NILS HARTMANN

Single-Page-Anwendungen mit

React

Eine praktische Einführung

Slides: https://nils.buzz/conceptpeople-react

NILS HARTMANN

nils@nilshartmann.net

Freiberuflicher Entwickler, Architekt, Trainer aus Hamburg

Trainings, Workshops und Beratung

Java
JavaScript, TypeScript
React
GraphQL

React

Grundlagen, fortgeschrittene Techniken und Praxistipps

2. Auflage, Dezember 2019

HTTPS://REACTBUCH.DE

REACT

https://reactjs.org

- Minimales API (?)
- Minimales Feature Set

REACT

https://reactjs.org

- Minimales API (?)
- Minimales Feature Set

👉 Man kann (muss ?) viele Entscheidungen selber treffen

👉 Sehr gut integrierbar in vorhandenen Technologie Stack

REACT

https://reactjs.org

- Minimales API (?)
- Minimales Feature Set

👉 Man kann (muss ?) viele Entscheidungen selber treffen

👉 Sehr gut integrierbar in vorhandenen Technologie Stack

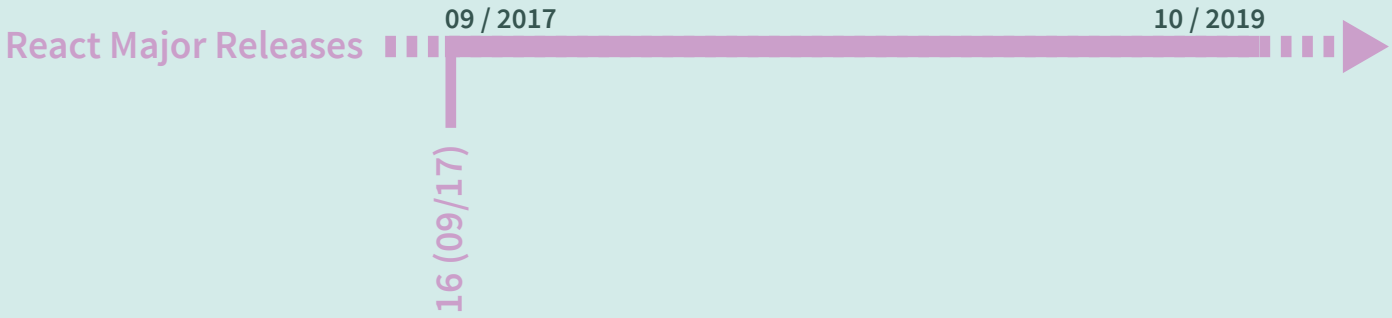
- Bewusste Verstöße gegen Best Practices

React ist sehr stabil

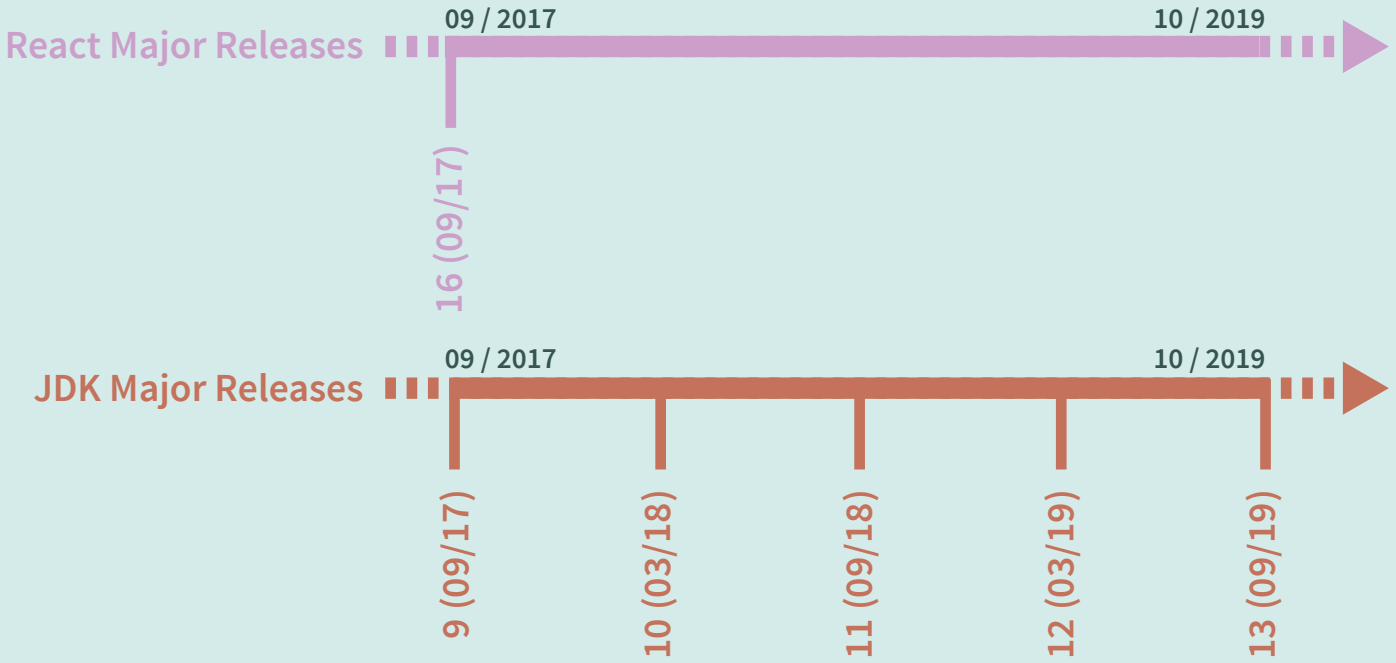
- Erste 16er-Version im September 2017
- Seitdem nur Minor-Versionen
- Trotzdem sehr viele neue Features, inklusive neuer Hooks API

👉 Gut geeignet für langlaufende Anwendungen

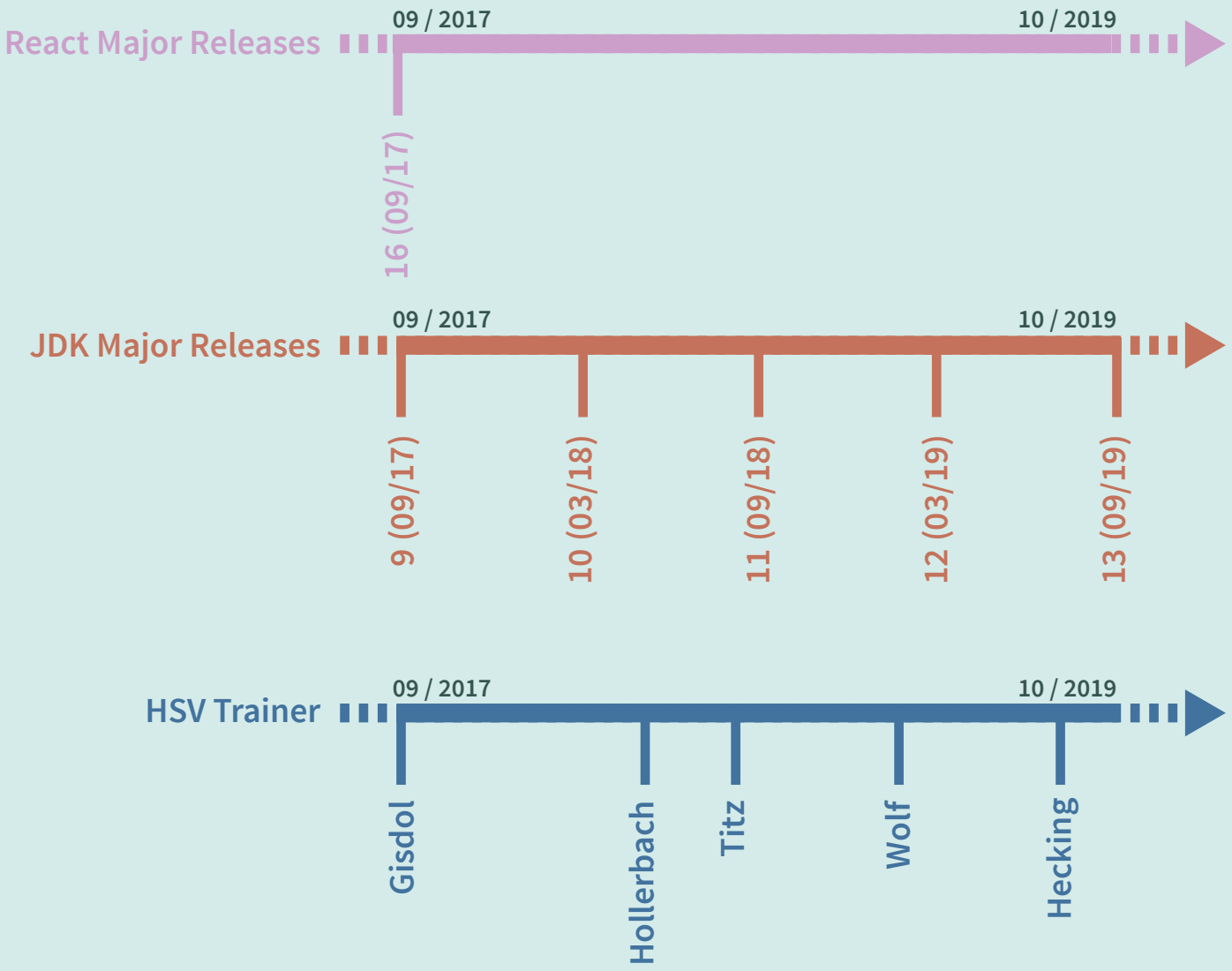
REACT - STABILITÄT



REACT - STABILITÄT



REACT - STABILITÄT



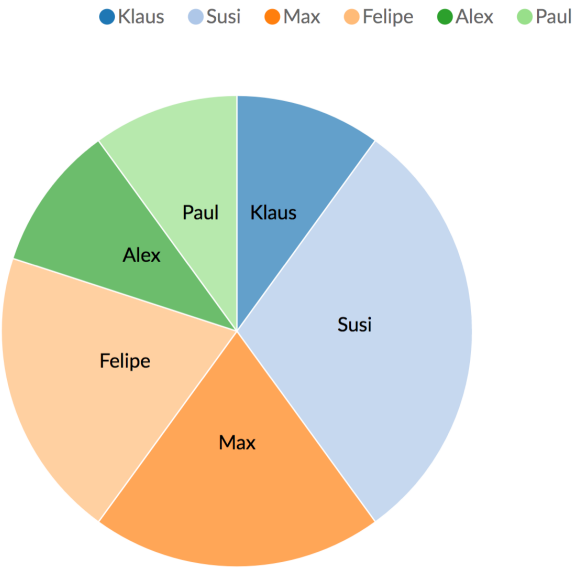
Greeting App

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)

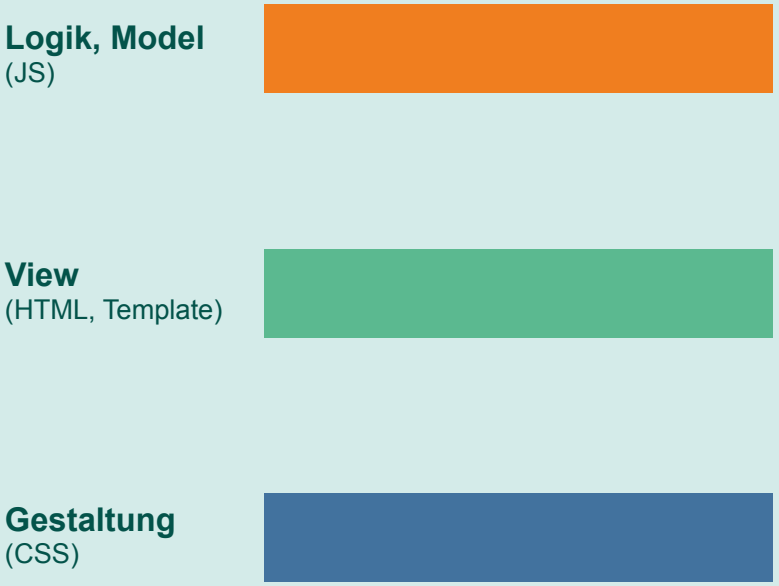
Add



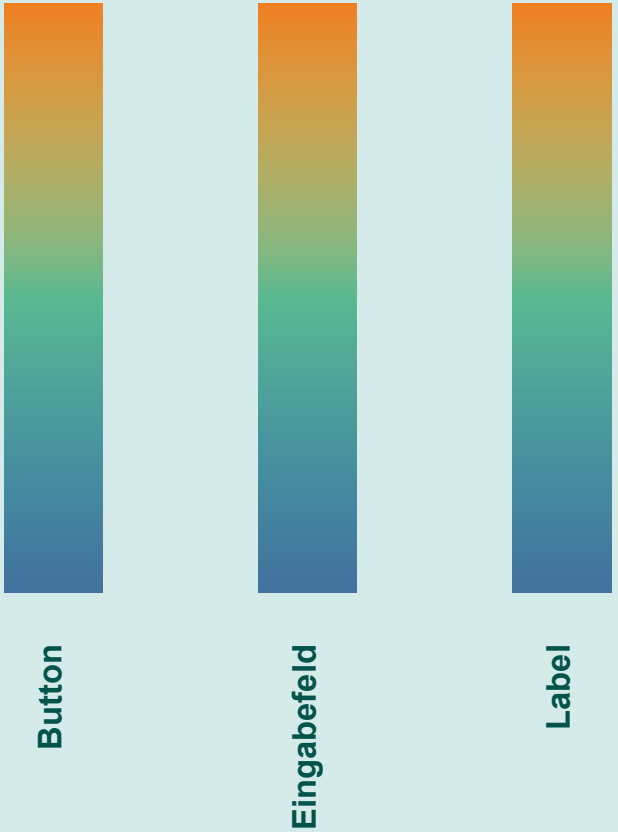
BEISPIEL: DIE GREETING APP

RETHINKING BEST PRACTICES

Klassische Aufteilung



Aufteilung in Komponenten



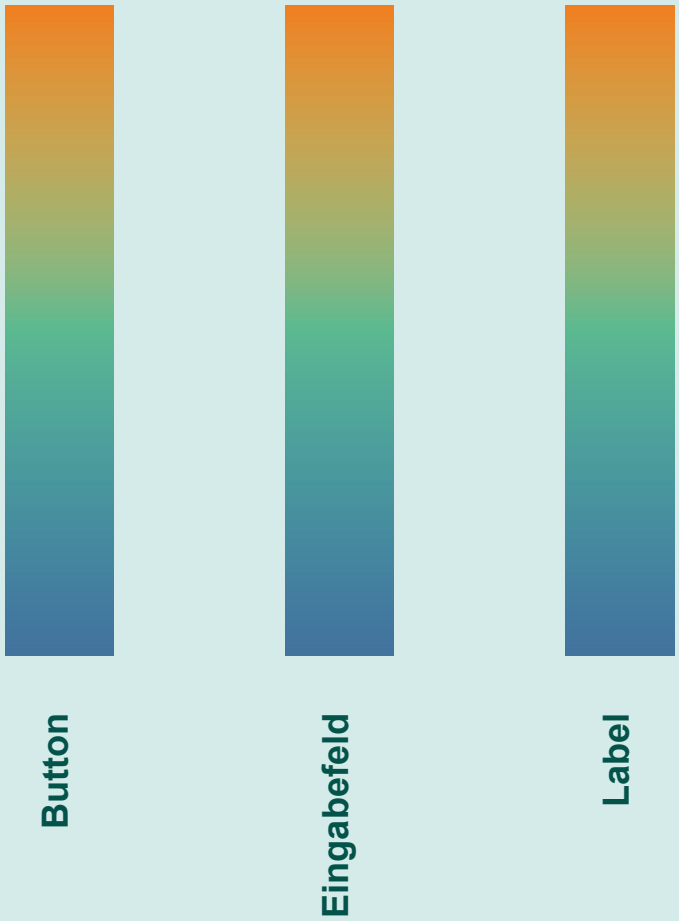
Grafik Inspiriert von: https://pbs.twimg.com/media/DCXJ_tjXoAAoBbu.jpg

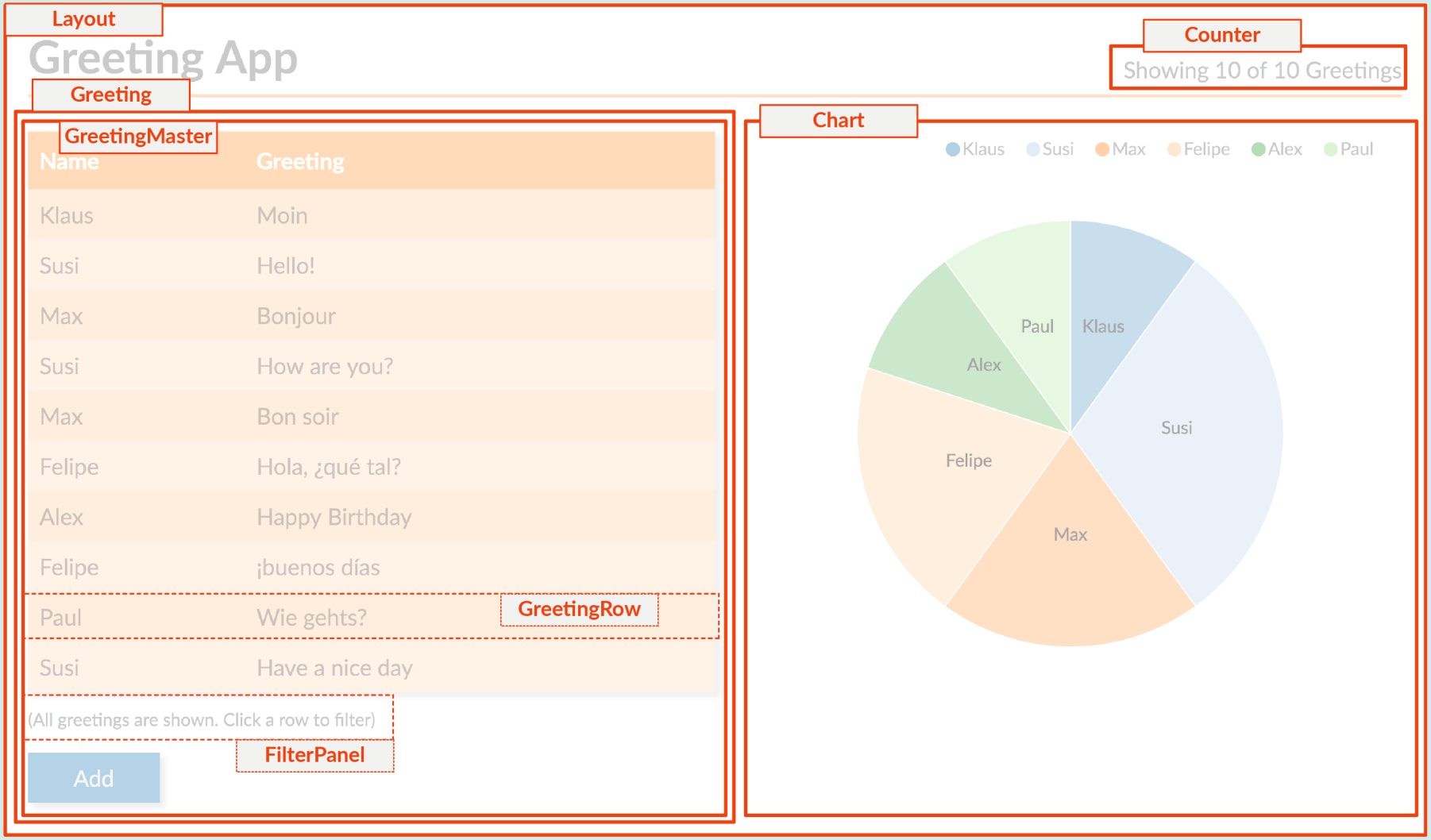
KOMPONENTEN

React-Komponenten

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

- können auf dem Server gerendert werden





GREETING APP: KOMPONENTEN

React

PRAKTISCH

EINE EINFACHE REACT KOMPONENTE

Komponente

Showing 3 of 11 Greetings

EINE EINFACHE REACT KOMPONENTE

Komponente

Komponenten sind JavaScript-Funktionen

```
function Counter() {


}
```

EINE EINFACHE REACT KOMPONENTE

Komponente

Showing 3 of 11 Greetings

```
function Counter() {
  return <div>Showing 3 of 11 Greetings</div>
}
```

UI in JavaScript ("JSX") 😱

EINE EINFACHE REACT KOMPONENTE

Komponente

Properties

```
function Counter({filtered, total}) {


}
```

EINE EINFACHE REACT KOMPONENTE

Komponente

Showing 3 of 11 Greetings

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

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} />
```

KOMPONENTEN WERDEN ZU APPLIKATIONEN AGGREGIERT

App.js

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

function App() {
  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="root"></div>
  </body>
  <script src="dist/index.js"></script>
</html>
```

index.js

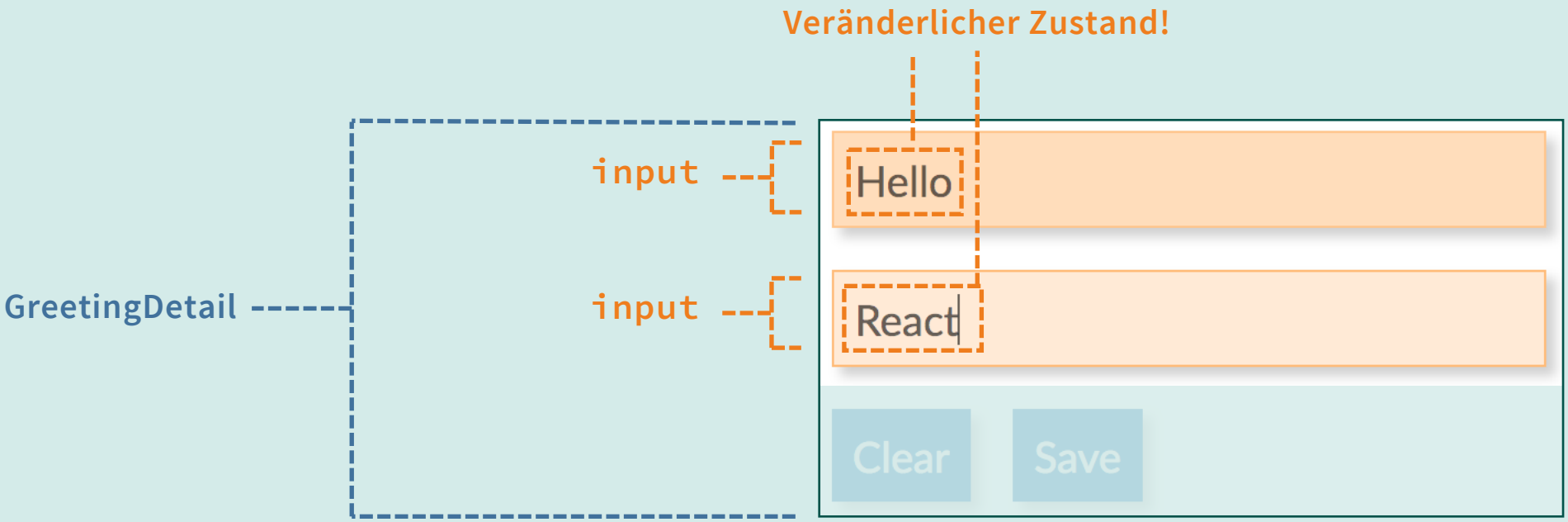
```
import React from 'react';
import ReactDOM from 'react-dom';

import App from './App';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

Model a.k.a

State

ARBEITEN MIT VERÄNDERLICHEN DATEN

BEISPIEL: EINGABEFELD



REACT HOOKS API

Hooks: Neue React API seit Anfang 2019

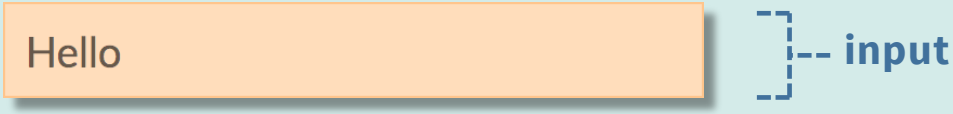
- Zugriff auf Zustand, Lebenszyklus etc. aus Funktionskomponenten

Hooks sind reguläre Funktionen, aber...

- **nur in** Funktionskomponenten (oder anderen Hooks) erlaubt
- **müssen** auf Top-Level-Ebene stehen (nicht in Schleife, if, ...)
- **müssen** mit "use" benannt werden

BEISPIEL: EINGABEFELD

Komponente



"Hook"-Funktion

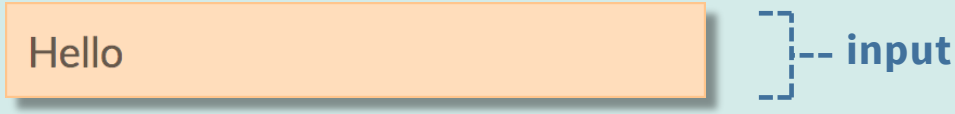
```
function GreetingDetail() {
  const [ phrase, setPhrase ] = React.useState("Hello");

}
```

Aktueller State **Setter** **Initialer Wert**

- Mit useState wird ein "Model" erzeugt (State in React)
- Es gibt kein zwei-Wege-Databinding

BEISPIEL: EINGABEFELD

Komponente



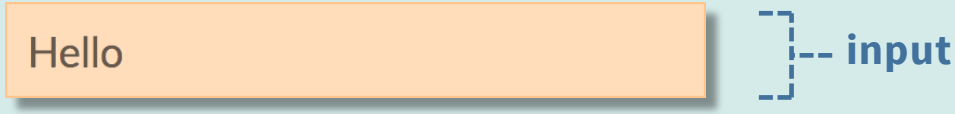
"Hook"-Funktion

```
function GreetingDetail() {
  const [ phrase, setPhrase ] = React.useState("Hello");

  return <div>
    <input
      value={phrase}

    />
  </div>
}
```

1. Input mit Wert aus State befüllen

BEISPIEL: EINGABEFELD

Komponente



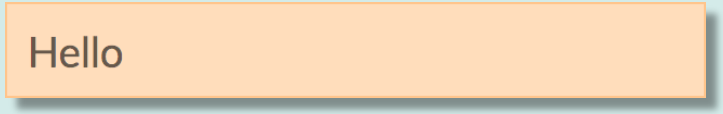
"Hook"-Funktion

1. Input mit Wert aus State befüllen
2. Zustand neu setzen

```
function GreetingDetail() {
  const [ phrase, setPhrase ] = React.useState("Hello");

  return <div>
    <input
      value={phrase}
      onChange={e => setPhrase(e.target.value)}
    />
  </div>
}
```

BEISPIEL: EINGABEFELD

Komponente



-- input

"Hook"-Funktion

```
function GreetingDetail() {
  const [ phrase, setPhrase ] = React.useState("Hello");

  return <div>
    <input
      value={phrase}
      onChange={e => setPhrase(e.target.value)}
    />
  </div>
}
```

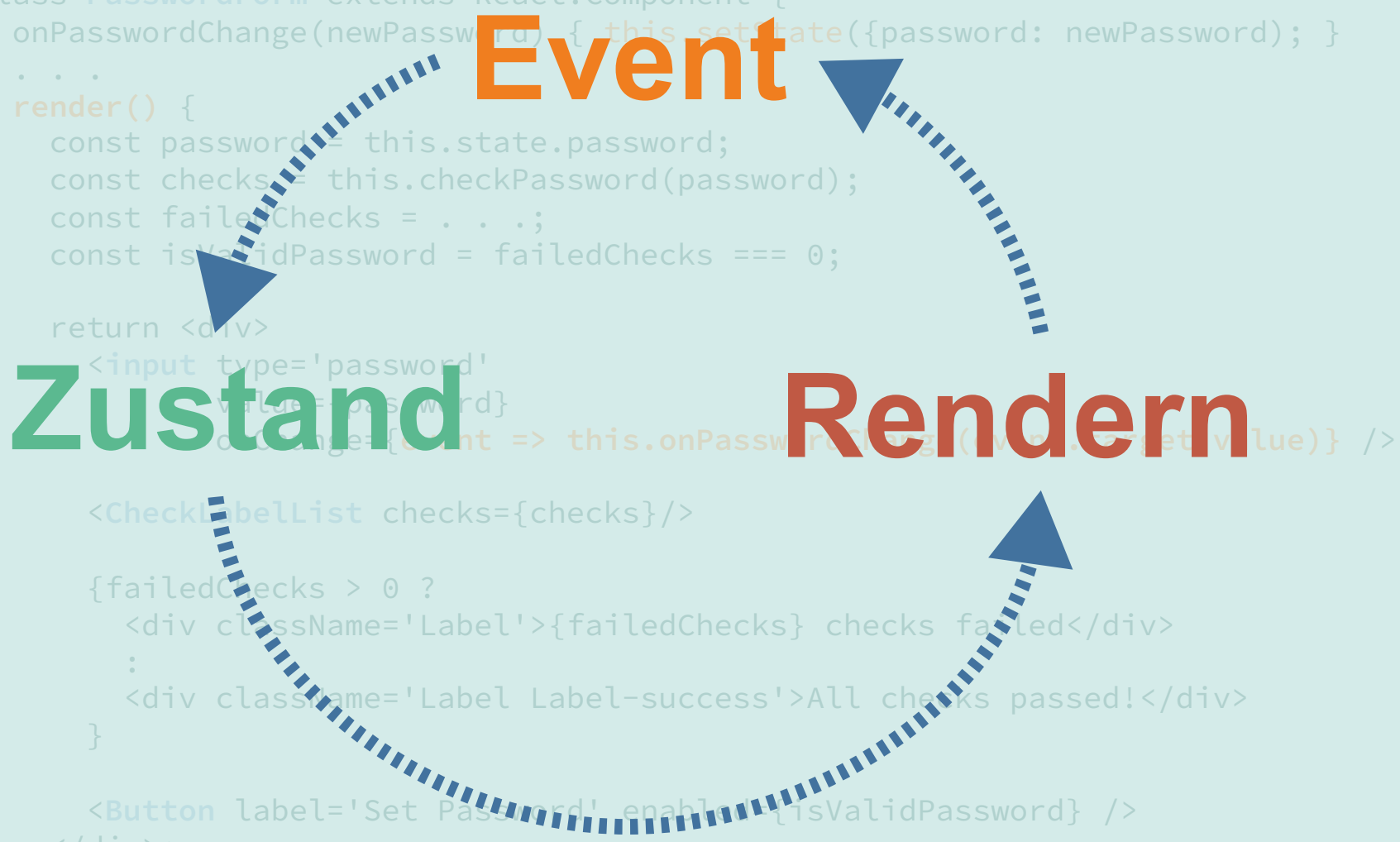
Neu rendern

Event

Zustand ändern

1. Input mit Wert aus State befüllen
2. 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
- Verhindert Inkonsistenzen
- "UI as a Function"


```
function GreetingDetail() {
  const [ phrase, setPhrase ] = React.useState("Hello");
  const [ name, setName ] = React.useState("React");
  const saveDisabled = !(phrase && name);

  return <div>
    <input value={phrase} />
    <input value={name} />
    <button disabled={saveDisabled}>Save</button>
  </div>
}
```

RENDERING VON KOMPONENTEN

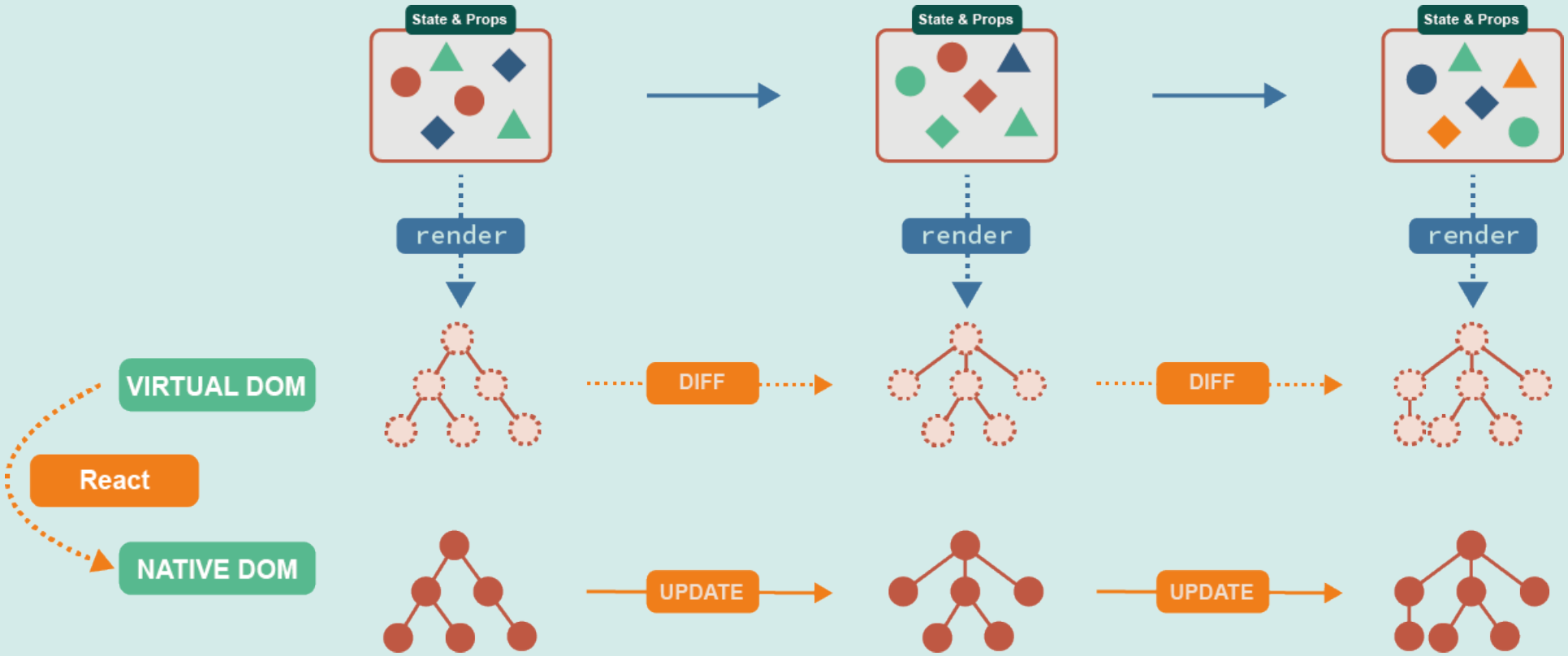
Beispiel

👉 DevTools: GreetingDetail mit Input-Komponente

RENDERING VON KOMPONENTEN

Virtual DOM

👆 Rendering hat doppelte Bedeutung!



BEISPIEL: DATEN VOM SERVER LADEN

- Gleiches Prinzip: geladene Daten werden in den State gesetzt
- Besonderheit: React Komponenten müssen Seiteneffekt-frei sein

```
function GreetingController() {
  // 1. Model (State) für geladene Daten erzeugen
  const [greetings, setGreetings] = React.useState(null);

  // 2. Leere Liste (oder Platzhalter) rendern
  return greetings <GreetingMaster greetings={greetings} />
                   ? : <LoadingIndicator />;
}
```

BEISPIEL: DATEN VOM SERVER LADEN

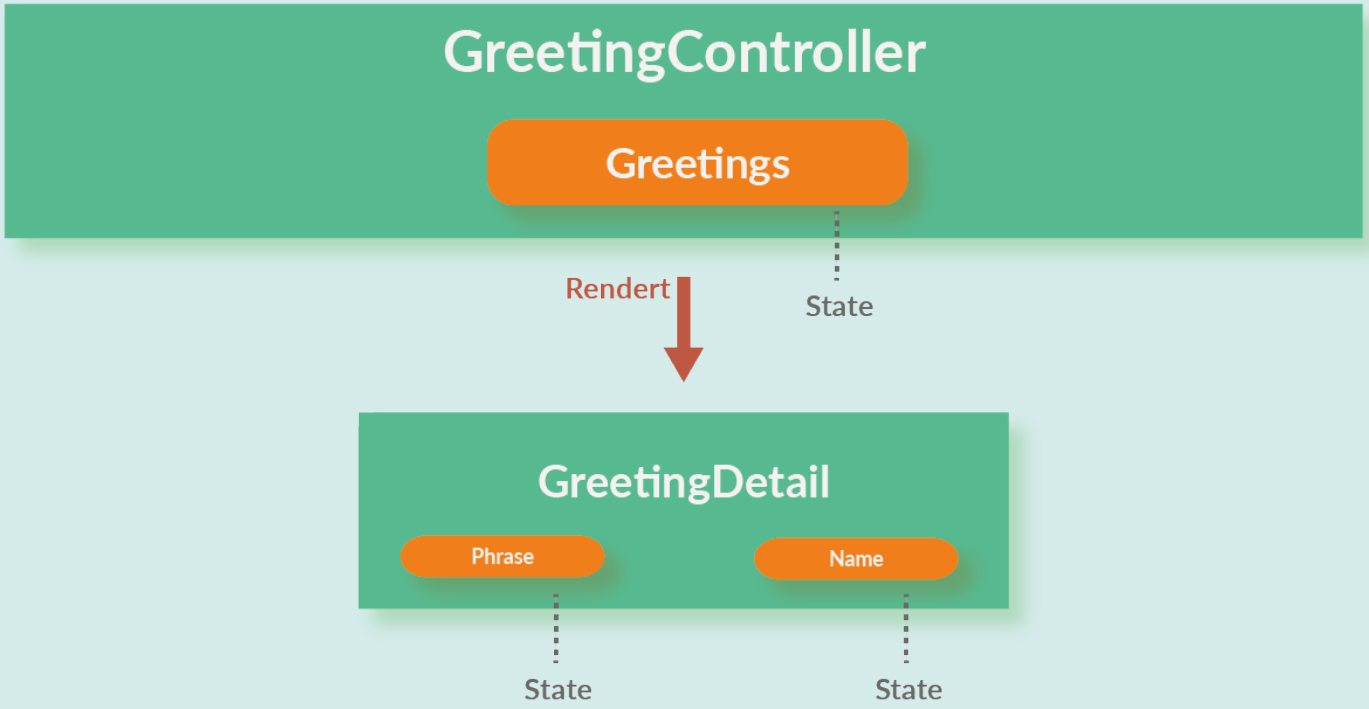
- Gleiches Prinzip: geladene Daten werden in den State gesetzt
- Besonderheit: React Komponenten müssen Seiteneffekt-frei sein

```
function GreetingController() {
  // 1. Model (State) für geladene Daten erzeugen
  const [greetings, setGreetings] = React.useState(null);

  // 2. Seiteneffekt registrieren
  React.useEffect( () => {
    fetch("/api/greetings");
      .then(response => response.json())
      .then(greetingsAsJson => {
          // 3. Daten sind geladen => State setzen => neu rendern
          setGreetings(greetingsAsJson);
      })
  }), []);

  // 4. Liste mit Grüßen (oder Platzhalter) rendern
  return greetings <GreetingMaster greetings={greetings} />
                  ? : <LoadingIndicator />;
}
```

VON DER KOMPONENTE ZUR ANWENDUNG

KOMPONENTENHIERARCHIEN

- **Komponenten werden in Hierarchien aggregiert**

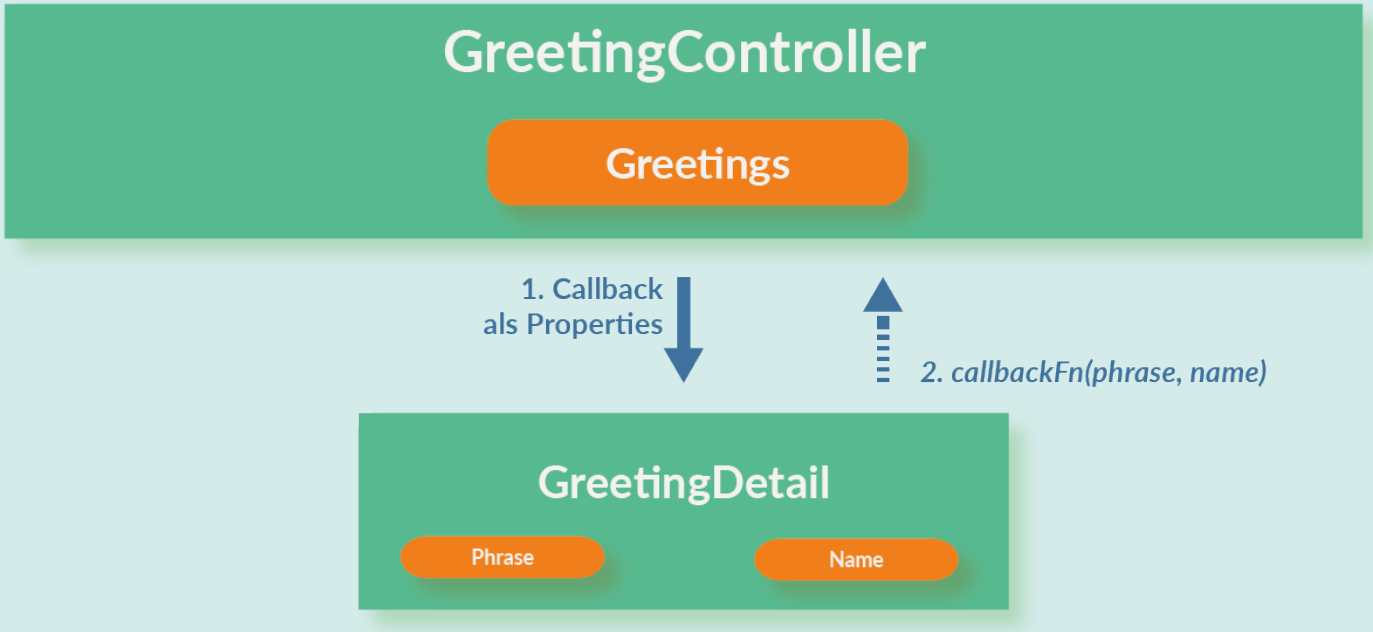


KOMPONENTENHIERARCHIEN

- **Kommunikation per Callback-Funktionen**

KOMPONENTENHIERARCHIEN

- **Kommunikation per Callback-Funktionen**
- Unterkomponente kann Daten per Callback nach oben kommunizieren



BEISPIEL: KOMMUNIKATION PER CALLBACK-FUNKTION

```
function GreetingDetail({onSave}) {
  const [ phrase, setPhrase ] = React.useState("Hello");
  const [ name, setName ] = React.useState("React");

  return <div>
    <input value={phrase} ... />
    <input value={name}  ... />
    <button onClick={() => onSave(phrase, name})>Save</button>
  </div>
}
```

BEISPIEL: KOMMUNIKATION PER CALLBACK-FUNKTION

```
function GreetingDetail({onSave}) {
  const [ phrase, setPhrase ] = React.useState("Hello");
  const [ name, setName ] = React.useState("React");

  return <div>
    <input value={phrase} ... />
    <input value={name}  ... />
    <button onClick={() => onSave(phrase, name})>Save</button>
  </div>
}

function GreetingController() {
  function onNewGreeting(phrase, name) {
    // ...
  }

  return ... <GreetingDetail onSave={onNewGreeting} /> ...
}
```

BEISPIEL: ROUTING UND DEEP LINKS

React hat keinen eigenen Router

- De-facto-Standard: React Router

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

function App() {
  return (
    <div>
      <h1>Greetings</h1>
      <Switch>
        <Route path="/greet/:greetingId"><GreetingDisplayPage/></Route>
        <Route exact path="/">           <GreetingMaster />    </Route>
        <Route>                          <NotFound />          </Route>
      </Switch>
    </div>
  );
}
```

BEISPIEL: TESTEN

Testen einer Komponente

Testen ohne Browser möglich, headless im CI-Build

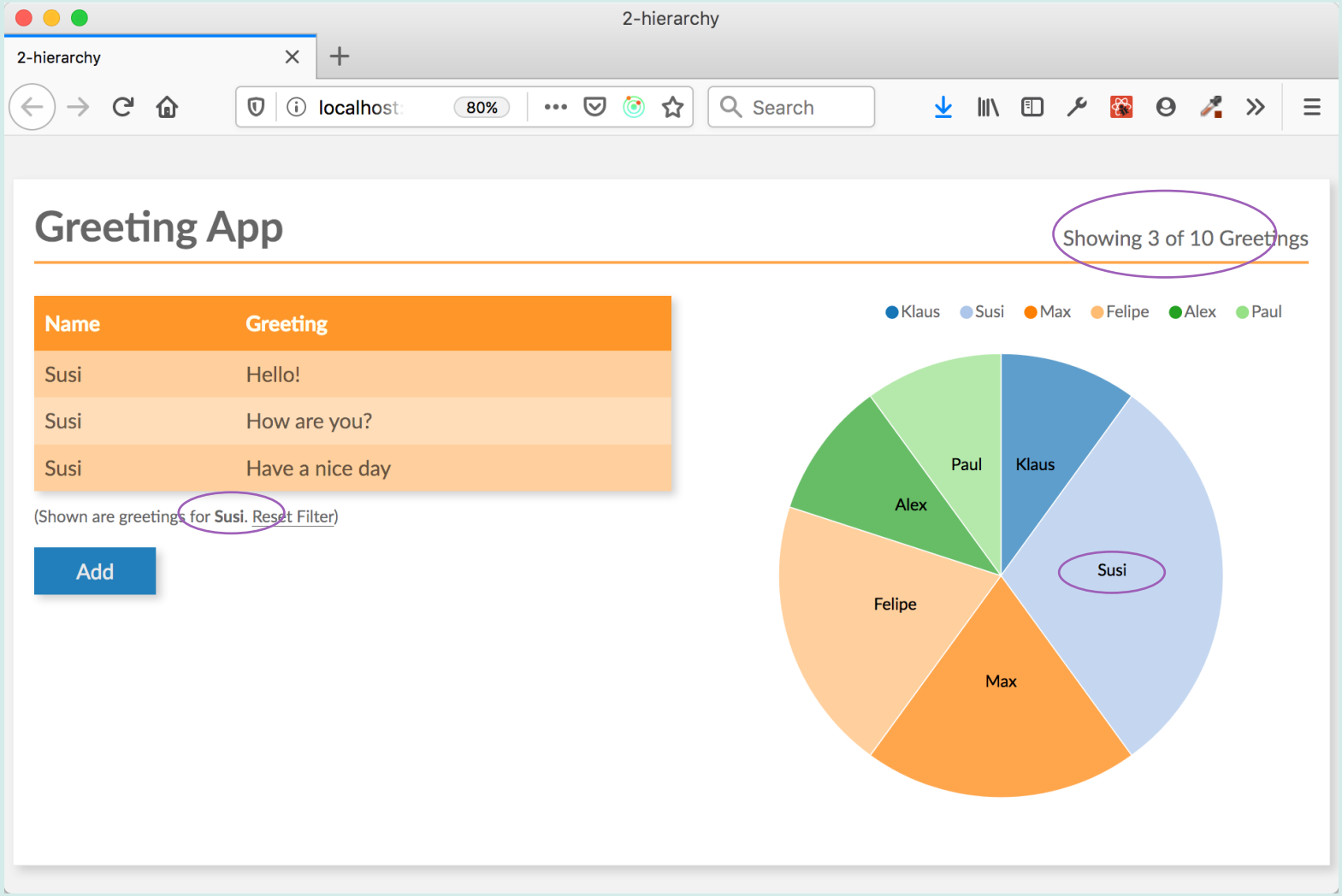
```
import React from "react";
import { render, fireEvent } from "@testing-library/react";
import GreetingDetails from "./GreetingDetails";

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

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

  // Komponente Rendern
  const { getByText, container } = render(
    <GreetingDetails initialPhrase="Hello" initialGreeting="React"
                     onSave={onSaveHandler} />
  );

  // Ereignis simulieren
  fireEvent.click(getByText("Save"));

  // Ergebnis überprüfen
  expect(onSaveHandler).toHaveBeenCalledWith("Hello", "React");
});
```

GLOBALE DATEN

ZUSTANDSMANAGEMENT

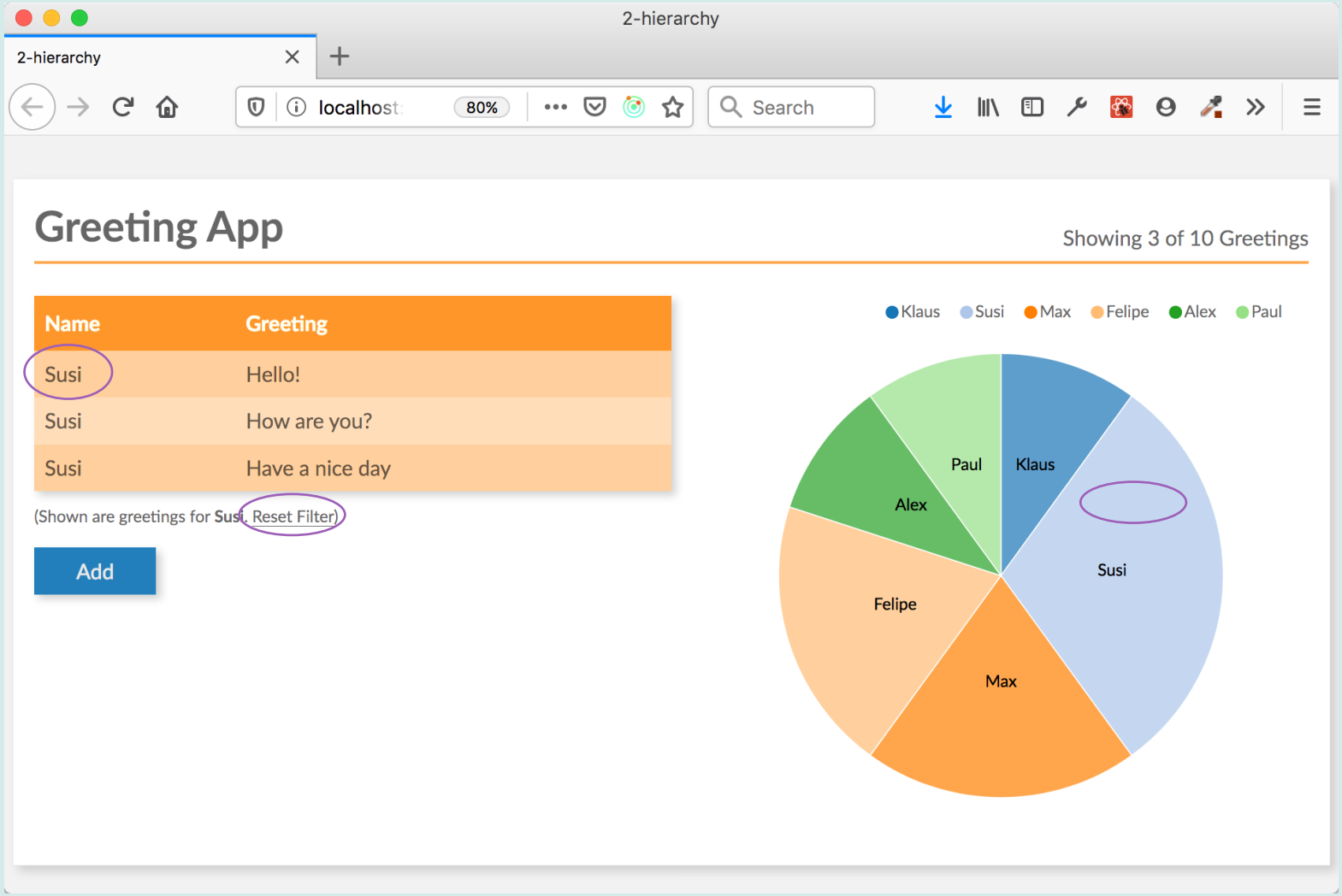
GLOBALE DATEN

- **Beispiel: Globale Daten in der Anwendung**

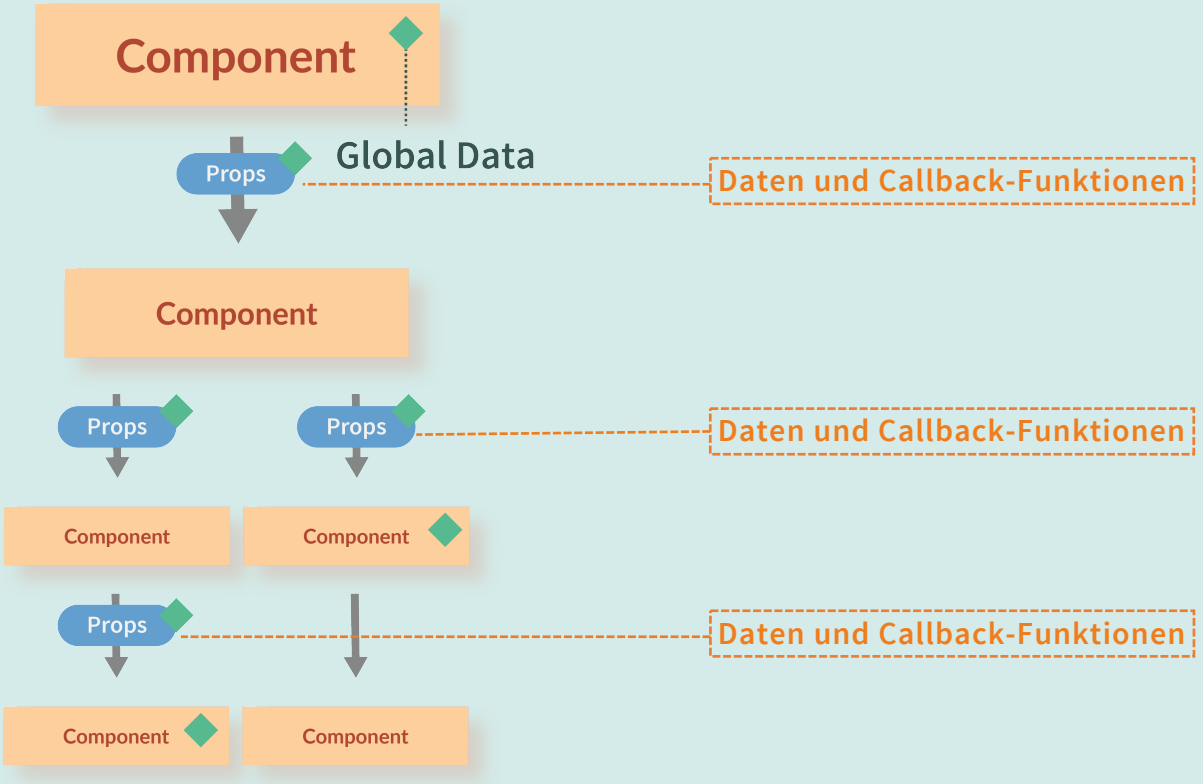


GLOBALE DATEN

- **Beispiel: Globale Aktionen in der Anwendung**

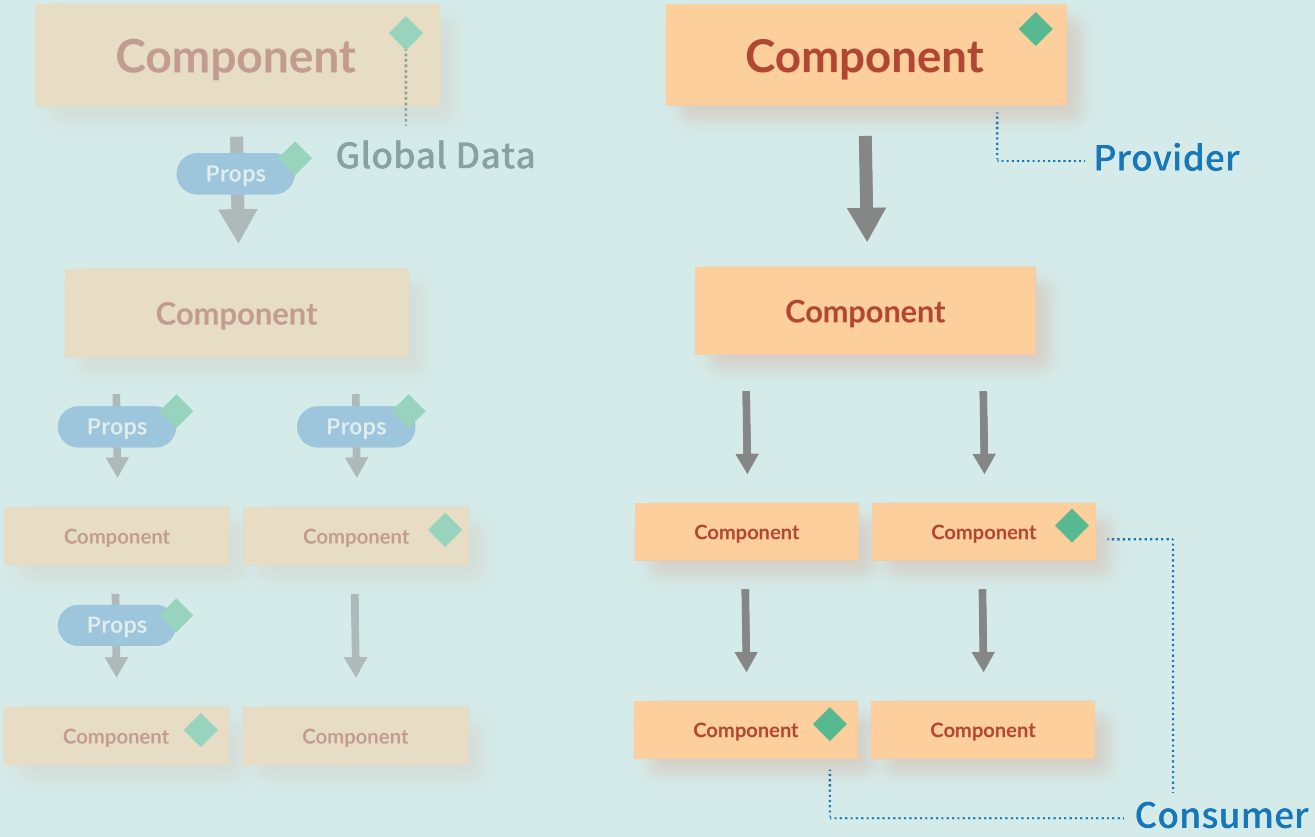


Globale Daten: Mit Properties (Durchreichen)



Globale Daten: Mittels **Context API**

- Provider bietet Daten an
- Consumer kann auf Daten zugreifen



BEISPIEL: REACT CONTEXT

Provider stellt globale Daten und Aktionen zur Verfügung

```
const GreetingContext = React.createContext();

function GreetingProvider({children}) {
  const [greetings, setGreetings] = React.useState([]);
  const [filter, setFilter] = React.useState("");

  return <GreetingContext.Provider
           value={{greetings, setGreetings, filter, setFilter}}>
             {children}
         </GreetingContext.Provider>;
}
```

Einbinden des Providers in die Komponenten-Hierarchie

```
import GreetingProvider from "./GreetingProvider";

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

Zugriff auf Context möglich

BEISPIEL: REACT CONTEXT

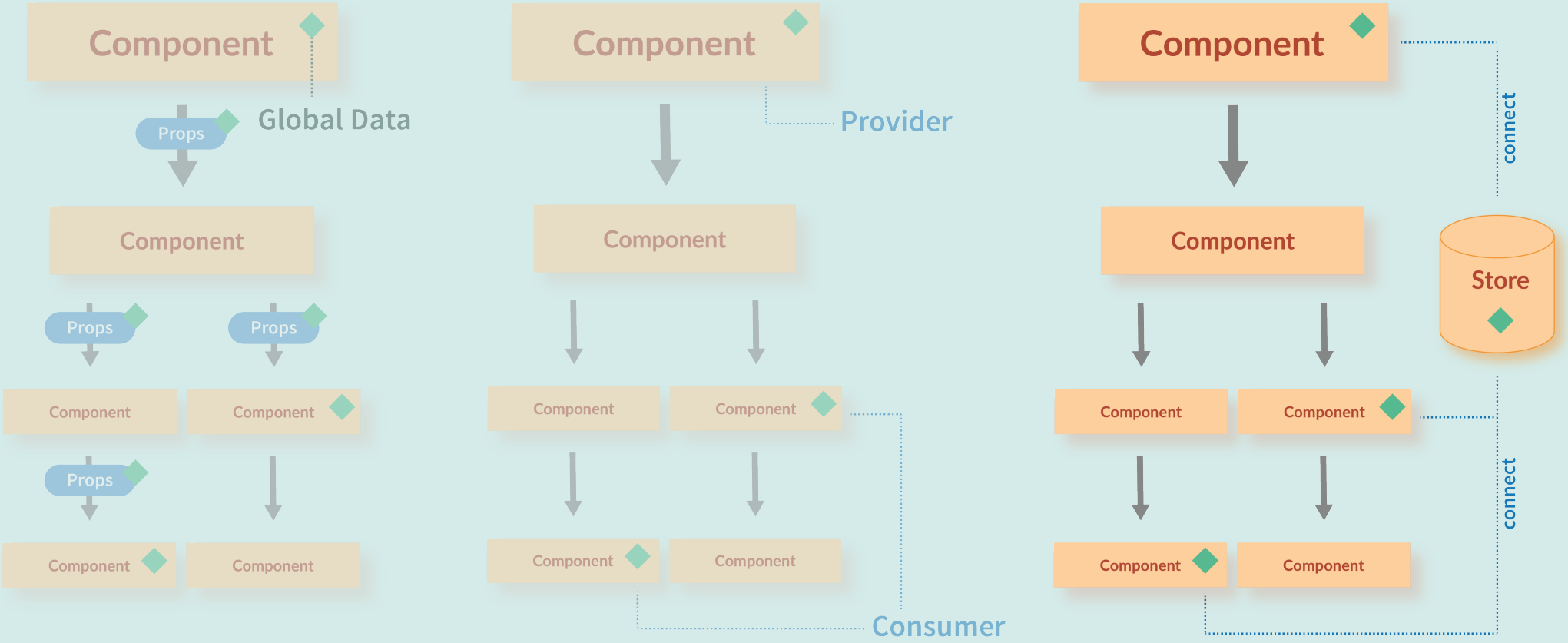
Verwenden des Contexts

```
import GreetingContext from "./GreetingProvider";

function GreetingMaster() {
  const {greetings, filter, setFilter} = React.useContext(GreetingContext);
  const filteredGreetings = filterGreetings(greetings, filter);

  return <table>
     {filteredGreetings.map(greeting =>
         <tr onClick={() => setFilter(greeting.name)}>
           <td>{greeting.name}</td><td>{greeting.phrase}</td>
         </tr>
     )}
     </table>
}
```

OPTIONEN

Globale Daten: Mit externem State Management

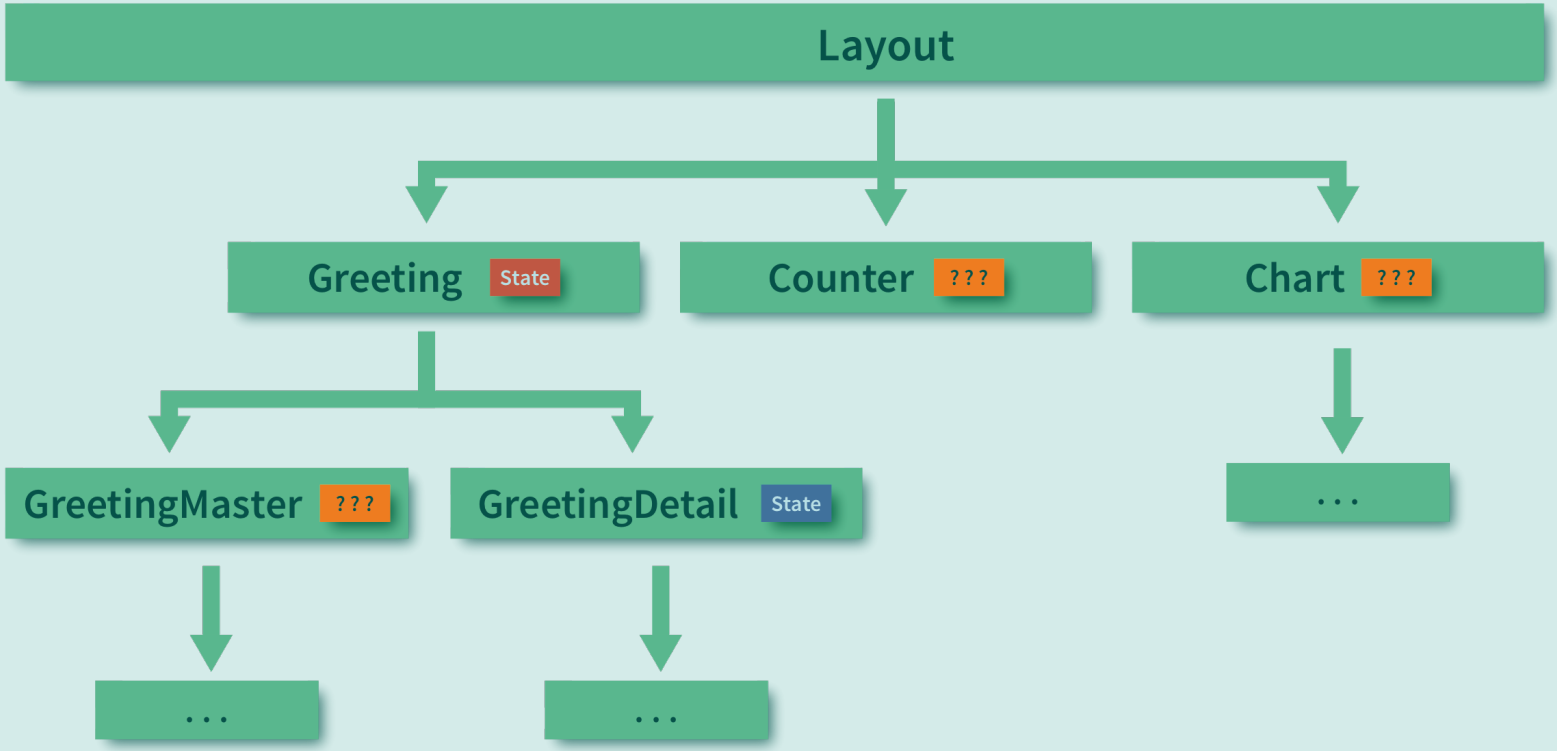
- MobX oder Redux



REDUX

EXTERNES STATEMANAGEMENT

KOMPONENTENHIERARCHIEN



Probleme:

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

EXTERNES STATE MANAGEMENT

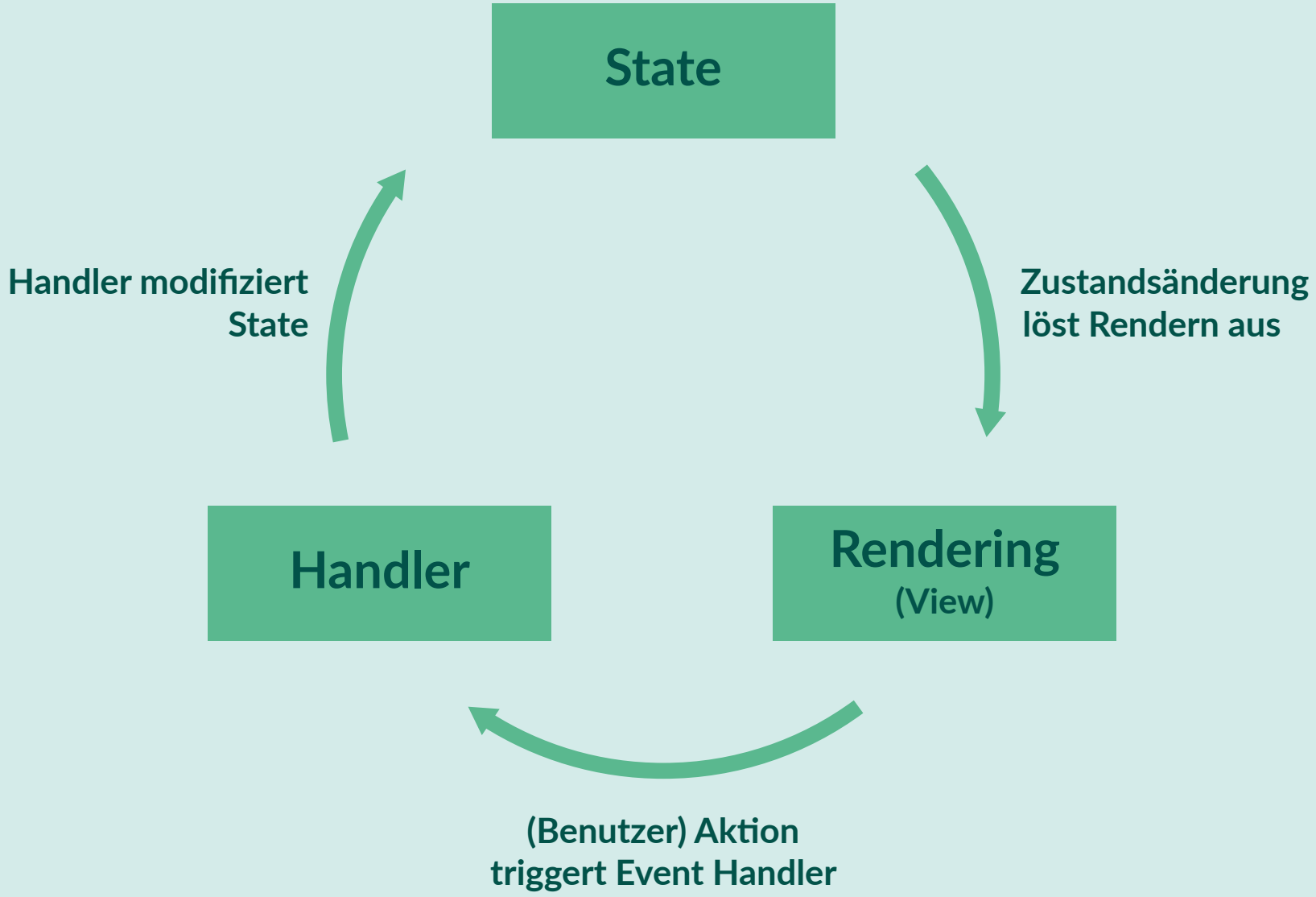
Beispiel: Redux

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

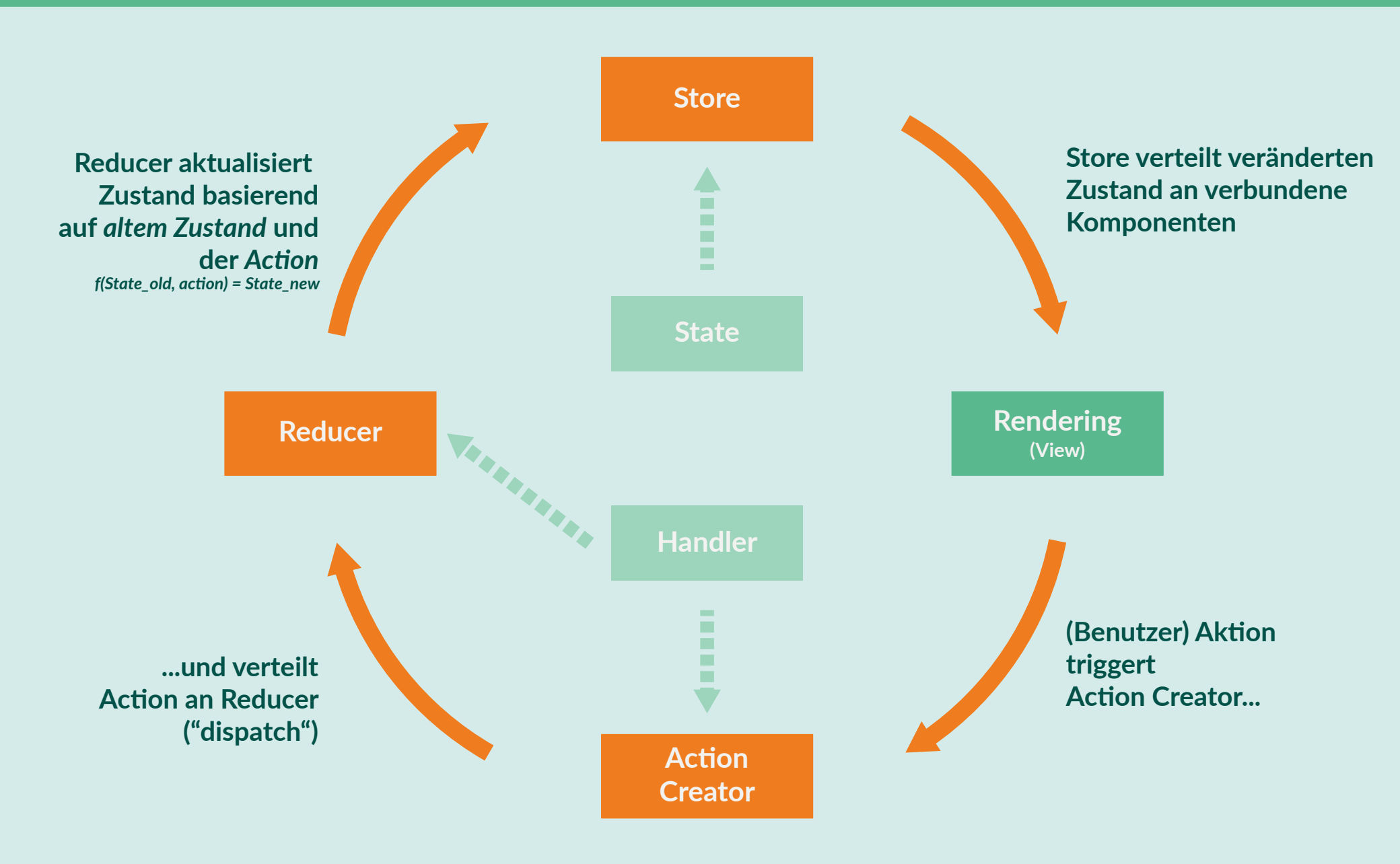
- Architektur Pattern und Implementierung

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

REACT RENDER ZYKLUS



REDUX-BASIERTE REACT-ANWENDUNG



Zugriff auf globalen Zustand mit useSelector-Hook

```
import {useSelector} from "react-redux";

function GreetingMaster() {
  const { greetings, filter } = useSelector(state => ({
    greetings: state.greetings,
    filter: state.filter
}));


  return ... // Tabelle mit gefilterten Greetings
}
```

Auslösen von Actions mit useDispatch

```
import {useSelector, useDispatch} from "react-redux";

function GreetingMaster() {
  const {greetings, filter} = useSelector(...);
  const dispatch = useDispatch();

  function rowClick(greeting) {
    dispatch({
      type: "FILTER_SELECTED", payload: { filter: greeting.filter }
    })

  return ...
    <tr onClick={() => rowClick(greeting)}>...Greeting...</tr>
}
```

BEISPIEL: REDUX

Reducer-Funktion

```
function filterReducer(filter = "", action) {
  switch (action.type) {
    case "FILTER_SELECTED":
      return action.payload;
    case "FILTER_CLEARED":
      return "";
    default:
      return filter;
}
```

TYPESCRIPT

TYPSICHERES REACT

TYPESCRIPT AUF EINEN BLICK

TypeScript: Obermenge von JavaScript

- Jeder gültige JavaScript Code ist gülter TypeScript Code (theoretisch...)
- **Ergänzt JS um Typ-System**, Sichtbarkeiten, Enums und Dekoratoren
- Compiler erzeugt aus TypeScript-Code JavaScript (ES3, ES5, ES6)-Code

- Entwickelt von Microsoft
  - http://www.typescriptlang.org/
- Sehr guter IDE Support
  - IntelliJ IDEA, Visual Studio Code

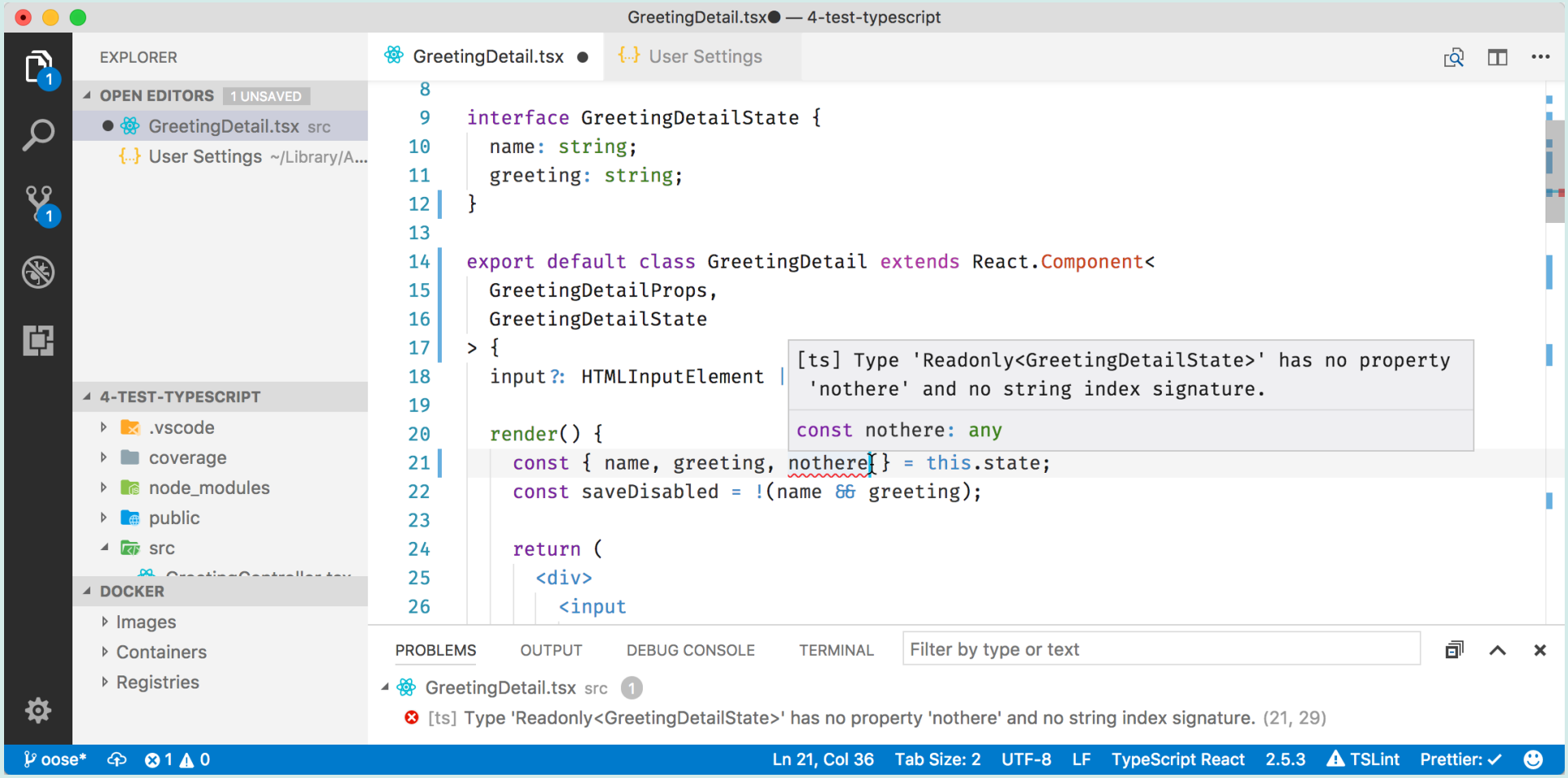
Typ-Angaben für typsichere Artefakte

- State (kann abgeleitet werden)
- Komponenten Properties
- Redux State und Actions

TYPESCRIPT DEMO

Typen für React Komponenten

Fehlermeldung und Code Completion in der IDE



NILS HARTMANN
https://nilshartmann.net

https://reactbuch.de

Vielen Dank!

Slides: https://nils.buzz/conceptpeople-react

Source Code: https://github.com/nilshartmann/react-training

Fragen & Kontakt: nils@nilshartmann.net