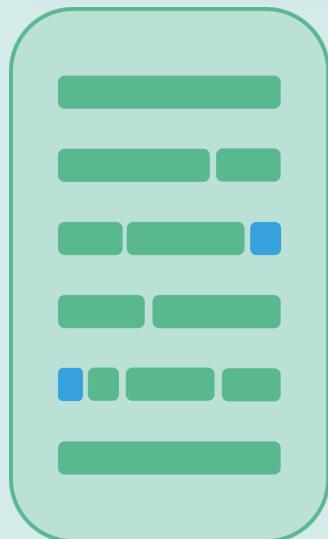
"Klassisch": Serverseitiges Rendern



HTML-Seite
mit
JavaScript

- Server liefert fertige **HTML-Seite** zum Client
- Für jede Interaktion dann Server-Roundtrip (Link, Formular)
- Typische Vertreter: PHP, Java, dotNET, ...

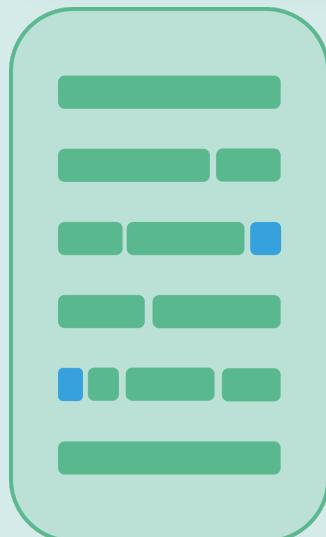
"Klassisch": Serverseitiges Rendern plus JavaScript "Schnipsel"



- Für feingranulare Interaktion wird **JavaScript** eingestreut

HTML-Seite
mit
JavaScript

"Klassisch": Serverseitiges Rendern plus JavaScript "Schnipsel"



HTML-Seite
mit
JavaScript

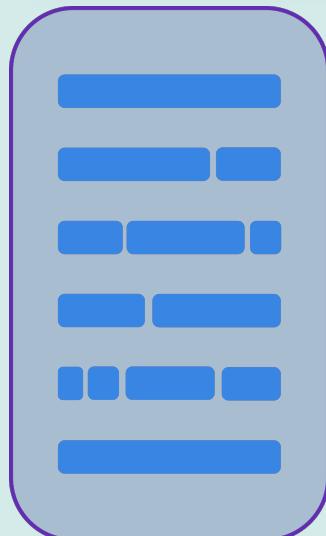
- Für feingranulare Interaktion wird **JavaScript** eingestreut
- Eigentlich optimal:
 - wir haben **JavaScript** nur da, wo wir es **wirklich** brauchen (für Interaktivität)
 - alles andere kann statisches HTML und CSS sein ❤️

Konsequenz Serverseitiges Rendern plus JavaScript "Schnipsel"



- Bunter Strauß an Server- und Client-Technologien (Backend-Sprache, Template-Sprache, JavaScript)

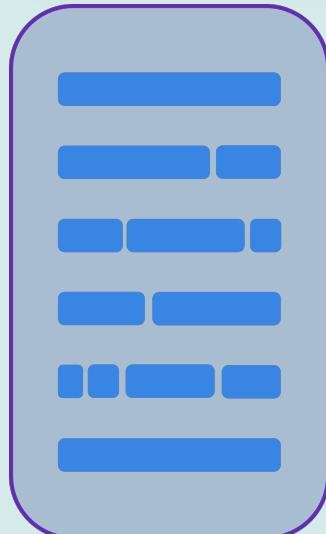
Single-Page-Anwendungen



JavaScript
im
Browser

- Vollständig in **JavaScript** implementiert
- Ausgeführt komplett **im Browser**
- Statisches HTML spielt (fast) keine Rolle

Konsequenz Single-Page-Anwendungen

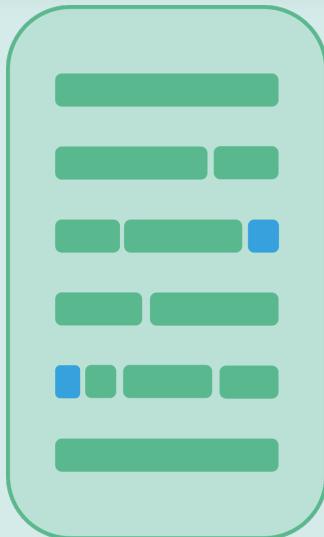


JavaScript
im
Browser

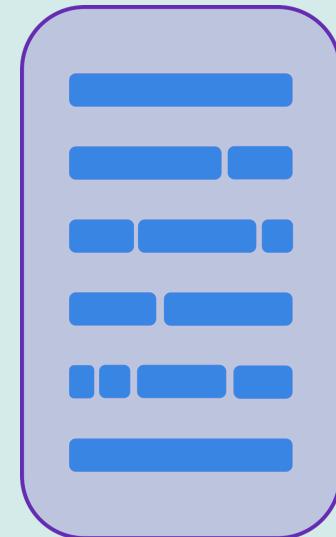
- Sehr viel **JavaScript** **zur Laufzeit**
- Auch für **statische** Inhalte
- Das JavaScript muss laufen, bevor etwas dargestellt wird
- 🚫 Die Anwendung funktioniert nicht ohne JavaScript

„Fullstack“ mit Next.js

„Fullstack“ mit Next.js

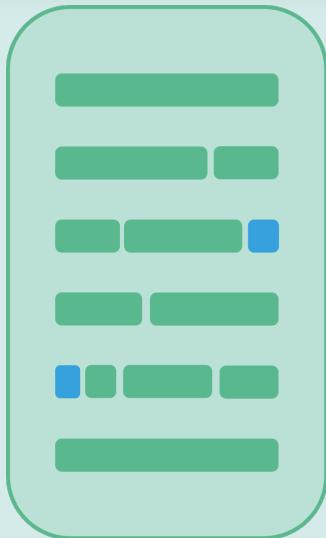


Klassisch
(im Browser: HTML+**JS**)

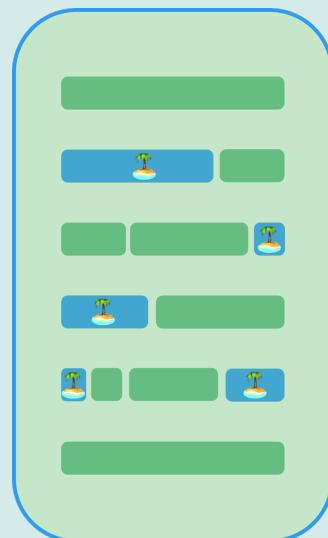


Single-Page-App
(im Browser: nur **JS**)

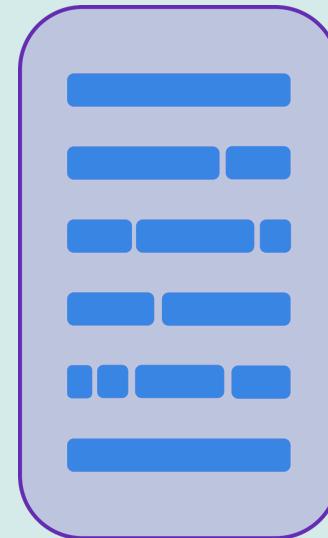
„Fullstack“ mit Next.js



Klassisch
(im Browser: HTML+**JS**)

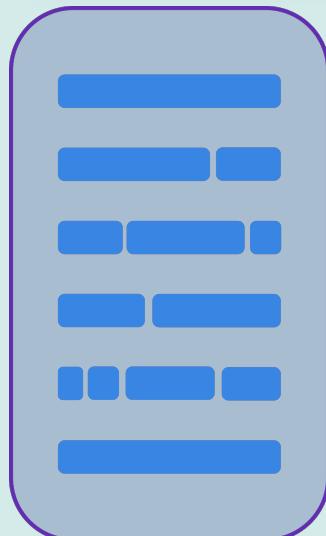


Next.js-Anwendung
(im Browser: HTML+**JS**)



Single-Page-App
(im Browser: nur **JS**)

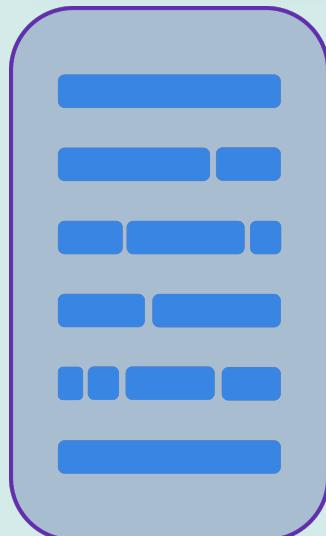
Next.js-Anwendungen



JavaScript
auf dem Server

- Vollständig in **JavaScript entwickelt**

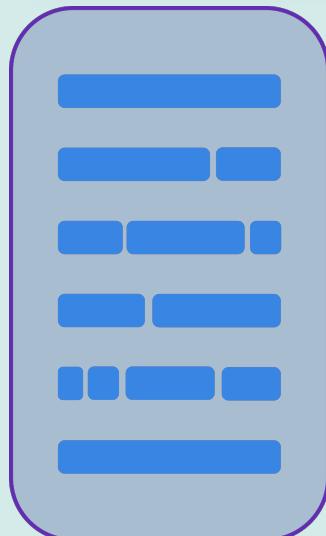
Next.js-Anwendungen



JavaScript
auf dem Server

- Vollständig in **JavaScript entwickelt**
- Erzeugt **HTML** auf dem **Server**

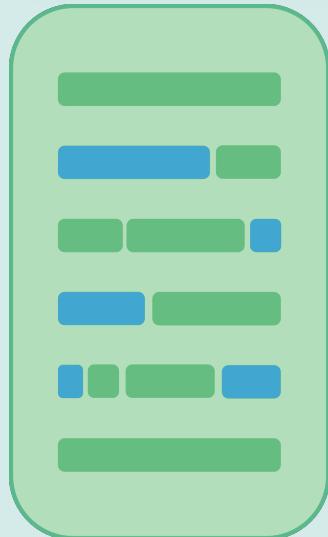
Next.js-Anwendungen



JavaScript
auf dem Server

- Vollständig in **JavaScript entwickelt**
- Erzeugt **HTML** auf dem **Server**
- Nur der **JavaScript-Code** für **Interaktionen** wird zum **Browser** geschickt

Next.js-Anwendungen



- Wir sind zurück zur **JavaScript-Schnipsel-Architektur**
 - aber: die Schnipsel werden von Next.js erzeugt
 - die Schnipsel existieren nur zur **Laufzeit**
 - In der **Entwicklung** eine Code-Basis



Bestes aus beiden Welten?

im Browser:
HTML+**JS**

NEXT.JS

<https://nextjs.org>

<https://nextjs.org>

- basiert auf React

<https://nextjs.org>

- basiert auf React
- fügt serverseitiges Rendern, Caching und Routing hinzu

<https://nextjs.org>

- basiert auf React
- fügt serverseitiges Rendern, Caching und Routing hinzu
- ähnlich wie Spring Boot, das z.B. Webserver + JSP/Thymeleaf integriert

<https://nextjs.org>

- basiert auf React
- fügt serverseitiges Rendern, Caching und Routing hinzu
- ähnlich wie Spring Boot, das z.B. Webserver + JSP/Thymeleaf integriert
- Typische Einsatzgebiete:
 - Eher Webseiten als Webanwendungen
 - Zum Beispiel eCommerce, Nachrichten- und Produktseiten

Live

Demo



[HTTP://LOCALHOST:8100](http://localhost:8100)

Vielen Dank!

Code: <https://react.schule/wjax2025-nextjs>

Fragen und Kontakt:

nils@nilshartmann.net

<https://nilshartmann.net/kontakt>

