

Keine Angst vor der

Single-Page- Anwendung

Slides: <https://nils.buzz/oose-keine-angst>

NILS HARTMANN

nils@nilshartmann.net

Freiberuflischer Entwickler, Architekt, Trainer aus Hamburg

Java
JavaScript, TypeScript
React
GraphQL

Trainings, Workshops und
Beratung



2. Auflage, Dez. 2019

HTTPS://NILSHARTMANN.NET

Single- Page- Anwendungen

BEISPIELE

E-Mail – Nils Hartmann – Outlook

https://outlook.live.com/mail/drafts

Outlook Suchen

Neue Nachricht

Favoriten

Ordner

Posteingang

Junk-E-Mail

Entwürfe 3

Gesendete Elemente

Gelöschte Elemente

Archiv

Notizen

Verlauf der Unterhaltung

Neuer Ordner

Gruppen

Neue Gruppe

Von ms@nilshartmann.net Cc Bcc

An nils@nilshartmann.net X

Betreff hinzufügen

[Entwurf] nils@nilshartmann.net (Kein Betreff) 17:19
Es ist keine Vorschau verfügbar.

[Entwurf] Frage 21.04.2019
Guten Tag, ... Viele Grüße!

[Entwurf] Hallo 08.11.2018
Es ist keine Vorschau verfügbar.

concierge

Zwei Bier, bitte!

HTTP://OUTLOOK.LIVE.COM

The screenshot shows a Gitpod session for a React project named "react-training". The browser tab is titled "nilshartmann/react-training - master" and the URL is "https://bb09e8a8-cc8e-4775-b340-55590f31900e.ws-eu0.g". The VS Code interface displays the file "GreetingController.tsx" with the following code:

```
1 import React from "react";
2
3 import GreetingMaster from "./GreetingMaster";
4 import GreetingDetail from "./GreetingDetail";
5 import useApi from "./useApi";
6
7 import { NewGreeting, Greeting } from "./types";
8
9 const BACKEND_URL = "http://localhost:7000/greetings";
10
11 type MODE = "MODE_MASTER" | "MODE_DETAIL";
12
13 export default function GreetingController() {
14   const [mode, setMode] = React.useState<MODE>("MODE_MASTER");
15   const [greetings, setGreetings, isLoading] = useApi(BACKEND_URL, []);
16
17   Remove variable statement
18   Extract to inner function in function 'GreetingController'
19   Extract to function in module scope
20   Extract to constant in enclosing scope
21   Extract to constant in module scope
22   Accept: "application/json",
23   "Content-Type": "application/json"
24 }
```

The "Problems" panel shows a list of compiled files and their sizes:

- [...]/node_modules/url/url.js /workspace/react-training/node_modules/url/url.js 22.8 KiB {main} [built]
- [...]/node_modules/webpack-dev-server/client/index.js?http://localhost (webpack)-dev-server/client?http://localhost 9.26 KiB {main} [built]
- [...]/node_modules/webpack-dev-server/client/overlay.js (webpack)-dev-server/client/overlay.js 3.59 KiB {main} [built]
- [...]/node_modules/webpack-dev-server/client/socket.js (webpack)-dev-server/client/socket.js 1.05 KiB {main} [built]
- [...]/node_modules/webpack/hot sync ^\.\log\$ (webpack)/hot sync nonrecursive ^\.\log\$ 170 bytes {main} [built]
- [...]/node_modules/webpack/hot/emitter.js (webpack)/hot/emitter.js 75 bytes {main} [built]
- ./src/GreetingDetail.js 1.8 KiB {main} [built]
- ./src/main.js 277 bytes {main} [built]
+ 23 hidden modules

The status bar at the bottom indicates "Ln 16, Col 1 LF UTF-8 Spaces: 4 Ports: 0 8080 TypeScript React 3.6.3".

HTTPS://GITPOD.IO

Sample File – Figma

Sicher | https://www.figma.com/file/pQLIW7fU1VQwYIJjs2epDl/Sample-File?node-id=0%...

Log-In Page

9:42 AM 42%

YOUR ART MUSEUM

151 3rd St
San Francisco, CA 94103

Email address

Password

Forgot your password?

Log In

Don't have an account?

Background Image

Instructions

- Intro Text
- Step 1
- Step 2
- Step 3
- Step 4
- Step 5
- Step 6
- Step 7
- Step 8

Log-In Page

- Status Bar
- App Info
 - 151 3rd St San Fran...
 - YOUR ART MUSEUM
- Log-In Fields
- Log-In Button

Home

Menu

Exhibition

Shop

Share

DESIGN PROTOTYPE CODE

Bring Forward ⌘]

Bring to Front ⌘[

Send Backward ⌘[

Send to Back ⌘]

Group Selection ⌘G

Ungroup Selection ⌘ShiftG

Flatten Selection ⌘E

Use as Mask ⌘M

Outline Stroke ⌘O

Create Component ⌘K

Go to Master Component

Reset Instance

Detach Instance ⌘B

Show/Hide Selection ⌘H

Lock/Unlock Selection ⌘L

Flip Horizontal ⌘H

Flip Vertical ⌘V

Copy Style as CSS

Copy as SVG

Copy Style ⌘C

Paste Style ⌘V

CONSTRANTS

LAYER

FILL

STROKE

EFFECTS

EXPORT

Click + to add an export setting

0 0

375 667

0° 0

Left Top

Pass Through 100%

Image 100%

Layer Blur

?

HTTPS://WWW.FIGMA.COM

EXKURS: PROGRESSIVE WEB APP

Progressive Web Apps: Trennung zwischen Web und nativer App aufheben

- Responsive Design, um wie native Apps auszusehen
- Offline-fähig durch Daten Caching
- Können (auf dem Home-Screen) lokal installiert werden
- Empfangen Push-Notifications
- Können als SPA entwickelt werden, ist aber kein Muss
- "Progressive Enhancement": Features werden je nach Browser zur Verfügung gestellt

EXKURS: PROGRESSIVE WEB APP

Beispiel: <https://mobile.twitter.com>



MODERNE WEB-ANWENDUNGEN

Aus Benutzersicht: bestes UI/UX

- Einheitliches Layout und Design
 - Konsistentes Verhalten in der ganzen Anwendung
 - Konsistente Darstellung der Daten
-
- Gewohntes Verhalten von Desktop Anwendungen
 - Kurze Reaktionszeiten

Für Entwicklung

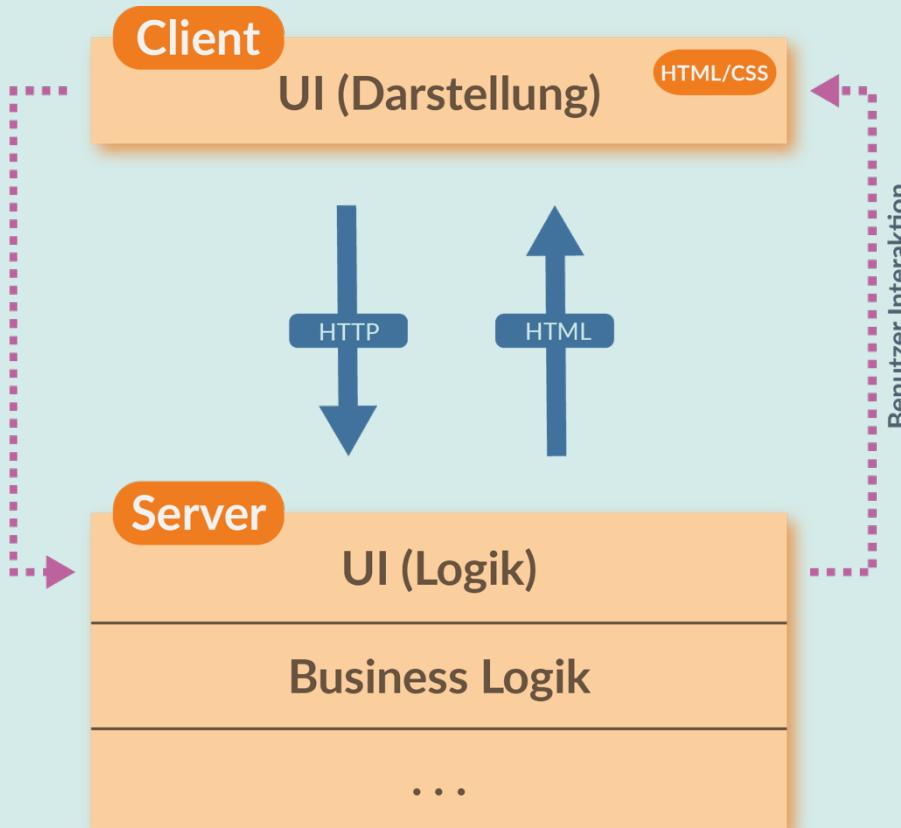
- Einfach und schnell
- Saubere und verständliche Architektur
- Wartbar auch bei großer und langlebiger Code-Basis

KLASSISCHE WEB-ANWENDUNGEN

RÜCKBLICK

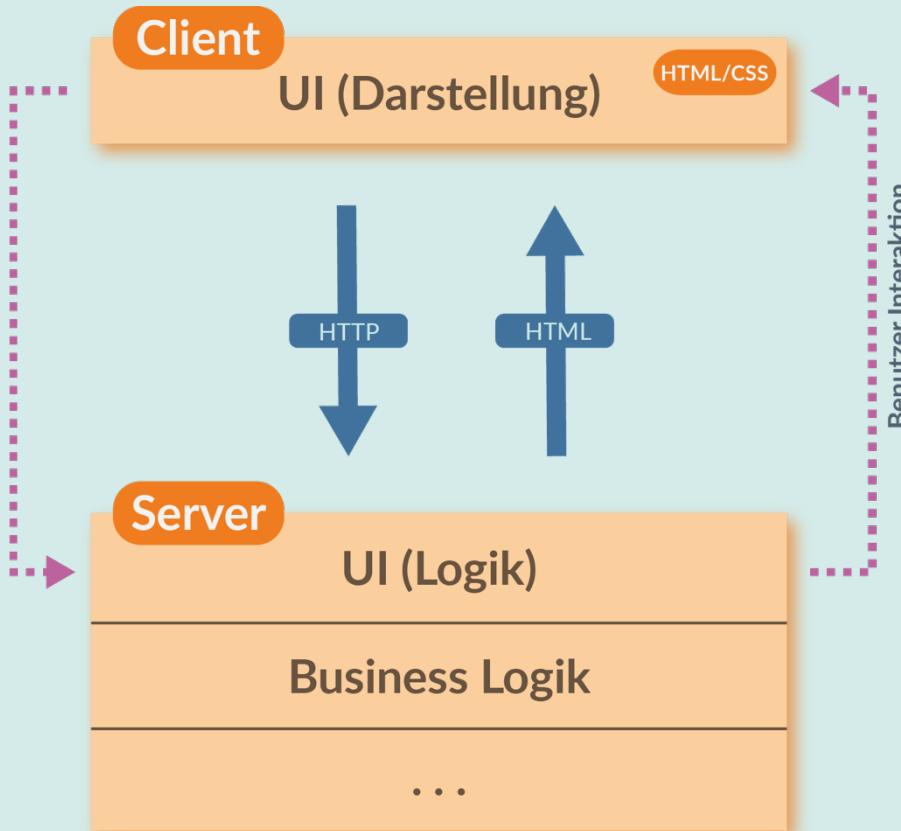
RÜCKBLICK: KLASSISCHE WEB-ANWENDUNG

Klassische Web-Anwendung: Bekannte Technologie und Sprache



RÜCKBLICK: KLASSISCHE WEB-ANWENDUNG

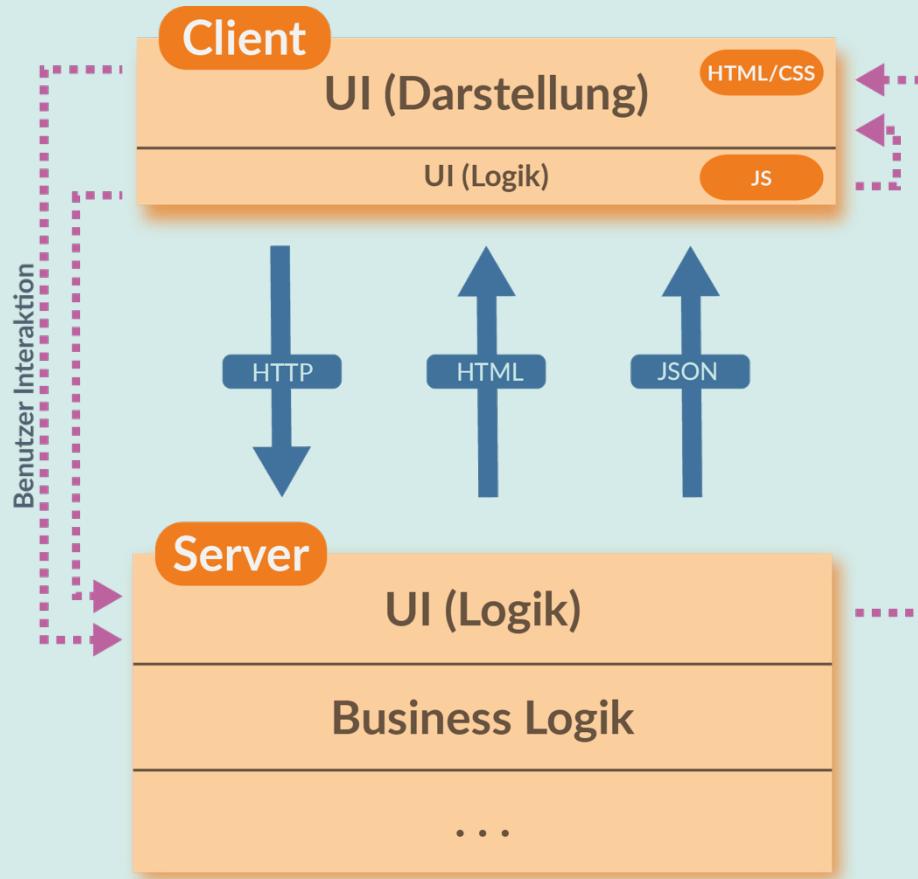
Klassische Web-Anwendung: Bekannte Technologie und Sprache



Konsequenzen: Roundtrips für jede Kleinigkeit

RÜCKBLICK: KLASSISCHE WEB-ANWENDUNG

Klassische Web-Anwendung: ...mit "ein bisschen" JavaScript

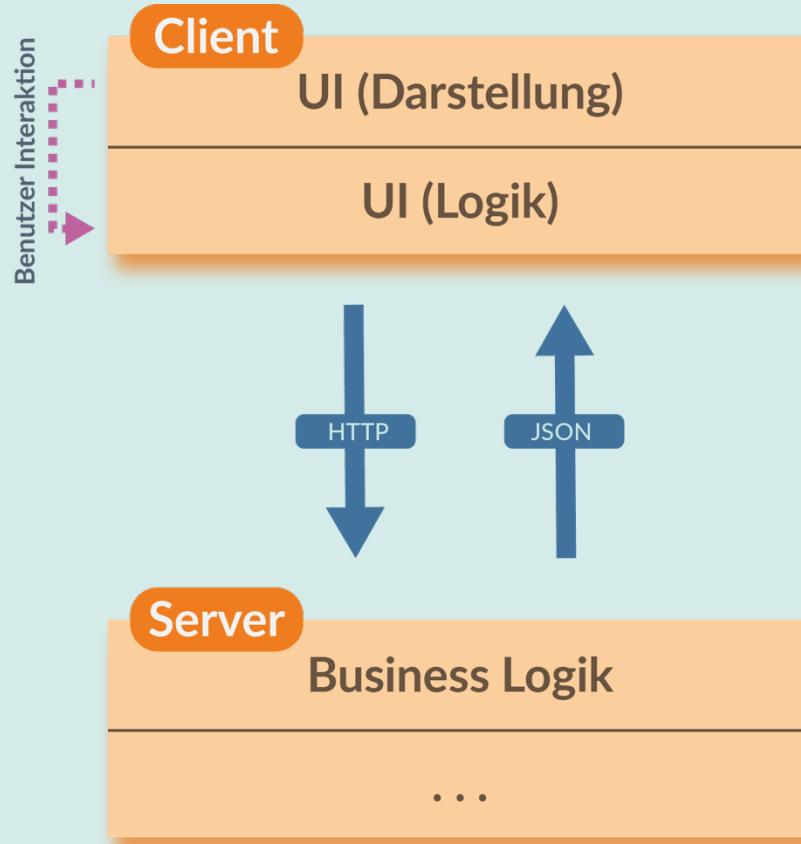


Konsequenzen: unsaubere Architektur, Wartungshölle

Ursache für miserablen Ruf von JavaScript?

SINGLE-PAGE-ANWENDUNG

Moderne Single-Page-Anwendung: Saubere Architektur



Konsequenzen: Kompletter Client in JavaScript...

SINGLE-PAGE-ANWENDUNGEN

Herausforderungen I

- JavaScript als Sprache "gewöhnungsbedürftig"
- Verwirrendes JavaScript Ökosystem
- Abhängigkeits- und Paket-Management
- Testen von Anwendungen
- Architektur, damit Code verständlich und wartbar bleibt
- Aufteilen der Anwendung in kleine Teile zur besseren Entwicklung

SINGLE-PAGE-ANWENDUNGEN

Anforderungen

- Gewohnte Navigation über Browser soll funktionieren
- Anwendung soll auf diversen Geräten "richtig" aussehen
- Anwendung soll konsistent dargestellt werden
- Anwendung soll schnell dargestellt werden
- Geräte (Kamera, Audio, Sensoren) sollen verwendet werden

Umsetzung

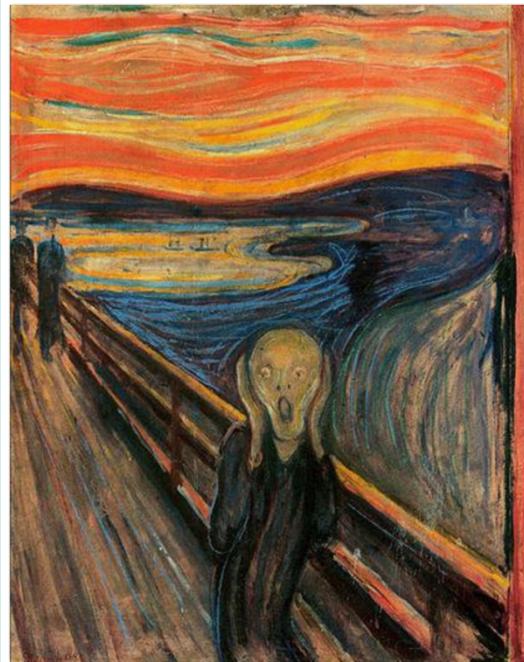
Lukas Eder
@lukaseder

Folgen

▼

Still one of my favourite paintings:
Edvard Munch: The JavaScript, 1893

Original (Englisch) übersetzen



11:00 - 15. Okt. 2016

285 Retweets 442 „Gefällt mir“-Angaben



7 285 442

JavaScript

<https://twitter.com/lukaseder/status/787216648642109441>

JavaScript: Nur Sprache

- Keine zentrale Organisation (abgesehen vom Sprachstandard)
- Keine Standard Bibliothek (nur minimal)
- Kein Typ-System
- Kein Compiler
- Keine einheitliche Laufzeitumgebung
- ~~Kein Modul-System~~
- Kein Threading

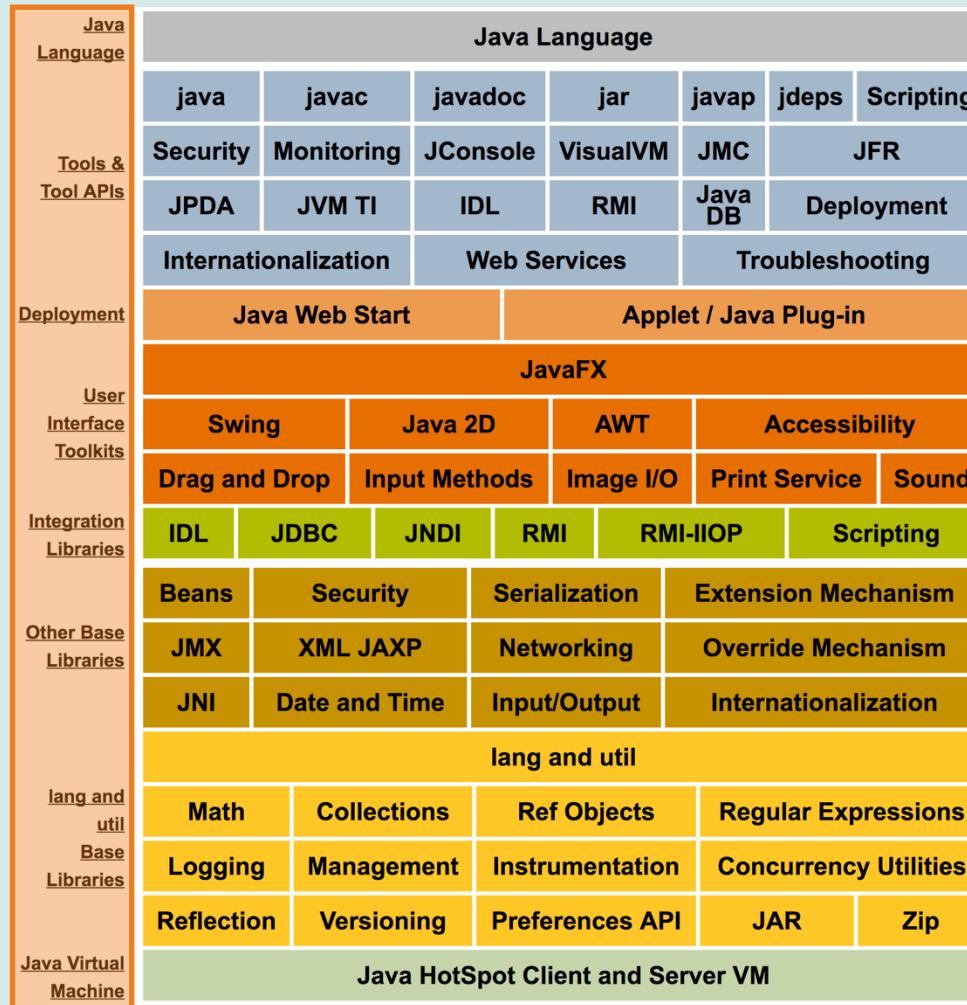
- Wenig Konventionen (keine klassische Maven Projektstruktur z.B.)

ZUM VERGLEICH: JAVA

Java: Bringt alles mit was wir zum Entwicklung brauchen

Tools, Bibliotheken, Laufzeitumgebung, ...

Alles aus einer Hand



Grafik: <https://docs.oracle.com/javase/8/docs/index.html>

"Dezentrale" Struktur, einerseits...

- Sehr hohes Innovationstempo
 - erfordert stetig neue Lösungen für neue Probleme

"Dezentrale" Struktur, einerseits...

- Sehr hohes Innovationstempo
 - erfordert stetig neue Lösungen für neue Probleme

...andererseits

- Für Tools, Bibliotheken etc. gibt es keine zentrale Instanz
- Probleme werden "dezentral" gelöst

"Dezentrale" Struktur, einerseits...

- Sehr hohes Innovationstempo
 - erfordert stetig neue Lösungen für neue Probleme

...andererseits

- Für Tools, Bibliotheken etc. gibt es keine zentrale Instanz
- Probleme werden "dezentral" gelöst

...es gibt mittlerweile De-facto-Standard Tool-Stacks

Werkzeuge

Modernes JavaScript: Releases ab 2015

- Deutlich besser als Versionen davor, viele Schwachstellen entfernt
- Zum Beispiel Block-Scoping, Klassen und Module, Promises
- Jährliche Releases mit kleineren Änderungen ("ES.Next")

```
class Person {  
  #name;  
  constructor(name) {  
    this.#name = name;  
  }  
  sayHi() { return `Hello, ${this.#name}`  
}  
export default Person;
```

Modernes JavaScript: Releases ab 2015

```
// Person.js
export default class Person {
  #name;

  constructor(name) {
    this.#name = name;
  }

  sayHi() { return `Hello, ${this.#name}` }
}
```

Modernes JavaScript: Releases ab 2015

```
// Person.js
export default class Person {
    #name;

    constructor(name) {
        this.#name = name;
    }

    sayHi() { return `Hello, ${this.#name}` }
}

// main.js
import Person from "./Person";

const p = new Person("Klaus");
p.sayHi(); // OK
p.#name = "Dieter"; // ERROR
```

Statische Code-Analyse: ESLint (<https://eslint.org/>)

- Findet typische JavaScript Programmierfehler
- Achtet auf Einhaltung von Konventionen (z.B. Semikolon ja/nein)
- Kann in den CI-Build eingebunden werden

The screenshot shows a code editor window with a tab labeled "example.js". The code contains several ESLint errors, indicated by red squiggly underlines and vertical error bars:

```
1
2  function identical(a, b) {
3    if (a ~ b) {
4      return
5      true;
6    }
7 }
```

Below the code editor, the "PROBLEMS" tab is selected in the bottom navigation bar. The list of errors is as follows:

- ✖ [eslint] Expected '===' and instead saw '=='. (eqeqeq) (3, 8)
- ✖ [eslint] Expected an assignment or function call and instead.. (5, 3)
- ✖ [eslint] Unreachable code. (no-unreachable) (5, 3)

TypeScript (<https://typescriptlang.org>)

- Mächtiges Typ-System für JavaScript von Microsoft
- Baut auf JavaScript auf, jeder JavaScript-Code auch TypeScript-Code
- Compiler für ES6 Code nach ES5

TYPESCRIPT

TypeScript (<https://typescriptlang.org>)

```
// Person.ts
export default class Person {
    private name: string;           // Typ-Angaben und Sichtbarkeiten

    constructor(name: string) {
        this.name = name;
    }

    sayHi() { return `Hello, ${this.#name}` } // Return-Typ wird abgeleitet
}
```

TypeScript (<https://typescriptlang.org>)

```
// Person.ts
export default class Person {
    private name: string;           // Typ-Angaben und Sichtbarkeiten

    constructor(name: string) {
        this.name = name;
    }

    sayHi() { return `Hello, ${this.#name}` } // Return-Typ wird abgeleitet
}

// main.ts
import Person from "./Person";

const p = new Person("Klaus"); // OK
const h = new Person(123);    // ERROR: Number is not a string

const hi = p.sayHi();
hi.toUpperCase(); // OK - hi ist ein String (abgeleitet)

hi.sayHello(); // ERROR: sayHello gibt's nicht auf String
```

TypeScript (<https://typescriptlang.org>)

- Sehr guter IDE Support (Visual Studio Code, IDEA)
- Refaktorings, Find Usages etc. zuverlässig möglich

The screenshot shows a code editor window for a file named `Greeting.ts`. The code defines a class `Greeting` with a private phrase and a constructor that sets it. It has a method `greet` that returns a string template with the phrase and a name. A variable `g` is created and initialized with a new `Greeting` instance. The code then attempts to call `g.greet(123)`, which triggers a tooltip: "Argument of type '123' is not assignable to parameter of type 'string'. ts(2345)". Below the code editor is a "PROBLEMS" panel showing three errors:

- ts Greeting.ts code 3
 - Argument of type '123' is not assignable to parameter of type 'string'. ts(2345) [18, 9]
 - Property 'phrase' is private and only accessible within class 'Greeting'. ts(2341) [20, 3]
 - Type 'string' is not assignable to type 'number'. ts(2322) [23, 7]

At the bottom, there are status icons for Go Live, Ln 16, Col 1, Spaces: 2, UTF-8, LF, TypeScript 3.4.3, and a notification bell with 1 message.

```
1 class Greeting {
2     private phrase: string;
3
4     constructor(phrase: string) {
5         this.phrase = phrase;
6     }
7
8     greet(name: string) {
9         return `${this.phrase}, ${name}`;
10    }
11 }
12
13 const g = new Greeting("Hello");
14
15
16     Argument of type '123' is not assignable to parameter of
17     type 'string'. ts(2345)
18     Quick Fix... Peek Problem
19     g.greet(123);
20
21     g.phrase = "goodbye";
22
23     const result:number = g.greet("Klaus");
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL Filter. Eg: text, /*/*ts... PROBLEMS 3 ts Greeting.ts code 3

- Argument of type '123' is not assignable to parameter of type 'string'. ts(2345) [18, 9]
- Property 'phrase' is private and only accessible within class 'Greeting'. ts(2341) [20, 3]
- Type 'string' is not assignable to type 'number'. ts(2322) [23, 7]

Go Live Ln 16, Col 1 Spaces: 2 UTF-8 LF TypeScript 3.4.3 1

PACKAGE MANAGER / EXTERNE ABHÄNGIGKEITEN

npm: node package manager (<https://npmjs.com>)

- Standard Package Manager für Node-Entwicklung
- Beschreibung externer Abhängigkeiten
- Eigene Packages können publiziert werden
 - In zentrale Registry (analog maven-central)
 - Private Registry (z.B. Nexus) möglich
- Über npm werden üblicherweise auch notwendige **Tools** deklariert und installiert
 - Zum Beispiel für Compiler und Build-Tools
 - Jeder im Team verwendet einheitliche Version

Webpack: Zentrales Build-Werkzeug

- <https://webpack.github.io/>
- Erstellt lauffähiges JavaScript-Modul ("Bundle"), quasi ein "Fat Jar" für den Browser
- Unterstützung für alle Modul-Systeme
- Sehr viele Möglichkeiten zur Optimierung der Bundle-Größen (Performance!)

DOM
API

Beispiel: Service Worker

- Service Worker erlauben Caching von Daten
- Daten werden dann aus (persistentem) Cache geholt
- Anwendung wird Offline-fähig

Beispiel: Zugriff auf Geräte

- Für viele Geräte mittlerweile Standard APIs verfügbar bzw. in Planung
 - z.B. Kamera, Audio, Device Orientation, ...
- 👉 <https://nilshartmann.github.io/react-talk/examples/>

Zugriff auf Geräte: Web APIs

- Für viele Geräte mittlerweile Standard APIs verfügbar bzw. in Planung
 - z.B. Kamera, Audio, Device Orientation, ...
- 👉 <https://nilshartmann.github.io/react-talk/examples/>

```
navigator.mediaDevices
  .getUserMedia(constraints)
    .then(stream => {
      video.srcObject = stream;
      video.play();
      startButton.addEventListener("click", () => { video.play(); });
      pauseButton.addEventListener("click", () => { video.pause(); });
    })
  }
```

Grenzen der DOM API

- DOM API sehr mächtig, aber auch sehr "sperrig"
- Nicht geeignet, damit ganze Anwendungen zu bauen
- 👉 <https://nilshartmann.github.io/react-talk/examples/dom-api.html>

Grenzen der DOM API

- DOM API sehr mächtig, aber auch sehr "sperrig"
- Nicht geeignet, damit ganze Anwendungen zu bauen
- 👉 <https://nilshartmann.github.io/react-talk/examples/dom-api.html>

```
const title = document.createElement("h1");
const titleText = document.createTextNode("Hello World");
title.appendChild(titleText);

const greetButton = document.createElement("button");
greetButton.innerHTML = "Greet!";
greetButton.disabled = true;
greetButton.addEventListener("click", event => { ...

const greetInput = document.createElement("input");
greetInput.addEventListener("input", event => { ...

document.body.appendChild(title);
document.body.appendChild(greetInput);
document.body.appendChild(greetButton);|
```

Code, der nur für die Darstellung der Hello-World-App benötigt wird 😱

SPA

Frameworks

FRAMEWORKS

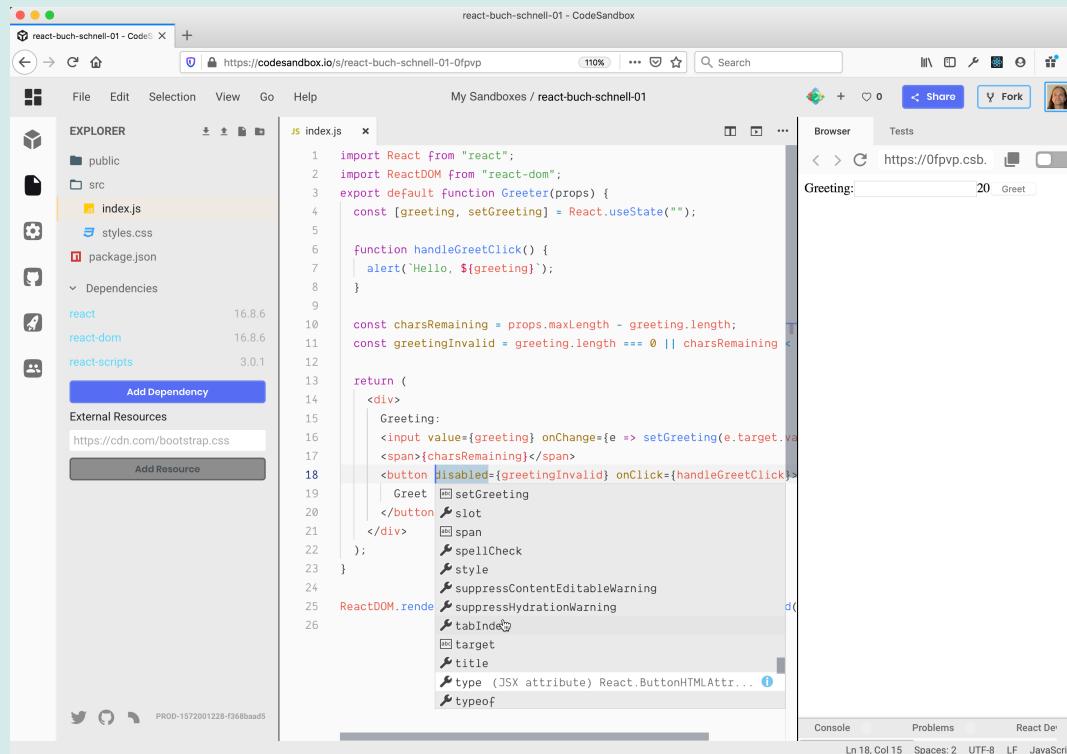
SPA-Frameworks

- es gibt sehr gute, ausgereifte Frameworks
- Vier prominente Vertreter
 - React
 - Angular
 - Vue
 - Web Components (Standard)
- CLIs helfen bei Installation und Verwaltung
- Frameworks und Tool-Stack mittlerweile relativ stabil und "Standard"

MAL KURZ AUSPROBIEREN...

Online-Editor: <https://codesandbox.io/>

- Vollständiger JavaScript/TypeScript-Editor im Web
- Vorkonfigurierte Projekte für Angular, React, Vue
- Zum ausprobieren, ohne was installieren müssen



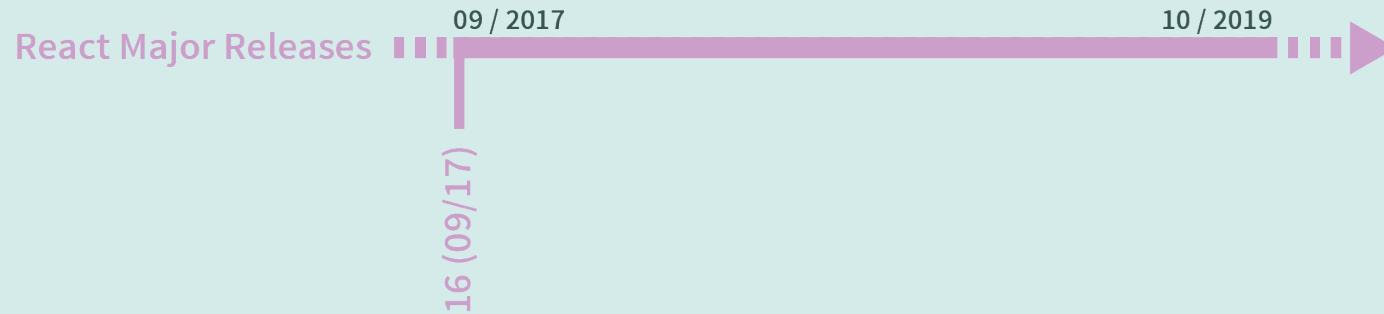
The screenshot shows the CodeSandbox web interface. On the left, the Explorer sidebar displays the project structure: a public folder, a src folder containing index.js and styles.css, and a package.json file. Below these are dependencies for react (16.8.6), react-dom (16.8.6), and react-scripts (3.0.1). A prominent blue button labeled "Add Dependency" is visible. The central area is a code editor for index.js, which contains the following code:

```
1 import React from "react";
2 import ReactDOM from "react-dom";
3 export default function Greeter(props) {
4   const [greeting, setGreeting] = React.useState("");
5
6   function handleGreetClick() {
7     alert(`Hello, ${greeting}`);
8   }
9
10  const charsRemaining = props.maxLength - greeting.length;
11  const greetingInvalid = greeting.length === 0 || charsRemaining <
12
13  return (
14    <div>
15      Greeting:
16      <input value={greeting} onChange={e => setGreeting(e.target.value)} />
17      <span>{charsRemaining}</span>
18      <button disabled={greetingInvalid} onClick={handleGreetClick}>
19        Greet
20        <slot name="setGreeting"></slot>
21      </button>
22    </div>
23  );
24  <ReactDom.render>(
25    <React.StrictMode>
26      <Greeter maxLength={20} />
27    </React.StrictMode>
28  );
29}
```

To the right of the code editor is a preview window titled "Browser". It shows a simple form with a text input field and a button. The input field has the placeholder "Greeting:" and contains the value "20". The button is labeled "Greet". Below the preview are tabs for "Tests" and "Console". At the bottom, there are links for "Problems" and "React DevTools".

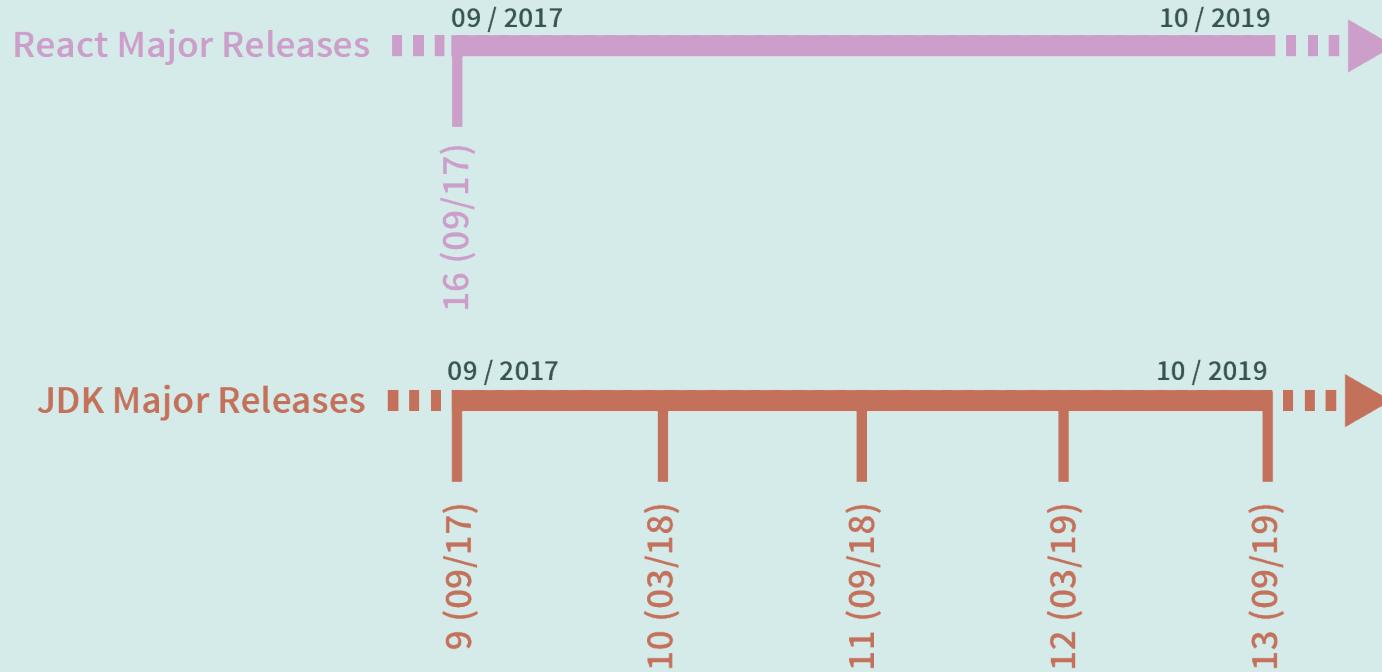
BEISPIEL: REACT

React - Stabilität



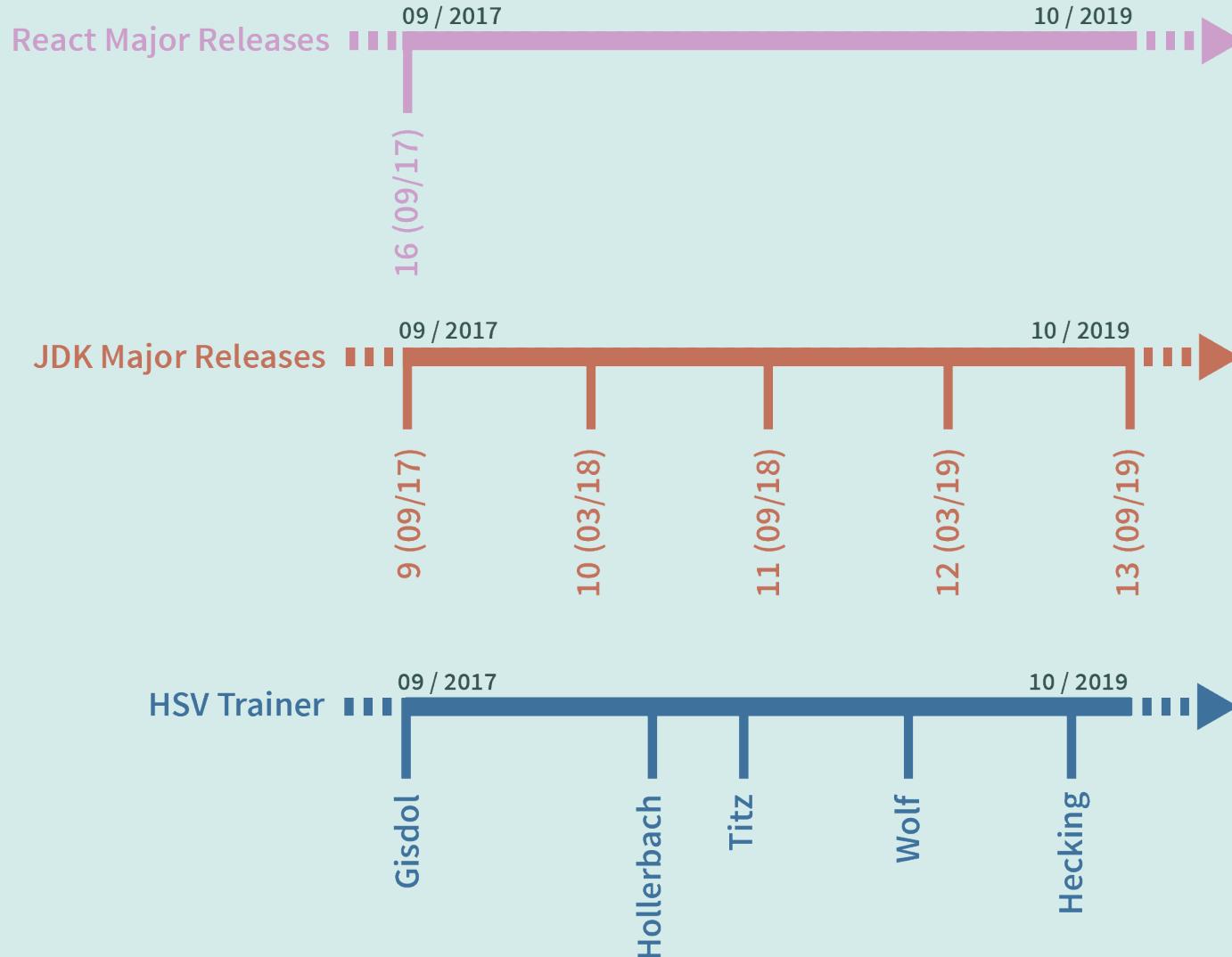
BEISPIEL: REACT

React - Stabilität



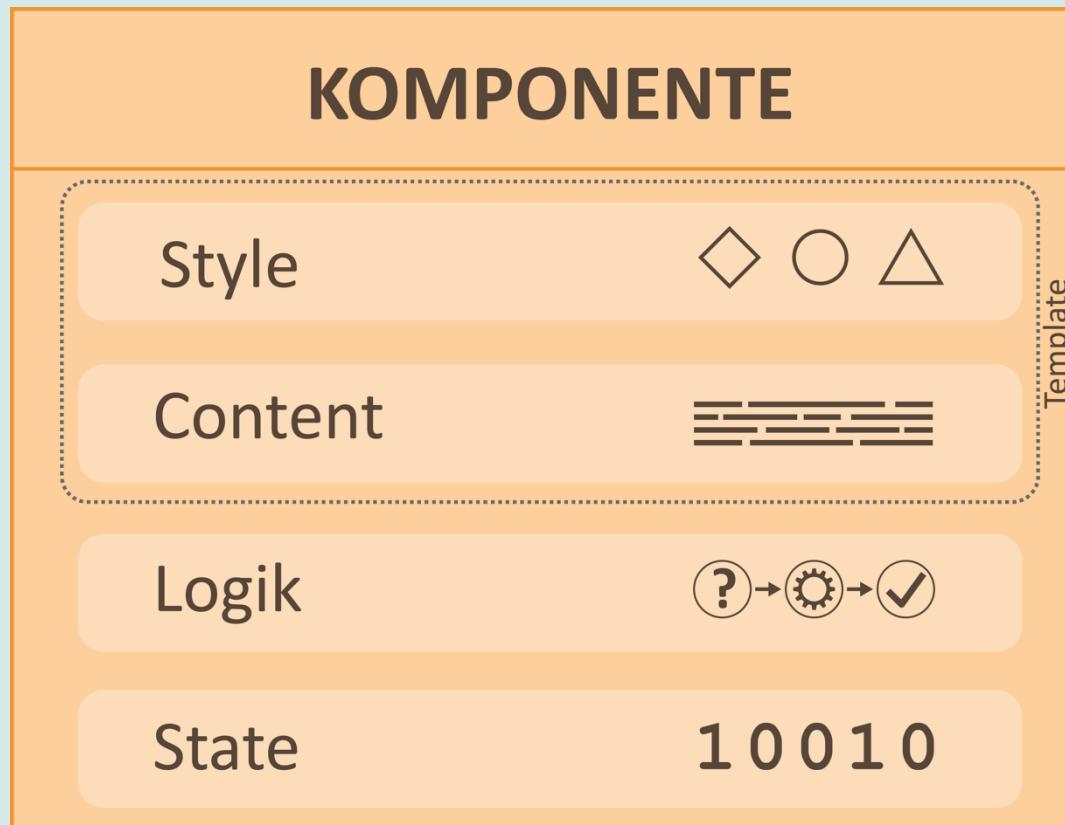
BEISPIEL: REACT

React - Stabilität



FRAMEWORKS

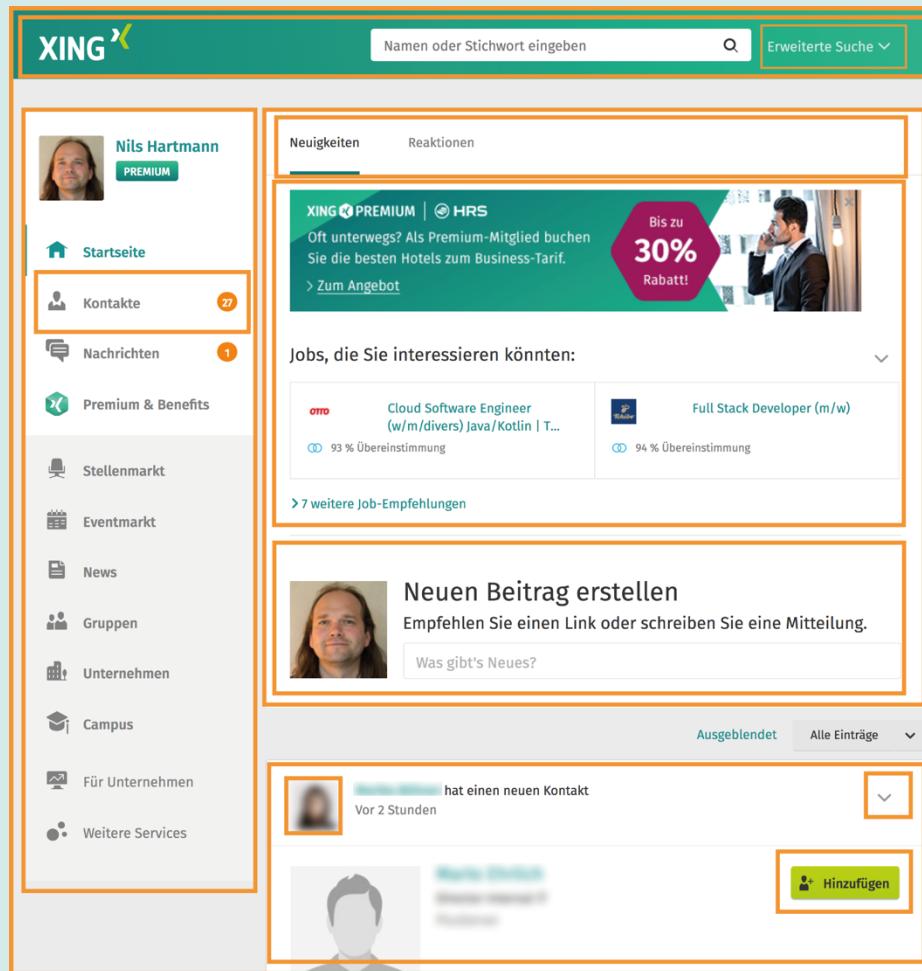
Frameworks: wir bauen Komponenten



FRAMEWORKS

Komponenten werden zu Anwendungen aggregiert

Beispiel: XING



BEISPIEL: REACT

React - Eine Komponente

```
import React from "react";

export default function Greeter(props) {
  const [greeting, setGreeting] = React.useState(props.initialGreeting);

  const buttonDisabled = greeting.length === 0;

  return (
    <div>
      <input
        value={greeting}
        onChange={e => setGreeting(e.target.value)}
      />

      <button disabled={buttonDisabled}
              onClick={() => props.onGreet(greeting)}>
        Greet!
      </button>
    </div>
  );
}
```

BEISPIEL: REACT

React - Eine Anwendung

```
import React from "react";

import {Header, Main, Footer} from "./Layout";
import LoginForm from "./forms/LoginForm";
import Greeter from "./forms/Greeter";

export default function App() {
  return (<>
    <Header><LoginForm /></Header>
    <Main>
      <Greeter initialGreeting ="Hello!" onGreet={greeting => ... } />
    </Main>
    <Footer>© Greeting Inc.</Footer>
  </>
);
}
```

BEISPIEL: REACT

React - Eine Komponente mit TypeScript

```
import React from "react";

type GreeterProps = {
    initialGreeting: string
    onGreet(greeting: string): void
}

export default function Greeter(props: GreeterProps) {
    // Rest unverändert,
    // TypeScript kann alle Typen herleiten
}
```

BEISPIEL: REACT

React – Mit TypeScript

Fehlermeldungen bei inkorrektener Verwendung

```
(alias) function Greeter(props: GreeterProps): JSX.Element
import Greeter

export
return
<He
<Ma
<Footer>© Greeting Inc.</Footer>
</>
);
}
```

Property 'onGreet' is missing in type '{ initialGreeting: string; }' but required in type 'GreeterProps'. ts(2741)

Greeter.tsx(5, 3): 'onGreet' is declared here.

Peek Problem No quick fixes available

<Greeter initialGreeting = "Hello!" />

BEISPIEL: REACT

React – Clientseitiges Routing

Eine Router-Komponente synchronisiert URL mit der Anwendung
"Deep-Links" funktionieren und gewohnte Navigation über Back-Button

```
import React from "react";
import { BrowserRouter, Router, Redirect } from "react-router";
import LoginForm from "./forms/Greeter";
import Greeter from "./forms/Greeter";

function App() {
  return ...
  <Main>
    <BrowserRouter>
      <Route path="/greet"> <Greet ... /> </Route>
      <Route path="/login"> <LoginForm /> </Route>
      <Route path="/"> <Redirect to="/greet"/> </Route>
      <Route> <NotFound /> </Route>
    </BrowserRouter>
  </Main>
  ...
}
```

BEISPIEL: REACT

Serverseitiges Rendering (SSR) – Anwendung auf dem Server vorrendern

- Schnelle First-Page-Impression
- Verbessertes SEO
- Vorschauen, z.B. in Social Media

```
import React from "react";
import ReactDOMServer from "react-dom/server";
import App from "./App";

ReactDOMServer.renderToString(<App />);
```

BEISPIEL: REACT

React – Testen einer Komponente

Testen ohne Browser möglich, headless im CI-Build

```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import Greeter from "./forms/Greeter";

test("it should behave fine", () => {

  // Mock für Event-Handler
  const onGreetHandler = jest.fn();

  // Komponente Rendern
  const { getByText, container } = render(
    <Greeter initialGreeting="Hello" onGreet={onGreetHandler} />
  );

  // Ereignis simulieren
  fireEvent.click(getByText("Greet!"));

  // Ergebnis überprüfen
  expect(onGreetHandler).toHaveBeenCalledWith("Hello");

});
```

BEISPIEL: REACT

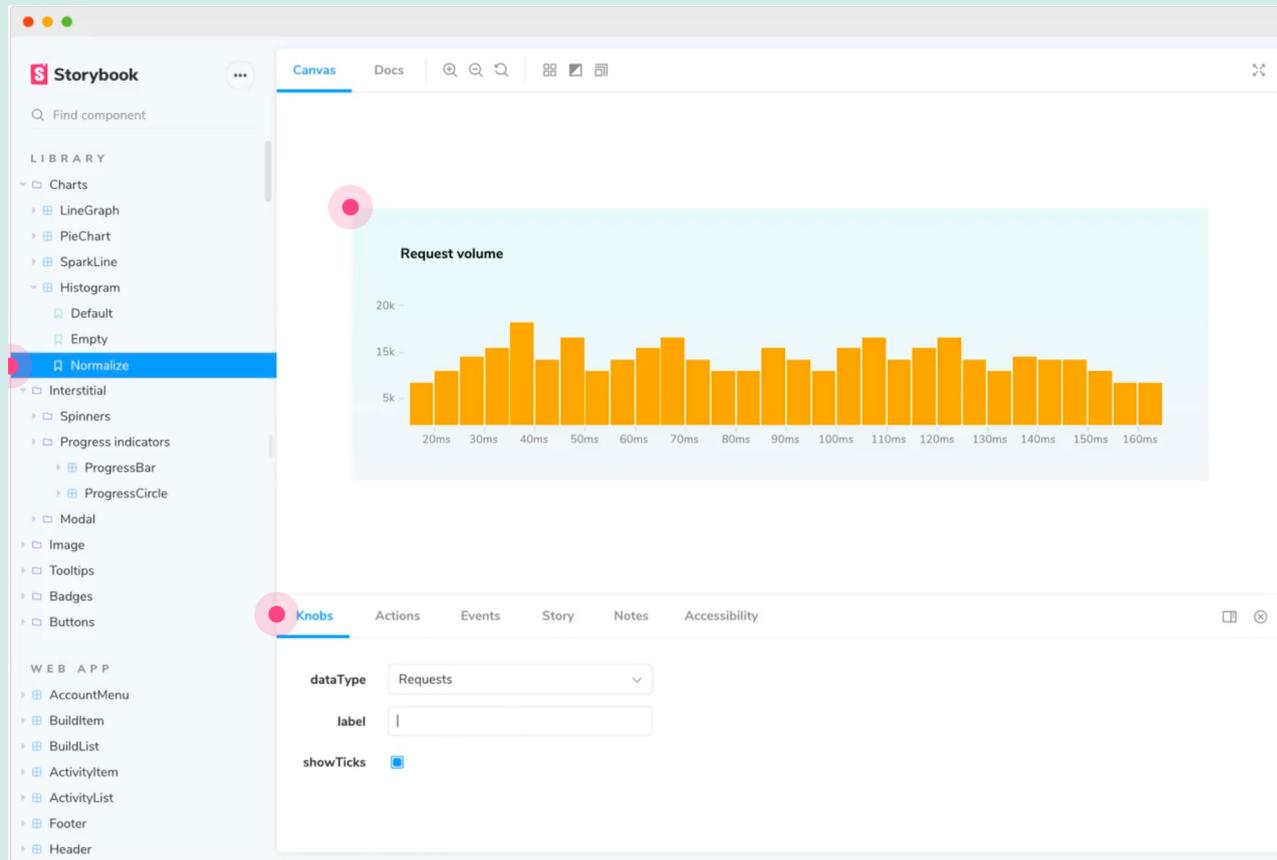
End-2-End-Tests im Browser

- TestCafe (<https://DevExpress.github.io/testcafe/>)
 - Cypress (<https://www.cypress.io/>)
 - Selenium natürlich weiterhin möglich
-
- Tests werden in JavaScript/TypeScript geschrieben
 - Guter Browser-Support (insb. TestCafe)
 - Gutes Debuggen und Fehleranalyse von Tests
 - Ausführung headless im CI-Build möglich

BEISPIEL: REACT

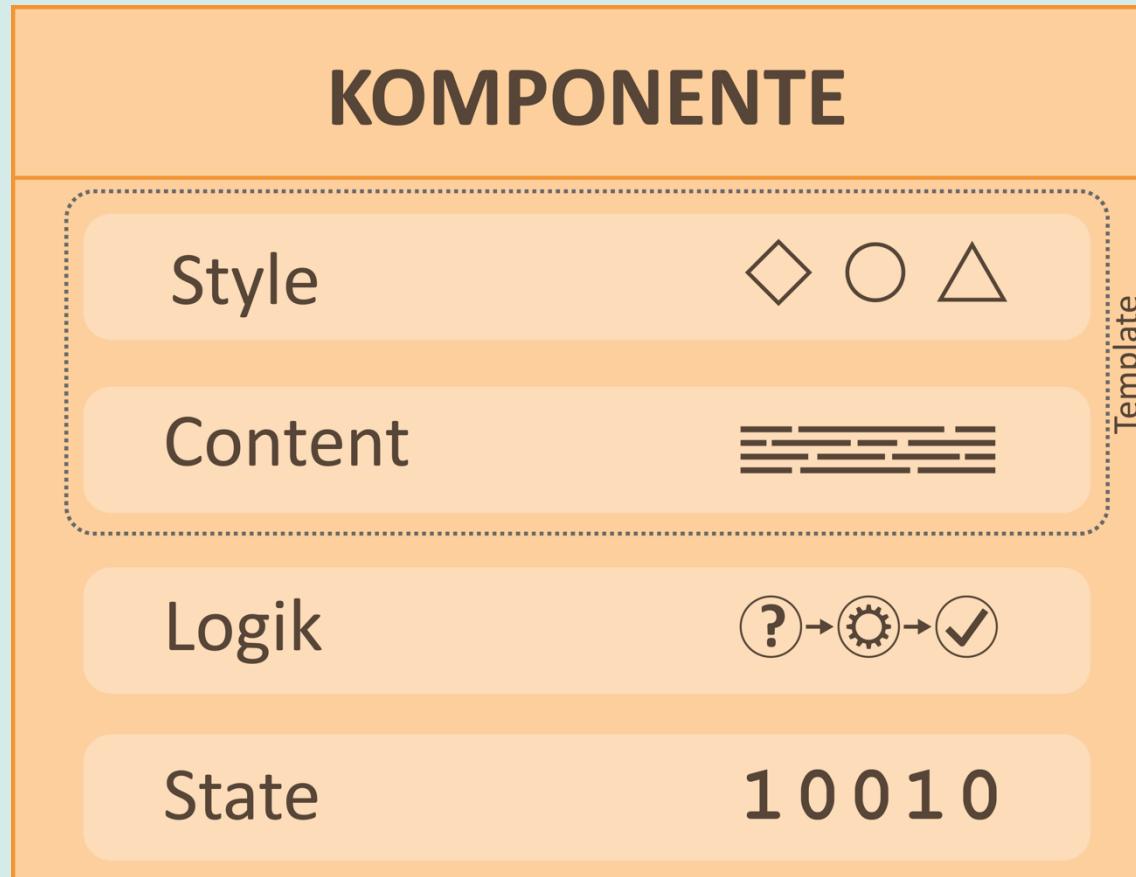
Storybook (<https://storybook.js.org/>)

- Testen, dokumentieren und präsentieren von Komponenten



Frontend- Architektur

Frameworks: wir bauen Komponenten



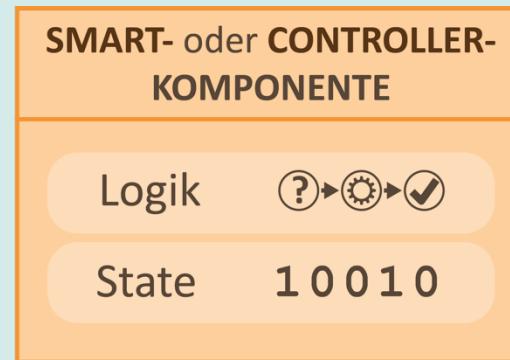
ARCHITEKTUR

Architektur-Muster: Smart- und Dumb-Komponenten



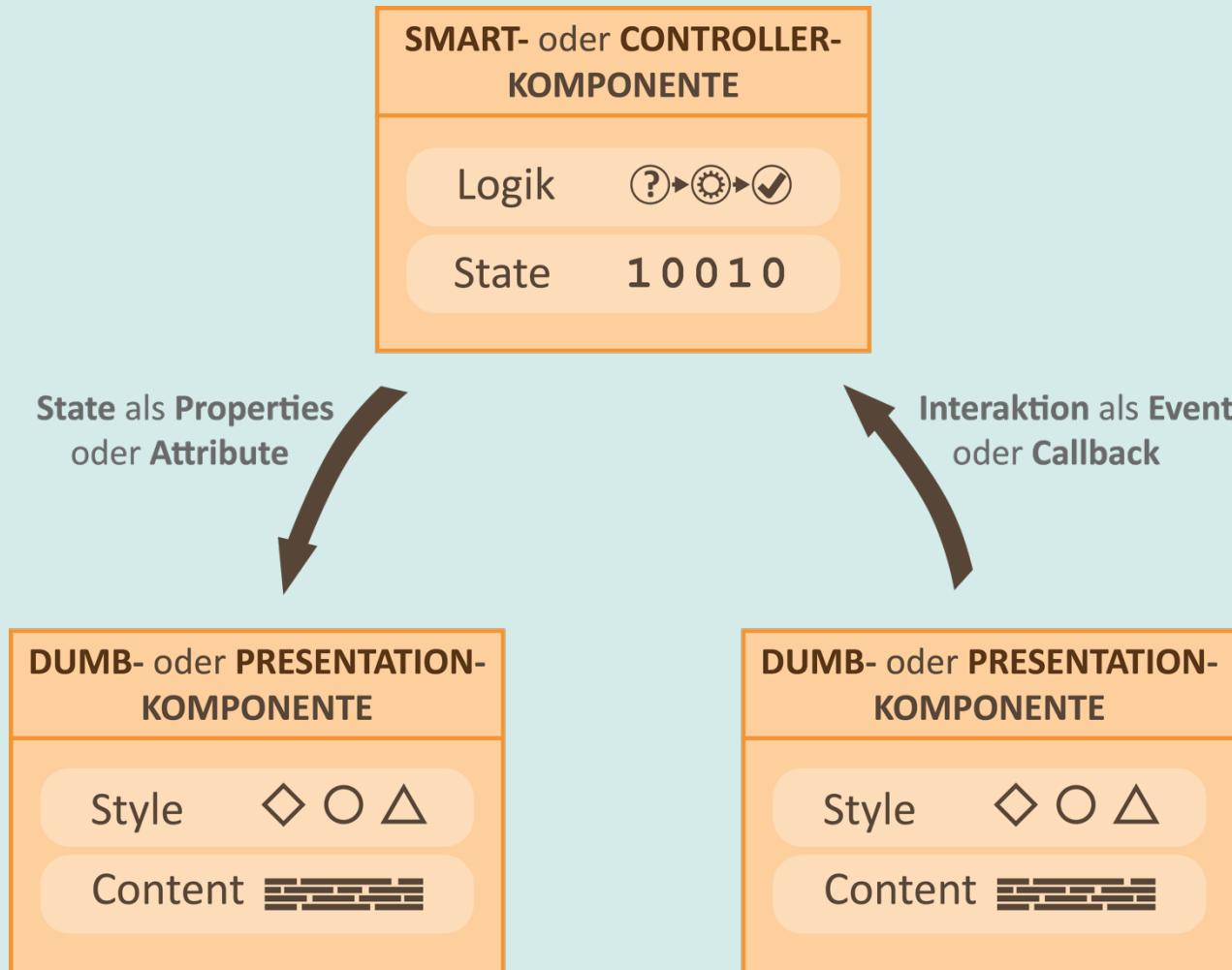
ARCHITEKTUR

Architektur-Muster: Smart- und Dumb-Komponenten



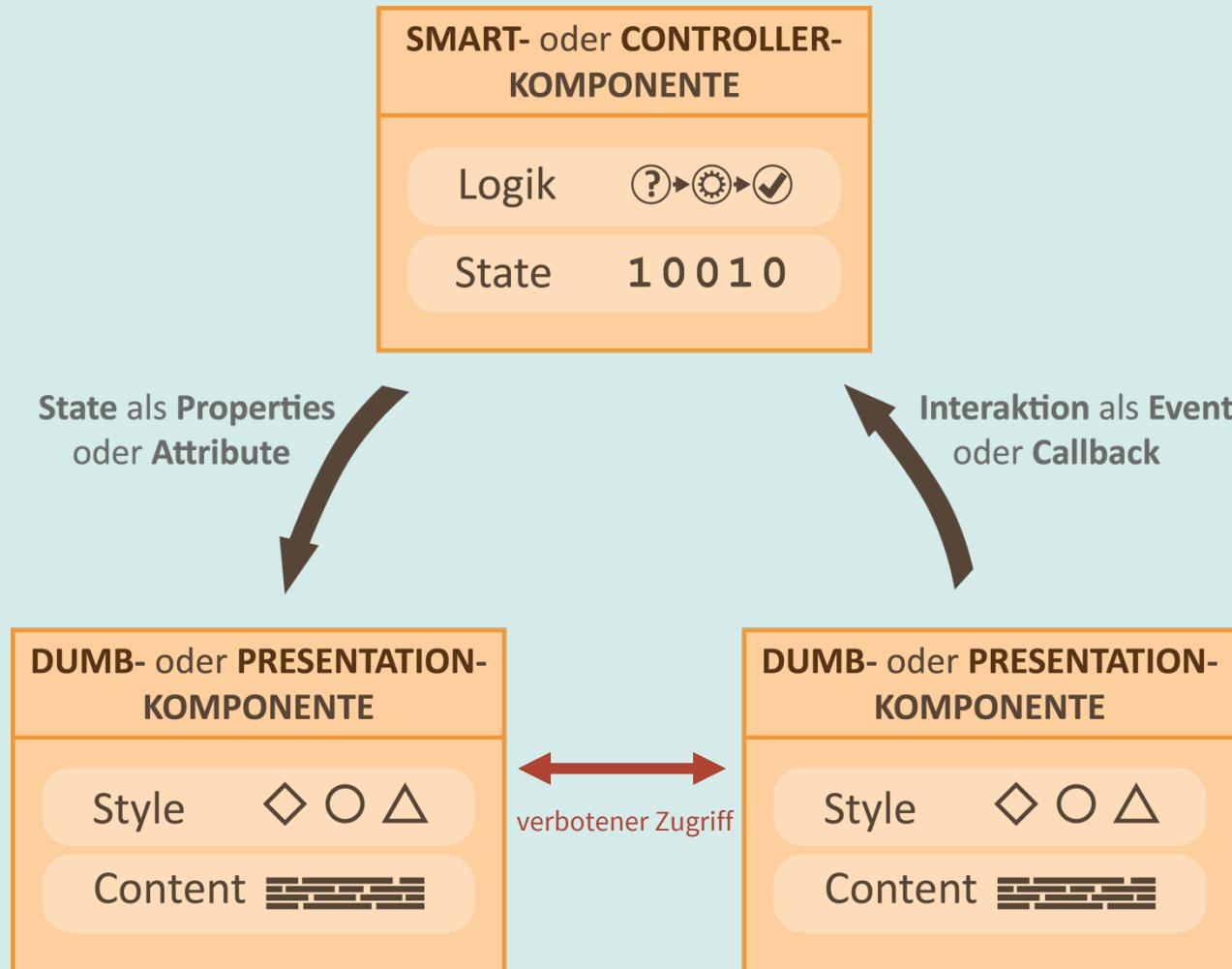
KOMPOSITION VON KOMPONENTEN

Architektur-Muster: Smart- und Dumb-Komponenten



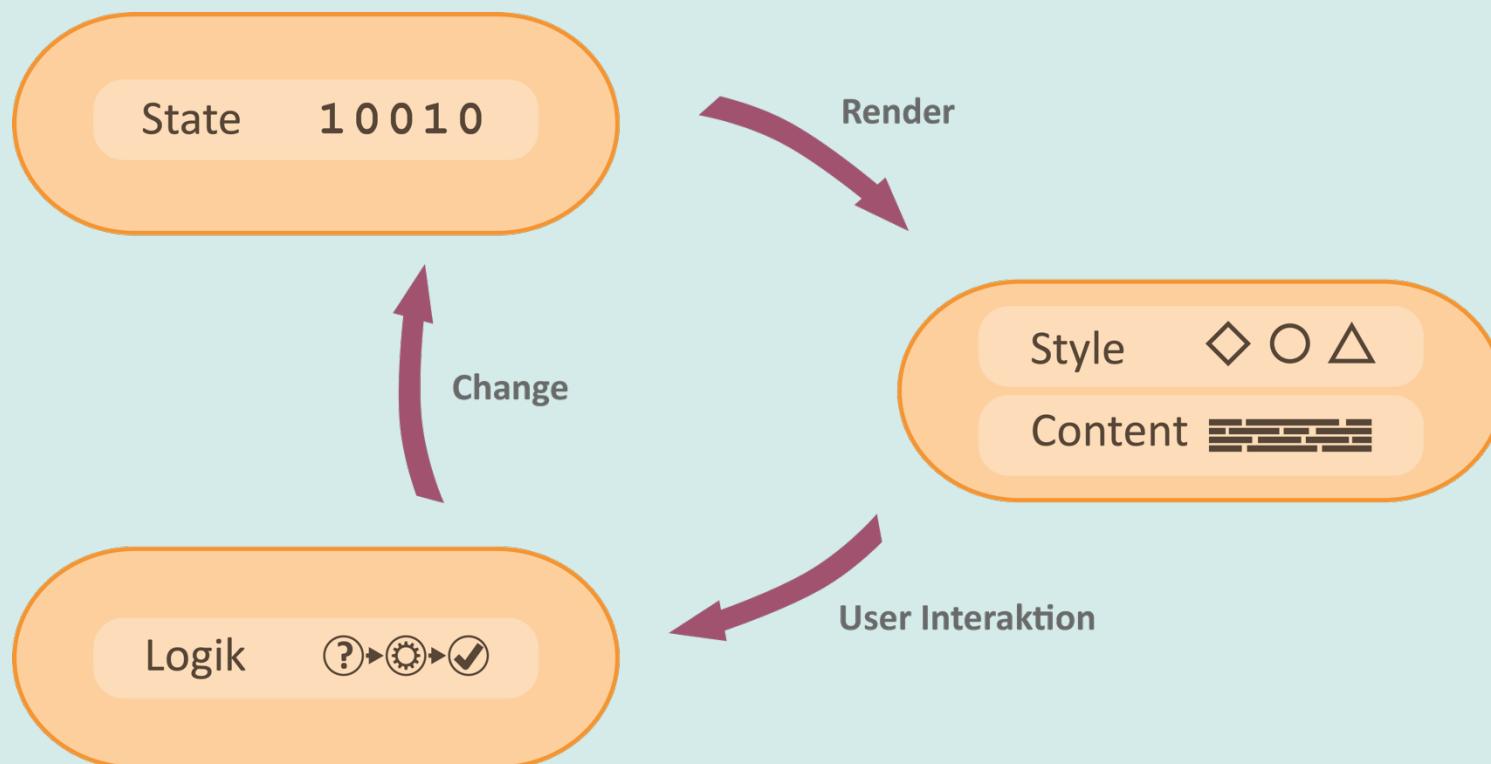
KOMPOSITION VON KOMPONENTEN

Architektur-Muster: Smart- und Dumb-Komponenten



ARCHITEKTUR MUSTER

Reaktives Verhalten: Nachvollziehbarer Renderzyklus



KOMPONENTEN

Grenzen dieser Architektur

- Große Komponenten entstehen
- Viele Controller-Komponenten => wieder Chaos
- Starke Abhängigkeit vom gewählten UI-Framework

KOMPONENTEN

Grenzen dieser Architektur

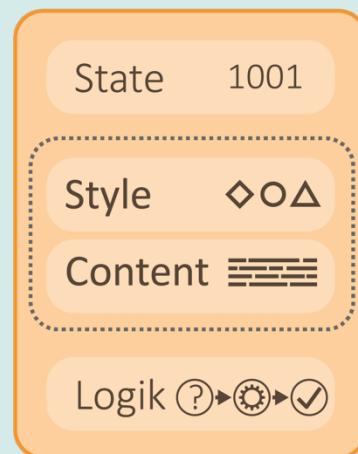
- Große Komponenten entstehen
- Viele Controller-Komponenten => wieder Chaos
- Starke Abhängigkeit vom gewählten UI-Framework

Externes Statemanagement

- Verschiebt Logik und Zustand aus der Komponente
- Prominente Vertreter: Redux, MobX

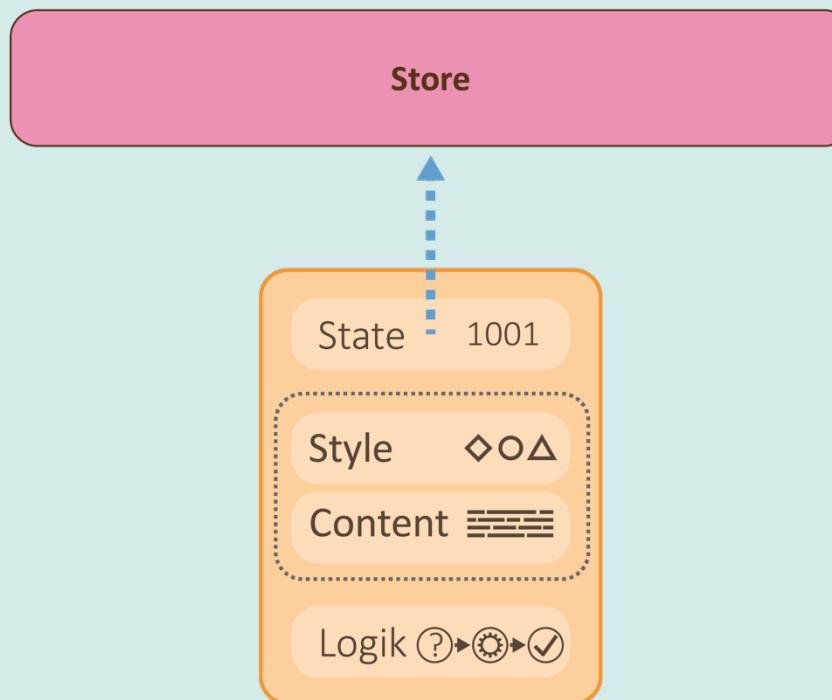
KOMPONENTEN

Externes Statemanagement



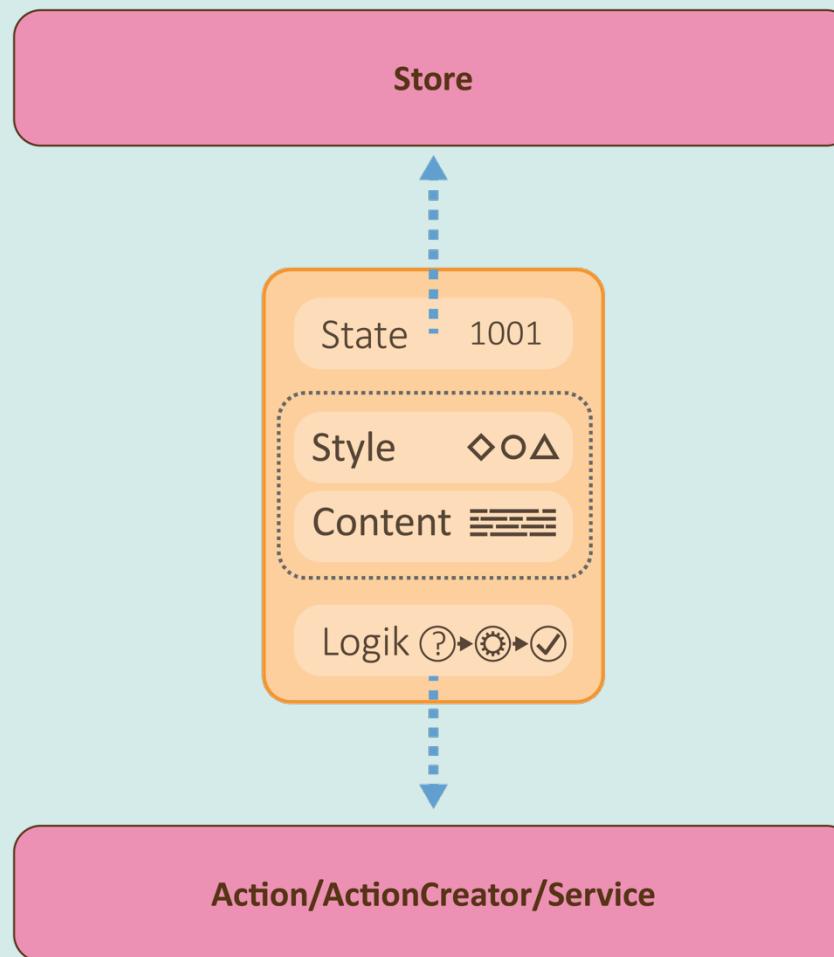
KOMPONENTEN

Externes Statemanagement



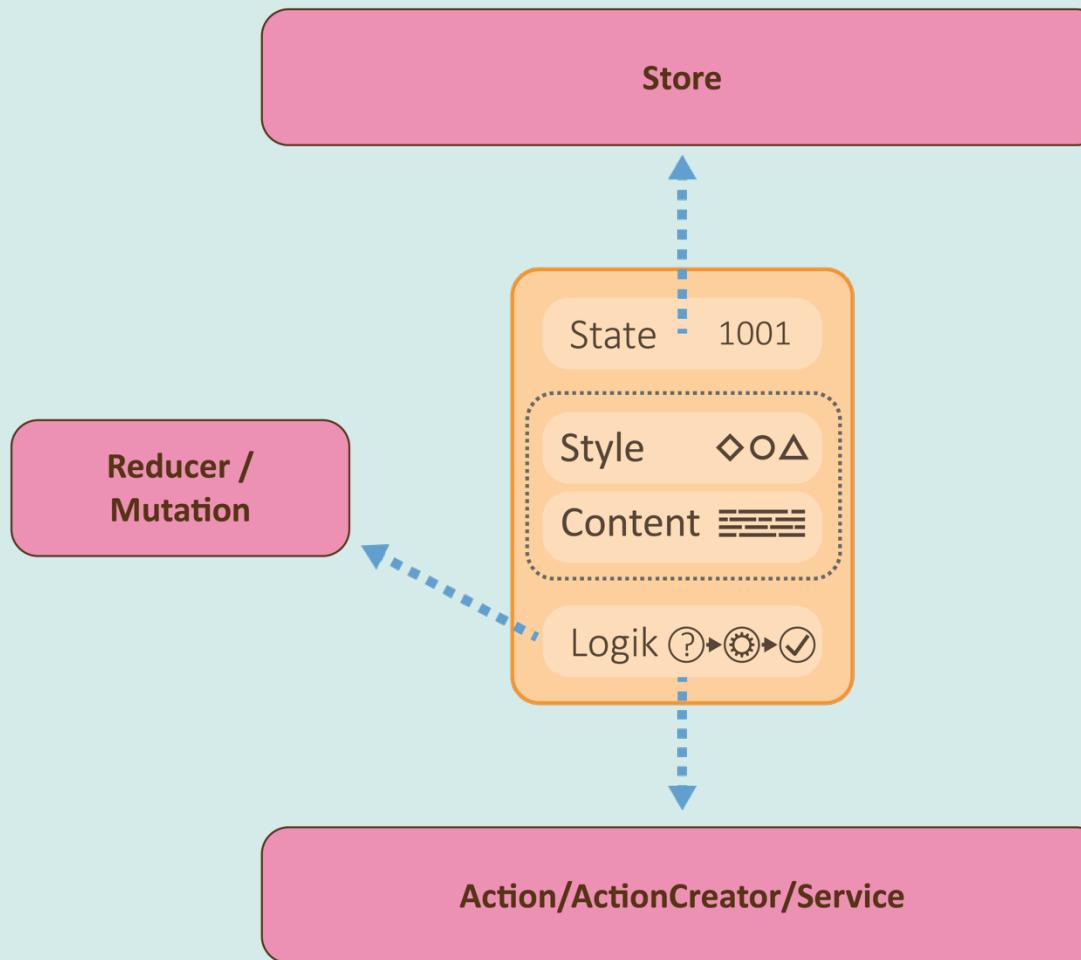
KOMPONENTEN

Externes Statemanagement



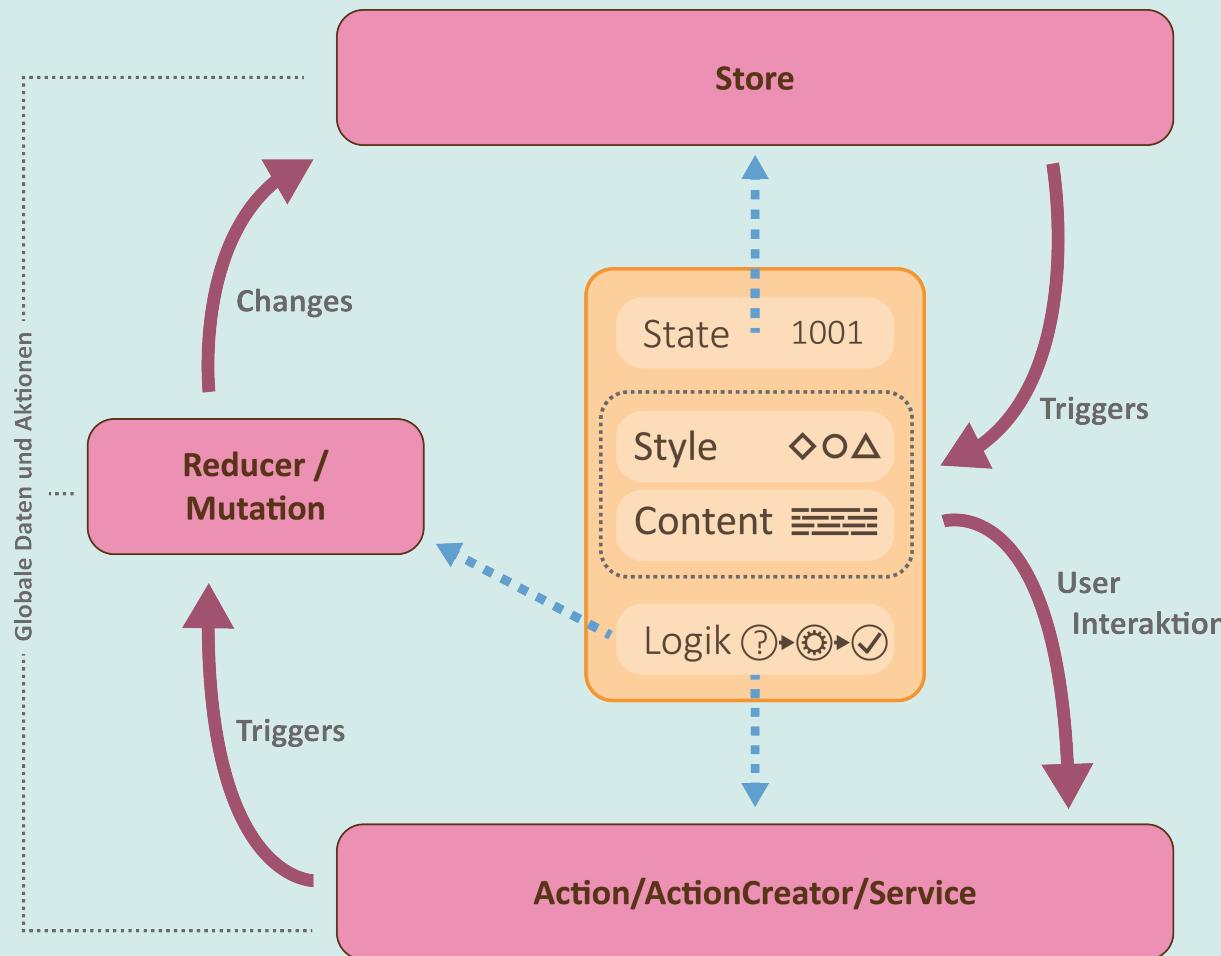
KOMPONENTEN

Externes Statemanagement



KOMPONENTEN

Externes Statemanagement



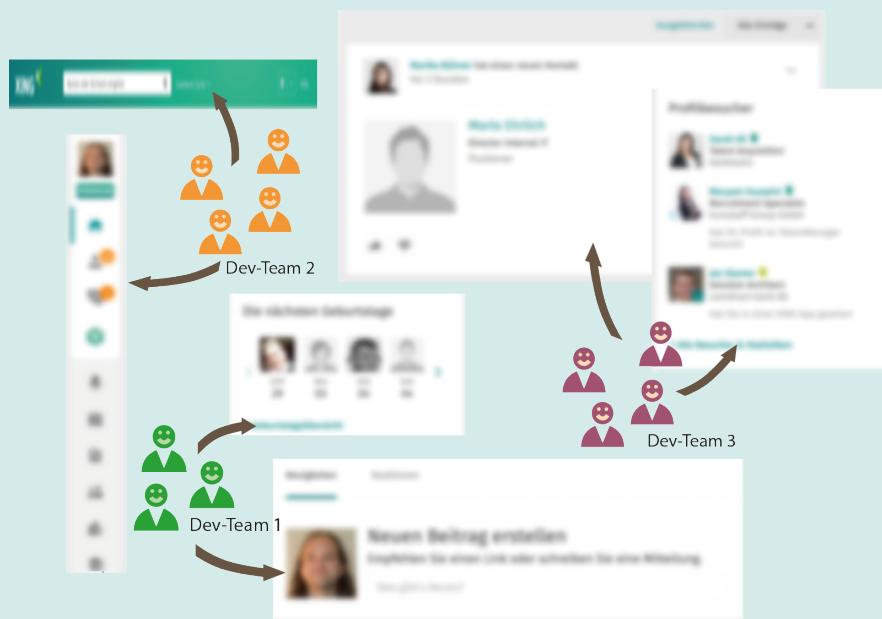
KOMPONENTEN

Grenzen dieser Architektur

- Sehr große Anwendungen
- Mehrere Teams
- Unabhängige Anwendungsteile, unabhängige Deployments
erwünscht

MODULARISIERUNG

Microfrontends: Aufteilen der Anwendung in kleine Teile

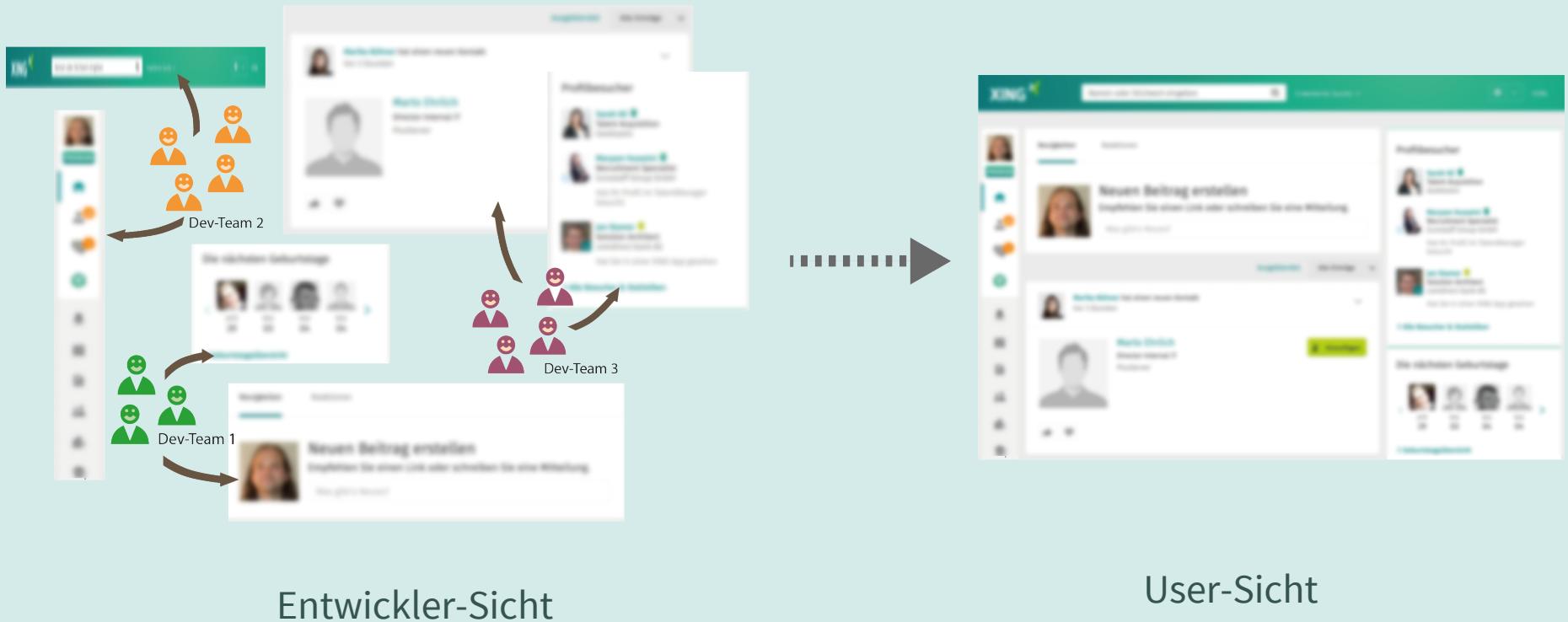


MODULARISIERUNG

Microfrontends: Aufteilen der Anwendung in kleine Teile

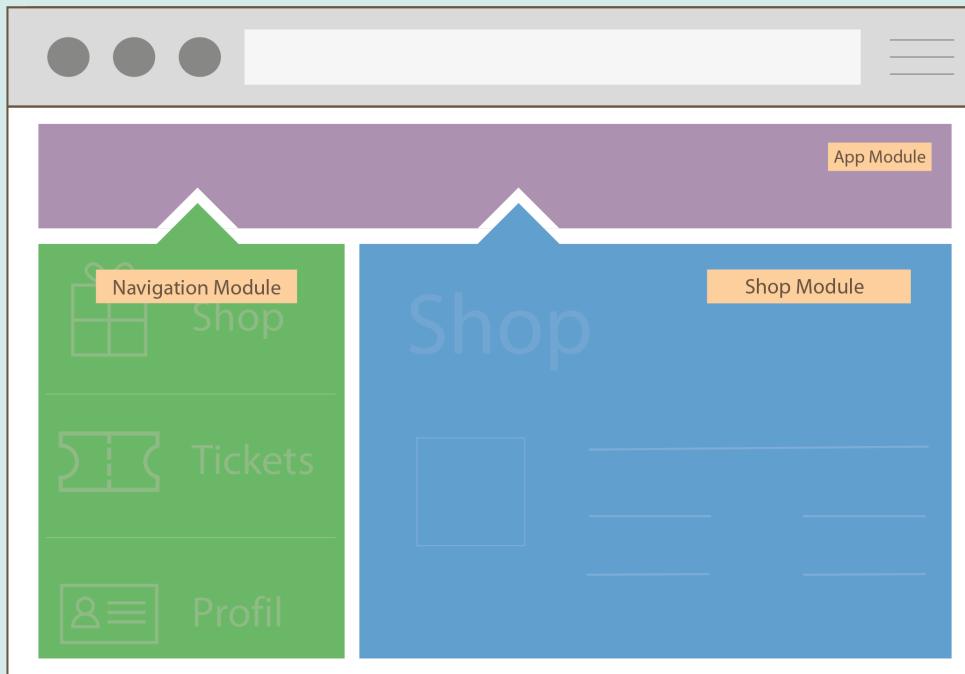
Widerspruch zwischen User-Sicht und Entwickler-Sicht

- wie wird die Anwendung wieder integriert?



MODULARISIERUNG

Modularer Monolith



ANWENDUNGSTEIL

Package (lerna)
Modul (npm)

INTEGRATION

Build

DEPLOYMENT

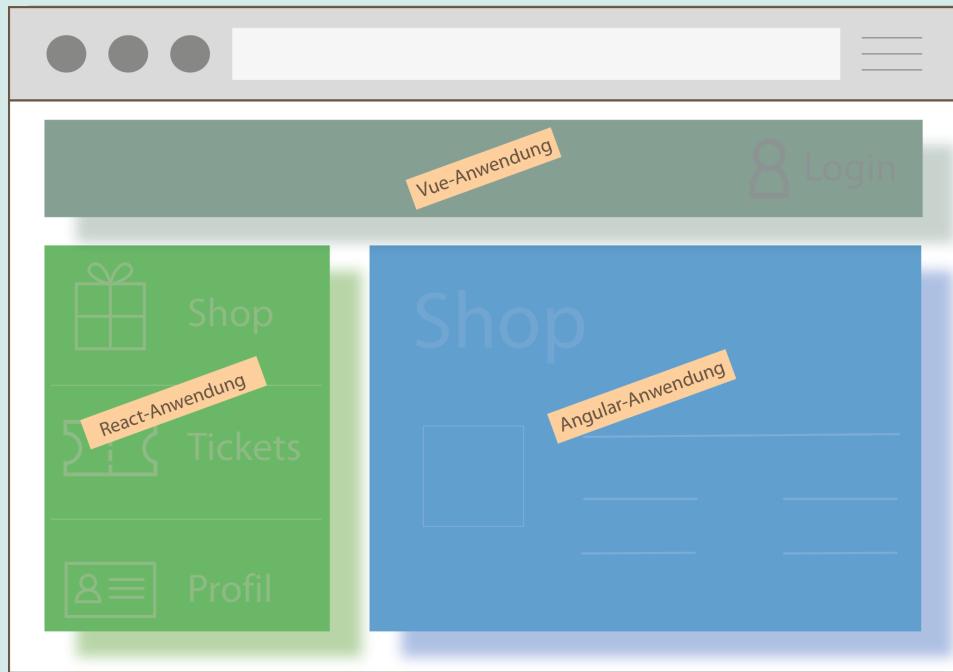
Monolith

SYNCHRONISATION

zentraler Zustand
Events

MODULARISIERUNG

Micro Components



ANWENDUNGSTEIL

SPA

INTEGRATION

Laufzeit/Browser
(iFrame, WebComponent)

DEPLOYMENT

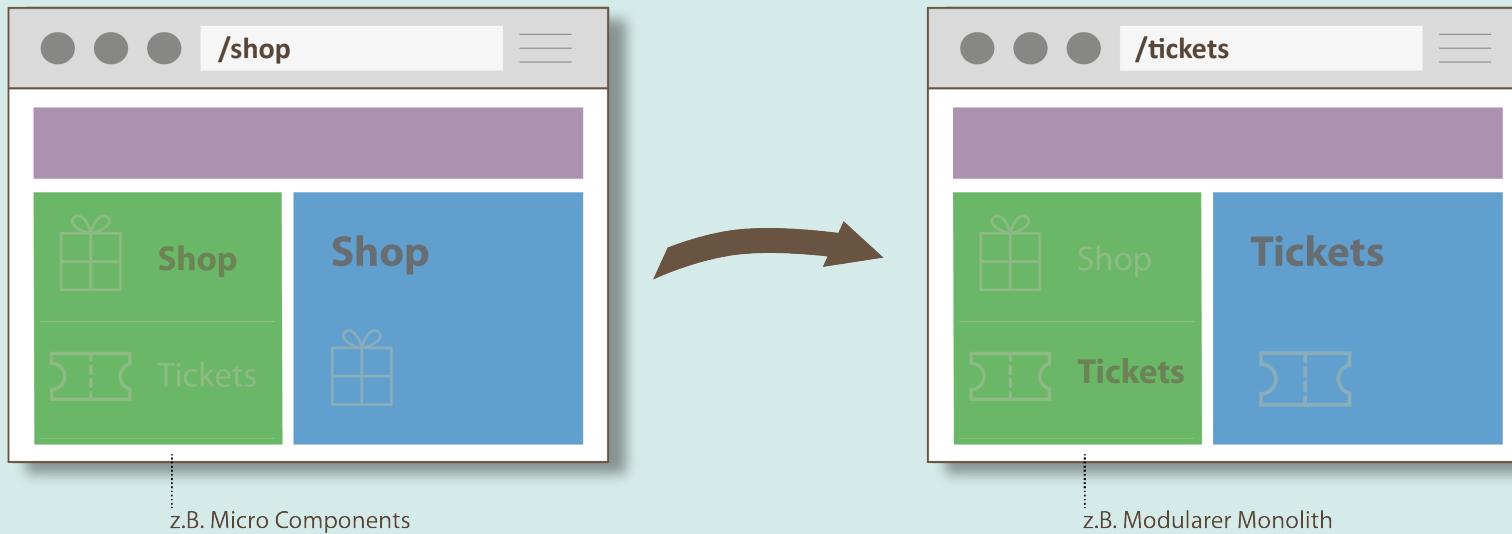
individuell
(jede SPA einzeln)

SYNCHRONISATION

Events
Cookies
Local Storage

MODULARISIERUNG

Links



ANWENDUNGSTEIL

SPA* oder
Klassische Web-App

INTEGRATION

Hyperlinks

DEPLOYMENT

individuell
(jede Teil einzeln)

SYNCHRONISATION

URL Parameter
Cookies
Local Storage

* z.B. als Micro Components
oder Modularer Monolith

Fazit

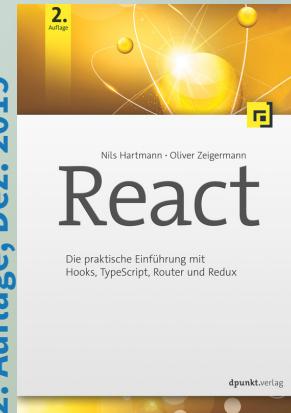
KEINE ANGST VOR DER SINGLE-PAGE-ANWENDUNG!

FAZIT

Keine Angst vor der Single-Page-Anwendung!

- Es hat sich in den letzten Jahren vieles geändert
 - und zum Guten gewendet
- Für viele Problemstellungen gibt es akzeptable Lösungen
 - Modernes JavaScript samt Modul-System
 - Typ-Checker (TypeScript)
 - Ausgereifte SPA-Frameworks
 - Etablierte Architektur-Muster für das Frontend
 - ...
- Gewissen Leidensbereitschaft müsst ihr trotzdem mitbringen

NILS HARTMANN
<https://nilshartmann.net>



2. Auflage, Dez. 2019

vielen Dank!

Slides: <https://nils.buzz/oose-keine-angst>

NILS@NILSHARTMANN.NET