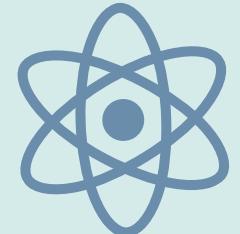


NILS HARTMANN | [HTTPS://NILSHARTMANN.NET](https://nilshartmann.net)

SINGLE-PAGE-ANWENDUNGEN MIT

React

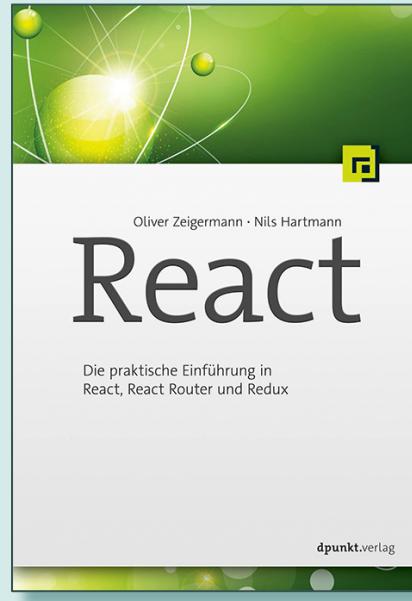


Slides: <http://bit.ly/oose-react>

# **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**

**MODERNE WEB ANWENDUNGEN**

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

X 0 Y 0

W 375 H 667

0° 0

CONSTRANTS

Left Top

LAYER

Pass Through 100%

FILL

Image 100%

STROKE

EFFECTS

Layer Blur

EXPORT

Click + to add an export setting

?

HTTPS://WWW.FIGMA.COM

# Greeting App

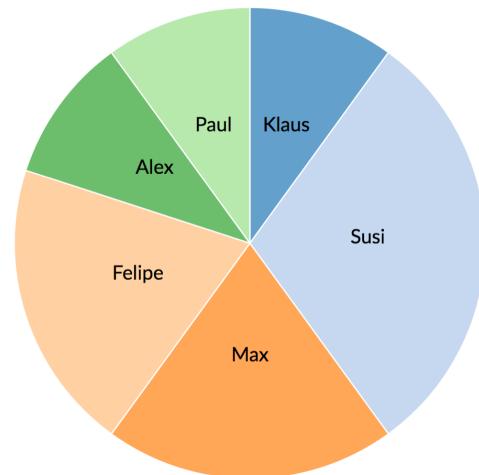
Showing 10 of 10 Greetings

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

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

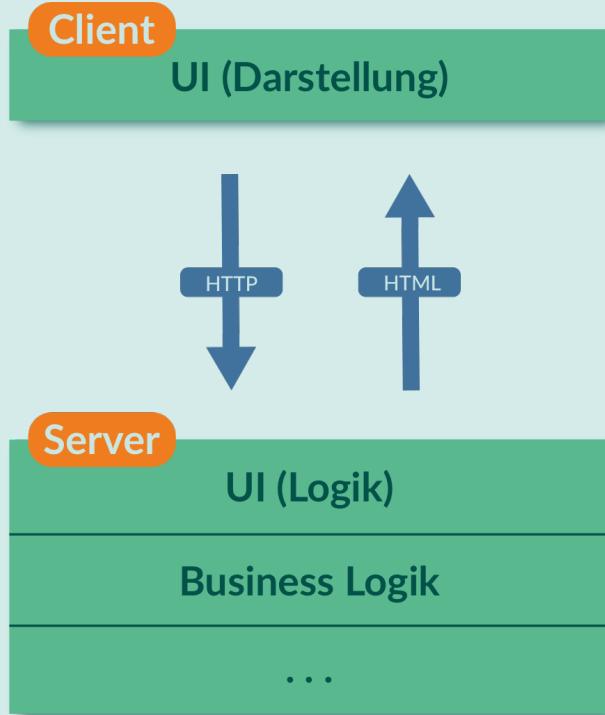


## BEISPIEL: DIE GREETING APP

# **KLASSISCHE WEB-ANWENDUNGEN**

**RÜCKBLICK**

# RÜCKBLICK: KLASSISCHE WEB ANWENDUNG



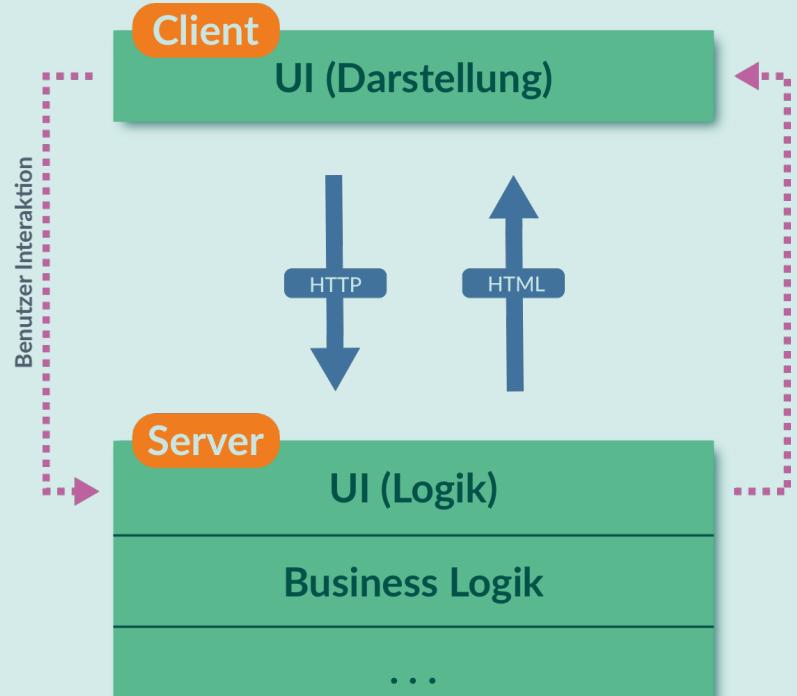
## Klassische Web Anwendung:

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

## Technologie

- JSP, Thymeleaf, JSF
- jQuery

# RÜCKBLICK: KLASSISCHE WEB ANWENDUNG



## Technologie

- JSP, Thymeleaf, JSF
- jQuery

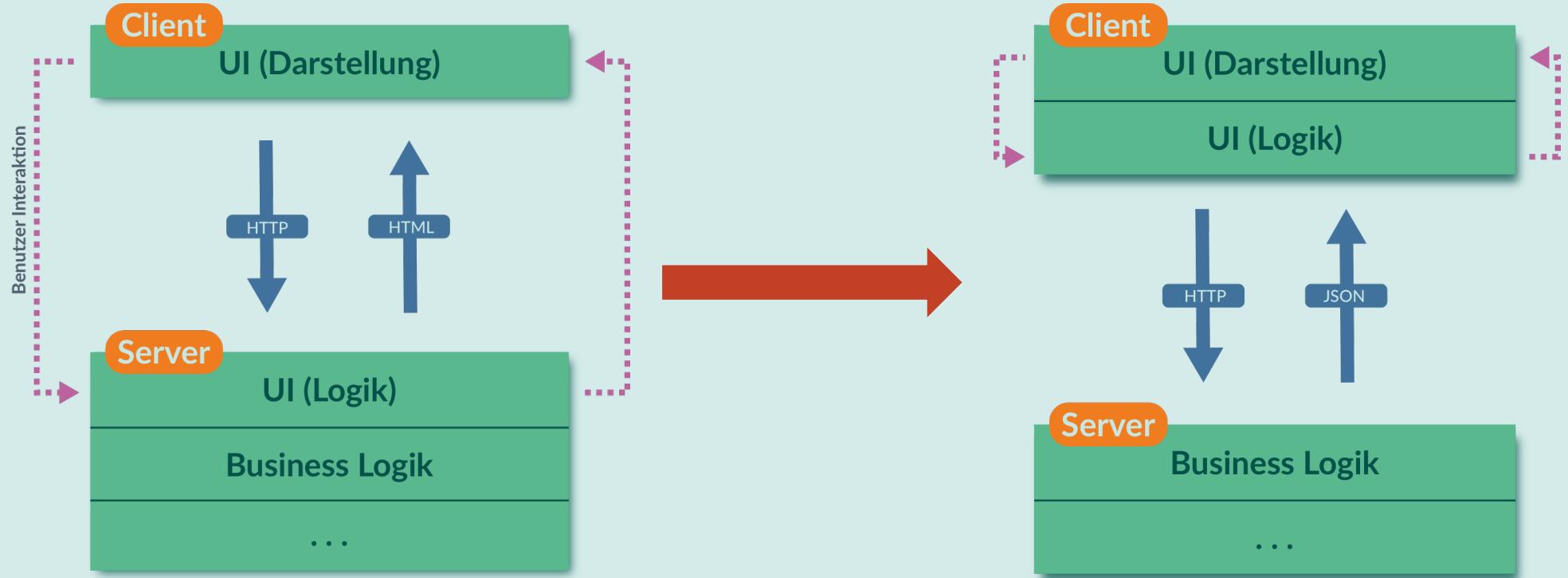
## Klassische Web Anwendung:

- 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

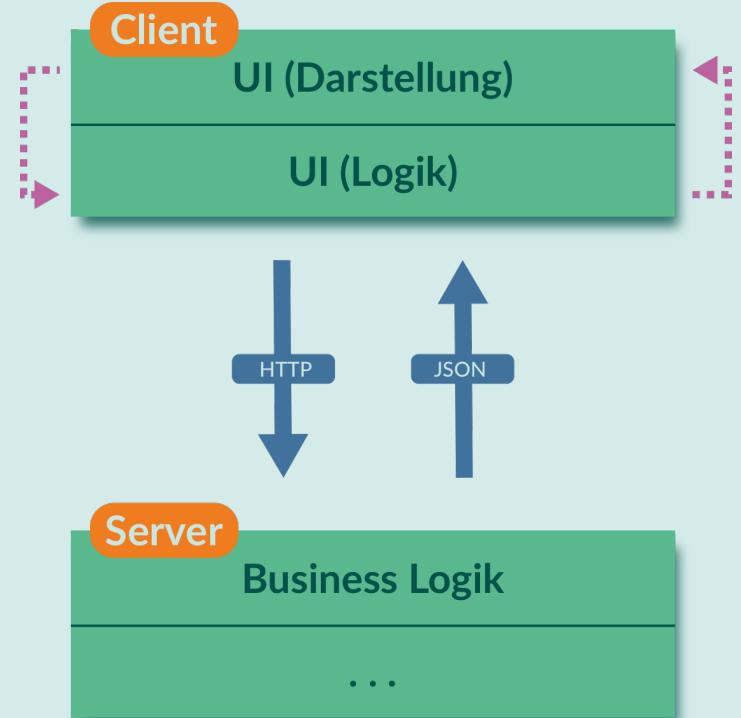
# SINGLE-PAGE-APPLICATION

## 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



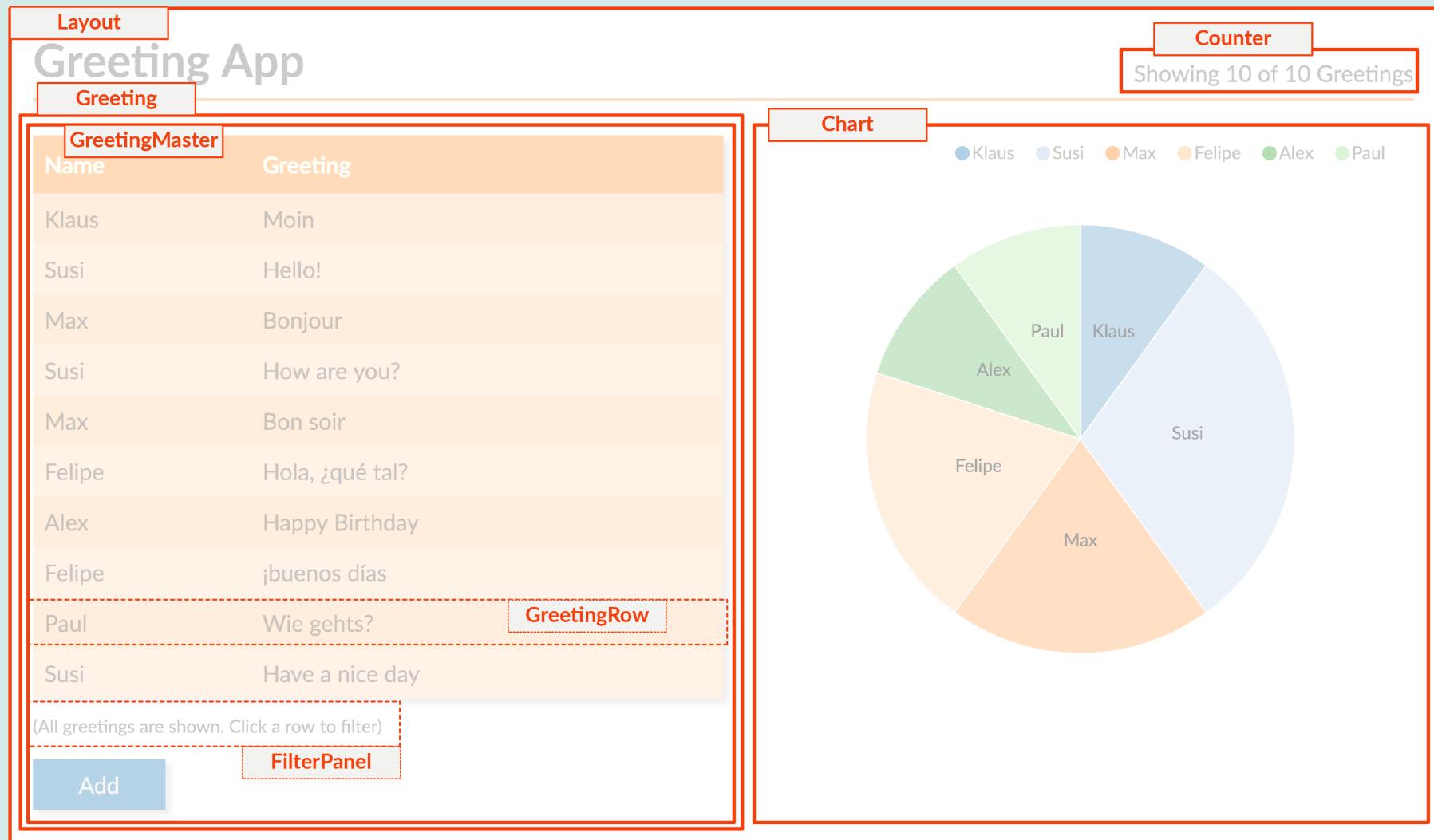
## Technologie

- REST API
- React, Angular, Vue

"A JAVASCRIPT LIBRARY FOR  
BUILDING USER INTERFACES"

React

[HTTPS://REACTJS.ORG/](https://reactjs.org/)



## GREETING APP: KOMPONENTEN

# Komponenten in React

Klassische Aufteilung

**Logik, Model**  
(Java)



**View**  
(HTML, Template)



**Gestaltung**  
(CSS)



Aufteilung in Komponenten



Button

Input

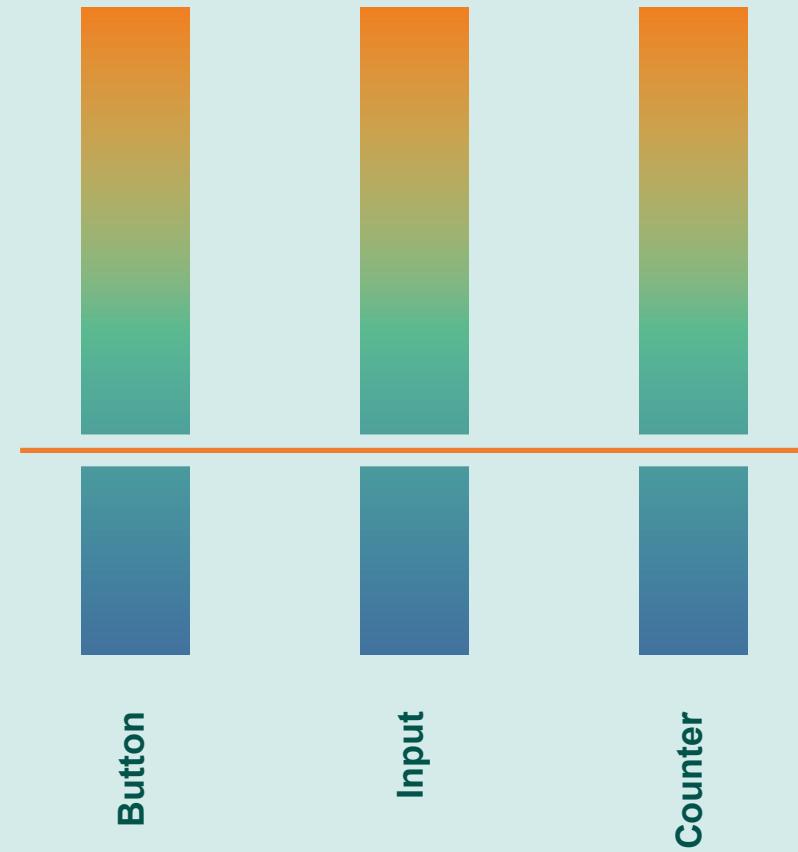
Counter

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

## SEPARATION OF CONCERNES

## 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

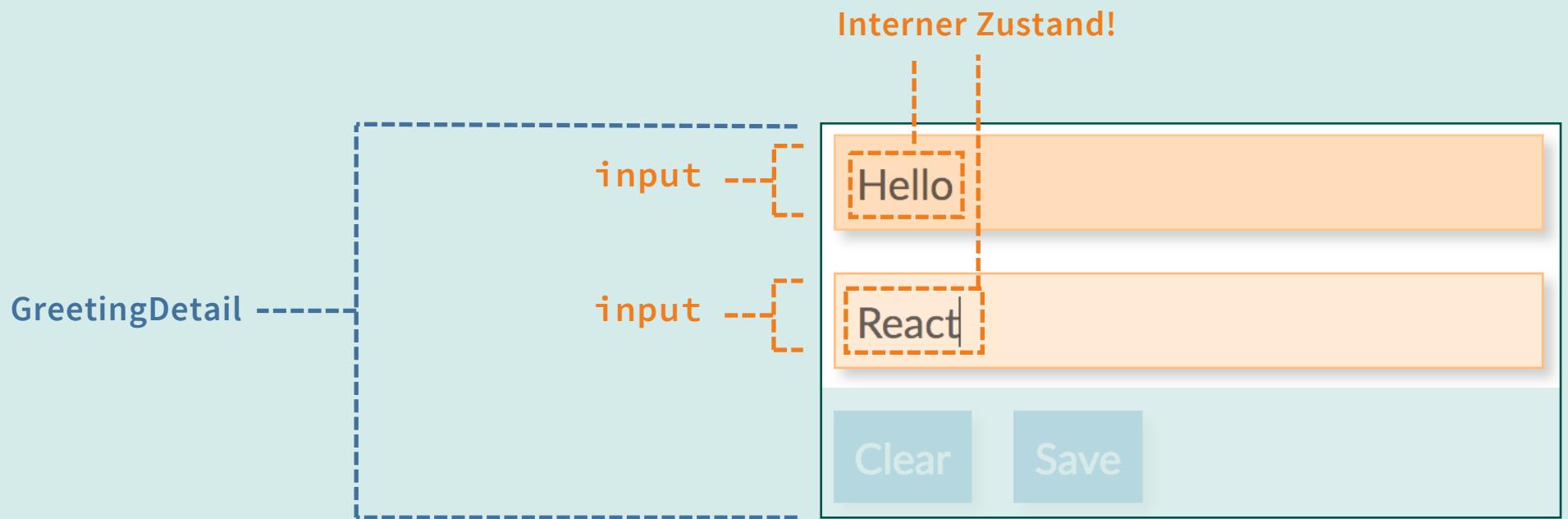
```
import React from 'react';
import ReactDOM from 'react-dom';

import Layout from './Layout';

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

# State

## BEISPIEL: EINGABEFELD



# ZUSTAND: EINGABEFELD

Komponente

Hello

-- input

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

Hello

-- input

1. Input mit Wert aus State befüllen

2a. Event Handler registrieren

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

2b. Event Handler

# ZUSTAND: EINGABEFELD

Komponente

Hello

-- input

1. Input mit Wert aus State befüllen

2a. Event Handler registrieren

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

2b. Event Handler

3. Zustand neu setzen

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

# ZUSTAND: EINGABEFELD

Komponente

Hello

-- input

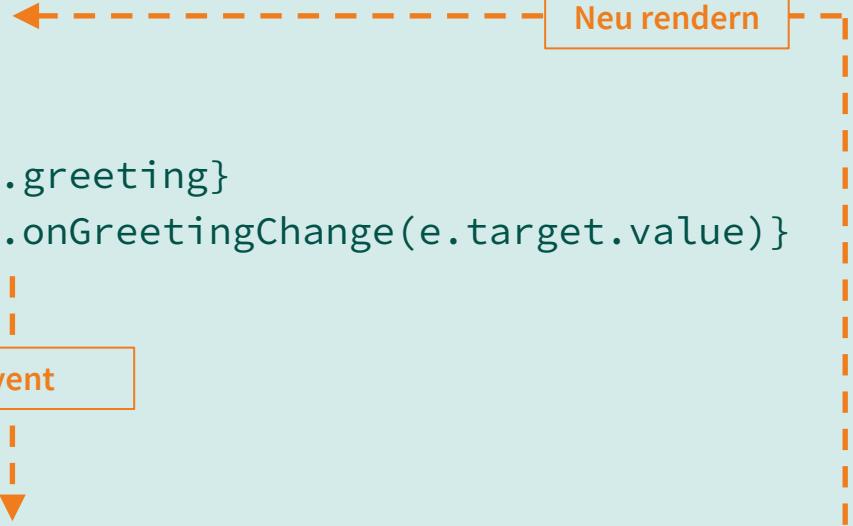
1. Input mit Wert aus State befüllen

2a. Event Handler registrieren

```
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});  
  }  
}
```

2b. Event Handler

3. Zustand neu setzen



# REACT: UNI DIRECTIONAL DATAFLOW

```
class PasswordForm extends React.Component {  
  onPasswordChange(newPassword) { this.setState({password: newPassword}); }  
  ...  
  render() {  
    const password = this.state.password;  
    const checks = this.checkPassword(password);  
    const failedChecks = ...;  
    const isValidPassword = failedChecks === 0;  
  
    return <div>  
      <input type='password'  
             value={password}  
             onChange={event => this.onPasswordChange(event.target.value)} />  
  
      <CheckLabelList checks={checks}>  
        {failedChecks > 0 ?  
          <div className='Label'>{failedChecks} checks failed</div>  
          :  
          <div className='Label Label-success'>All checks passed!</div>  
        }  
  
      <Button label='Set Password' enabled={isValidPassword} />  
    </div>;  
  }  
}
```

The diagram illustrates the unidirectional data flow in React. It features three main components: **Zustand** (green), **Event** (orange), and **Rendern** (red). A blue dashed arrow points from **Zustand** to **Event**, representing the flow of state changes. Another blue dashed arrow points from **Event** to **Rendern**, representing the flow of events. A third blue dashed arrow points from **Rendern** back to **Zustand**, representing the flow of UI updates that trigger state changes. The text at the bottom, **RESPOND TO EVENTS & RENDER UI**, summarizes the process.

**Event**

**Zustand**

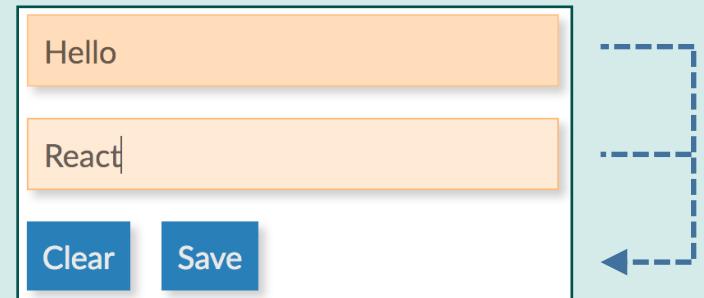
**Rendern**

**RESPOND TO EVENTS & RENDER UI**

# RENDERING VON KOMPONENTEN

## Gerendert wird immer die **ganze** Komponente

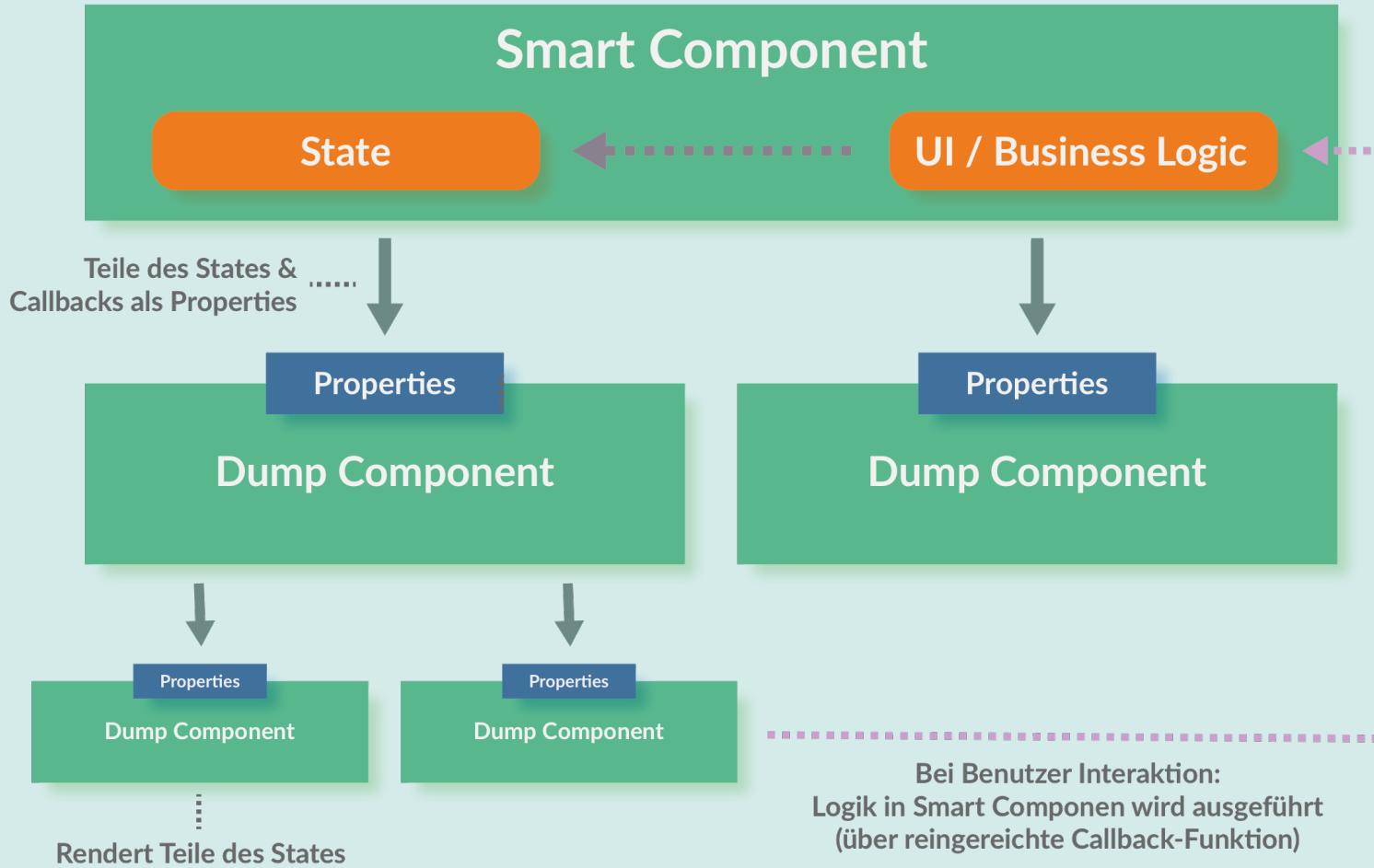
- Inklusive aller Unterkomponenten
- Bei Zustandsänderung
- Vermeidet Inkonsistenzen



```
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"

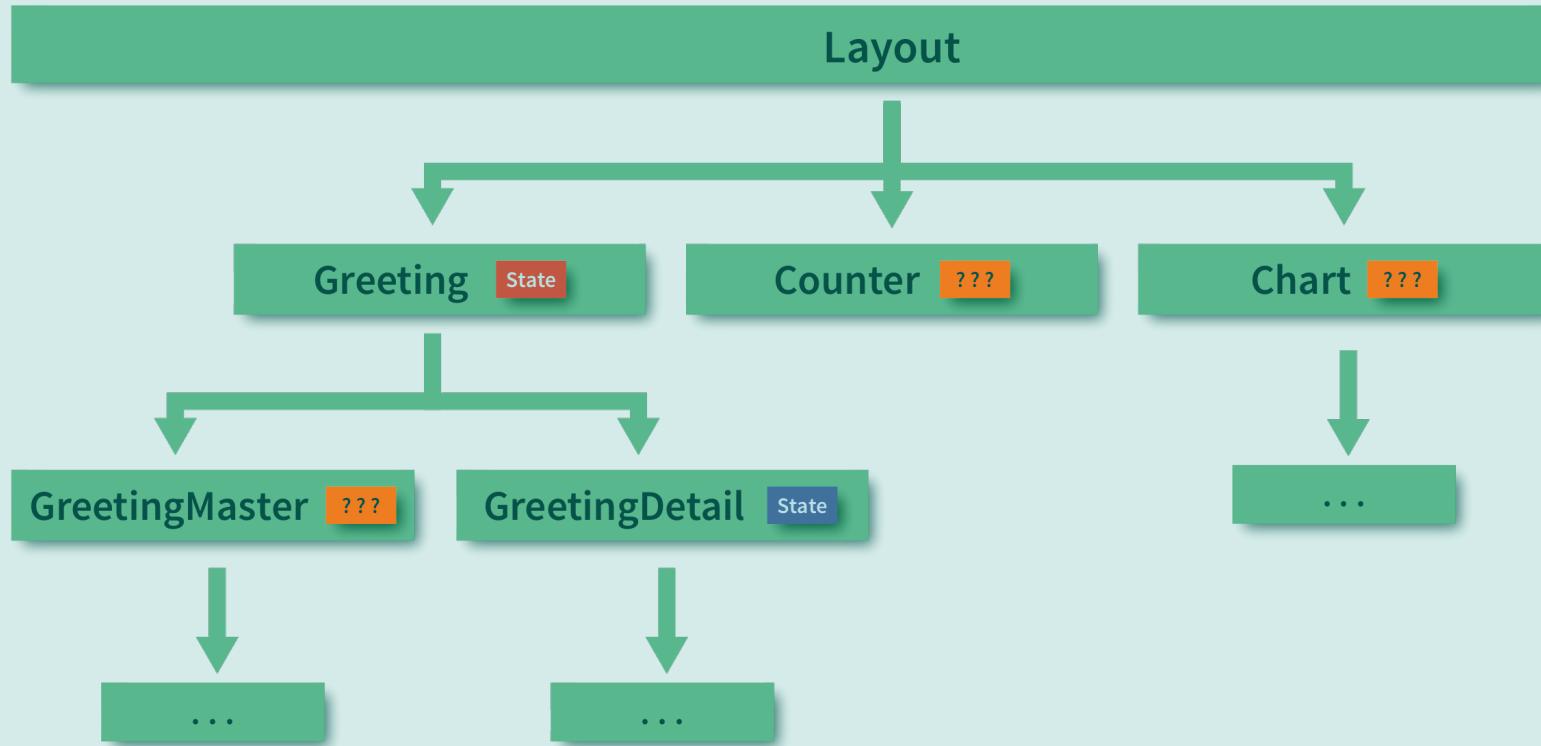


**Typische React Anwendungen:** Hierarchisch aufgebaut

- 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**

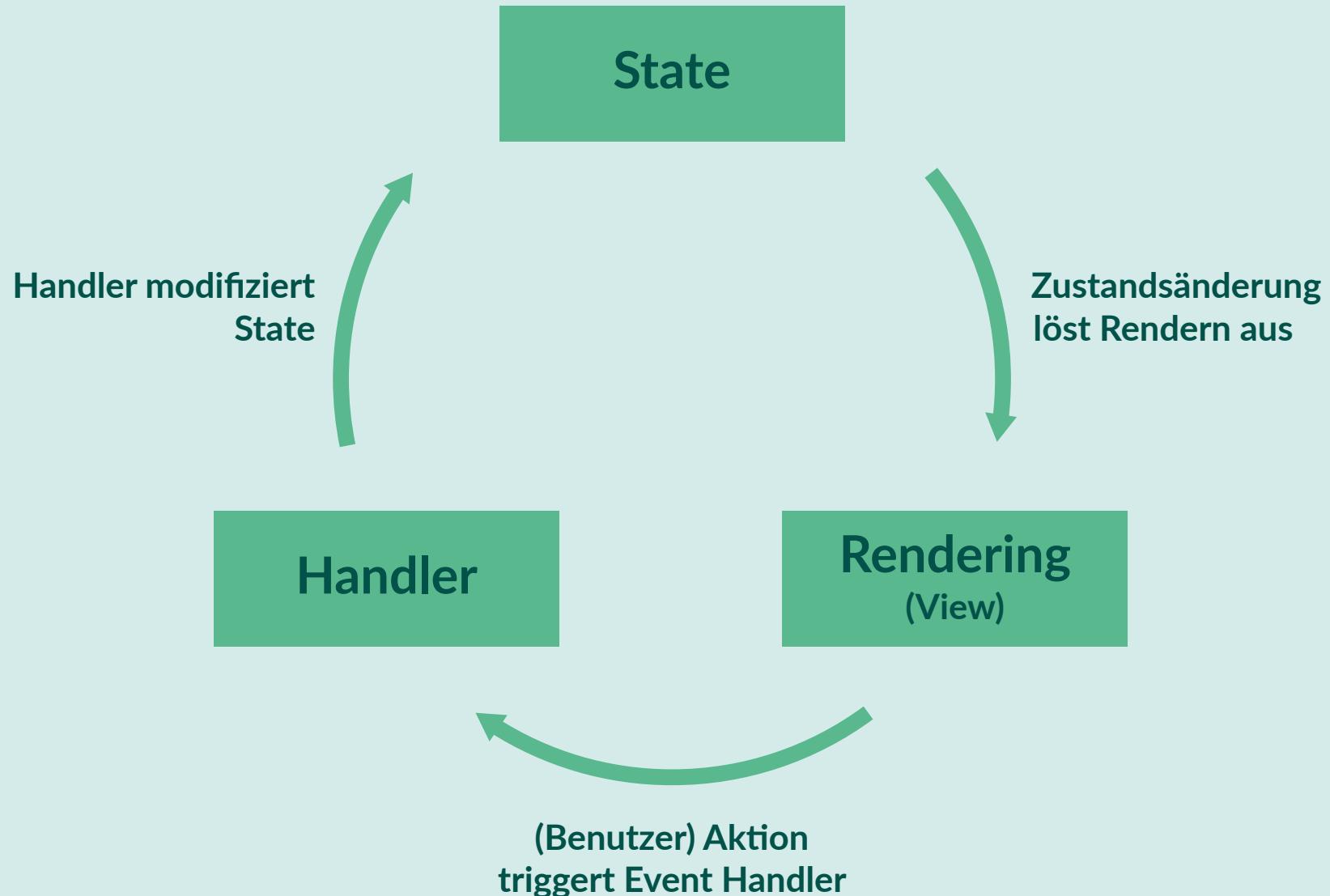
**EXTERNES STATE MANAGEMENT**

# EXTERNES STATE MANAGEMENT

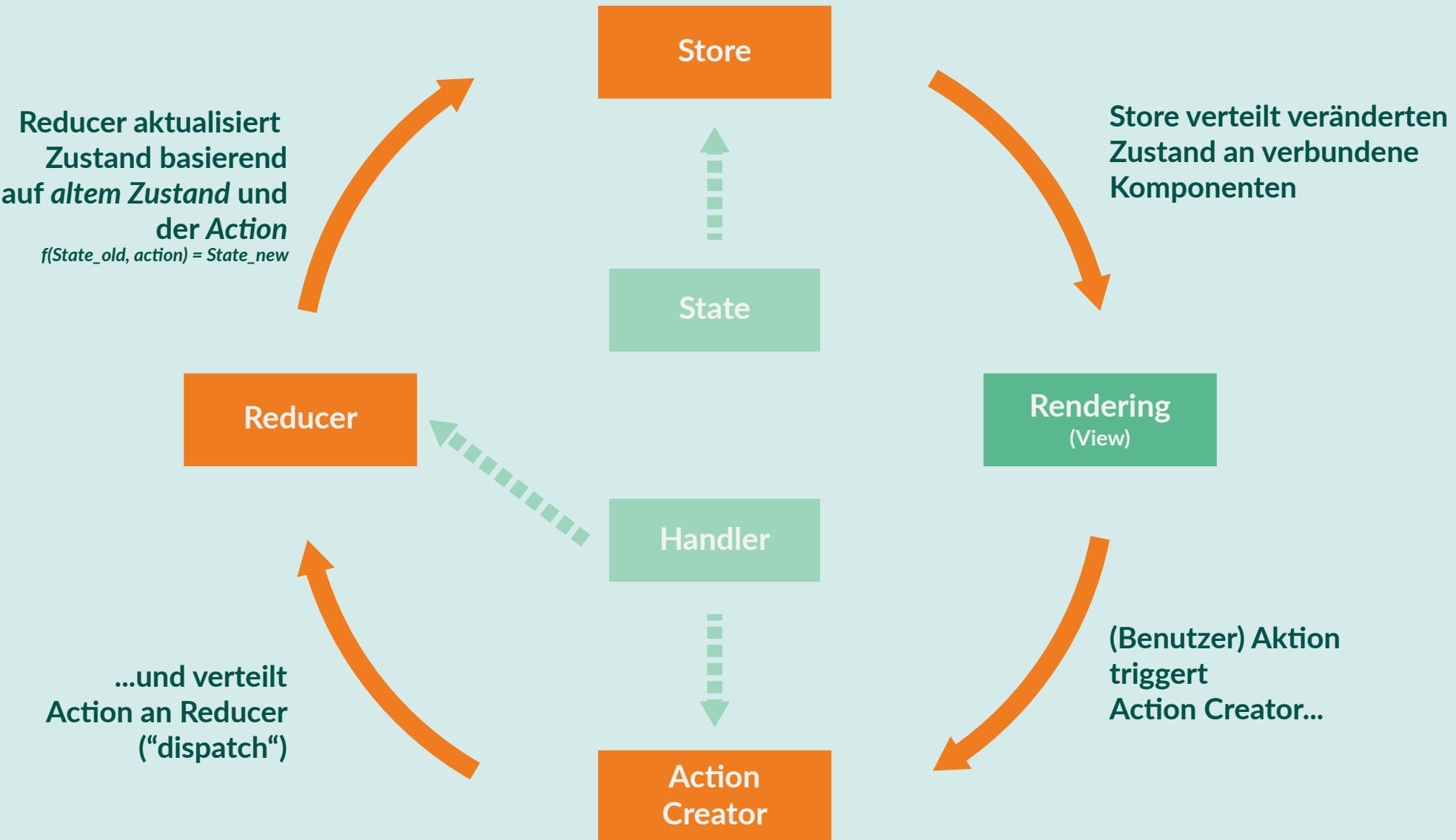
## Beispiel: Redux

- 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

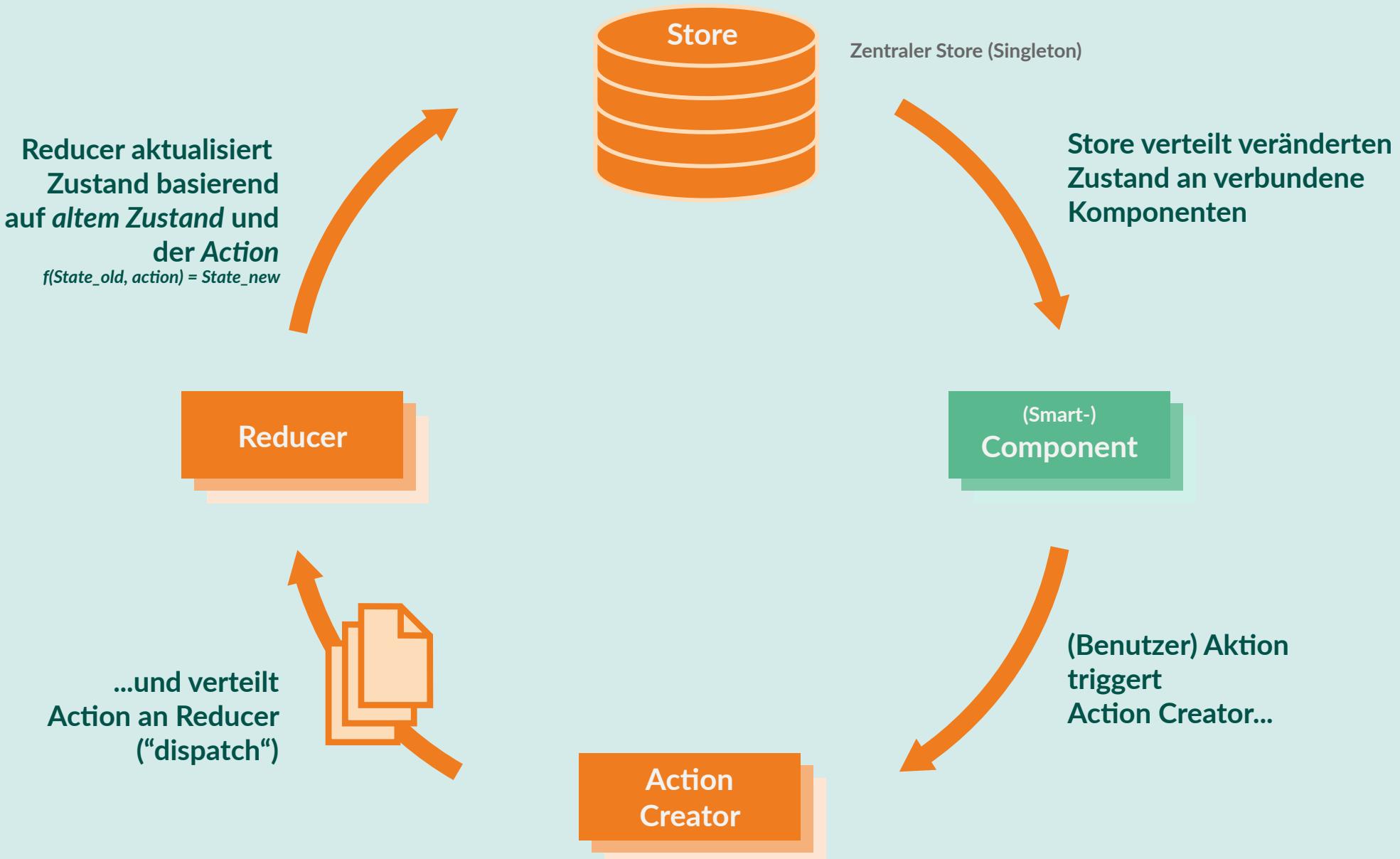
# REACT RENDER ZYKLUS



# 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

The screenshot shows the VS Code interface with the following details:

- File:** GreetingDetail.tsx — 4-test-typescript
- Editor Content:**

```
8
9  interface GreetingDetailState {
10    name: string;
11    greeting: string;
12  }
13
14  export default class GreetingDetail extends React.Component<
15    GreetingDetailProps,
16    GreetingDetailState
17  > {
18    input?: HTMLInputElement | [ts] Type 'Readonly<GreetingDetailState>' has no property
19    'nothere' and no string index signature.
20    render() {
21      const { name, greeting, nothere } = this.state;
22      const saveDisabled = !(name && greeting);
23
24      return (
25        <div>
26          <input
```
- Code Completion Tooltip:** A tooltip appears over the `nothere` variable in line 21, stating: "[ts] Type 'Readonly<GreetingDetailState>' has no property 'nothere' and no string index signature." It also shows a suggestion: "const nothere: any".
- Problems Bar:** Shows one error: "[ts] Type 'Readonly<GreetingDetailState>' has no property 'nothere' and no string index signature. (21, 29)".
- Status Bar:** Shows the current line and column: Ln 21, Col 36.

# Vielen Dank!

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

# Fragen?

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

# 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>
  );
}
```