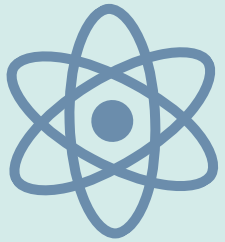
NILS HARTMANN | HTTPS://NILSHARTMANN.NET

SINGLE-PAGE-ANWENDUNGEN MIT

React

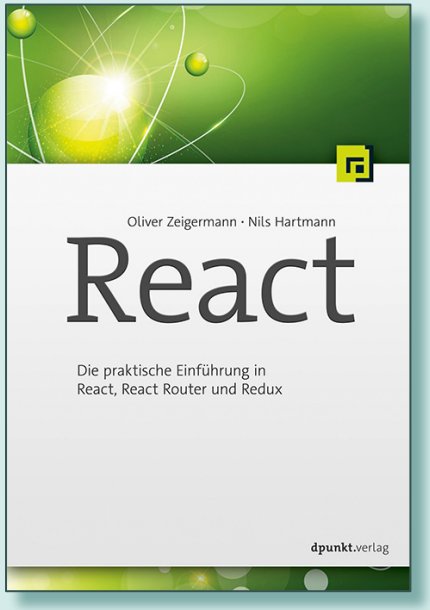


Slides: http://bit.ly/oose-react

NILS HARTMANN

Programmierer aus Hamburg

JavaScript, Java
Trainings und Workshops



@NILSHARTMANN

MODERNE WEB-ANWENDUNGEN

ANFORDERUNGEN

MODERNE WEB-ANWENDUNGEN

Aus Benutzersicht: bestes UI/UX

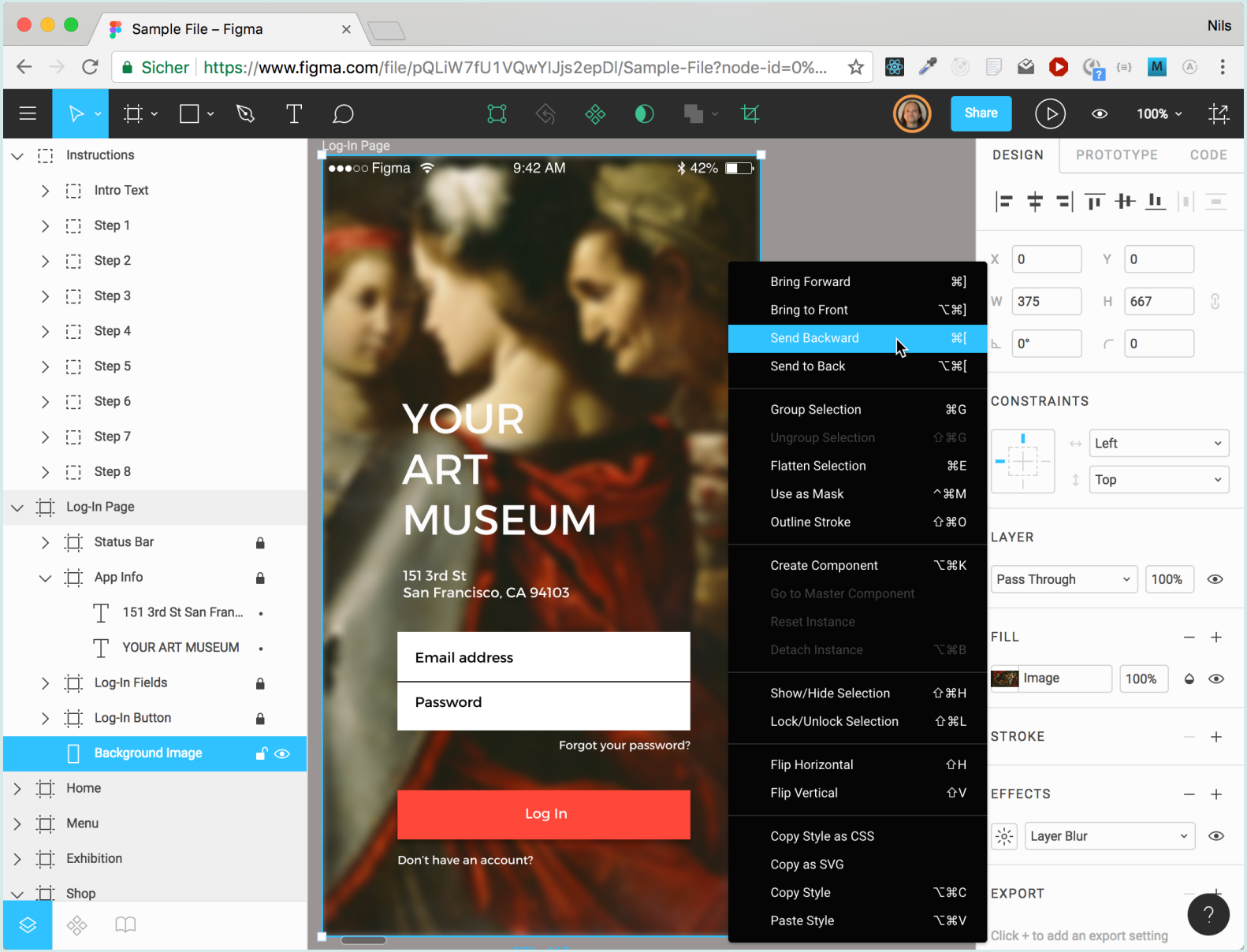
- Einheitliches Layout und Design
- Konsistentes Verhalten in der ganzen Anwendung
- Konsistente Darstellung der Daten

- Gewohntes Verhalten von Desktop Anwendungen
- Kurze Reaktionszeiten

Für Entwicklung

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

BEISPIELE



HTTPS://WWW.FIGMA.COM

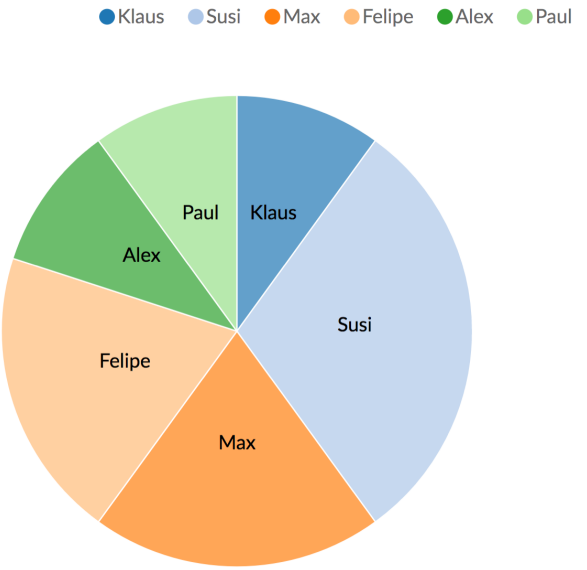
Greeting App

Showing 10 of 10 Greetings

Name	Greeting
Klaus	Moin
Susi	Hello!
Max	Bonjour
Susi	How are you?
Max	Bon soir
Felipe	Hola, ¿qué tal?
Alex	Happy Birthday
Felipe	¡buenos días
Paul	Wie gehts?
Susi	Have a nice day

(All greetings are shown. Click a row to filter)

Add

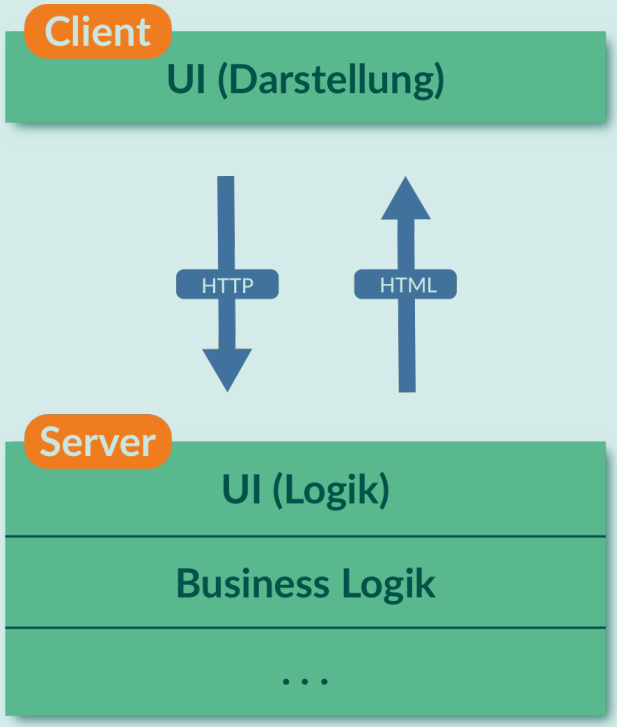


BEISPIEL: DIE GREETING APP

KLASSISCHE WEB-ANWENDUNGEN

RÜCKBLICK

RÜCKBLICK: KLASSISCHE WEB ANWENDUNG



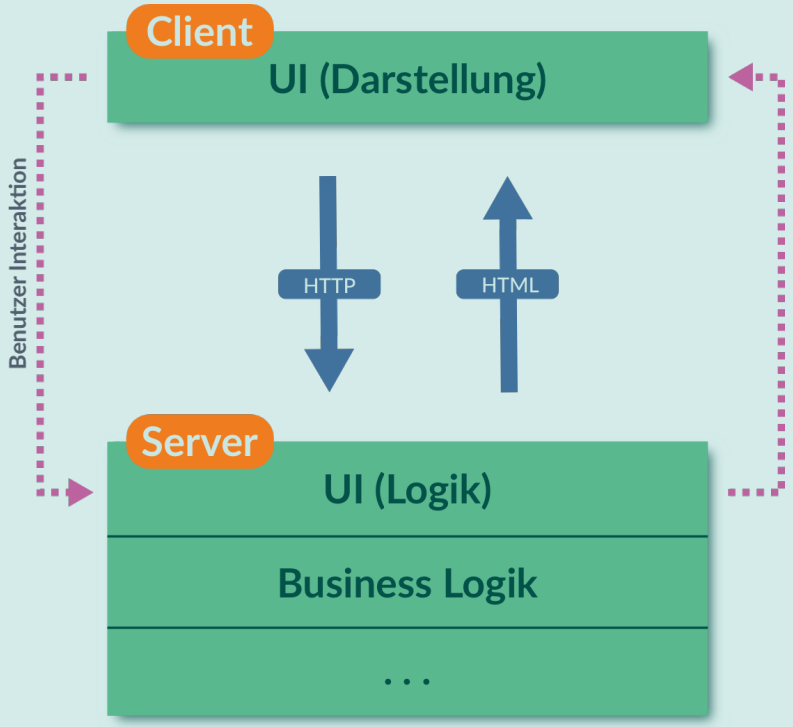
Technologie

- JSP, Thymeleaf, JSF
- jQuery

Klassische Web Anwendung:

- Bekannte Technologie und Sprache
- Kleinere Erweiterungen per JS/jQuery
  - "SPA light"

RÜCKBLICK: KLASSISCHE WEB ANWENDUNG



Technologie

- JSP, Thymeleaf, JSF
- jQuery

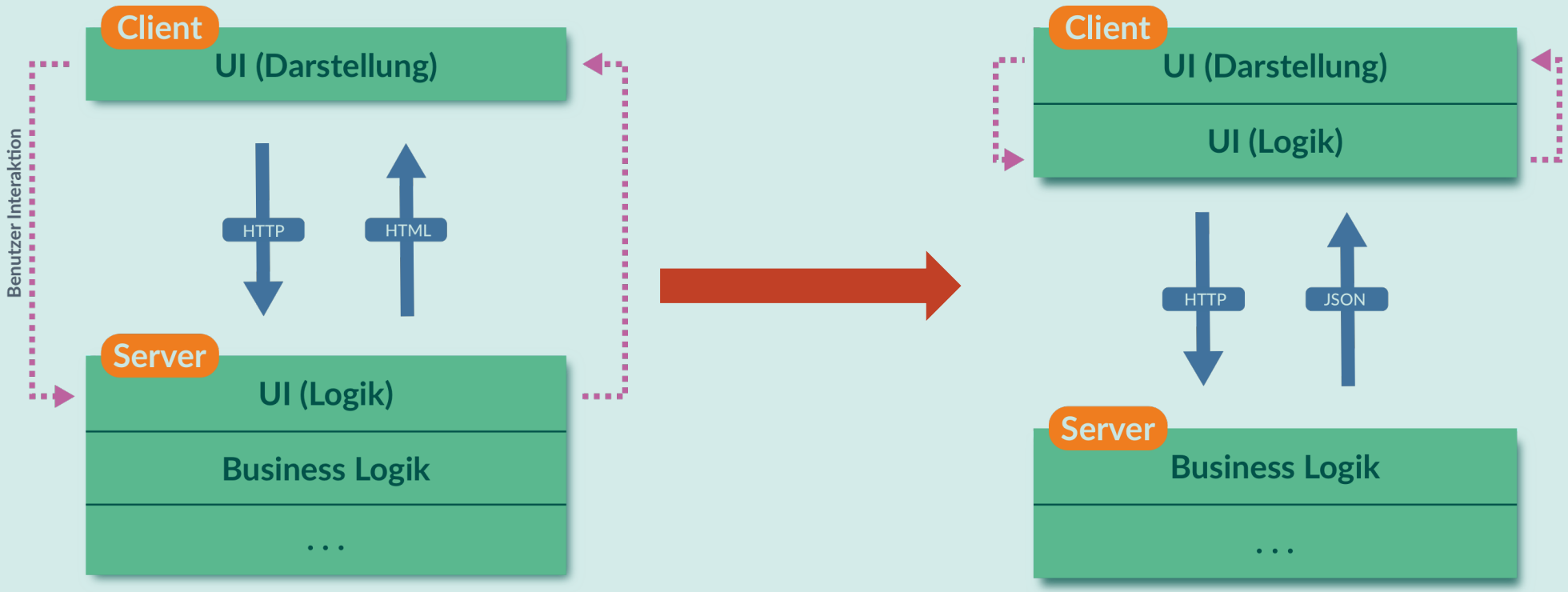
Klassische Web Anwendung:

- Bekannte Technologie und Sprache
- Kleinere Erweiterungen per JS/jQuery
  - "SPA light"

...aber problematisch:

- Lange round-trips durch Server Aufrufe
- Benutzer-Interaktion schwierig (ohne JavaScript)...
- ...deswegen wird JavaScript "dazu gehackt"
- Wartungshölle: welcher Code ist wo?

SINGLE-PAGE-APPLICATION



Technologie

- JSP, Thymeleaf, JSF
- jQuery

Technologie

- REST API
- React, Angular, Vue

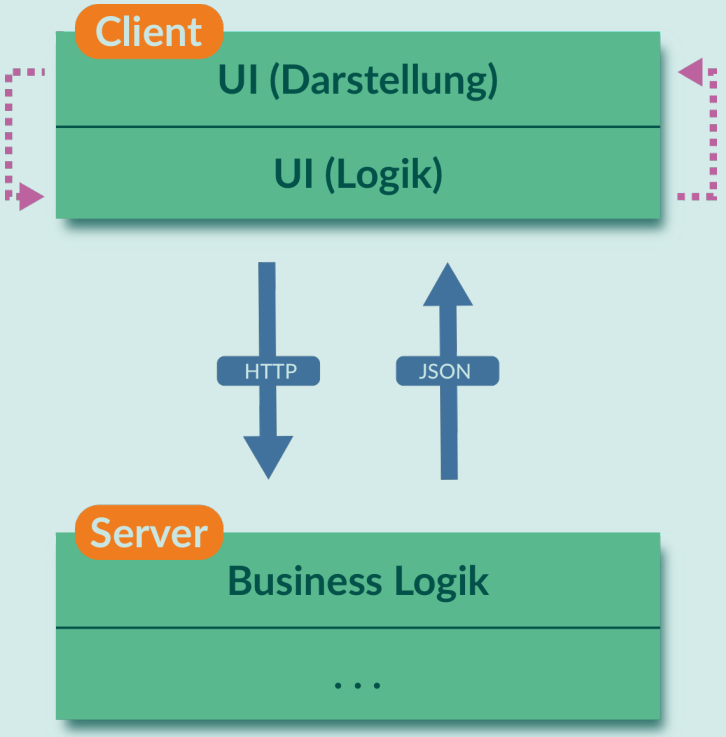
SINGLE-PAGE-APPLICATION

Single-Page-Anwendung

- Ermöglicht UI/UX wie auf dem Desktop
- Funktioniert sogar offline
  - Daten können im Browser (zwischen) gespeichert werden

JavaScript "first-class citizen"

- Klare Trennung der Verantwortlichkeiten nach Client und Server
- Entwicklung im / für den Browser
- Keine unnötigen Abstraktionen



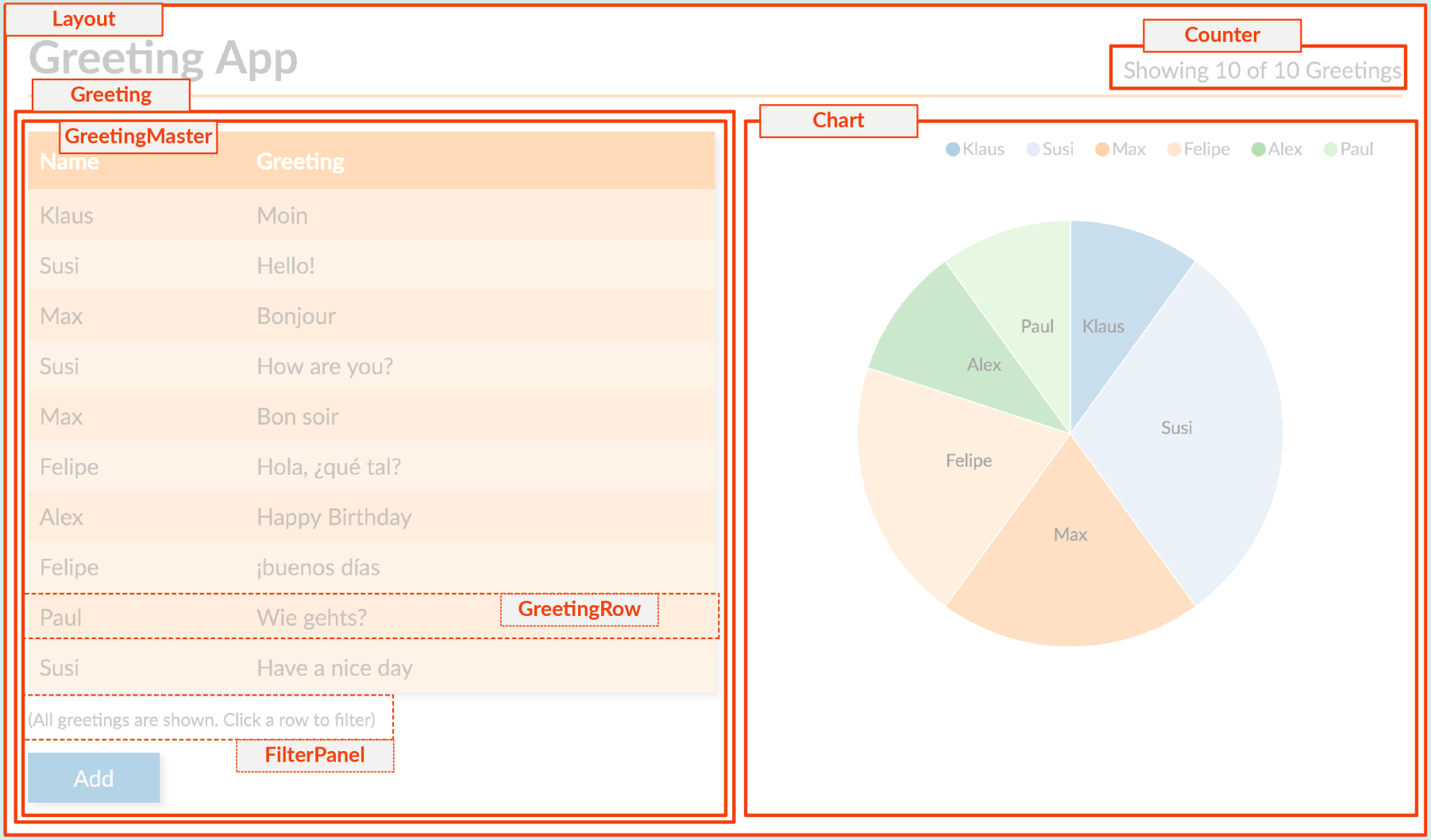
Technologie

- REST API
- React, Angular, Vue

"A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES"

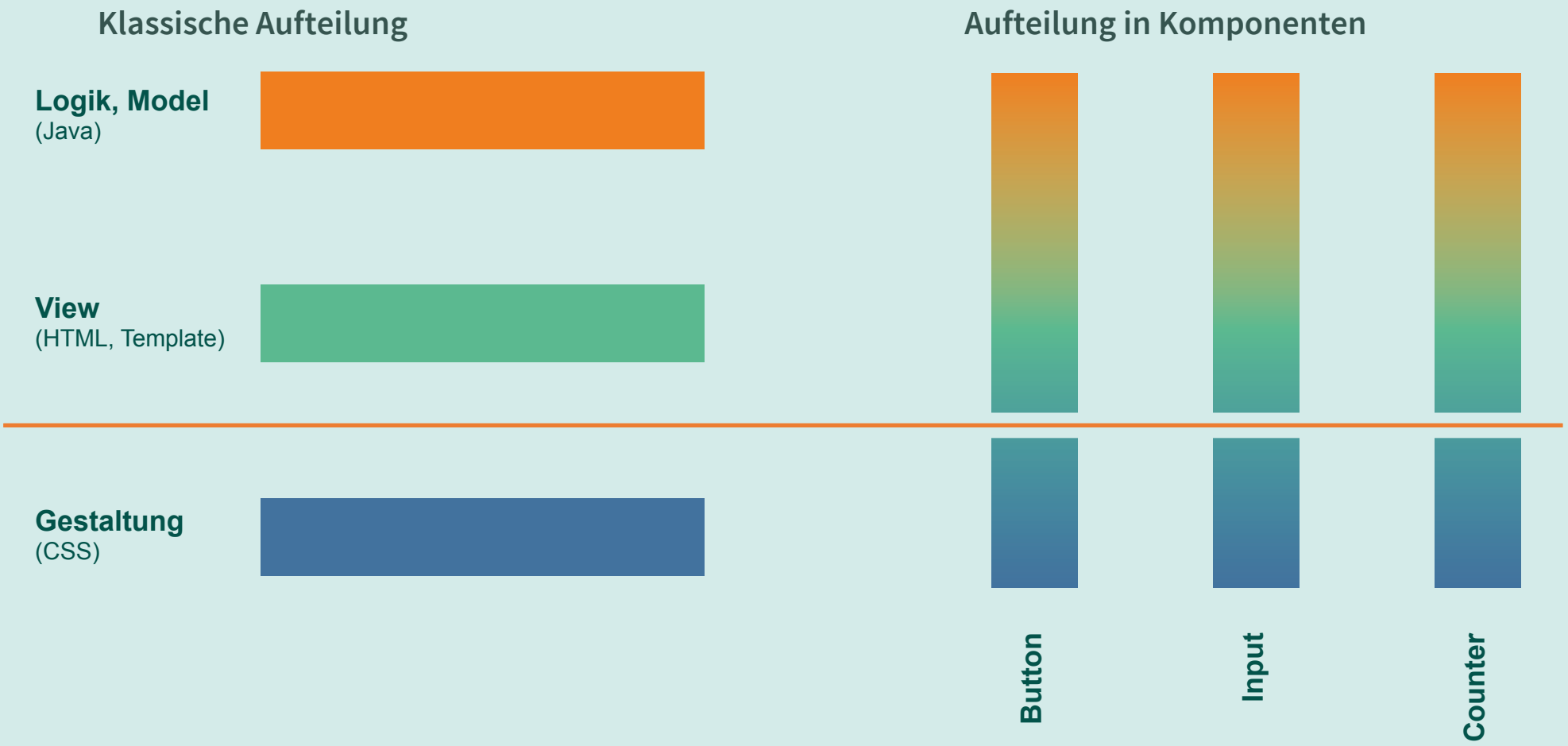
React

HTTPS://REACTJS.ORG/



GREETING APP: KOMPONENTEN

Komponenten in React



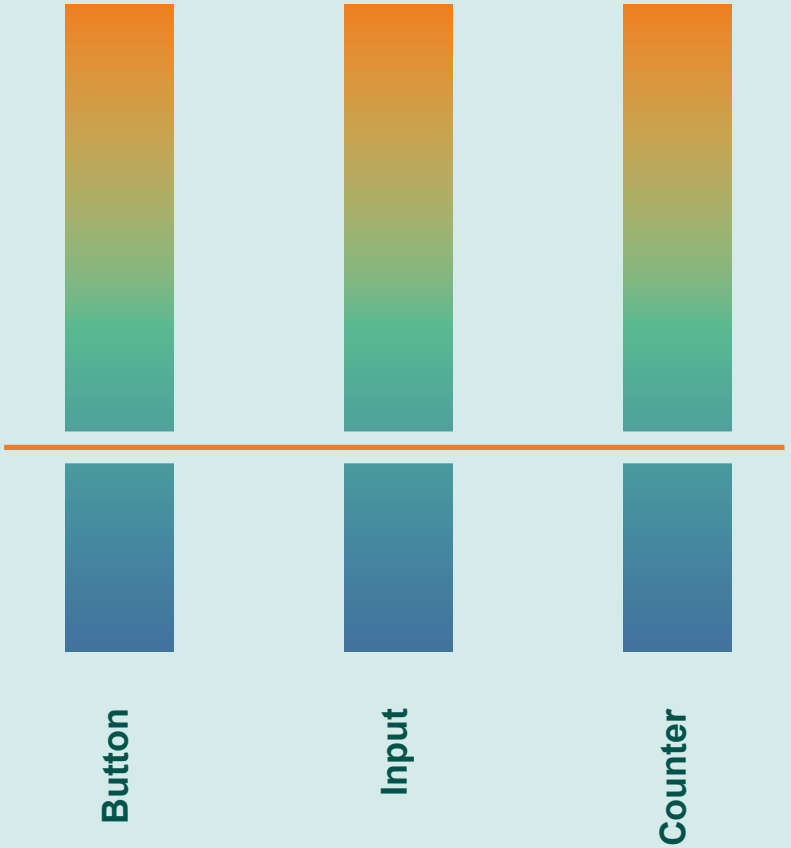
Grafik Inspiriert von: https://pbs.twimg.com/media/DCXJ_tjXoAAoBbu.jpg

SEPARATION OF CONCERNS

React-Komponenten

- bestehen aus Logik und UI
- werden deklarativ beschrieben
- keine Templatesprache
- werden immer komplett gerendert

- können auf dem Server gerendert werden („universal webapps“)



EINE EINFACHE REACT KOMPONENTE

Komponente

Showing 3 of 11 Greetings

Counter.js

```
function Counter({filtered, total}) {
  return filtered === total ?
    <div>Showing all {total} Greetings</div>
  :
    <div>Showing {filtered} of {total} Greetings</div>
}
```

Verwendung

```
<Counter filtered={3} total={11} />
```

APPLIKATIONEN WERDEN AGGREGIERT

Layout.js

```
import Counter from './Counter';
import Greeting from './Greeting';
import Chart from './Chart';

function Layout() {
  return
    <div className="Main">
      <div className="Title">
        <Counter filtered={3} total={11} />
      </div>
      <div className="Left">
        <Greeting />
      </div>
      <div className="Right">
        <Chart />
      </div>
    </div>
}
```

KOMPONENTE EINBINDEN

index.html

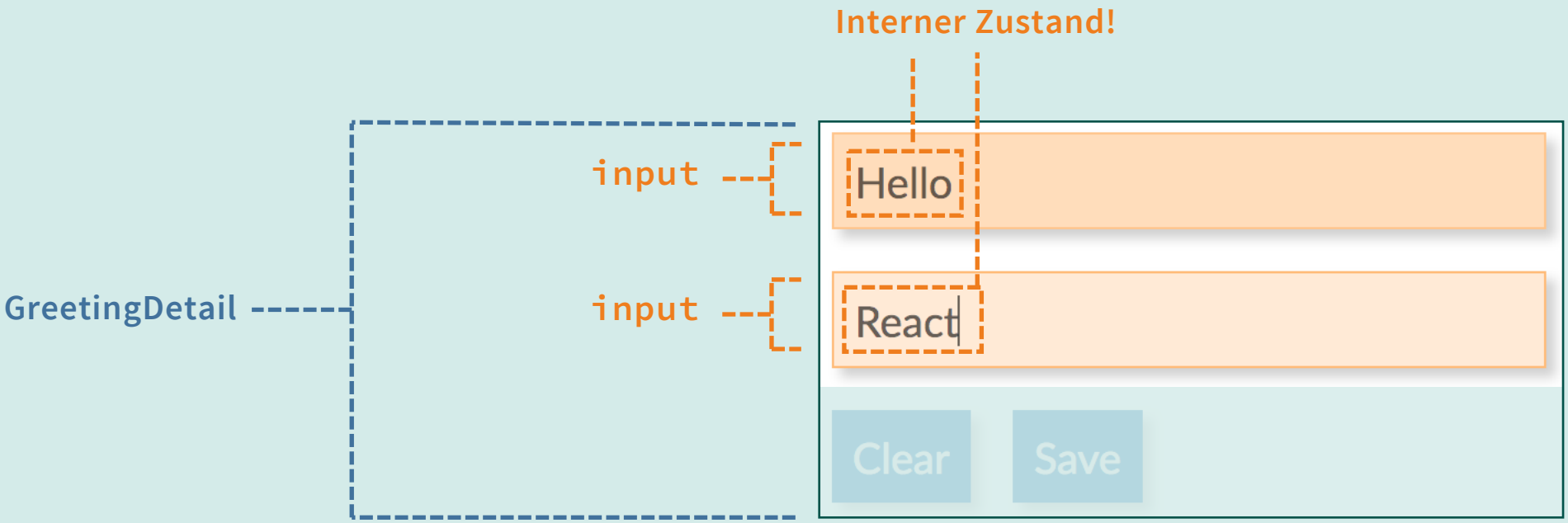
```
<html>
  <head>. . .</head>
  <body>
    <div id="mount"></div>
  </body>
  <script src="dist/app.js"></script>
</html>
```

app.js

```
import React from 'react';
import ReactDOM from 'react-dom';

import Layout from './Layout';

ReactDOM.render(
  <Layout />,
  document.getElementById('mount')
);
```

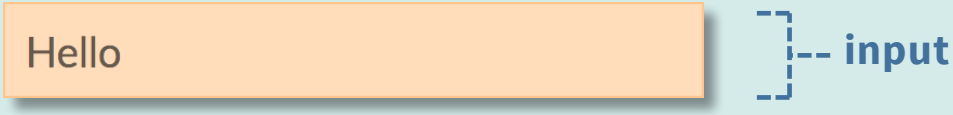
State

BEISPIEL: EINGABEFELD



ZUSTAND: EINGABEFELD

Komponente

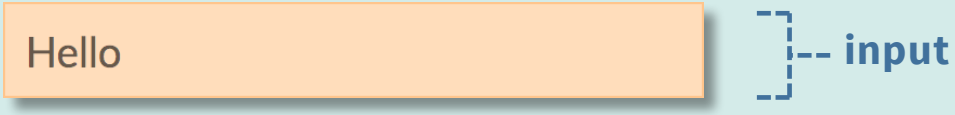


1. Input mit Wert aus State befüllen

```
class GreetingDetail extends React.Component {
  render() {
   return <div>
     <input
       value={this.state.greeting}

     />
     <input .../>
   </div>;
  }



}
```

ZUSTAND: EINGABEFELD

Komponente




```
class GreetingDetail extends React.Component {
  render() {
   return <div>
     <input
       value={this.state.greeting}
       onChange={e=>this.onGreetingChange(e.target.value)}
     />
     <input .../>
   </div>;
  }

  onGreetingChange(newGreeting) {

  }
}
```

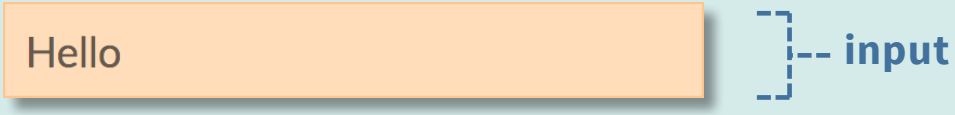
1. Input mit Wert aus State befüllen

2a. Event Handler registrieren

2b. Event Handler

ZUSTAND: EINGABEFELD

Komponente




```
class GreetingDetail extends React.Component {
  render() {
   return <div>
     <input
       value={this.state.greeting}
       onChange={e=>this.onGreetingChange(e.target.value)}
     />
     <input .../>
   </div>;
  }

  onGreetingChange(newGreeting) {
    this.setState({greeting: newGreeting});
  }
}
```

1. Input mit Wert aus State befüllen

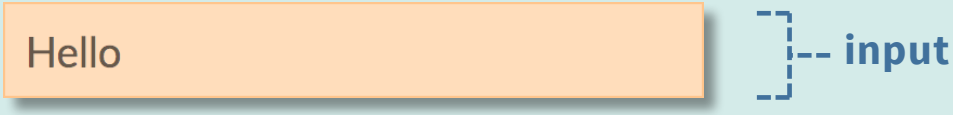
2a. Event Handler registrieren

2b. Event Handler

3. Zustand neu setzen

ZUSTAND: EINGABEFELD

Komponente




```
class GreetingDetail extends React.Component {
  render() {
   return <div>
     <input
       value={this.state.greeting}
       onChange={e=>this.onGreetingChange(e.target.value)}
     />
     <input .../>
   </div>;
  }

  onGreetingChange(newGreeting) {
    this.setState({greeting: newGreeting});
  }
}
```

Neu rendern

1. Input mit Wert aus State befüllen

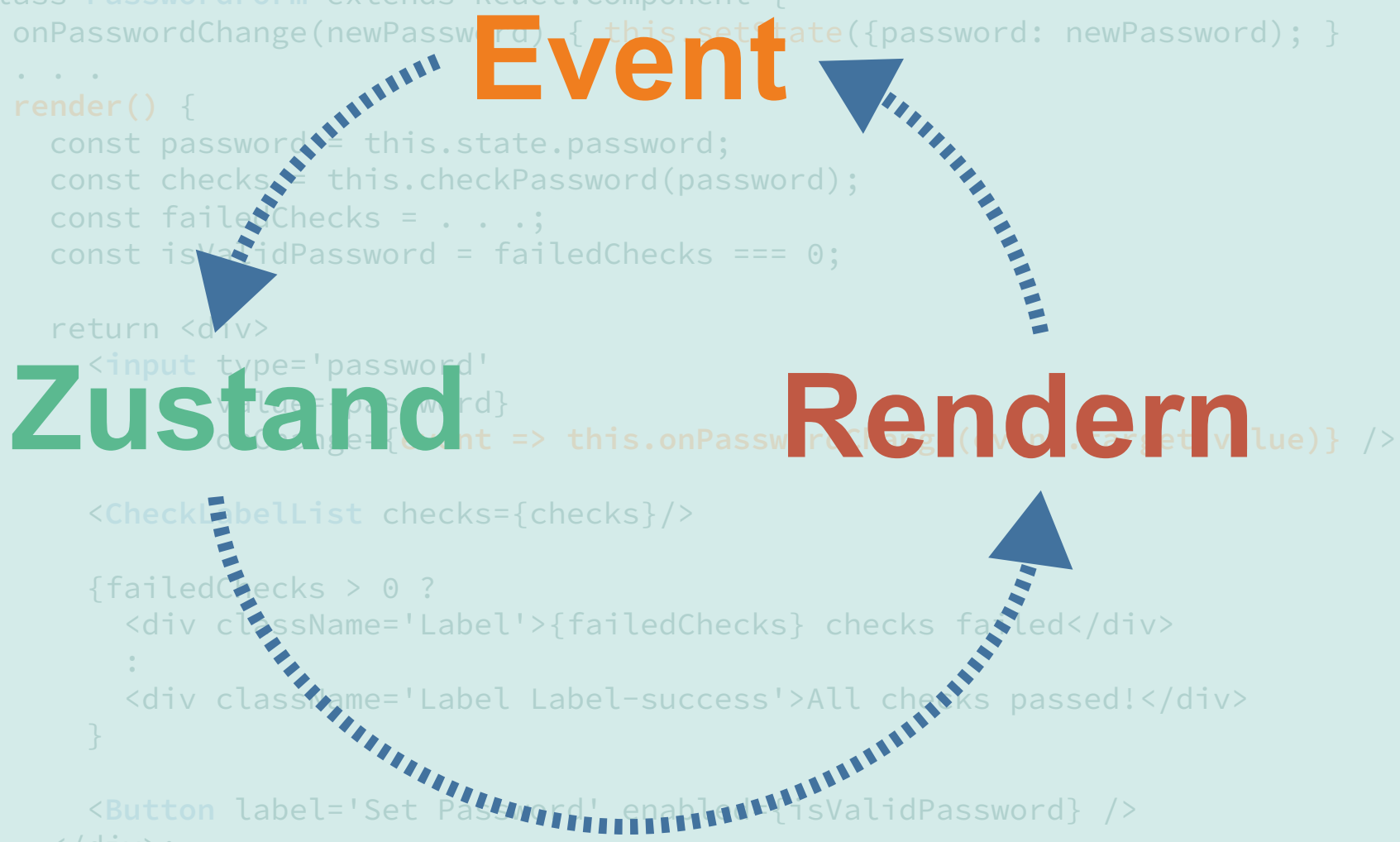
2a. Event Handler registrieren

Event

2b. Event Handler

3. Zustand neu setzen

REACT: UNI DIRECTIONAL DATAFLOW



RESPOND TO EVENTS & RENDER UI

RENDERING VON KOMPONENTEN

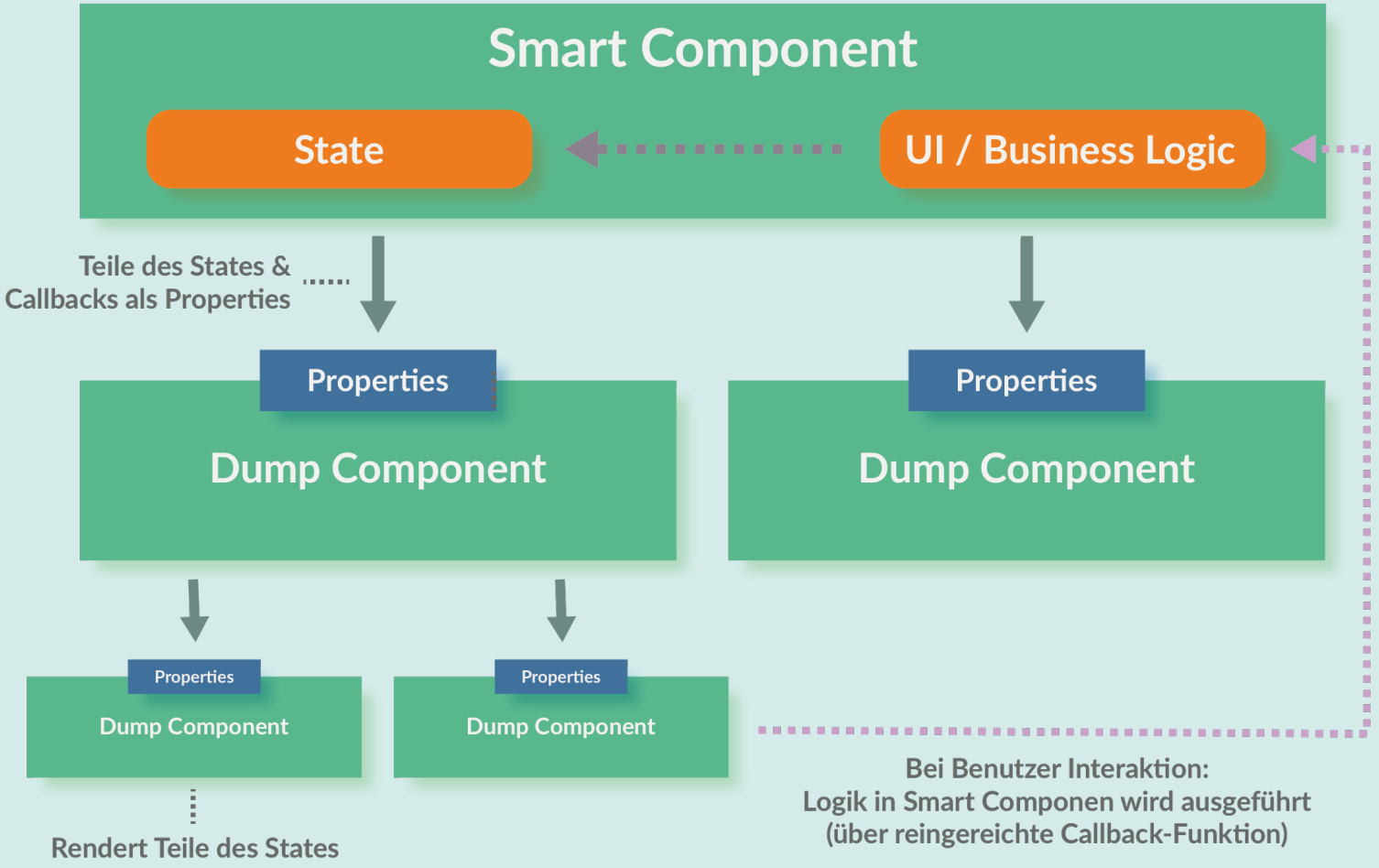
Gerendert wird immer die ganze Komponente

- Inklusive aller Unterkomponenten
- Bei Zustandsänderung
- Vermeidet Inkonsistenzen


```
class GreetingDetail extends React.Component {
  render() {
    const saveDisabled = !(this.state.name && this.state.greeting);

    return <div>
      <input ... />
      <input .../>
      <button disabled={saveDisabled}>Save</button>
    </div>;
  }
}
```

TYPISCHE ARCHITEKTUREN

"SMART UND DUMP COMPONENTS"

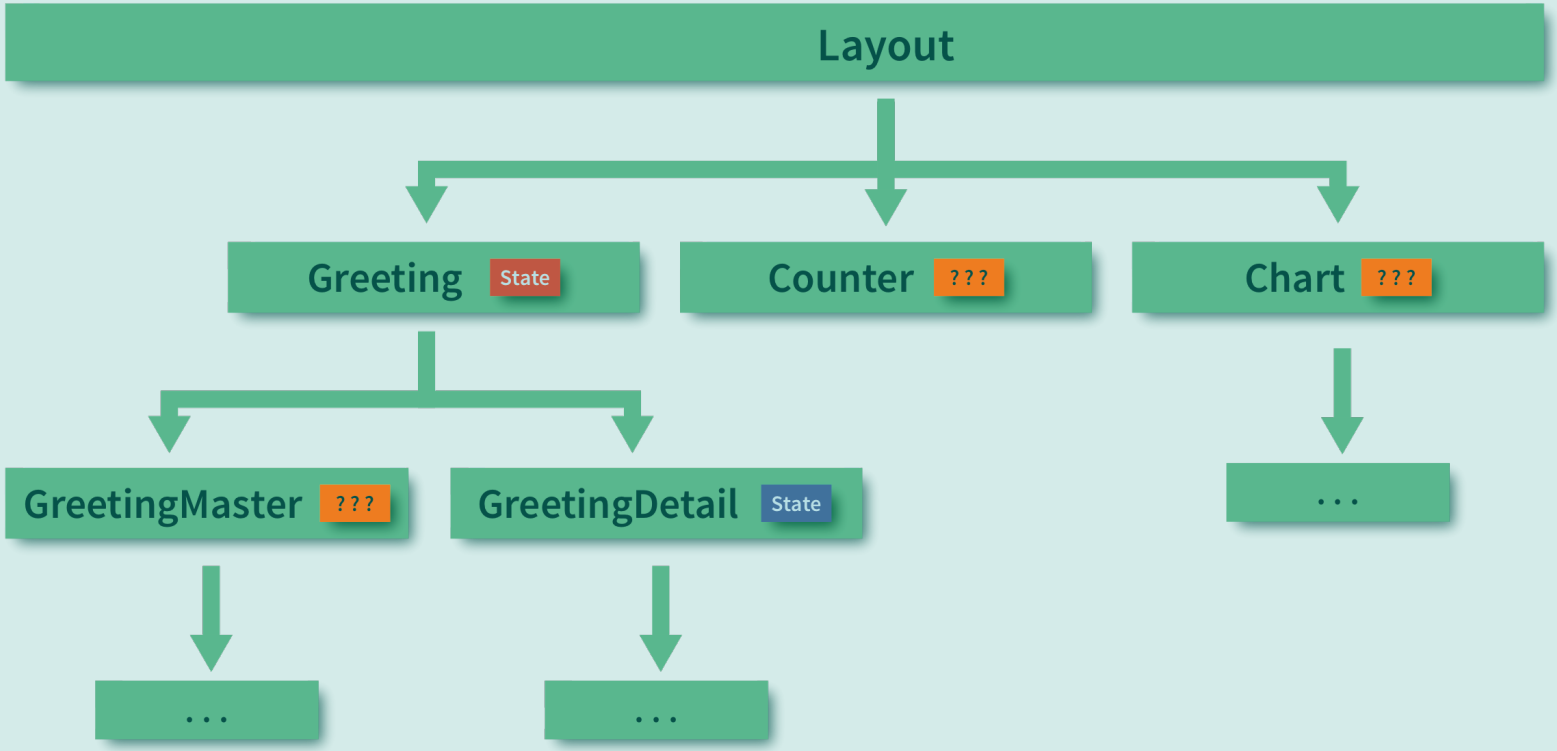


Typische React Anwendungen: Hierarchisch aufgebaut

- Kommunikation über Properties und **Callbacks**

TEIL 2

KOMPONENTENHIERARCHIEN



Probleme:

- Wohin mit gemeinsamen Zustand? (Greetings in 3 Komponenten!)
- Verteilter Zustand vs "Gott-Komponenten"
- Kopplung UI und Fach-Logik

REDUX

EXTERNES STATE MANAGEMENT

EXTERNES STATE MANAGEMENT

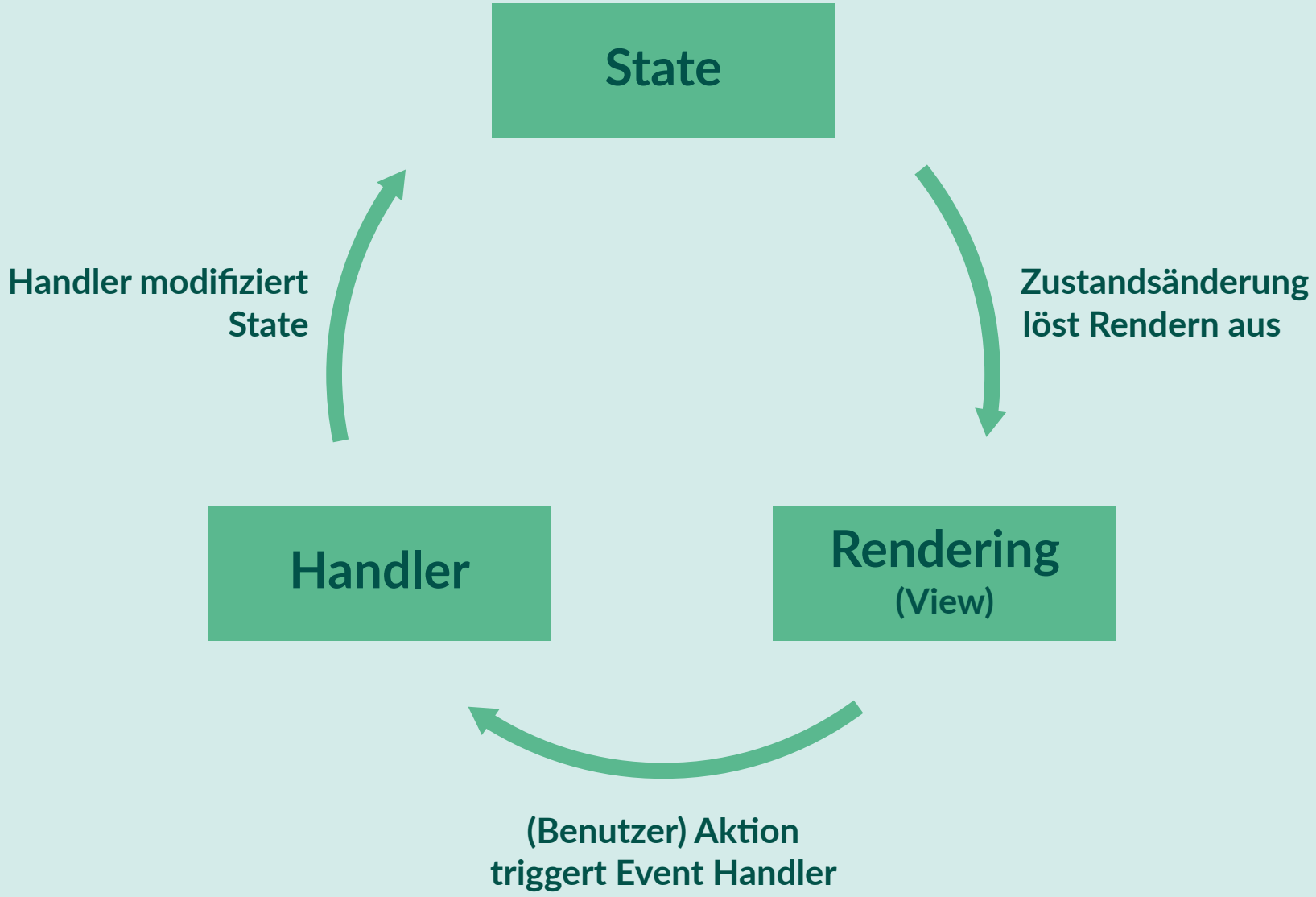
Beispiel: Redux

- Extern: Zustand wandert aus den UI Komponenten
- Zentral: Zustand wandert in einen Store

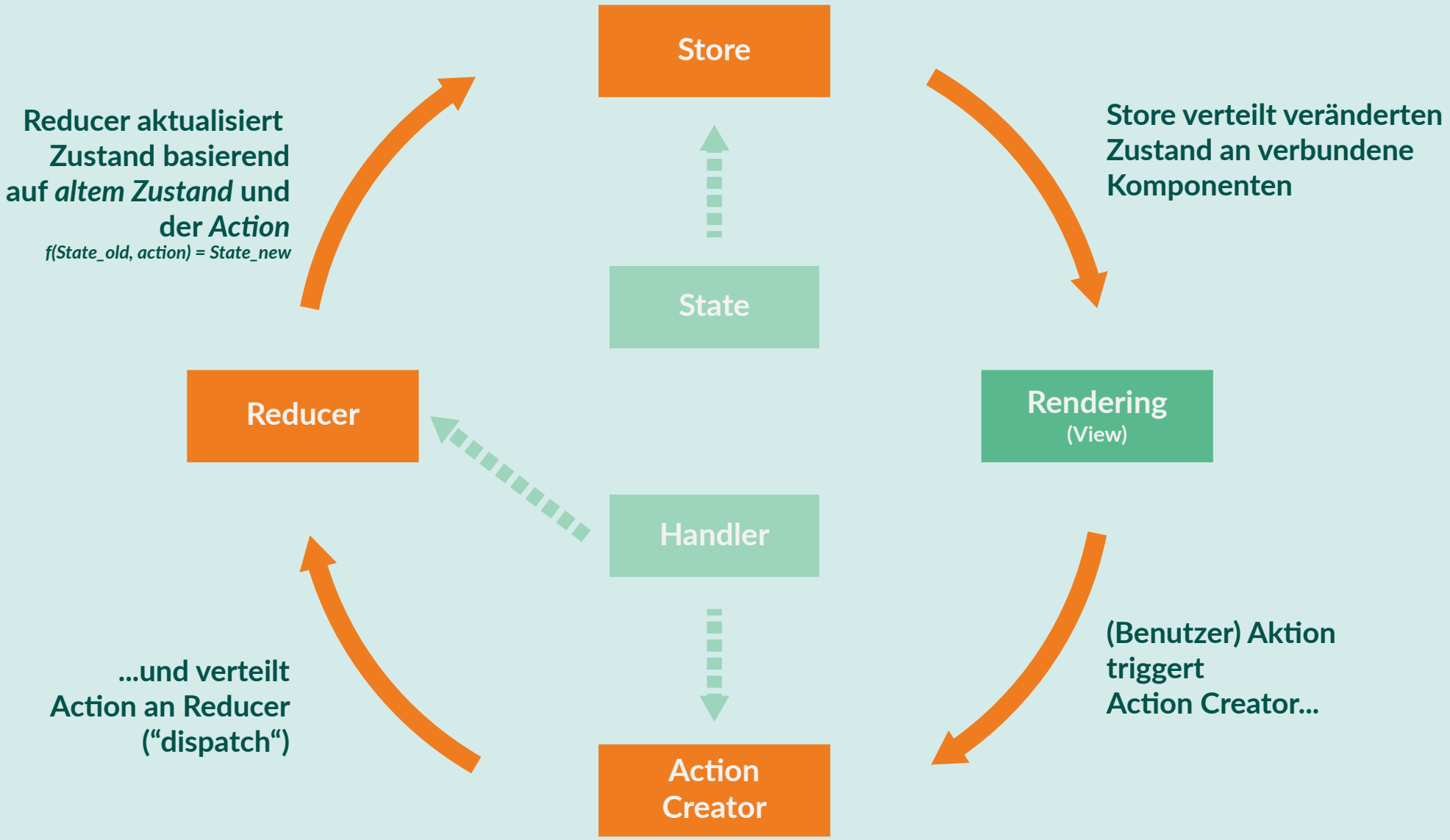
- Architektur Pattern und Implementierung

- React-unabhängig
  - Bindings u.a. auch für Angular und Vue

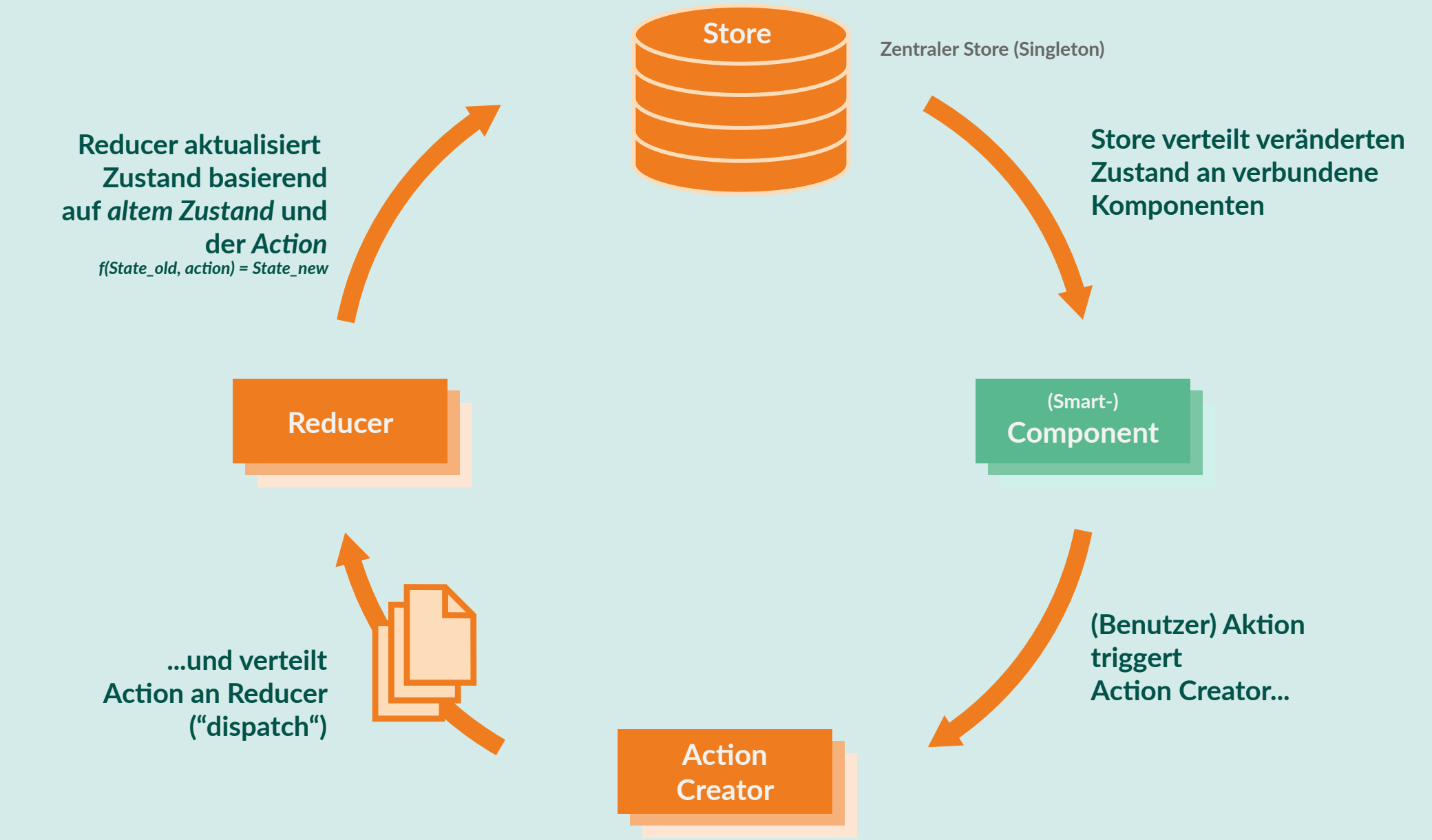
REACT RENDER ZYKLUS



REDUX-BASIERTE REACT-ANWENDUNG



REDUX ARCHITEKTUR



"JavaScript that scales"

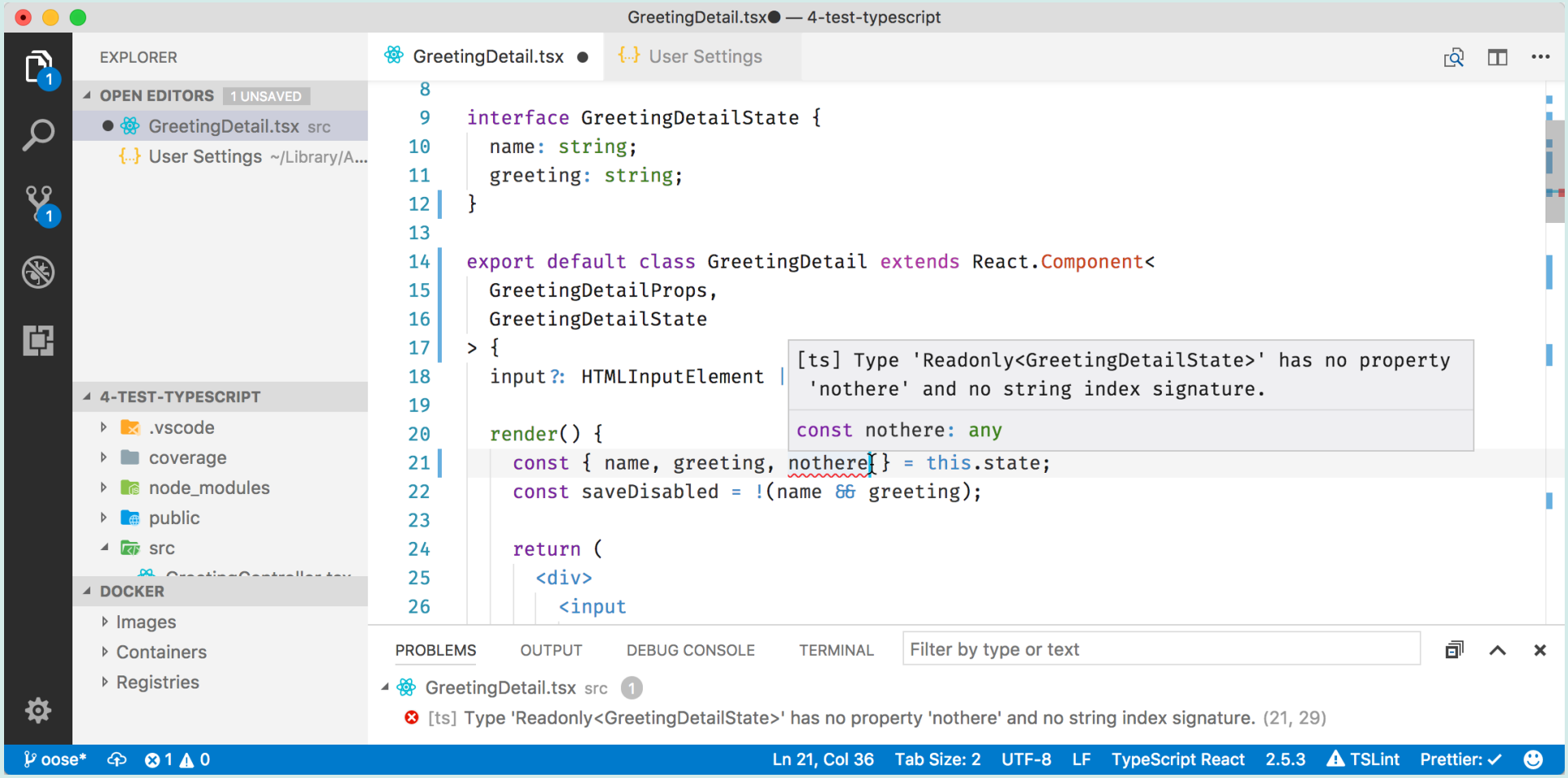
TypeScript

FÜR REACT-ANWENDUNGEN

TYPESCRIPT DEMO

Typen für React Komponenten

Fehlermeldung und Code Completion in der IDE



Vielen Dank!

Slides: http://bit.ly/oose-react

Fragen?

Weitere Beispiele

TYPISCHE USE-CASES

BEISPIEL: DATEN VOM SERVER LADEN

```
class GreetingController extends React.Component {

  // 1. Komponente wurde im DOM gerendert
  async componentDidMount() {
    const response = await fetch("/api/greetings");
    const greetingsAsJson = await response.json();

    // 2. Daten sind geladen => State setzen => neu rendern
    this.setState({greetings: greetingsAsJson });
  }

  render() {

    // 3. Daten anzeigen
    return <GreetingMaster greetings={this.state.greetings} />;
  }
}
```

BEISPIEL: ROUTING UND DEEP LINKS

```
import GreetingMaster from "./GreetingMaster";
import GreetingDisplay from "./GreetingDisplay";
import NotFound from "./NotFound";
import {HashRouter as Router, Route, Switch} from "react-router-dom";

function Layout() {
  return (
    <Router>
      <div>
        <h1>Greetings</h1>
        <Switch>
          <Route path="/greet/:greetingId" component={GreetingDisplay} />
          <Route path="/" component={GreetingMaster} />
          <Route component={NotFound} />
        </Switch>
      </div>
   </Router>
  );
}
```