

NILS HARTMANN

Einstegen und loslegen

# React

Eine praktische Einführung

Slides: <https://react.schule/cd2022-react>

# NILS HARTMANN

Kontakt: [nilshartmann.net](mailto: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**

# React

# Einführung

<https://reactjs.org>

- Minimales API
- Minimales Feature Set

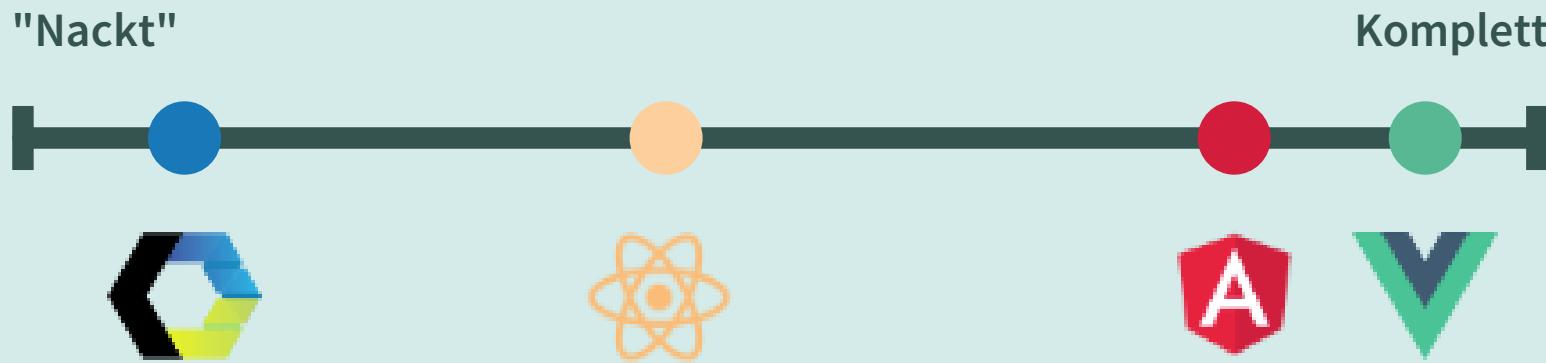
**Neue Doku: <https://beta.reactjs.org>**

# REACT

<https://reactjs.org>

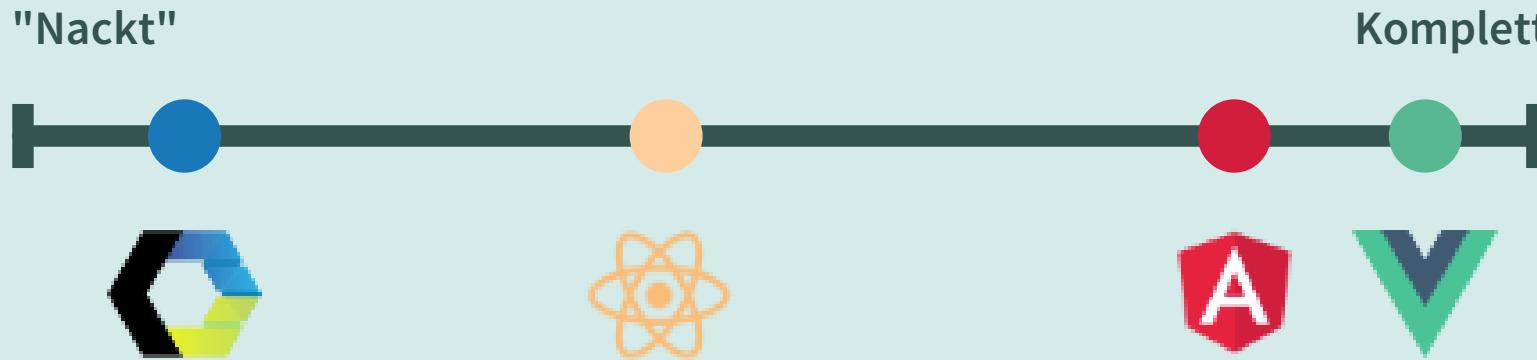
- Minimales API
- Minimales Feature Set

**Neue Doku: <https://beta.reactjs.org>**



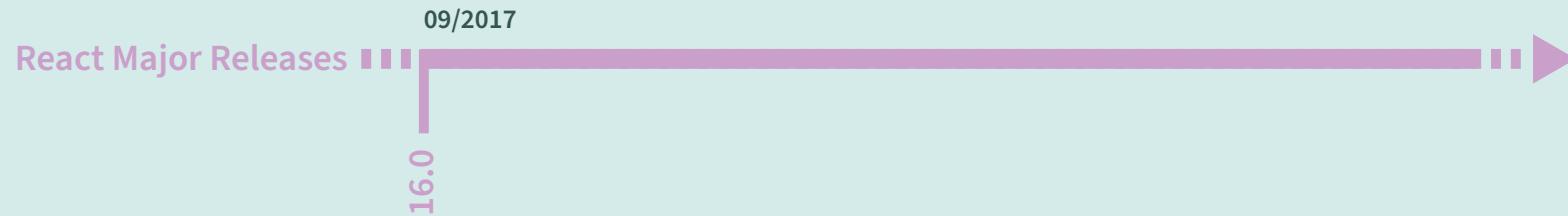
<https://reactjs.org>

- Minimales API
- Minimales Feature Set
- 👉 Man kann (muss ?) viele Entscheidungen selber treffen
- 👉 In ernsthaften Anwendungen werden weitere Bibliotheken benötigt



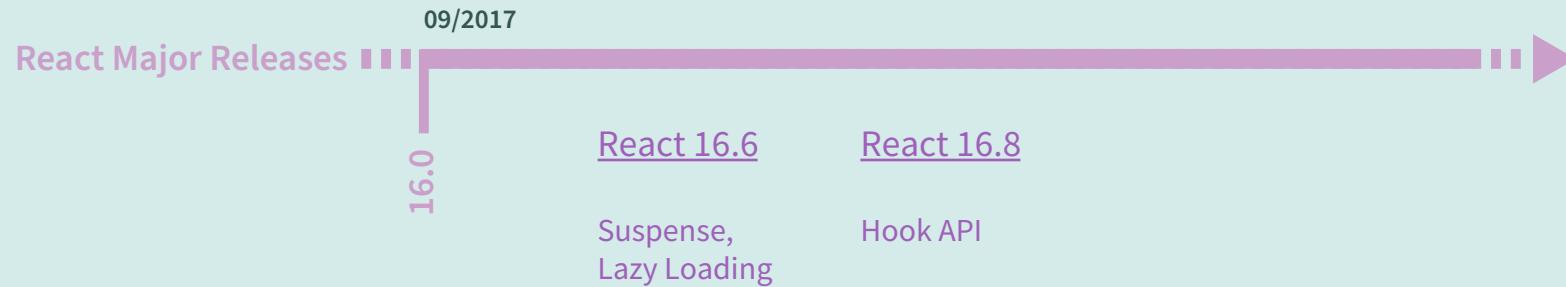
# RELEASEZYKLEN

## Sehr lange Release-Zyklen



# RELEASEZYKLEN

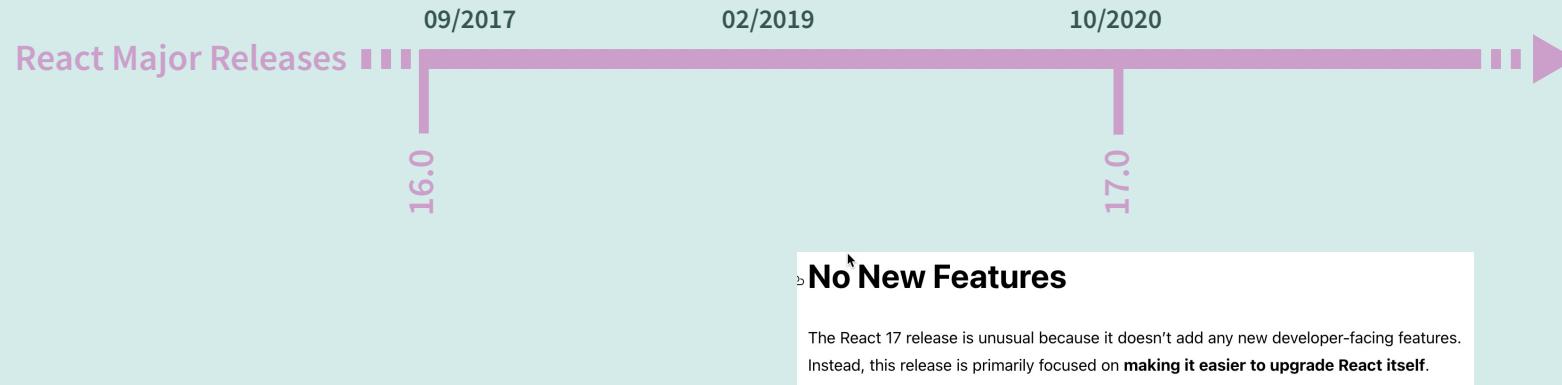
## Sehr lange Release-Zyklen



Neue Features und APIs in Minor-Versionen!

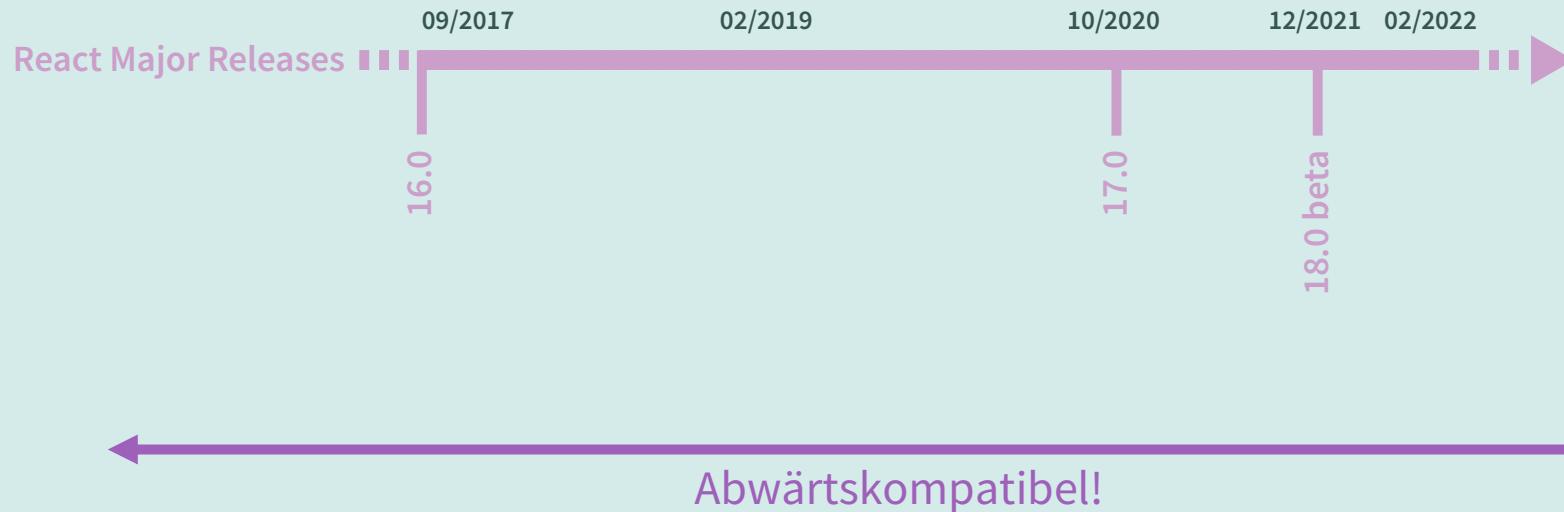
# RELEASEZYKLEN

## Sehr lange Release-Zyklen



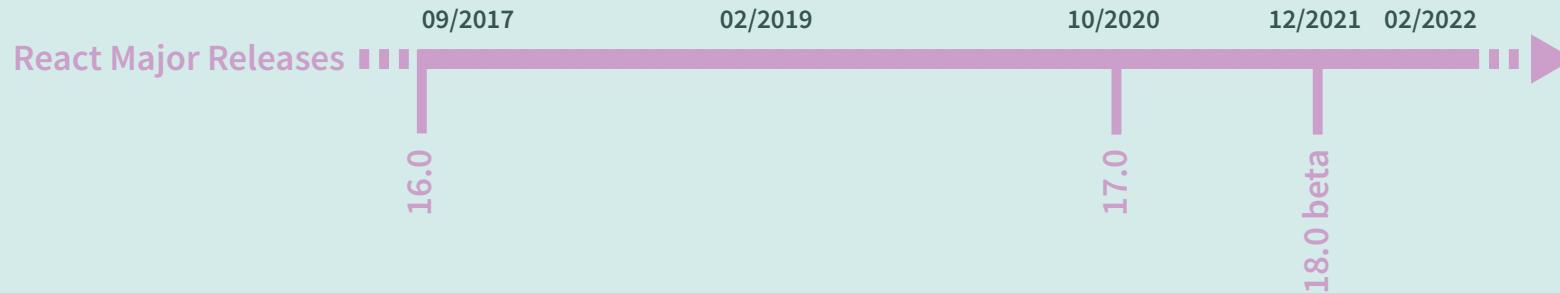
# RELEASEZYKLEN

## Sehr lange Release-Zyklen



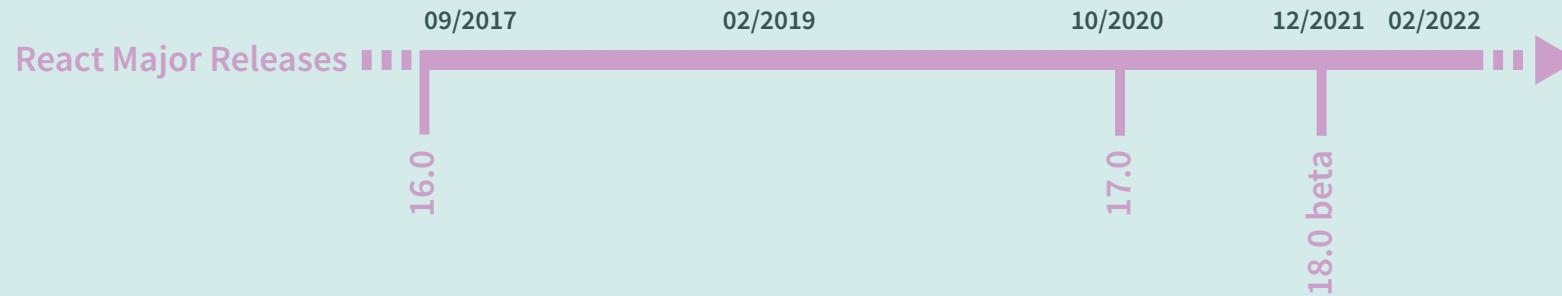
# RELEASEZYKLEN

## Sehr lange Release-Zyklen

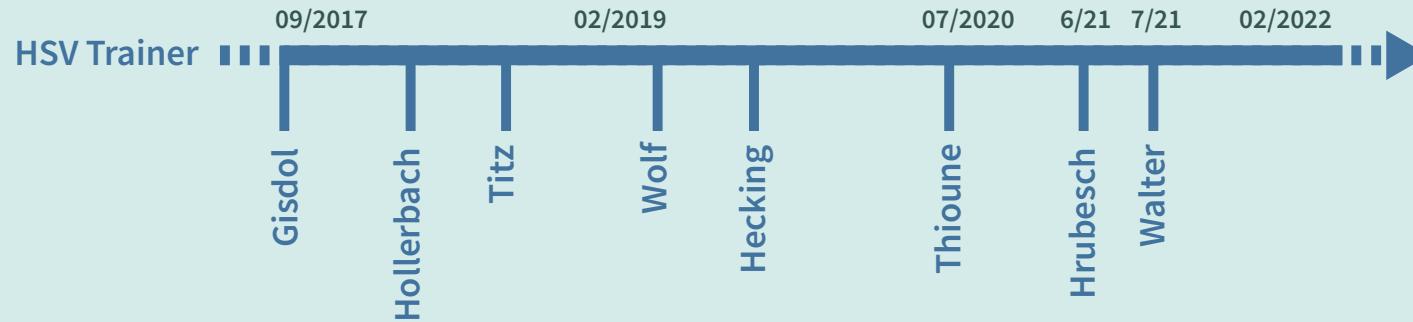


# RELEASEZYKLEN

## Sehr lange Release-Zyklen



## Sehr kurze Release-Zyklen (Symbolbild)



# Greeting App

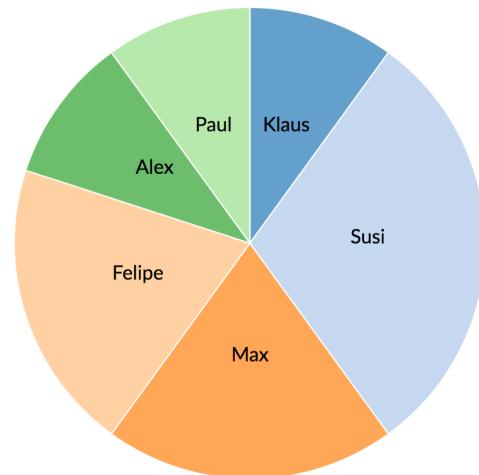
Showing 10 of 10 Greetings

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

App

## Greeting App

GreetingController

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

FilterPanel

GreetingMaster

Counter

Showing 10 of 10 Greetings

Chart

The pie chart displays the proportion of greetings for each person. The segments are labeled with the names: Klaus, Susi, Max, Felipe, Alex, and Paul. The sizes of the segments correspond to the frequency of their greetings in the list.

Name	Proportion
Klaus	~10%
Susi	~25%
Max	~15%
Felipe	~20%
Alex	~10%
Paul	~10%

GREETING APP: KOMPONENTEN

# React Komponenten

Klassische Aufteilung

**Logik, Model**  
(Java, JS)



**View**  
(HTML, Template)



**Gestaltung**  
(CSS)



Aufteilung in Komponenten



Button



GreetingEditor  
Counter

Grafik Inspiriert von: [https://pbs.twimg.com/media/DCXJ\\_tjXoAAoBbu.jpg](https://pbs.twimg.com/media/DCXJ_tjXoAAoBbu.jpg)

**KOMPONENTEN**

## React-Komponenten

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



Button



GreetingEditor



Counter

# 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';

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

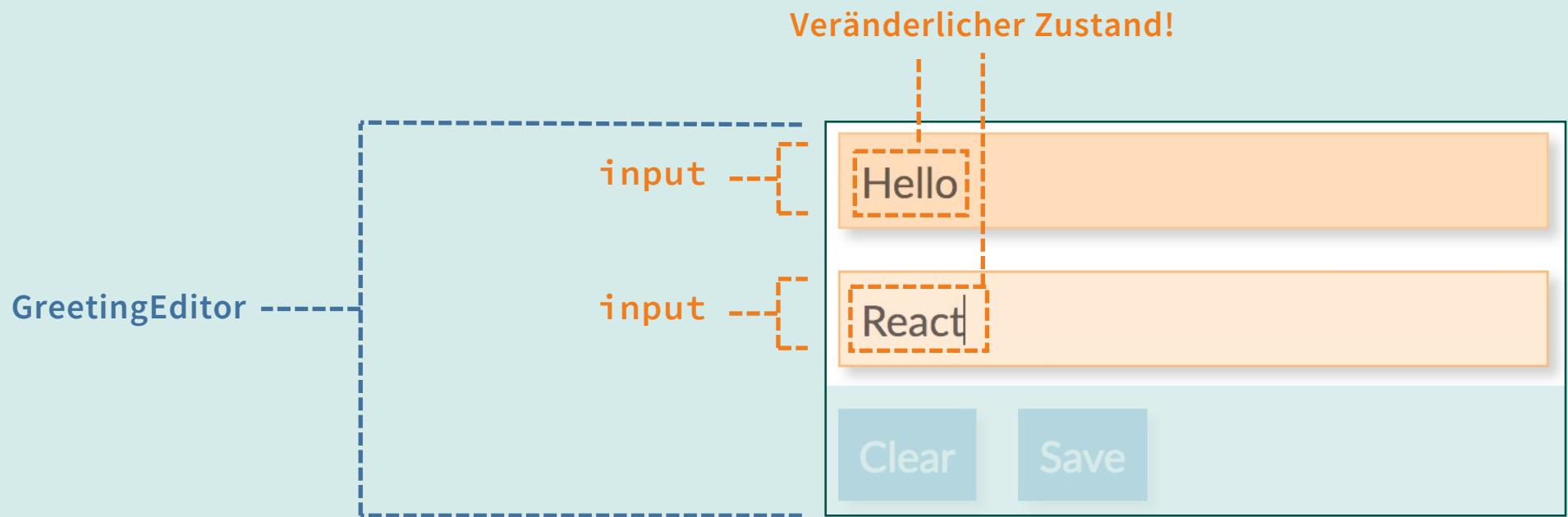
Model a.k.a

State

ARBEITEN MIT VERÄNDERLICHEN DATEN

# STATE

## Beispiele: Eingabefeld



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

## BEISPIEL: EINGABEFELD

Komponente

Hello

-- input

"Hook"-Funktion

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

|

Aktueller State    Setter

|

Initialer Wert

```
}
```

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

## BEISPIEL: EINGABEFELD

Komponente

"Hook"-Funktion

1. Input mit Wert aus State befüllen

2. Zustand neu setzen

Hello

]}-- input

```
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

1. Input mit Wert aus State befüllen

2. Zustand neu setzen

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

```
  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

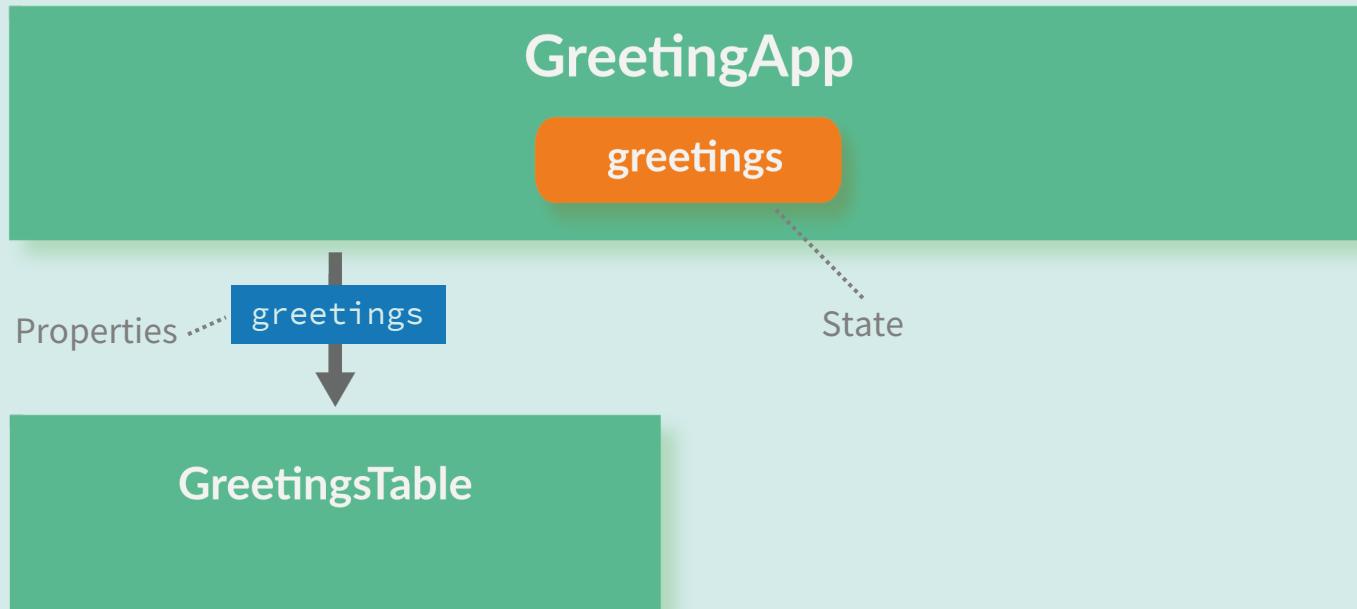


```
function GreetingEditor() {  
  const [ phrase, setPhrase ] = React.useState("Hello");  
  const [ name, setName ] = React.useState("React");  
  const saveDisabled = phrase === "" || name === "";  
  
  return <div>  
    <input value={phrase} onChange={e => setPhrase(e.target.value)} />  
    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

# HIERARCHIEN VON KOMPONENTEN

## Gemeinsame Daten wandern in gemeinsame Oberkomponenten

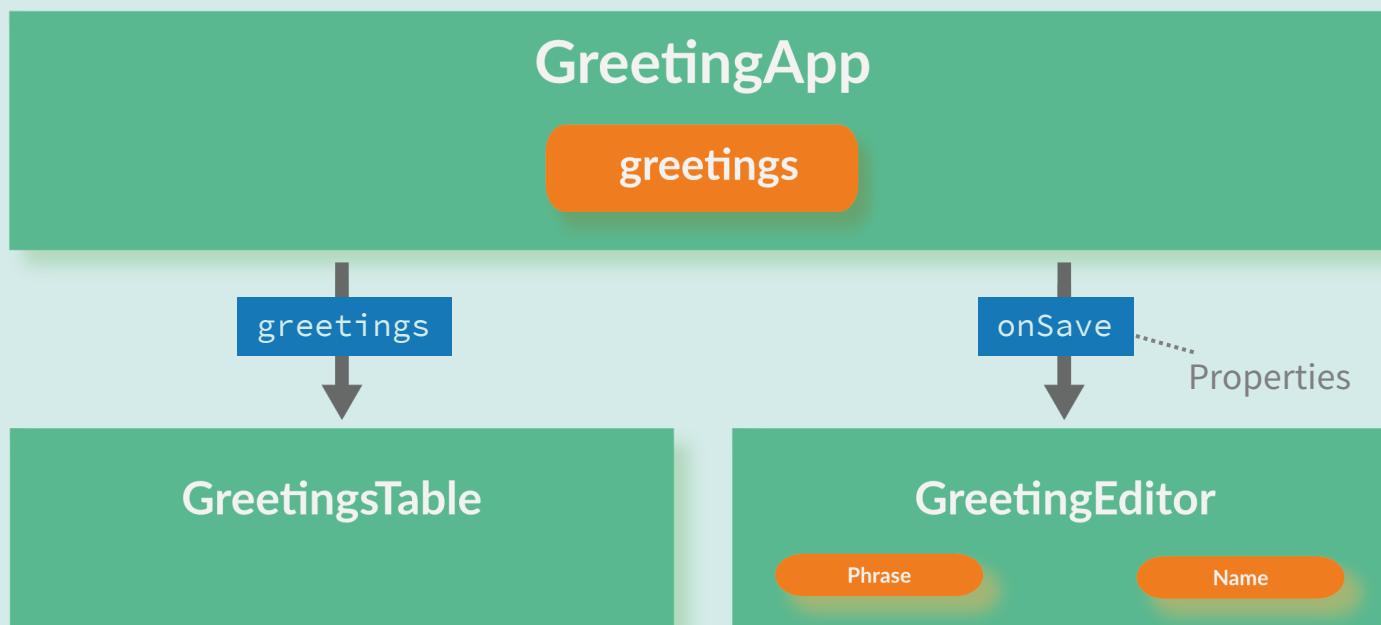
- Von dort per Properties nach unten



# HIERARCHIEN VON KOMPONENTEN

## Gemeinsame Daten wandern in gemeinsame Oberkomponenten

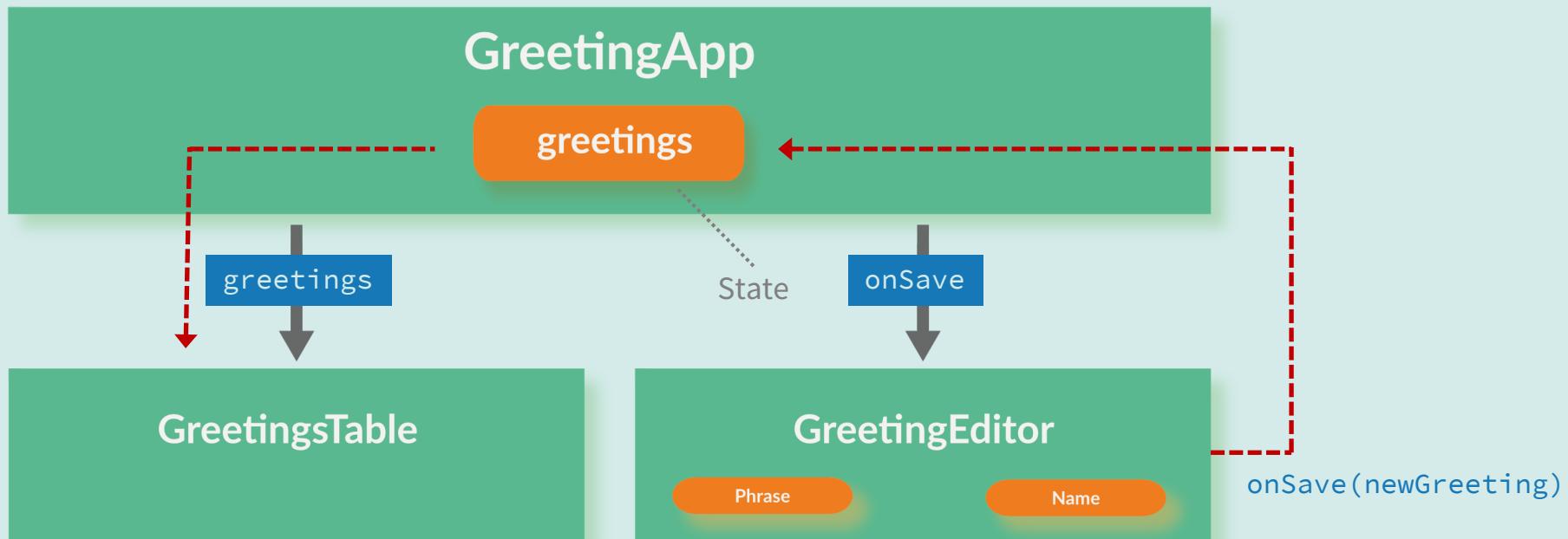
- Von dort per Properties nach unten
- Unterkomponenten informieren nach oben per Callback-Funktion



# 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

The screenshot shows a code editor window for a file named `GreetingEditor.tsx`. The code defines a component with TypeScript interfaces and functions. A tooltip is displayed over the word `name` in the `value` prop of an `<input>` element, indicating that the name 'nam' cannot be found, suggesting a misspelling of 'name'. The code editor's bottom bar shows various developer tools like Wallaby, ESLint, Prettier, and Apollo.

```
react-training — GreetingEditor.tsx — https://nilshartmann.net
GreetingEditor.tsx 3, U ●

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
19         type="text"
19         value="nam"
19         name="name"
19         placeholder="Name"
19       />
20
21     </div>
22   );
23
24

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

any
View Problem (F8) Quick Fix... (F3)
GreetingEditor.tsx(9, 10): 'name' is declared here.

any
Value 'nam' is not assignable to parameter of type 'SetStateAction<string>'. ts(2345) [13, 13]
GreetingEditor.tsx(5, 3): 'initialName' is declared here.

any
Property 'invalidGreeting' does not exist on type 'GreetingDetailProps'. Did you mean 'initialGreeting'? ts(2551) [14, 23]
GreetingEditor.tsx(5, 3): 'initialGreeting' is declared here.

any
Cannot find name 'nam'. Did you mean 'name'? ts(2552) [21, 16]
GreetingEditor.tsx(9, 10): 'name' is declared here.

OUTPUT PROBLEMS 3 DEBUG CONSOLE
PROBLEMS
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]
Cannot find name 'nam'. Did you mean 'name'? ts(2552) [21, 16]
```

## Empfehlung:

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

# MIT REACT LOSLEGEN

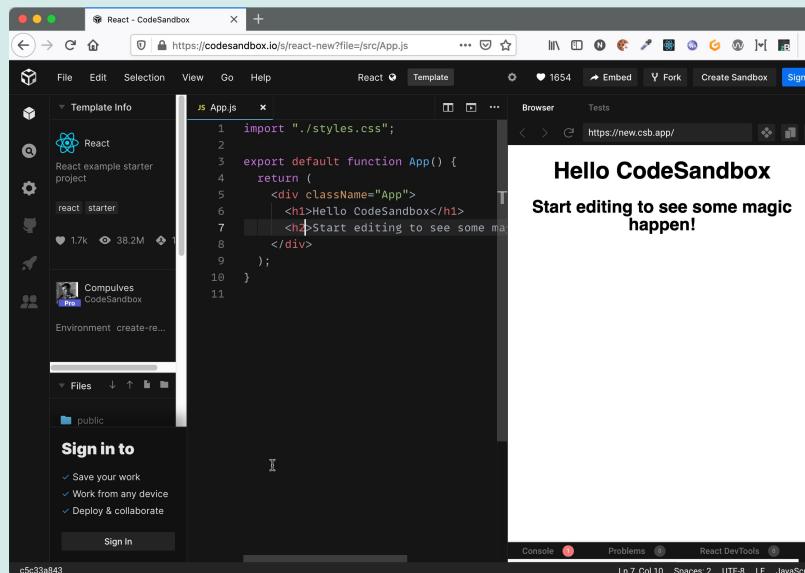
## Erzeugen vom Project mit `create-react-app`

- <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

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

```
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>
    </>
  );
}
```

### Lesen und Schreiben von Daten

- React macht keine Aussage, wie das geht
- Typische Vertreter:
  - SWR (<https://swr.vercel.app/>)
  - React Query (<https://react-query.tanstack.com/>)
  - Apollo GraphQL (<https://www.apollographql.com/docs/react/>)

# DATA FETCHING

## Beispiel: SWR

```
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

## Beispiel: SWR

```
import useSWR from "swr";

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

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



Daten werden gecached und stehen allen Komponenten zur Verfügung  
=> schnelle und konsistente Darstellung

# 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
- Funktioniert im CI-Build

```
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");  
});
```

# TESTEN VON KOMPONENTEN

## Browser Testing

- „Black Box“-Tests im Browser
  - TestCafe (<https://testcafe.io/>)
  - Cypress (<https://www.cypress.io/>)
  - (Selenium...)
- Tests werden in JavaScript/TypeScript geschrieben
- Headless Ausführung im CI-Build möglich

# **GLOBALER ZUSTAND**

# GLOBALER ZUSTAND

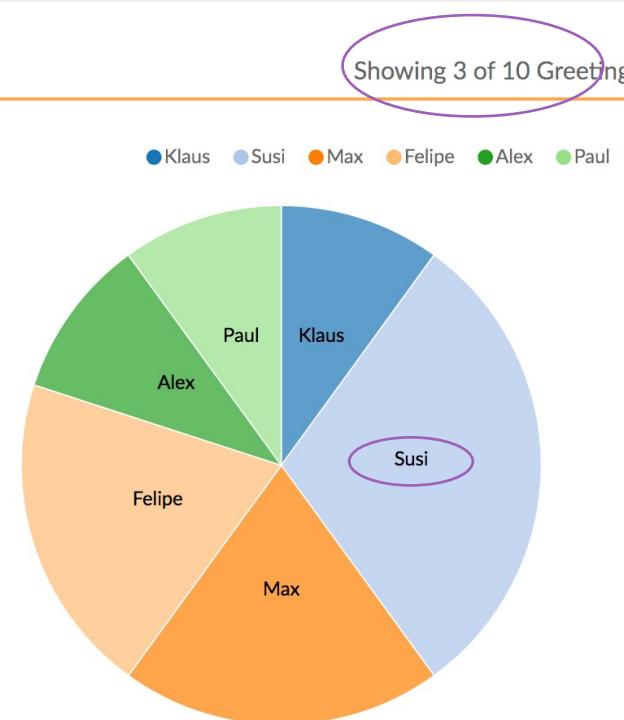
- Beispiel: Globale Daten in der Anwendung

Screenshot of a web application titled "Greeting App" showing global state management.

The application interface includes:

- A header bar with tabs: "2-hierarchy" (active), "localhost" (address bar), and a search bar.
- A message: "Showing 3 of 10 Greetings".
- A table showing greetings for "Susi":

Name	Greeting
Susi	Hello!
Susi	How are you?
Susi	Have a nice day
- A note: "(Shown are greetings for Susi. Reset Filter)".
- An "Add" button.
- A pie chart showing the distribution of greetings by sender:
  - Klaus (blue)
  - Susi (light blue)
  - Felipe (orange)
  - Max (dark orange)
  - Alex (green)
  - Paul (light green)



# GLOBLAER ZUSTAND

- Beispiel: Globale Aktionen und Logik in der Anwendung

2-hierarchy

localhost 80% Search

## Greeting App

Showing 3 of 10 Greetings

Name	Greeting
Susi	Hello!
Susi	How are you?
Susi	Have a nice day

(Shown are greetings for Susi. Reset Filter)

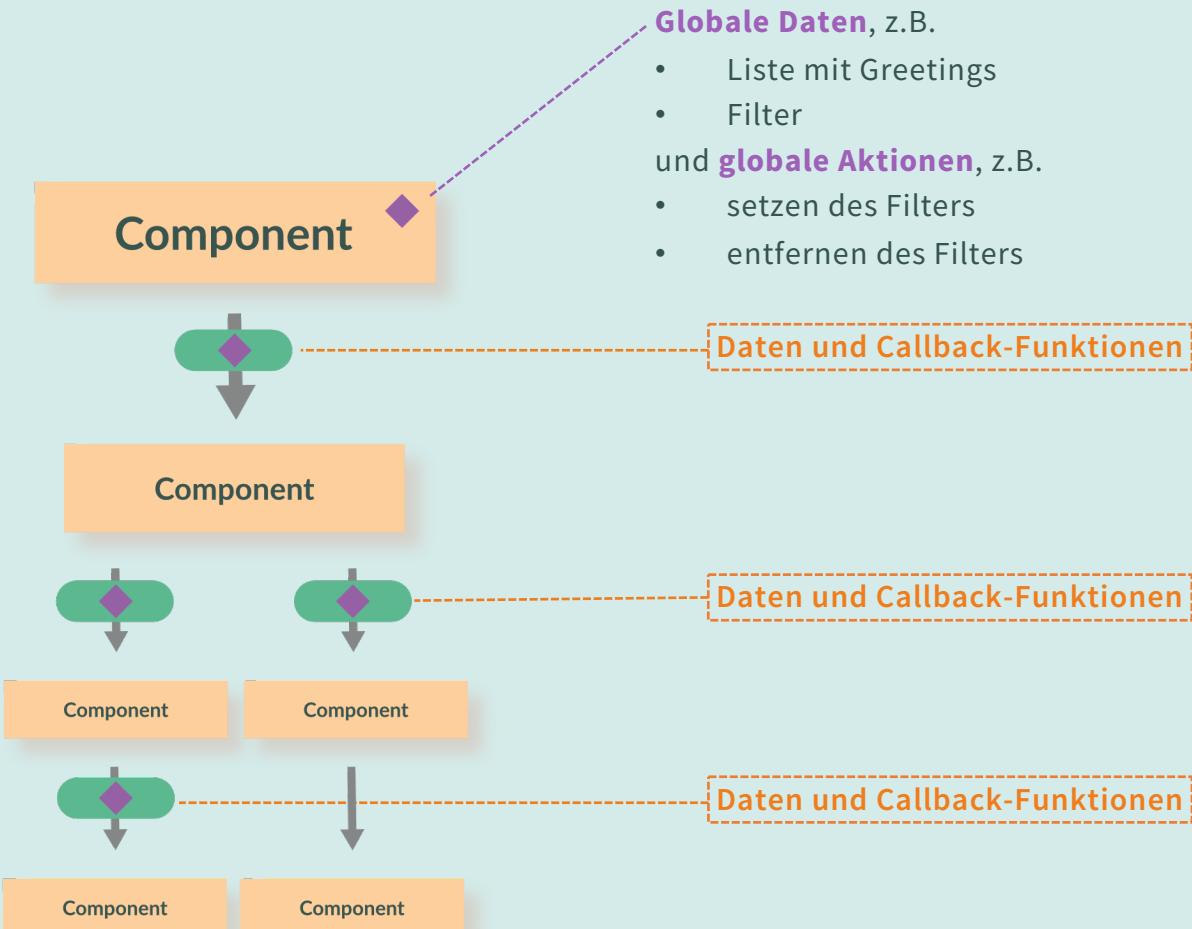
Add

A pie chart illustrating the distribution of greetings. The slices represent different names: Klaus (blue), Susi (light blue), Felipe (orange), Max (dark orange), Alex (green), and Paul (light green). The slice for Susi is highlighted with a purple oval.

Name	Percentage
Klaus	~10%
Susi	~30%
Felipe	~20%
Max	~15%
Alex	~10%
Paul	~5%

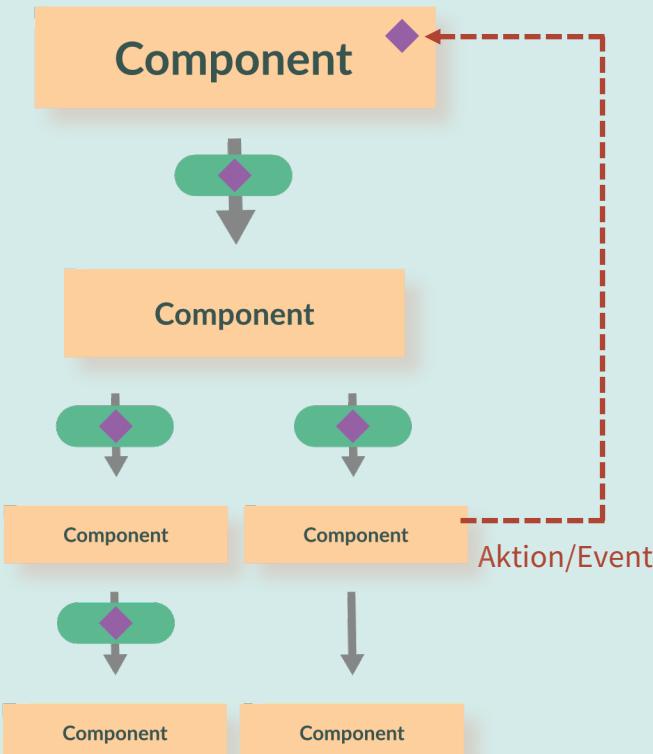
# GLOBALE DATEN

## Der klassische React Weg: Properties durchreichen



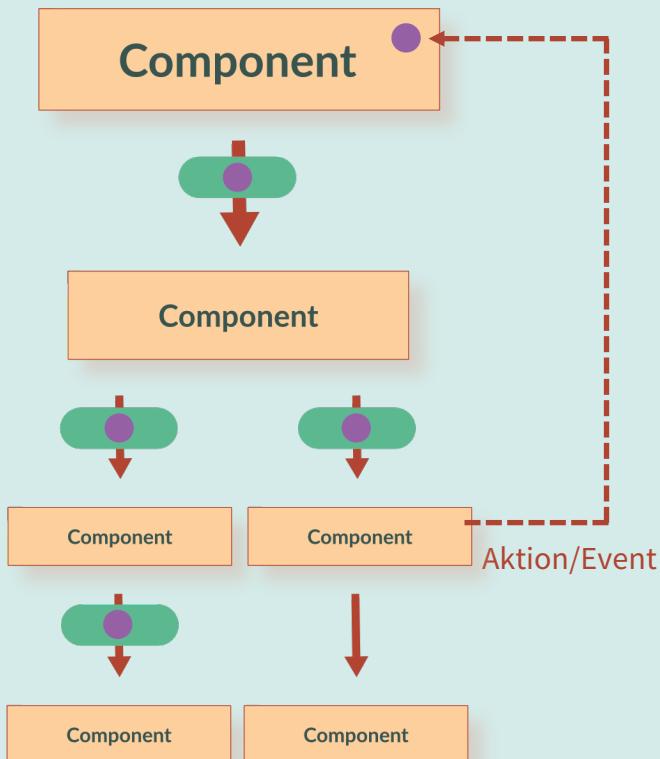
## Der klassische React Weg: Properties durchreichen

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



## Der klassische React Weg: Properties durchreichen

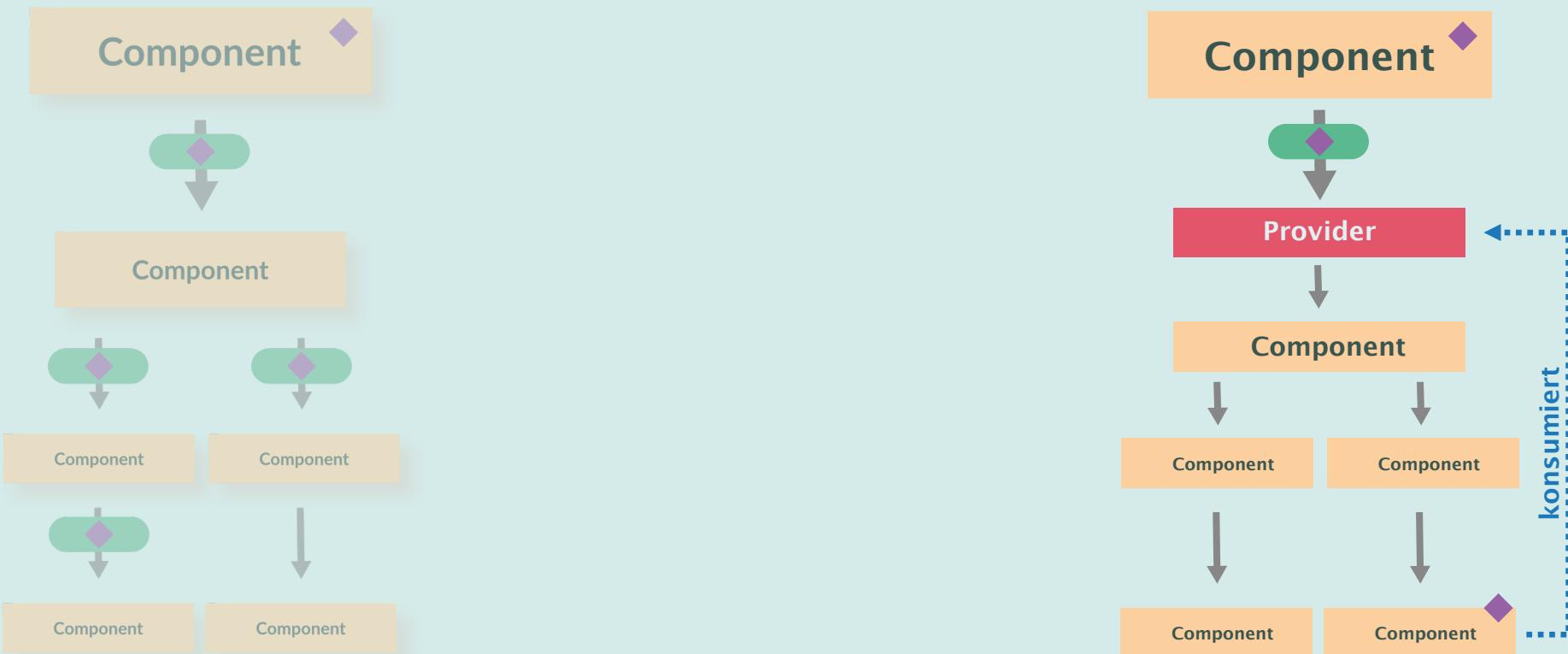
Typischer Datenfluss und Renderzyklus bleibt erhalten: Zustandsänderung -> Rendern



# GLOBALER ZUSTAND

## **React Context API:** Provider stellt Daten und Aktionen zur Verfügung

Verhalten ähnlich wie Properties durchreichen,  
aber ohne Properties durch zu reichen

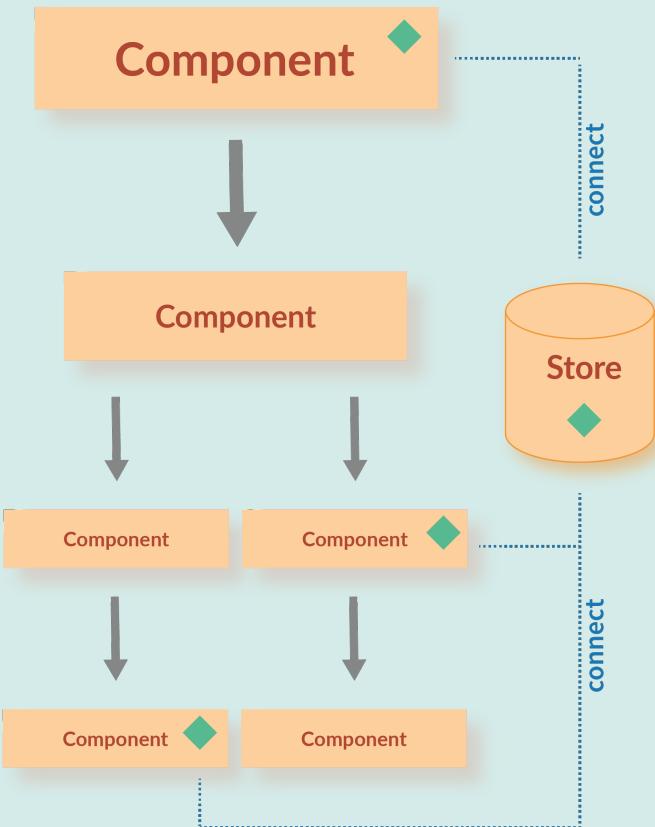
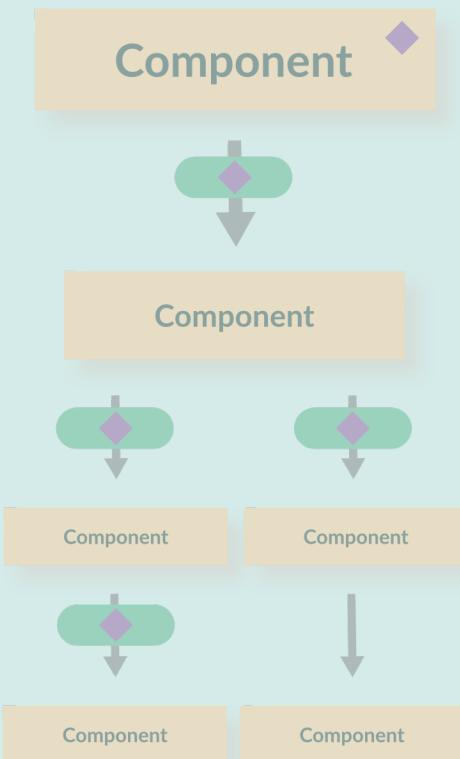


# GLOBALER ZUSTAND

**Externes Statemanagement:** Zustand wandert aus den Komponenten

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

**Store** hält globale Daten und Logik, Komponenten lesen direkt daraus



## 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>  
                      )  
                      }

### Option 2: CSS-in-JS

- Styles werden direkt in JavaScript-Code geschrieben
- Vielleicht konsequenterste 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>

# STYLING (CSS)

## 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

## Empfehlung

- Verwenden, was besser gefällt

# 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"
}
```

## ZUSAMMENFASSUNG

### Klein und einfach starten...

- React Bordmittel lernen und verwenden
- Für die allermeisten Use-Cases gibt mittlerweile De-facto-Standards

**NILS HARTMANN**

<https://nilshartmann.net>

# Vielen Dank!

Slides: <https://react.schule/cd2022-react>

Meine Workshops: <https://react.schule>

Fragen & Kontakt: [nils@nilshartmann.net](mailto:nils@nilshartmann.net)

Feedback



@NILSHARTMANN