

NILS HARTMANN

React

Einsteigen und loslegen - eine praktische Einführung

Slides: <https://react.schule/jax2021-react>

JAX ONLINE | 6. MAI 2021 | @NILSHARTMANN

NILS HARTMANN

Kontakt: nils@nilshartmann.net

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

Freiberuflicher Entwickler, Architekt, Trainer aus Hamburg

Java
JavaScript, TypeScript
React
GraphQL

Trainings & Workshops



<https://reactbuch.de>

[HTTPS://NILSHARTMANN.NET](https://nilshartmann.net)

React

Einführung

REACT

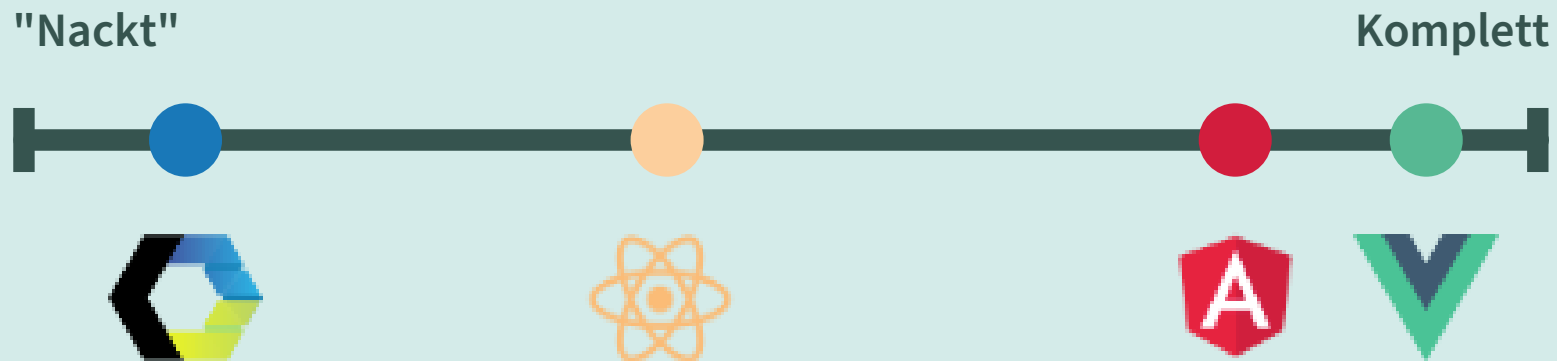
[**https://reactjs.org**](https://reactjs.org)

- Minimals API
- Minimals Feature Set

REACT

<https://reactjs.org>

- Minimales API
- Minimales Feature Set



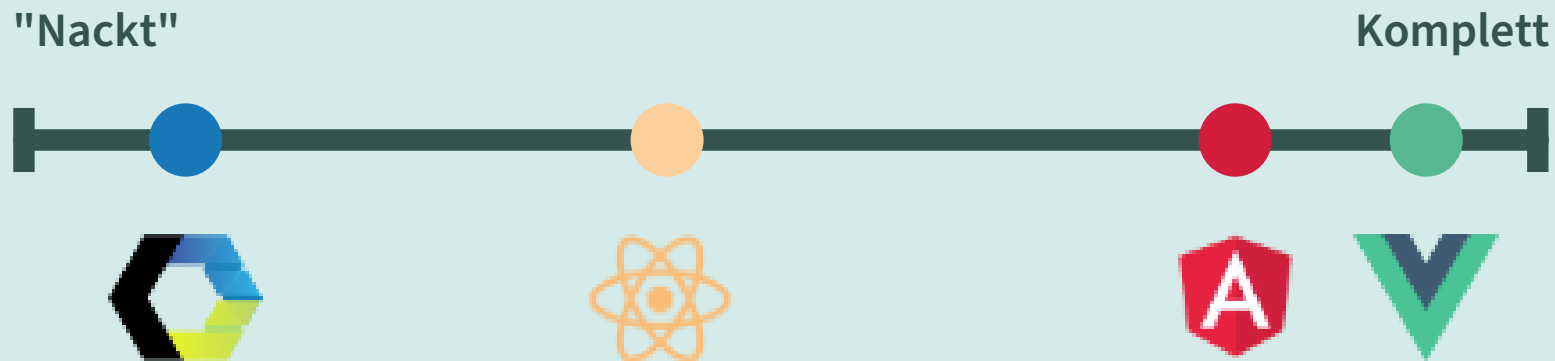
REACT

<https://reactjs.org>

- Minimales API
- Minimales Feature Set

👉 Man kann (muss ?) viele Entscheidungen selber treffen

👉 In ernsthaften Anwendungen werden weitere Bibliotheken benötigt



Greeting App

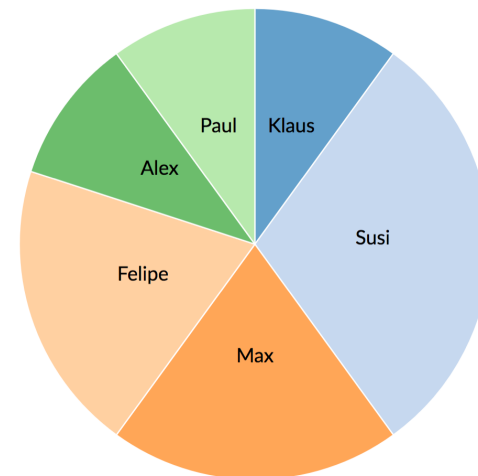
Showing 10 of 10 Greetings

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)

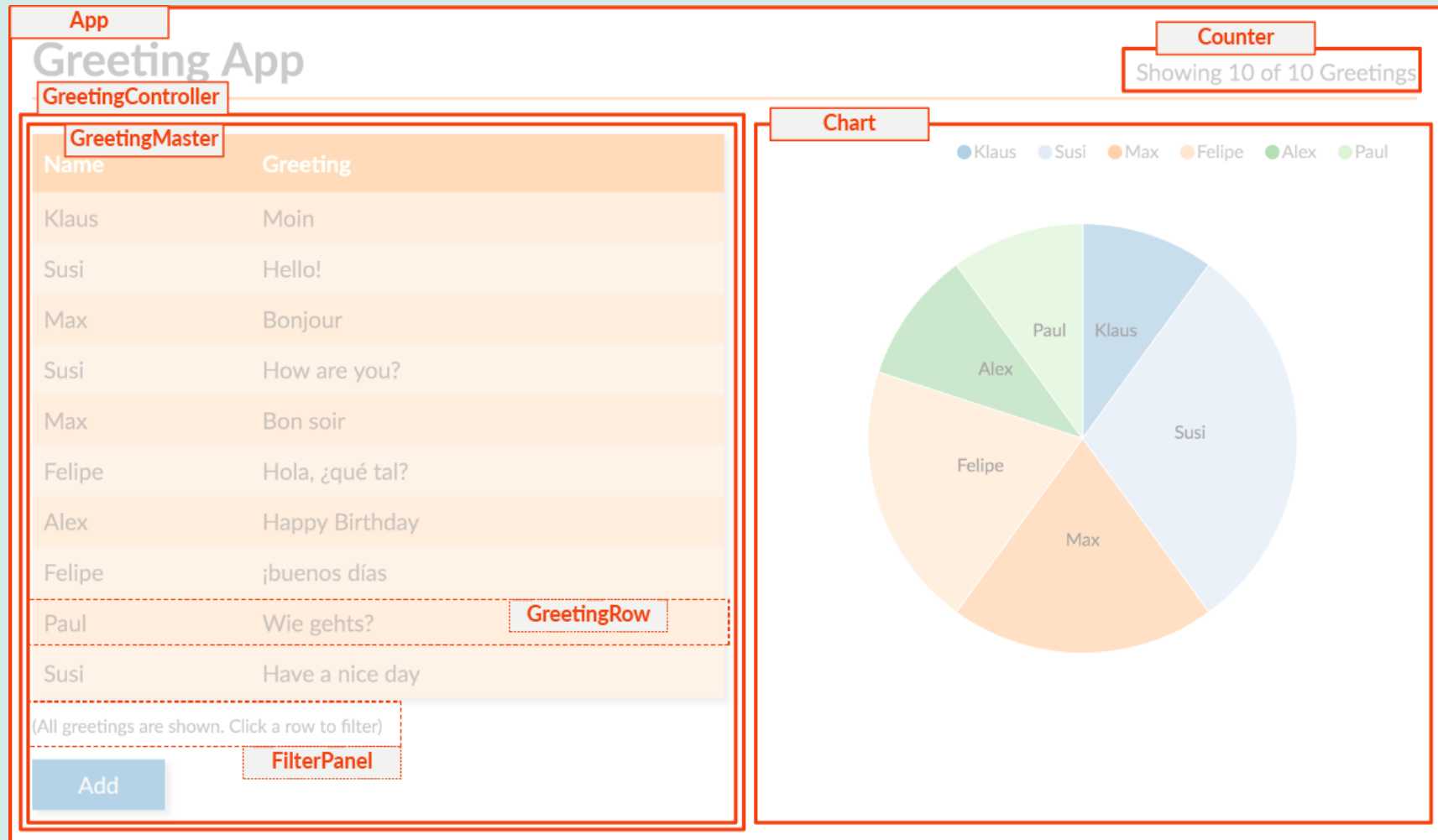
Add

● Klaus ● Susi ● Max ● Felipe ● Alex ● Paul



BEISPIEL: DIE GREETING APP

React Komponenten



GREETING APP: KOMPONENTEN

React Komponenten

Klassische Aufteilung

Logik, Model
(Java, JS)



View
(HTML, Template)



Gestaltung
(CSS)



Aufteilung in Komponenten



Button



GreetingEditor



Counter

KOMPONENTEN

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

React-Komponenten

- bestehen aus Logik und UI
- keine Templatesprache
- werden deklarativ beschrieben



Button

GreetingEditor

Counter

KOMPONENTEN

EINE EINFACHE REACT KOMPONENTE

Darstellung


Showing 3 of 11 Greetings

EINE EINFACHE REACT KOMPONENTE

Darstellung

Showing 3 of 11 Greetings

Komponenten sind JavaScript-Funktionen



```
function Counter() {  
  
}
```

EINE EINFACHE REACT KOMPONENTE

Darstellung

Showing 3 of 11 Greetings

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

UI in JavaScript ("JSX") 🤖

EINE EINFACHE REACT KOMPONENTE

Darstellung

Showing 3 of 11 Greetings

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

Properties

EINE EINFACHE REACT KOMPONENTE

Darstellung

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

Darstellung

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

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

App.js

```
function App() {  
  return (  
    <main>  
      <header>  
        <Counter filtered={3} total={11} />  
      </header>  
      <div className="Left">  
        <GreetingTable />  
      </div>  
      <div className="Right">  
        <Chart />  
      </div>  
    </main>  
  )  
}
```

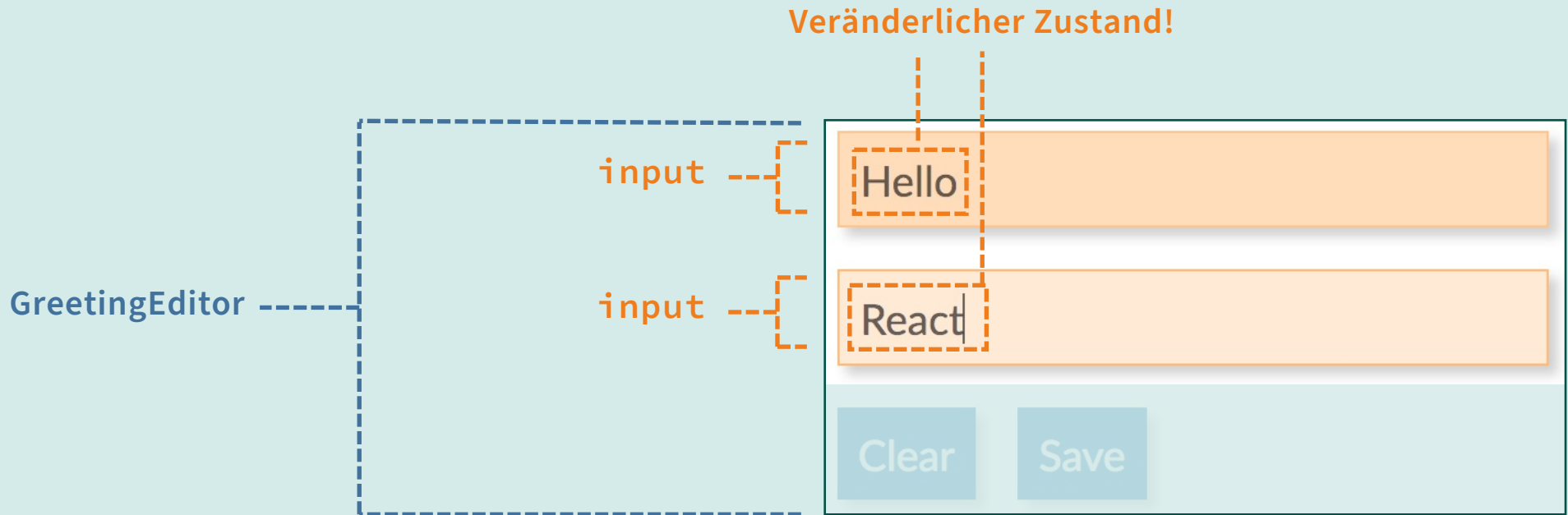
Model a.k.a

State

ARBEITEN MIT VERÄNDERLICHEN DATEN

STATE

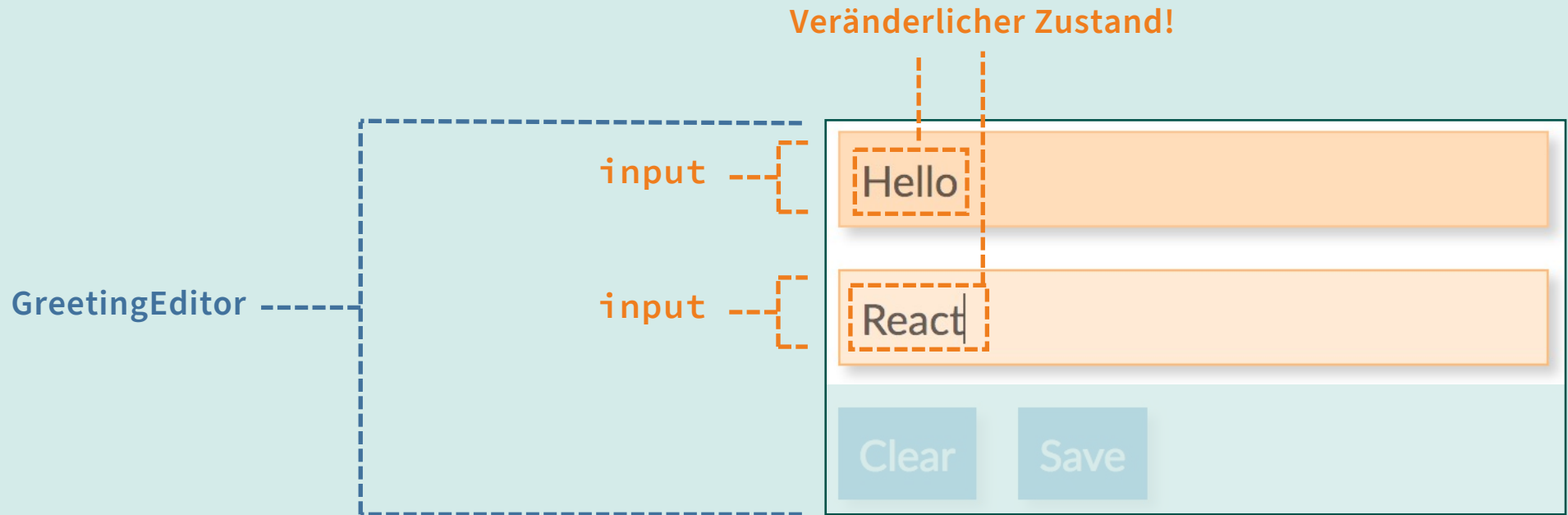
Beispiel: Eingabefeld



- Mit `useState` wird ein "Model" erzeugt ("State" in React genannt)

STATE

Beispiel: Eingabefeld



Andere typische Beispiele:

- geladene Daten vom Server
- Menü aufgeklappt/zugeklappt
- Modaler Dialog sichtbar/unsichtbar

BEISPIEL: EINGABEFELD

Komponente

Hello

input

"Hook"-Funktion

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

}

- Mit useState wird ein "Model" erzeugt ("State" in React genannt)

BEISPIEL: EINGABEFELD

Komponente

Hello

}-- input

"Hook"-Funktion

1. Input mit Wert aus State befüllen

2. Zustand neu setzen

```
function GreetingEditor() {  
  const [ phrase, setPhrase ] = React.useState("Hello");  
  
  return <div>  
    <input  
      value={phrase}  
      onChange={e => setPhrase(e.target.value)}  
    />  
  </div>  
}
```

BEISPIEL: EINGABEFELD

Komponente

Hello

}-- input

"Hook"-Funktion

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

Neu rendern

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

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

Event



Zustand ändern

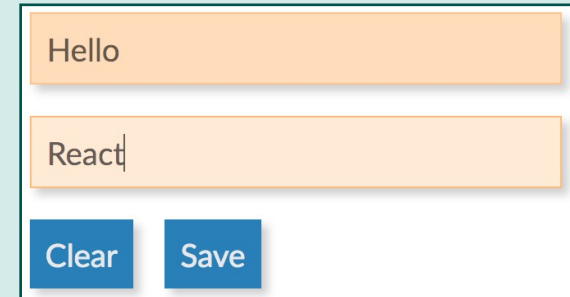
Besonderheiten:

- Kein 2-Wege-Databinding
- Bei Änderung am Zustand wird Funktion neu ausgeführt

RENDERING VON KOMPONENTEN

Gerendert wird immer die **ganze** Komponente

- Inklusive aller Unterkomponenten
- Bei jeder Zustandsänderung
- **Verhindert Inkonsistenzen**

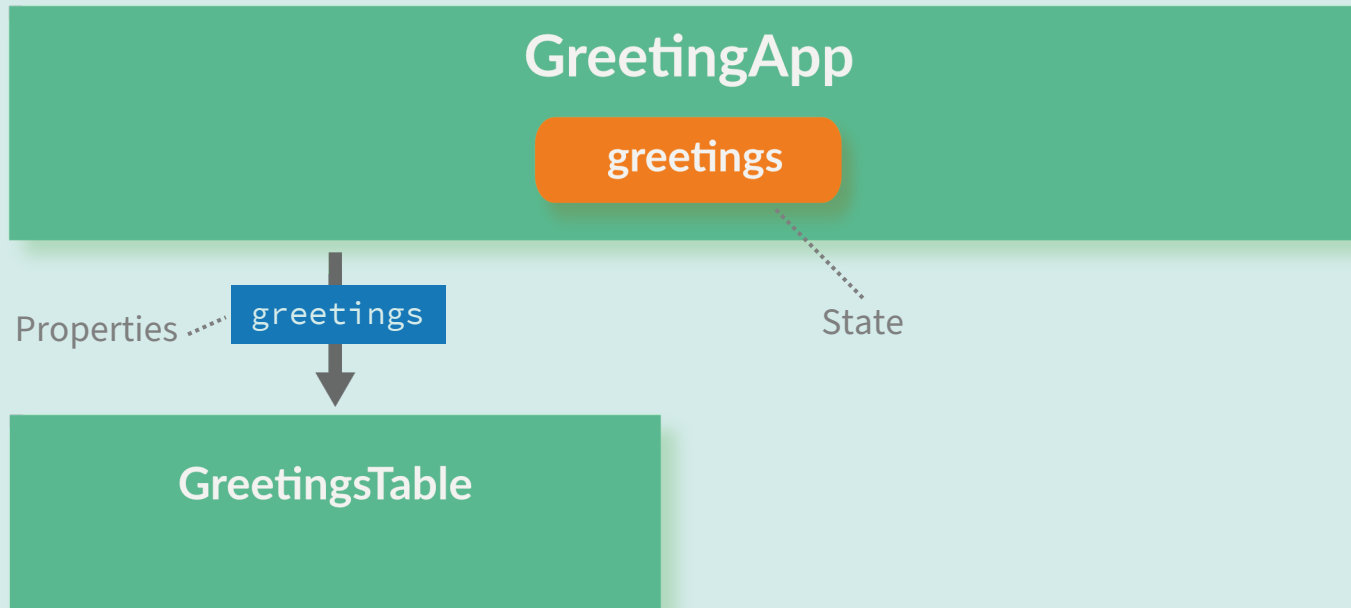


```
function GreetingEditor() {  
  const [ phrase, setPhrase ] = React.useState("Hello");  
  const [ name, setName ] = React.useState("React");  
  const saveDisabled = phrase === "" || name === "";  
  
  return <div>  
    <input value={phrase} onChange={e => setPhrase(e.target.value)} />  
    <input value={name} onChange={e => setName(e.garget.value)} />  
    <button disabled={saveDisabled}>Save</button>  
  </div>  
}
```


HIERARCHIEN VON KOMPONENTEN

Gemeinsame Daten wandern in gemeinsame Oberkomponenten

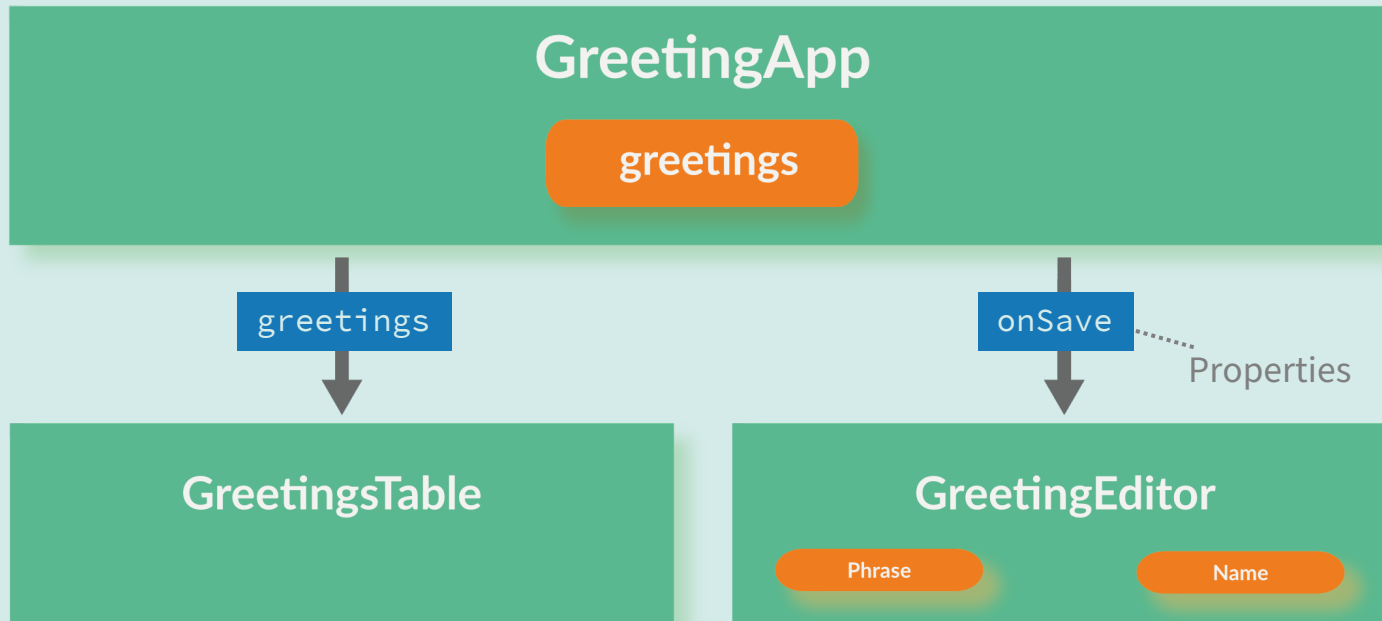
- Von dort per Properties nach unten



HIERARCHIEN VON KOMPONENTEN

Gemeinsame Daten wandern in gemeinsame Oberkomponenten

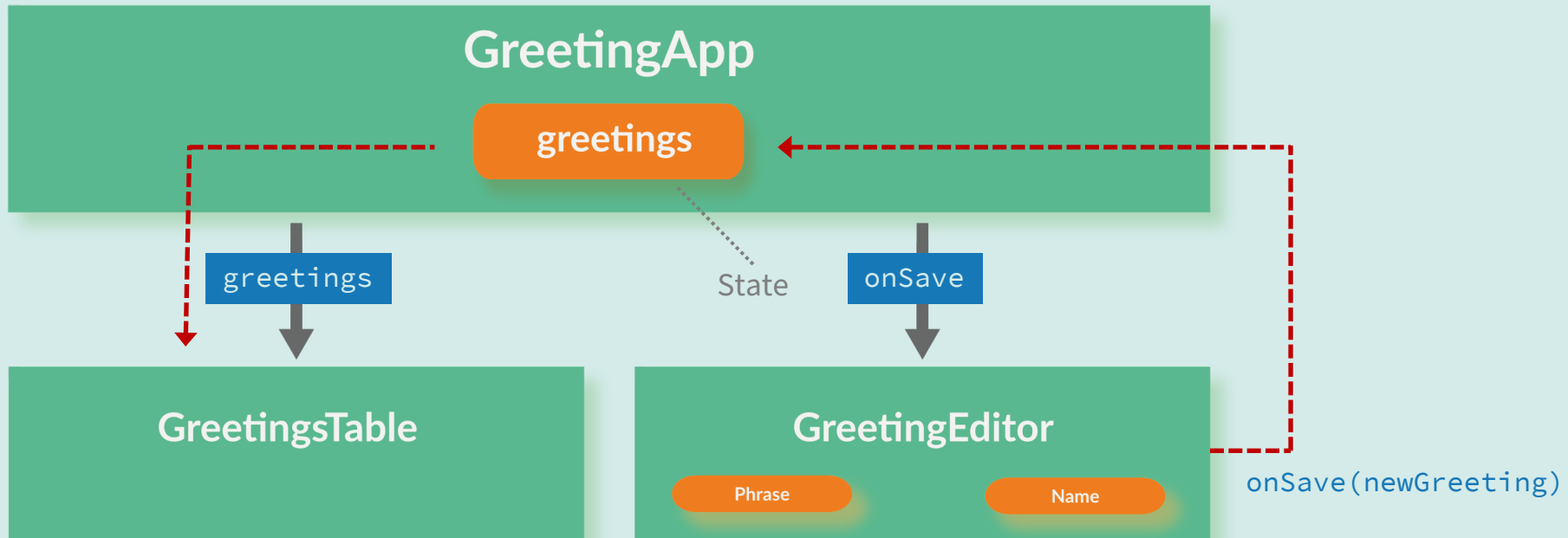
- Von dort per Properties nach unten
- Unterkomponenten informieren nach oben per Callback-Funktion



HIERARCHIEN VON KOMPONENTEN

Gemeinsame Daten wandern in gemeinsame Oberkomponenten

- Von dort per Properties nach unten
- Unterkomponenten informieren nach oben per Callback-Funktion
- Anwendung wird neu gerendert, alles konsistent!



TYPESCRIPT

Typsichere React-Anwendungen

- out-of-the-box Support für TypeScript

```
2
3 interface GreetingDetailProps {
4   initialName: string;
5   initialGreeting: string;
6 }
7
8 export default function GreetingDetail(props: GreetingDetailProps) {
9   const [name, setName] = React.useState<string>(props.initialName);
10  const [greeting, setGreeting] = React.useState<string>(props.initialGreeting);
11
12  function reset() {
13    setName(null);
14    setGreeting(props.invalidGreeting);
15  }
16
17  return (
18    <div>
19      <input
20        onChange
21        value={name}
22        name="name"
23        placeholder="Name"
24      />
25    </div>
26  );
27}
```

any
Cannot find name 'nam'. Did you mean 'name'? ts(2552)
GreetingEditor.tsx(9, 10): 'name' is declared here.

OUTPUT PROBLEMS 3 DEBUG CONSOLE

TS GreetingEditor.tsx advanced/steps/5a-redux-hello-world/src 3

- Argument of type 'null' is not assignable to parameter of type 'SetStateAction<string>'. ts(2345) [13, 13]
- Property 'invalidGreeting' does not exist on type 'GreetingDetailProps'. Did you mean 'initialGreeting'? ts(2551) [14, 23]
GreetingEditor.tsx[5, 3]: 'initialGreeting' is declared here.
- Cannot find name 'nam'. Did you mean 'name'? ts(2552) [21, 16]
GreetingEditor.tsx[9, 10]: 'name' is declared here.

Empfehlung:

- verwenden!
- zum Lernen evtl. erst "nur" JS

MIT REACT LOSLEGEN

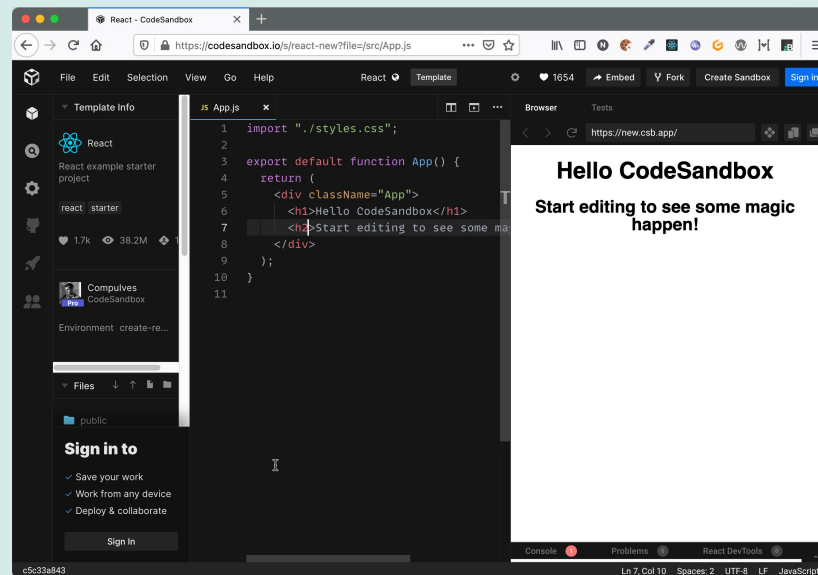
Erzeugen vom Project mit create-react-app

- <https://create-react-app.dev/>
- Offizielles Starter Kit
- Erzeugt fertig konfiguriertes Projekt
 - Build, Linter (statische Code Überprüfung), Test, TypeScript

MIT REACT LOSLEGEN

Erzeugen vom Project mit create-react-app

- <https://create-react-app.dev/>
- Offizielles Starter Kit
- Erzeugt fertig konfiguriertes Projekt
 - Build, Linter (statische Code Überprüfung), Test, TypeScript
- Alternative zum schnellen ausprobieren: <https://codesandbox.io/s/react-new>
 - React App im Browser entwickeln



React

Ökosystem

Arbeiten mit URLs

- De-facto-Standard: React Router

ROUTING

Arbeiten mit URLs

- React Router <https://reactrouter.com/>
- Wie auf dem Server: Pfade auf Komponenten mappen

```
import {Route, Switch} from "react-router";

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

TESTEN VON KOMPONENTEN

Die Basis: Jest

- <https://jestjs.io/>
- Standard JavaScript Test-Framework

```
function greet() { return "Hello!" }

// entspricht Test-Methode in Junit @Test
test("it should work", () => {

  const greeting = greet();

  // entspricht Assertions
  expect(greeting).toEqual("Hello");

});
```

TESTEN VON KOMPONENTEN

React Testing Library

- <https://testing-library.com/docs/react-testing-library/intro/>
- (Unit-)Testen ohne Browser

TESTEN VON KOMPONENTEN

React Testing Library

- <https://testing-library.com/docs/react-testing-library/intro/>
- (Unit-)Testen ohne Browser

```
test("it should behave fine", () => {  
  
  // Mock für Event-Handler  
  const onSaveHandler = jest.fn();  
  
  // Komponente rendern  
  render(  
    <GreetingEditor initialPhrase="Hello" initialGreeting="React"  
      onSave={onSaveHandler} />  
  );  
  
  // Ereignis simulieren  
  fireEvent.click(screen.getByText("Save"));  
  
  // Ergebnis überprüfen  
  expect(onSaveHandler).toHaveBeenCalledWith("Hello", "React");  
});
```

DATA FETCHING

Daten laden vom Server

- React macht keine Aussage, wie das geht

DATA FETCHING

Server-Zugriffe (per HTTP)

- Low-Level (Browser Standard): fetch API
 - https://developer.mozilla.org/de/docs/Web/API/Fetch_API
- Etwas weniger Low-Level: axios
 - <https://github.com/axios/axios>

DATA FETCHING

Server-Zugriffe (per HTTP)

- Low-Level (Browser Standard): fetch API
 - https://developer.mozilla.org/de/docs/Web/API/Fetch_API
- Etwas weniger Low-Level: axios
 - <https://github.com/axios/axios>
- High-Level, React-spezifisch:
 - SWR: <https://swr.vercel.app/>
 - React Query: <https://react-query.tanstack.com/>

DATA FETCHING

Server-Zugriffe (per HTTP)

- Low-Level (Browser Standard): fetch API
 - https://developer.mozilla.org/de/docs/Web/API/Fetch_API
- Etwas weniger Low-Level: axios
 - <https://github.com/axios/axios>
- High-Level, React-spezifisch:
 - SWR: <https://swr.vercel.app/>
 - React Query: <https://react-query.tanstack.com/>
- Oder für GraphQL:
 - Apollo: <https://www.apollographql.com/>

DATA FETCHING

SWR und React Query: vollständige Lösungen für React

```
import useSWR from "swr";

function GreetingApp() {
  const { data, error } = useSWR("/api/v1/greetings");

  if (error)
    return <Error msg="Loading failed" />

  if (!data)
    return <h1>Greetings loading...</h1>

  return <GreetingTable greetings={data} />
}
```

DATA FETCHING

SWR und React Query: vollständige Lösungen für React

```
import useSWR from "swr";
```

```
function GreetingApp() {  
  const { data, error } = useSWR("/api/v1/greetings");
```

```
  if (error)  
    return <Error msg="Loading failed" />
```

```
  if (!data)  
    return <h1>Greetings loading...</h1>
```

```
  return <GreetingTable greetings={data} />  
}
```

Wenn Request Status sich ändert,
wird Komponente neu gerendert,
=> neue Daten kommen zurück!

DATA FETCHING

SWR und React Query: vollständige Lösungen für React

- Folgen React Programmiermodell
- Globales Caching, Re-fetching
- Error Handling

```
import useSWR from "swr";
```

```
function GreetingApp() {  
  const { data, error } = useSWR("/api/v1/greetings");
```

```
  if (error)  
    return <Error msg="Loading failed" />
```

```
  if (!data)  
    return <h1>Greetings loading...</h1>
```

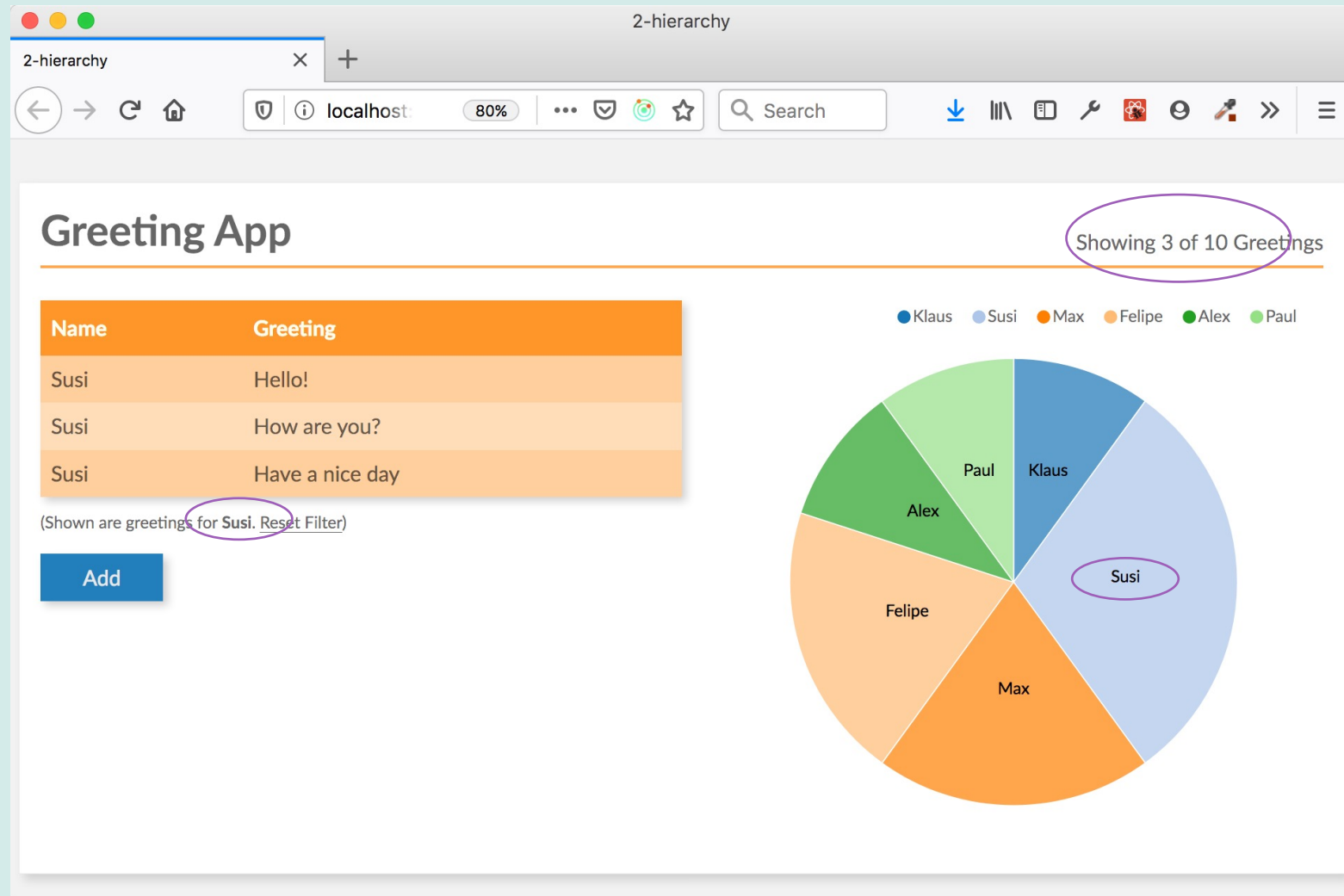
```
  return <GreetingTable greetings={data} />  
}
```

Wenn Request Status sich ändert,
wird Komponente neu gerendert,
=> neue Daten kommen zurück!

GLOBALER ZUSTAND

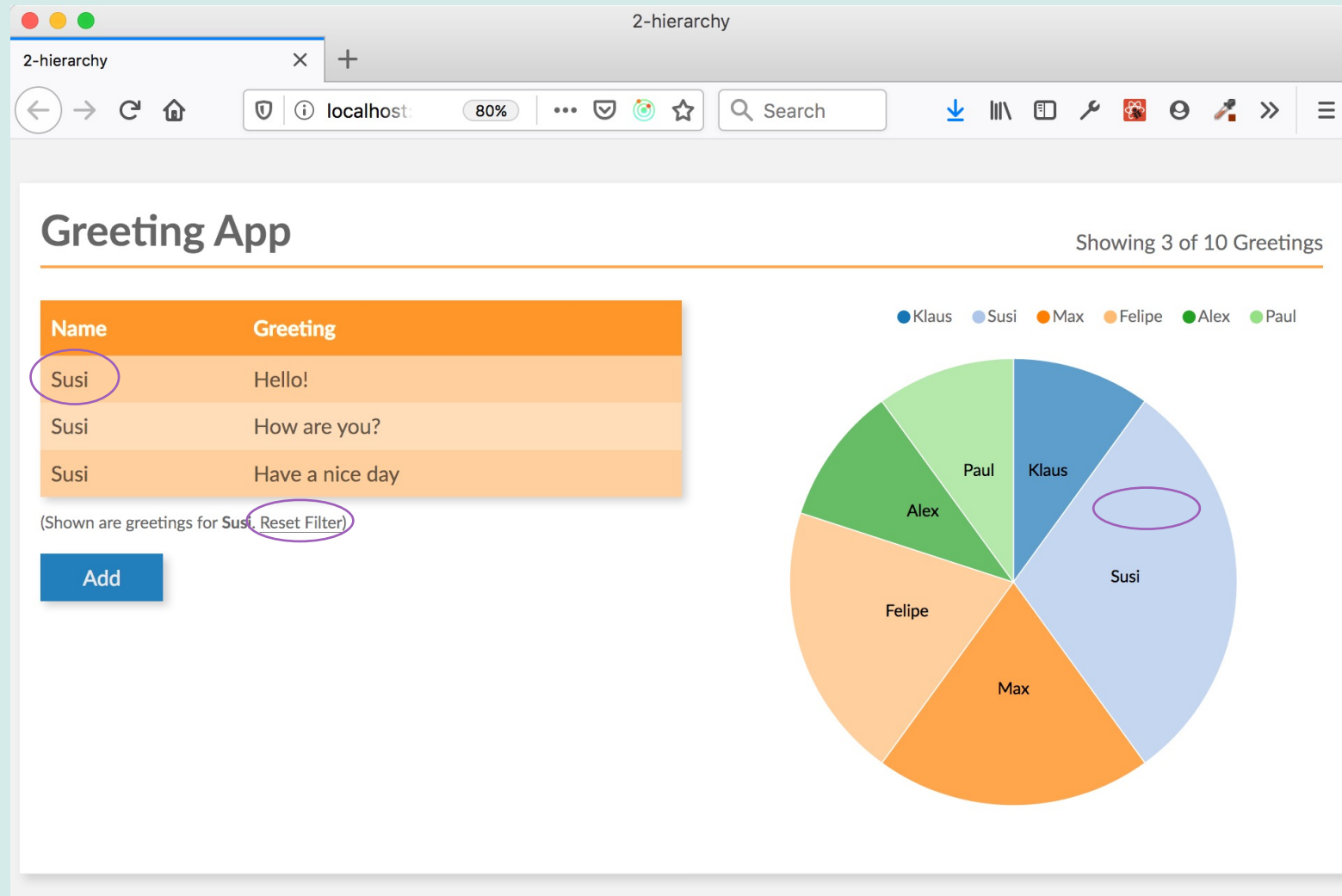
GLOBALER ZUSTAND

- Beispiel: Globale Daten in der Anwendung



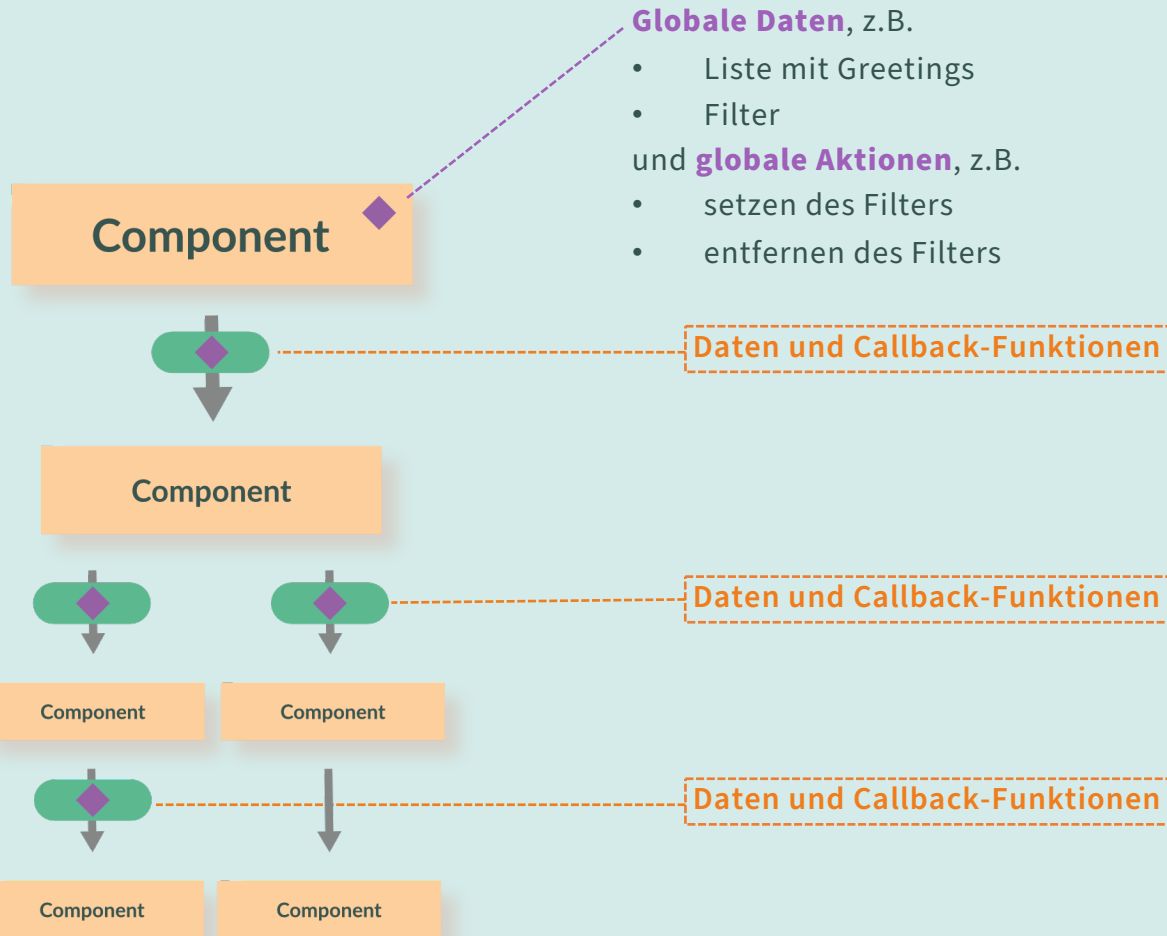
GLOBLAER ZUSTAND

- Beispiel: Globale Aktionen und Logik in der Anwendung



Globale Daten

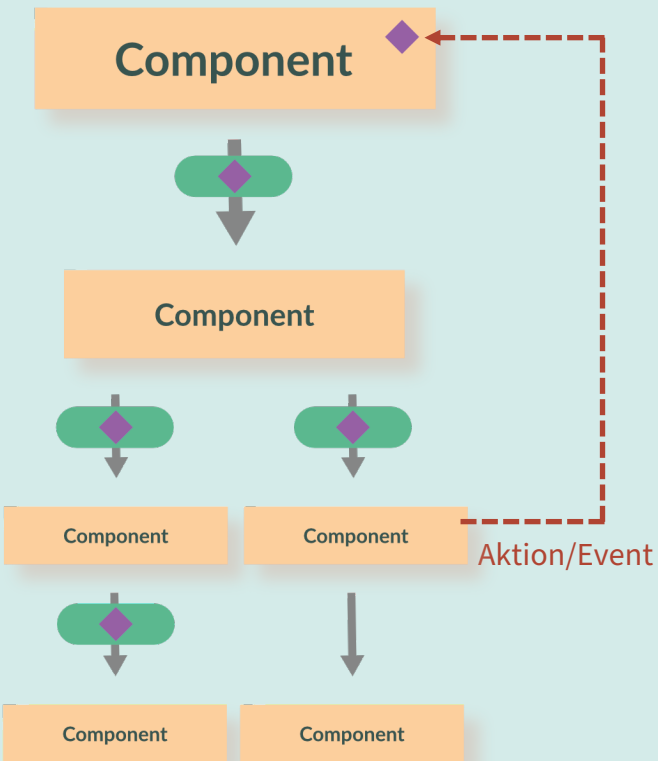
Der klassische React Weg: Properties durchreichen



Globale Daten

Der klassische React Weg: Properties durchreichen

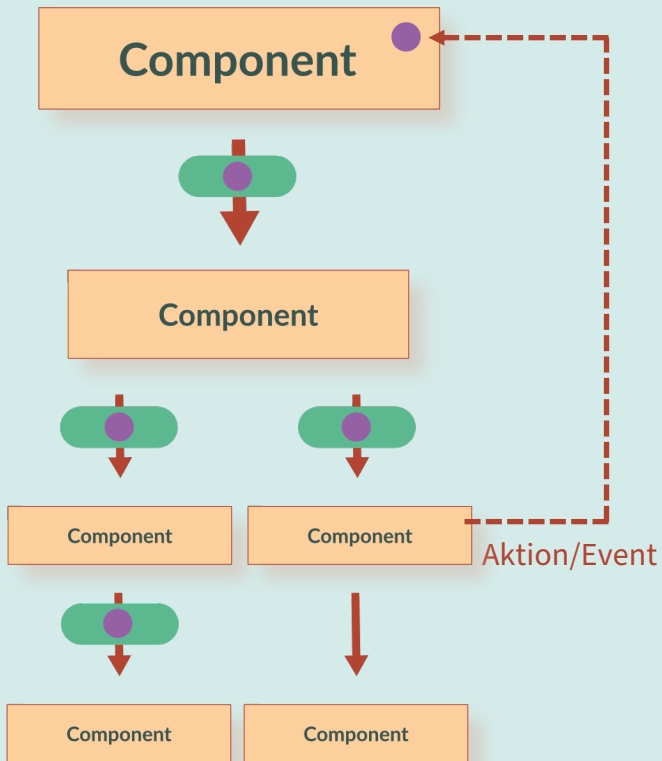
Unterkomponenten können Aktionen/Events auslösen, Aufruf von Callback-Funktionen



Globale Daten

Der klassische React Weg: Properties durchreichen

Typischer Datenfluss und Renderzyklus bleibt erhalten: Zustandsänderung -> Rendern

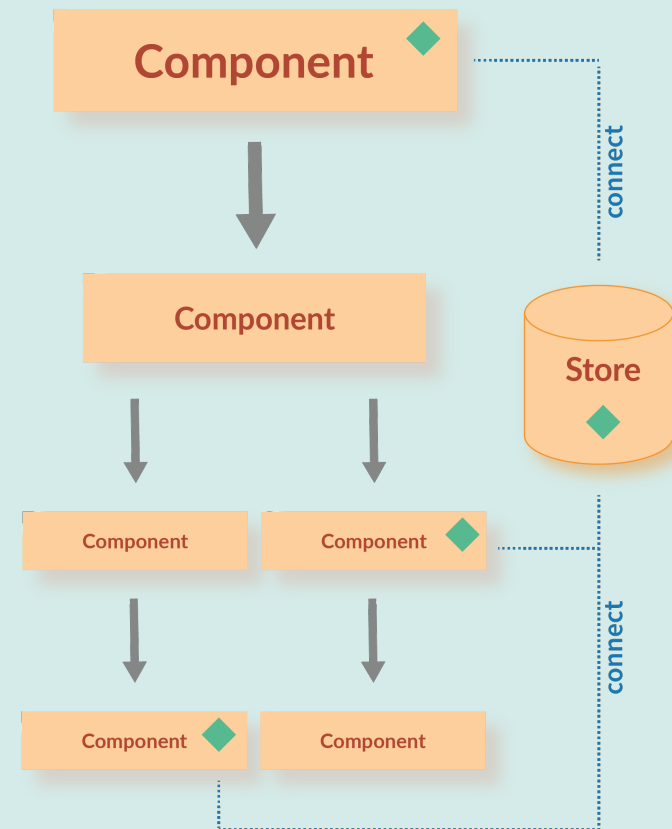
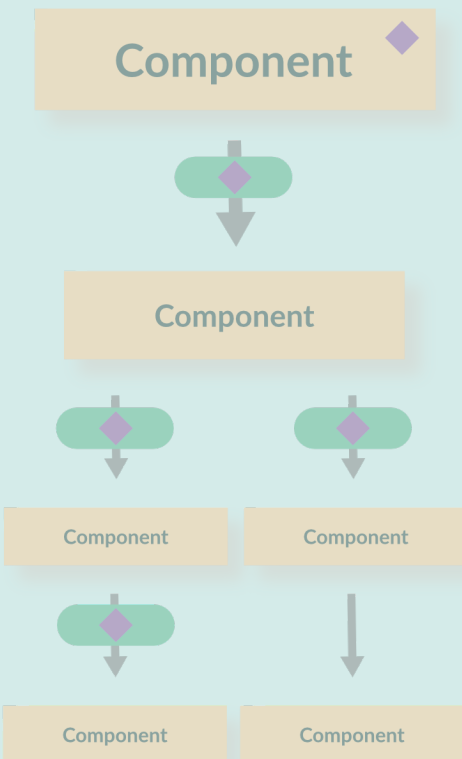


GLOBALER ZUSTAND

Externes Statemanagement: Zustand wandert aus den Komponenten

Prominente Vertreter: [Redux](#) oder [MobX](#)

[Store](#) hält globale Daten und Logik, Komponenten lesen direkt daraus



Beispiel: Redux

Zugriff mit Selector auf globalen Store

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

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

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

BEISPIEL: REDUX

Verarbeitung von Aktionen

Logik ist in Reducer-Funktionen untergebracht

Keine Abhängigkeit auf React oder andere Bibliothek

Verarbeitet Aktionen und liefert neuen Zustand zurück

```
function greetingsReducer(oldState, action) {  
  if (action.type === "removeGreeting") {  
    return {  
      ...oldState,  
      greetings: oldState.greetings.filter(g => g.id !== action.greetingId)  
    }  
  }  
}  
  
// weitere Actions behandeln...  
}
```

STYLING (CSS)

STYLING (CSS)

STYLING (CSS)

Option 1: CSS und CSS Modules

Support für CSS (Modules) ist eingebaut

Unterstützt wird dabei auch SASS

GreetingTable.css
(oder .scss)

```
.GreetingTable {  
  background-color: orange;  
}
```

GreetingTable.js
(oder .tsx)

```
import styles from "../GreetingTable.css";  
  
function GreetingTable() {  
  return (  
    <table className={styles.GreetingTable}>  
      ...  
    </table>  
  )  
}
```

STYLING (CSS)

Option 2: CSS-in-JS

- Styles werden direkt in JavaScript-Code geschrieben
- Vielleicht konsequenteste Umsetzung der Komponenten Idee
- Ermöglicht Anpassung der Styles zur Laufzeit, z.B. abhängig von Props
- Nicht unumstritten (was machen Leute, die CSS, aber kein JS können?)
- Bekannte Vertreter:
 - Styled Components <https://styled-components.com>
 - Emotion <https://emotion.sh/docs/@emotion/react>

Option 2: CSS-in-JS

- Beispiel: Styled Components

"Reguläre"
Komponente

```
function RawGreetingTable(props) {  
  return <table className={props.className}> ... </table>;  
}
```


STYLING (CSS)

Option 2: CSS-in-JS

- Beispiel: Styled Components

"Reguläre"
Komponente

```
function RawGreetingTable(props) {  
  return <table className={props.className}> ... </table>;  
}
```

Styling Angaben
mit dynamischen
Werten

```
import styled from "styled-components";  
  
const GreetingTable = styled(RawGreetingTable)`  
  background-color: orange  
  font-size: ${props => props.greetings.length > 10 ? "15px" : "20px"}  
`
```

STYLING (CSS)

Option 2: CSS-in-JS

- Beispiel: Styled Components

"Reguläre"
Komponente

```
function RawGreetingTable(props) {  
  return <table className={props.className}> ... </table>;  
}
```

Styling Angaben
mit dynamischen
Werten

```
import styled from "styled-components";  
  
const GreetingTable = styled(RawGreetingTable)`  
  background-color: orange  
  font-size: ${props => props.greetings.length > 10 ? "15px" : "20px"}  
`
```

Verwendung
wie gewohnt

```
function GreetingApp() {  
  return <GreetingTable greetings={...} />  
}
```

Texte und Daten übersetzen und korrekt formatieren

Typische Vertreter:

- react-intl (<https://formatjs.io/docs/react-intl/>)
- react-i18next (<https://react.i18next.com/>)

Sehr ähnliches Feature-Set

- Einlesen von Sprach-Dateien vom Server
- Übersetzen von Texten inkl. Platzhaltern, Plural etc.
- Formatieren von Datum und Zahlen
- Dynamisches Umstellen des Locales zur Laufzeit

INTERNATIONALISIERUNG

Beispiel: react-i18next

```
function TableHeader(greetings) {
  const { t } = useTranslation();

  return <h1>
    {t("tableHeader", { count: greetings.length })}
  </h1>
}

// i18n-Datei (en)
{
  "tableHeader": "Showing one greeting",
  "tableHeader_plural": "Showing {{count}} greetings"
}

// i18n-Datei (de)
{
  "tableHeader": "Ein Gruß",
  "tableHeader_plural": "Angezeigt werden {{count}} Grüße"
}
```

NILS HARTMANN

<https://nilshartmann.net>

<https://reactbuch.de>



Vielen Dank!

Slides: <https://react.schule/jax2021-react>

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

Fragen & Kontakt: nils@nilshartmann.net

@NILSHARTMANN