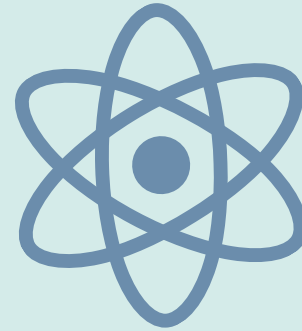


NILS HARTMANN

EINSTIEG IN

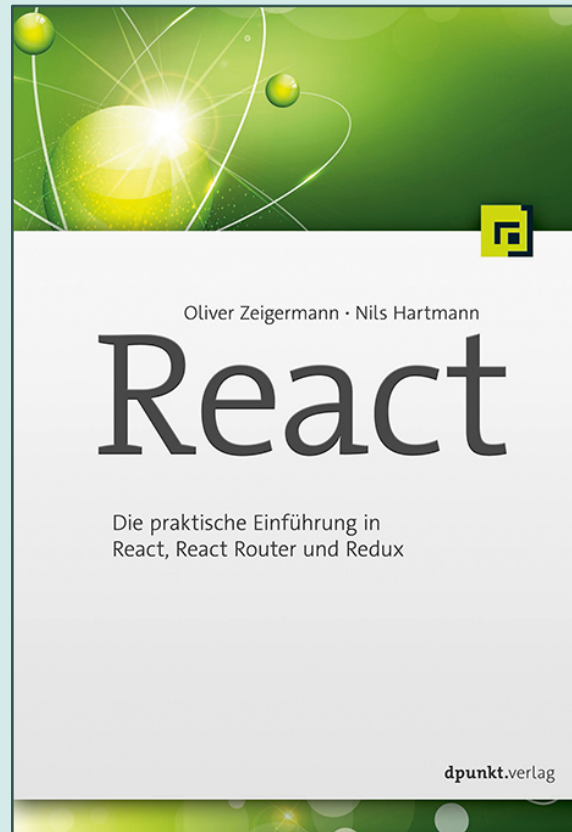


React

<http://bit.ly/react-codetalks>

NILS HARTMANN

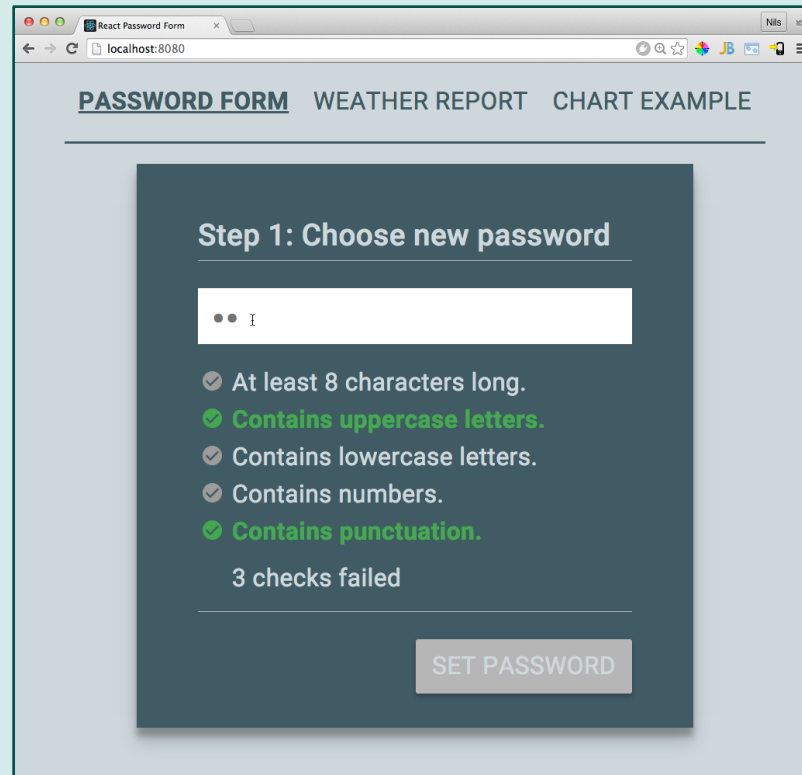
@NILSHARTMANN



[HTTP://REACT-BUCH.DE](http://react-buch.de) | [HTTP://REACT-WORKSHOP.DE](http://react-workshop.de)

SINGLE PAGE APPLICATIONS

React



Code: <https://github.com/nilshartmann/react-example-app>

Demo: <https://nilshartmann.github.io/react-example-app/>

BEISPIEL ANWENDUNG

Step 1: Choose new password

R I

- ✓ At least 8 characters long.
- ✓ **Contains uppercase letters.**
- ✓ Contains lowercase letters.
- ✓ Contains numbers.
- ✓ Contains punctuation.

4 checks failed

SET PASSWORD

```
<PasswordView>
  <PasswordForm>
    <input />
    <CheckLabelList>
      <CheckLabel />
      <CheckLabel />
    </CheckLabelList>
    <Label />
    <Button />
  </PasswordForm>
</PasswordView>
```

PASSWORD FORM WEATHER REPORT CHART EXAMPLE

Step 1: Choose new password

R I

- ✓ At least 8 characters long.
- ✓ Contains uppercase letters.
- ✓ Contains lowercase letters.
- ✓ Contains numbers.
- ✓ Contains punctuation.

4 checks failed

SET PASSWORD

```
<Application>
  <Navigation />
  <ViewController>
    <PasswordView>
      . . .
      . . .
    </PasswordView>
  </ViewController>
</Application>
```

ANWENDUNGEN AUS KOMPONENTEN KOMPONIERT

React-Komponenten

- werden deklarativ beschrieben
 - bestehen aus Logik und UI
 - keine Templatesprache
 - werden immer komplett gerendert
-
- können auf dem Server gerendert werden („universal webapps“)

✓ At least 8 characters long.

✓ At least 8 characters long.

✓ Contains uppercase letters.

REACT! I

✓ At least 8 characters long.

✓ Contains uppercase letters.

REACT SCHRITT FÜR SCHRITT

DIE JSX SPRACHERWEITERUNG

Anstatt einer Template Sprache: HTML in JavaScript integrieren

- Erlaubt Schreiben von HTML-artigen Ausdrücken im JavaScript-Code
- Wird zu regulärem JavaScript Code kompiliert (z.B. Babel, TypeScript)
- Optional

JSX

```
const name = 'Lemmy';  
const greeting = <h1>Hello, {name}</h1>;
```

Übersetztes JavaScript

```
var name = 'Lemmy';  
var greeting = React.createElement('h1', null, 'Hello, ', name);
```

EINE REACT KOMPONENTE: ALS FUNKTION

Komponente CheckLabel

✓ At least 8 characters long.

Komponentenfunktion

```
function CheckLabel() {                                     JSX
  return <div
    className="CheckLabel-unchecked">
    At least 8 characters long.
  </div>;
}
```

KOMPONENTE EINBINDEN

✓ At least 8 characters long.

index.html

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

KOMPONENTE EINBINDEN

✓ At least 8 characters long.

app.js

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

import CheckLabel from './CheckLabel';

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

KOMPONENTEN: PROPERTIES

✓ At least 8 characters long.

```
{  
  checked: false,  
  label: 'At least 8 characters long.'  
}
```



```
function CheckLabel(props) {  
  return <div  
    className=  
      {props.checked? 'CheckLabel-checked' : 'CheckLabel-unchecked'}>  
    {props.label}  
  </div>;  
}
```

KOMPONENTEN: PROPERTIES

✓ At least 8 characters long.

```
function CheckLabel(props) {  
  . . .  
}
```

Properties beschreiben

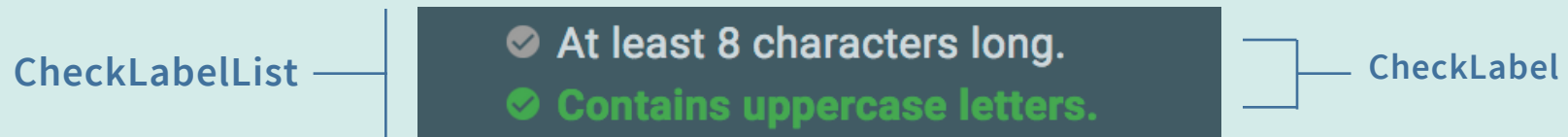
```
CheckLabel.propTypes = {  
  label:    React.PropTypes.string.isRequired,  
  checked:  React.PropTypes.bool  
};
```

Überprüfung zur Laufzeit

✖ Warning: Failed propTypes: Required prop `label` was not specified in `CheckLabel`. Check the render method of `CheckLabelList`. [main.js:12889](#)

KOMPONENTEN VERWENDEN

- Komponenten sind **zusammensetzbar**



```
function CheckLabelList() {  
  return <div>  
    <CheckLabel checked={false}  
      label='At least 8 characters long' />  
    <CheckLabel checked={true}  
      label='Contains uppercase letters.' />  
  </div>;  
}
```

```
function CheckLabel(props) {  
  // . . .  
}
```


BEISPIEL: KOMPONENTENLISTEN

✓ At least 8 characters long.

✓ Contains uppercase letters.

```
checks: [  
  { checked: false, label: 'At least 8 characters long.' },  
  { checked: true,  label: 'Contains uppercase letters' }  
]
```



```
function CheckLabelList(props) {  
  return <div>  
    {props.checks.map(c => <CheckLabel  
      label={c.label}  
      checked={c.checked}  
      key={c.label} />)  
    }  
  </div>;  
}
```

KOMPONENTEN KLASSEN

ECMAScript 2015 Klasse

Properties über Konstruktor
(optional)

Lifecycle Methoden
(optional)

Render-Methode (pflicht)

Properties über **props** Objekt

```
class CheckLabelList extends React.Component {  
  constructor(props) {  
    super(props);  
  }  
  
  componentDidMount() { . . . }  
  componentWillReceiveProps() { . . . }  
  shouldComponentUpdate() { . . . }  
  
  render() {  
    return <div>  
      {this.props.checks.map(c => <CheckLabel . . . />)}  
    </div>;  
  }  
}
```

ZUSTAND VON KOMPONENTEN

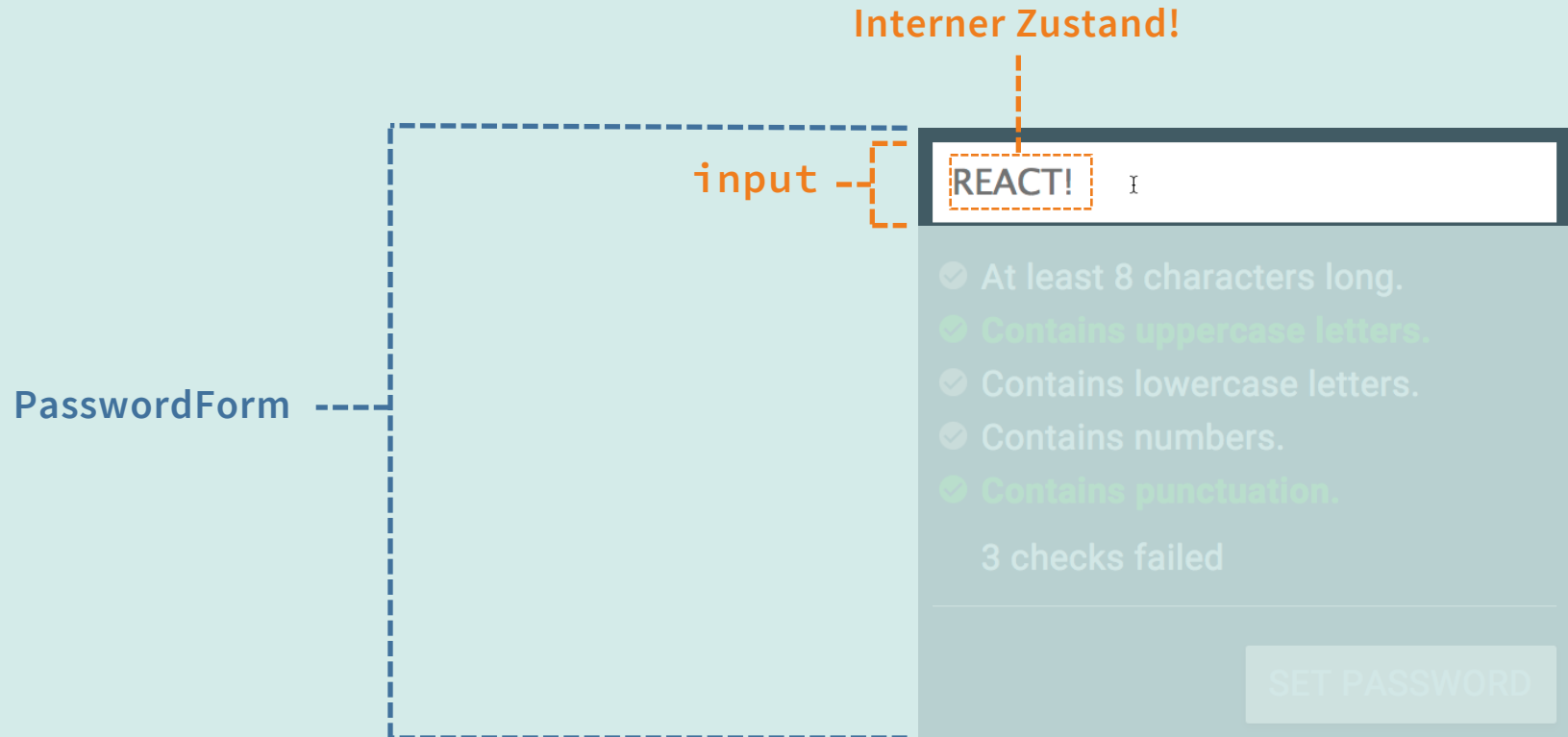
Zustand („state“): Komponenten-intern

- Beispiel: Inhalt von Eingabefeld, Antwort vom Server
- Objekt mit Key-Value-Paaren
- Zugriff über `this.state` / `this.setState()`
- Nur in Komponenten-Klassen verfügbar
- **`this.setState()` triggert erneutes Rendern**
 - auch alle Unterkomponenten

Zum Vergleich: Properties

- Von außen übergeben
- Unveränderlich
- Zugriff über `this.props` (Key-Value-Paare)

BEISPIEL: EINGABEFELD



BEISPIEL: EINGABEFELD



```
class PasswordForm extends React.Component {  
  render() {  
    return <div>  
      <input  
        value={this.state.password}  
        onChange={e=>this.onPasswordChange(e.target.value)}  
      />  
      . . .  
    </div>;  
  }  
  
  onPasswordChange(newPassword) {  
    this.setState({password: newPassword});  
  }  
}
```

1. Input mit Wert aus State befüllen

2. Event Listener

ZUSTAND: EINGABEFELD



```
class PasswordForm extends React.Component {  
  render() {  
    return <div>  
      <input  
        value={this.state.password}  
        onChange={e=>this.onPasswordChange(e.target.value)}  
      />  
      . . .  
    </div>;  
  }  
  
  onPasswordChange(newPassword) {  
    this.setState({password: newPassword});  
  }  
}
```

1. Input mit Wert aus State befüllen
2. Event Listener

3. Zustand neu setzen

Event

Neu rendern

ZUSTAND & RENDERING

Beispiel: Password Formular

The diagram illustrates a password form with a text input field containing "REACT!". Below the input field is a list of validation rules, each preceded by a checkmark icon. The first rule, "At least 8 characters long.", is marked with a grey checkmark. The second rule, "Contains uppercase letters.", is marked with a green checkmark. The third rule, "Contains lowercase letters.", is marked with a grey checkmark. The fourth rule, "Contains numbers.", is marked with a grey checkmark. The fifth rule, "Contains punctuation.", is marked with a green checkmark. Below the list of rules, the text "3 checks failed" is displayed. At the bottom right of the form is a button labeled "SET PASSWORD". To the right of the form, a vertical line labeled "beeinflusst" (influences) has arrows pointing to each of the five validation rules and the "SET PASSWORD" button, indicating that the state of the form (e.g., the text in the input field) influences these elements.

REACT! |

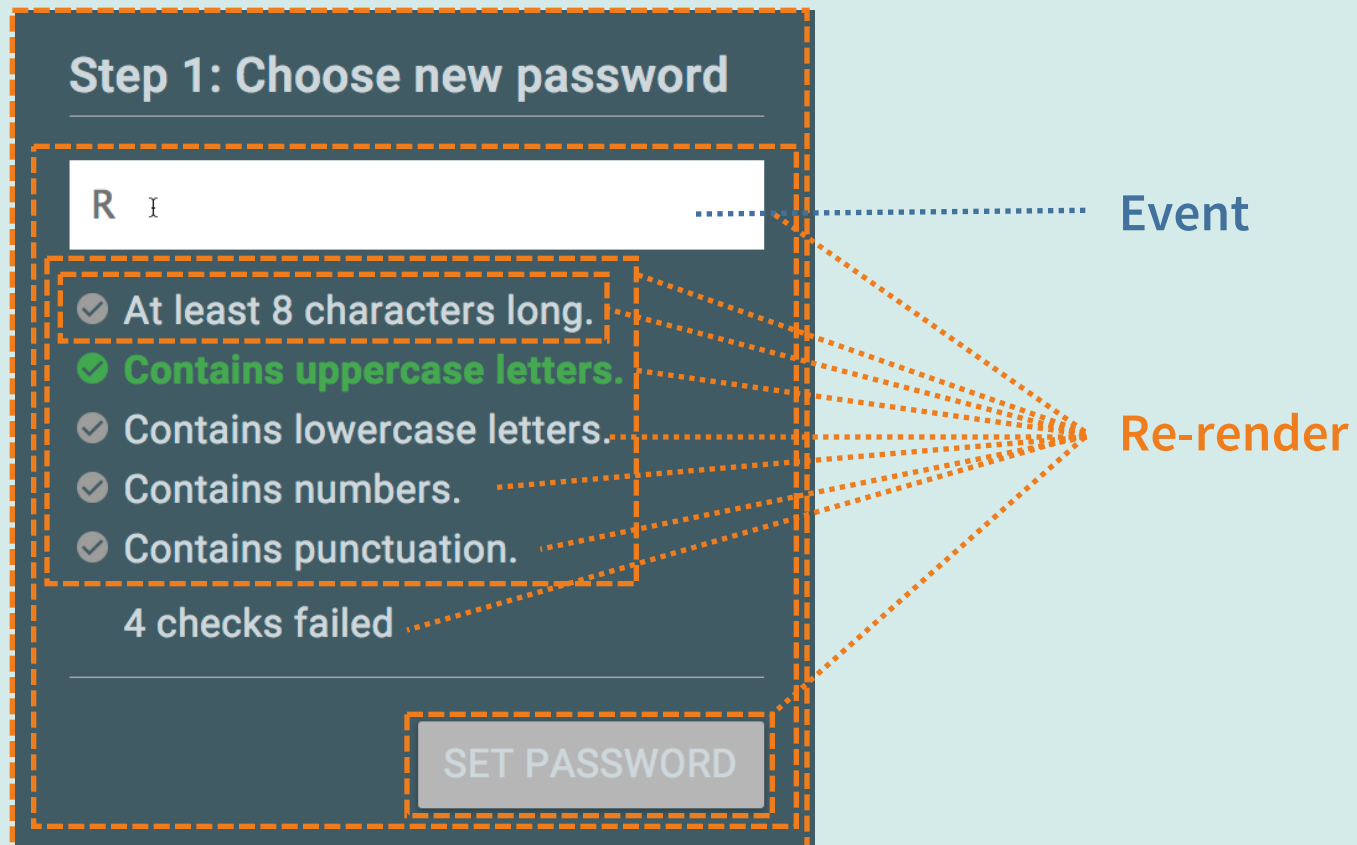
- ✓ At least 8 characters long.
- ✓ Contains uppercase letters.
- ✓ Contains lowercase letters.
- ✓ Contains numbers.
- ✓ Contains punctuation.

3 checks failed

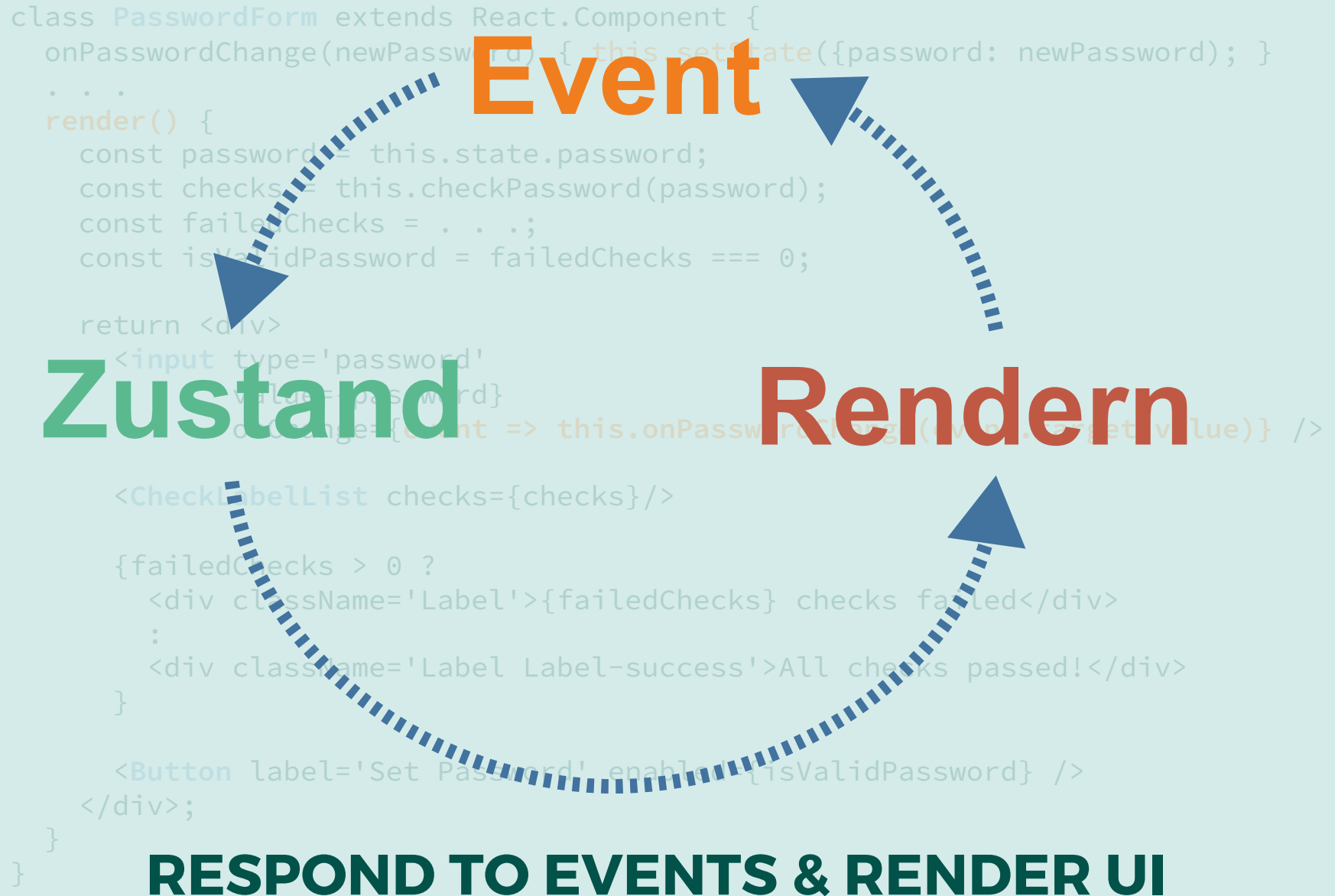
SET PASSWORD

beeinflusst

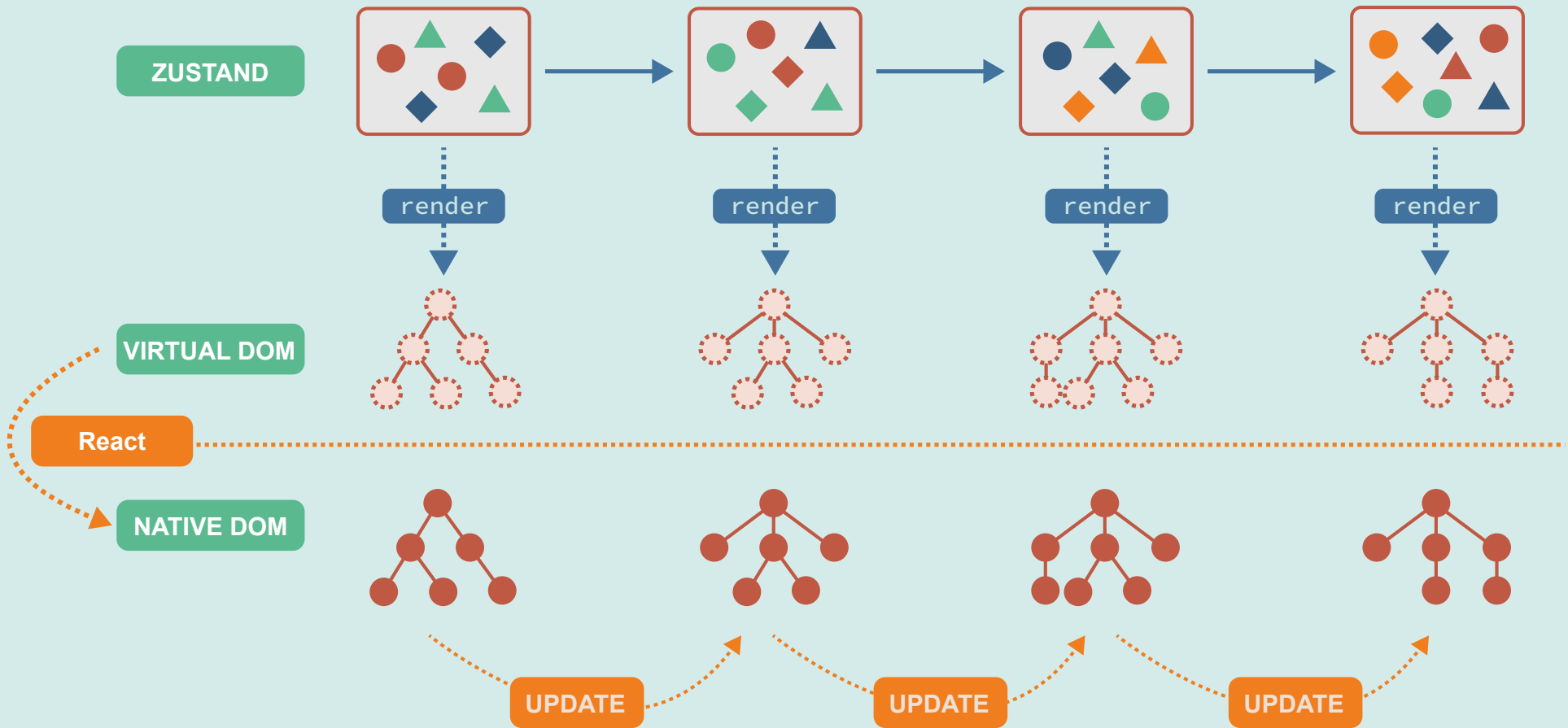
GANZ EINFACH: ALLES RENDERN



REACT: UNI DIRECTIONAL DATAFLOW



HINTERGRUND: VIRTUAL DOM



HINTERGRUND: VIRTUAL DOM

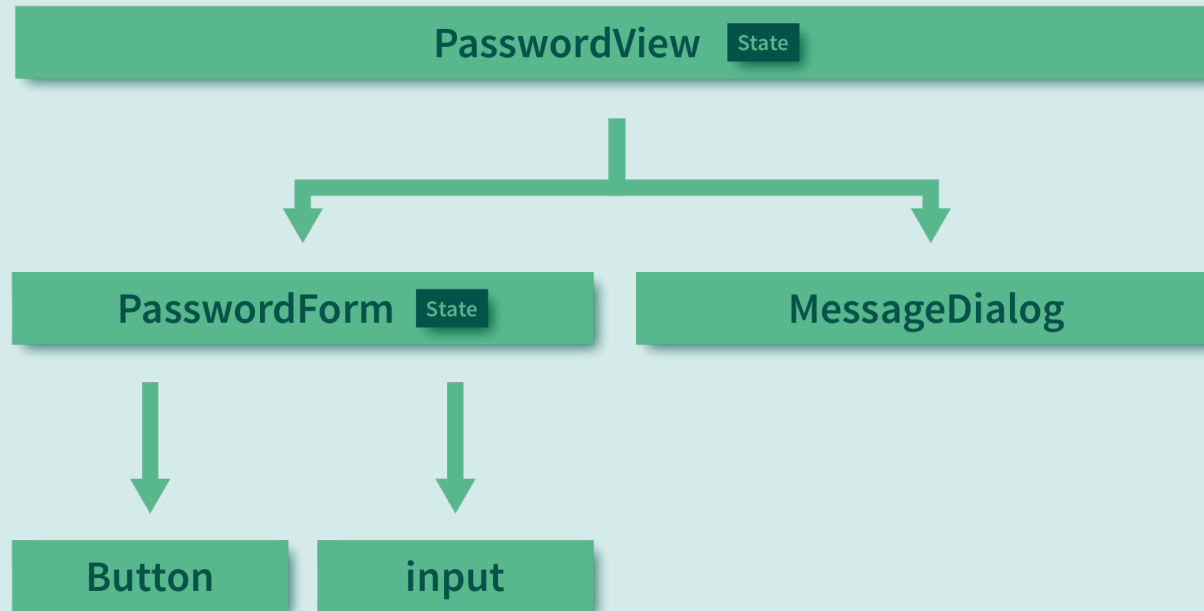
Virtual DOM

- Render-Methode liefert ein **virtuelles** DOM-Objekt zurück
- Trennung von Darstellung (DOM) und Repräsentation (virtueller DOM)

Vorteile

- Erlaubt performantes neu rendern der Komponente
- Ausgabe in andere Formate (z.B. String) möglich
- Kann auf dem Server gerendert werden (Universal Webapps)
- Kann ohne DOM/Browser getestet werden

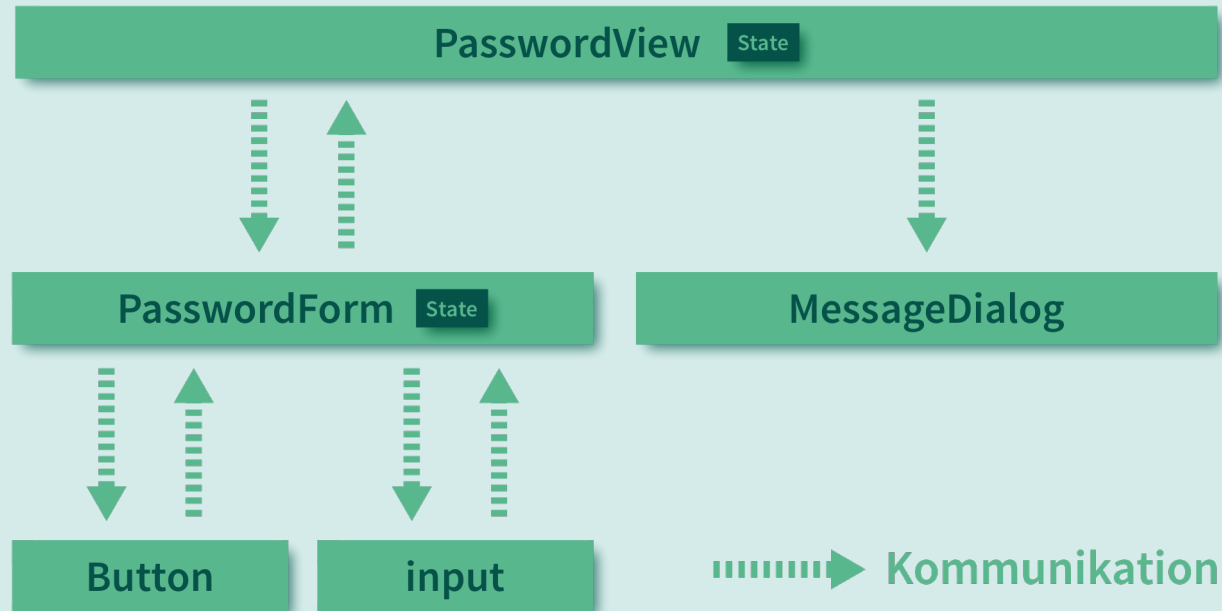
KOMPONENTENHIERARCHIEN



Typische React Anwendungen: Hierarchisch aufgebaut

- State möglichst weit oben („Container Komponenten“)
- Mehrere Komponenten mit State möglich
 - Beim neu rendern bleibt State erhalten

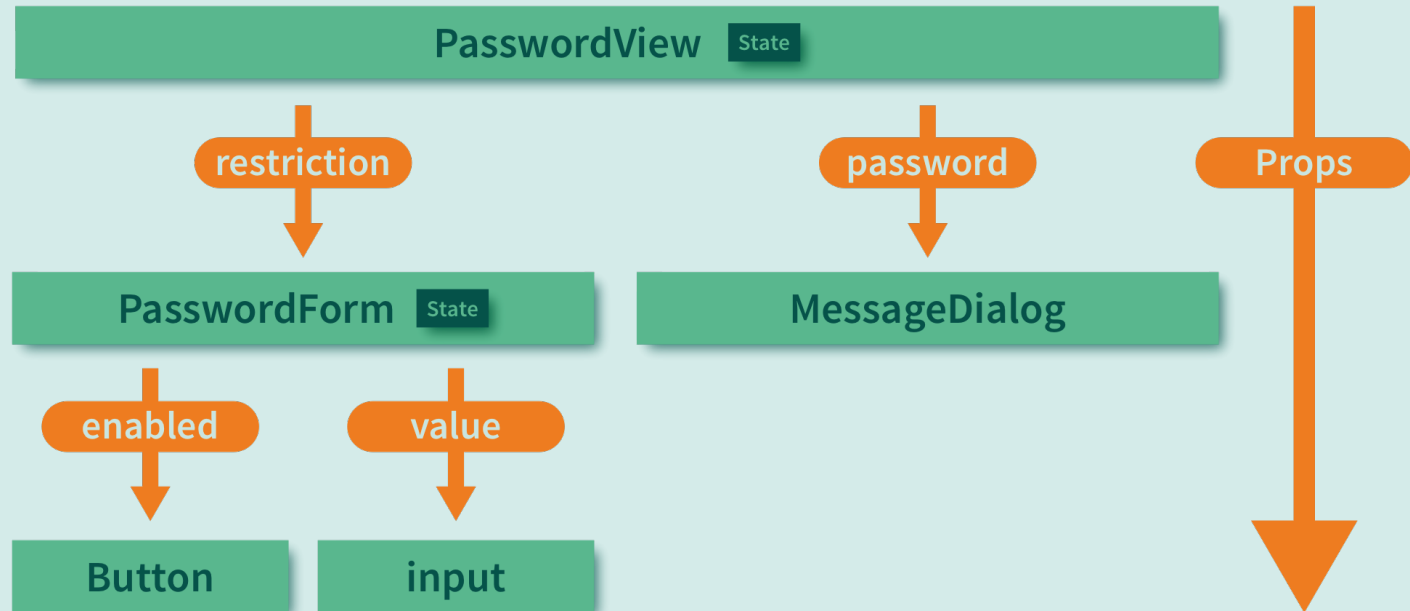
KOMMUNIKATION ZWISCHEN KOMPONENTEN



Typische React Anwendungen: Hierarchisch aufgebaut

- State möglichst weit oben („Container Komponenten“)
- Mehrere Komponenten mit State möglich
 - Beim neu rendern bleibt State erhalten
- Wie wird kommuniziert?

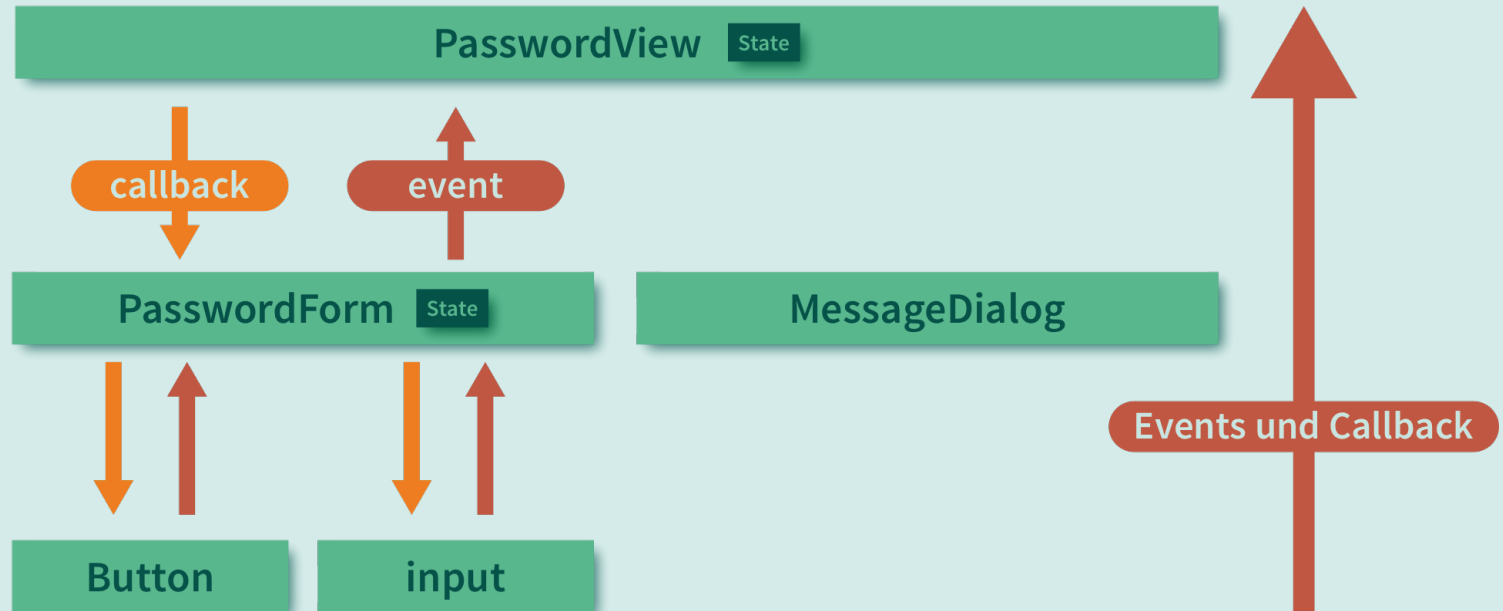
KOMMUNIKATION: PROPERTIES



Von oben nach unten: **Properties**

```
<Button enabled={...}>Set Password</Button>
```

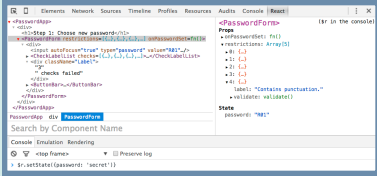
KOMMUNIKATION: EVENTS



Von unten nach oben: **Events und Callbacks**

- Callback-Funktion als **Property**
- **Event**: Aufruf der Callback-Funktion

ÖKOSYSTEM



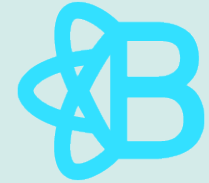
Developer Tools

material-design



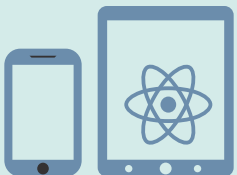
Flux Architekturpattern

Bootstrap



GraphQL & Relay

React Router



React Native

Fertige Komponenten



React

- Nur View-Schicht (Komponenten)
 - Gut integrierbar mit anderen Frameworks
 - Einfache Migrationspfade möglich
- JSX statt Templatesprache („HTML in JavaScript“)
- Deklarative UI
 - Komponenten werden immer komplett gerendert
 - Kein 2-Wege-Databinding
 - Komponenten typischerweise organisiert in Hierarchien

Vielen Dank!

<http://bit.ly/react-codetalks>

Fragen?

@NILSHARTMANN