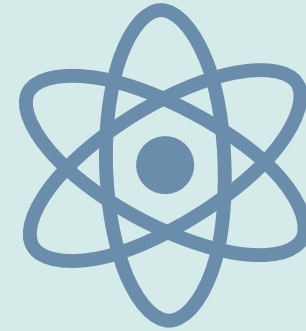


EINSTIEG IN

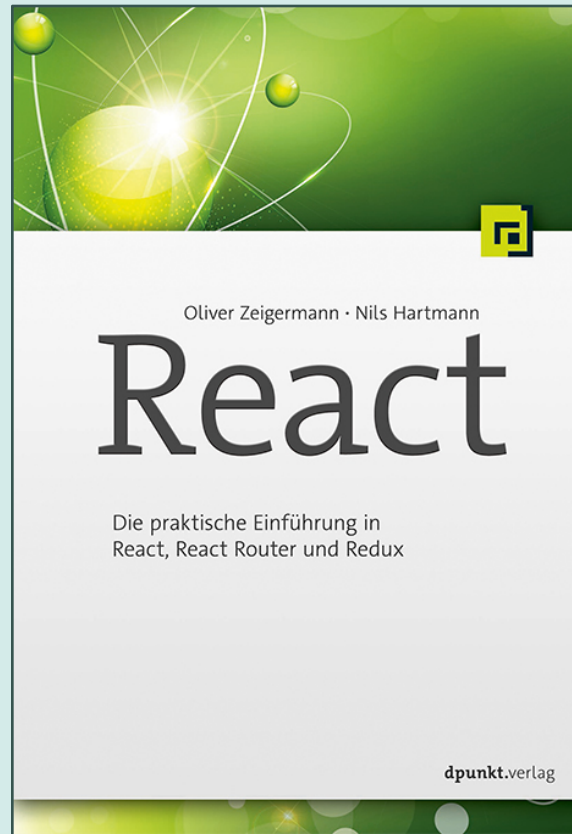


React

<http://nilshartmann.net/react-talk>

NILS@NILSHARTMANN.NET

[HTTP://NILSHARTMANN.NET/REACT-TALK](http://nilshartmann.net/react-talk)



[HTTP://REACTBUCH.DE](http://reactbuch.de)

A JAVASCRIPT LIBRARY FOR BUILDING
USER INTERFACES

React

SINGLE PAGE APPLICATIONS

React

OPEN SOURCE VON FACEBOOK

<https://facebook.github.io/react>

React

0.3

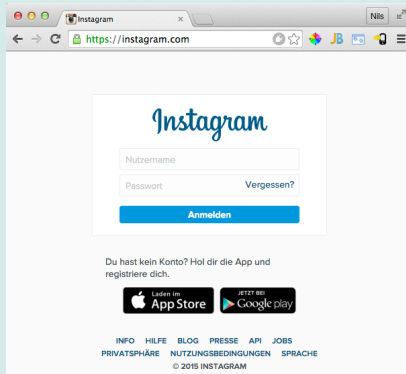
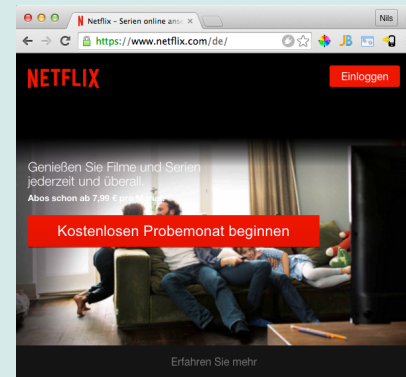
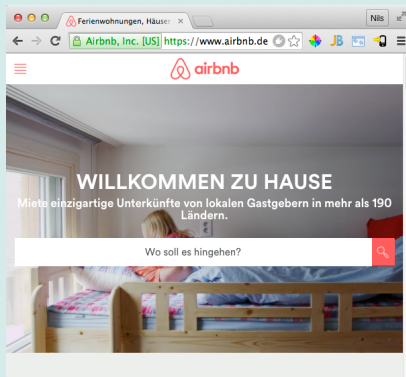
05 | 2013 – OPEN SOURCE

0.14.8

v 15.0

05 | 2016 – NEUE VERSIONIERUNG

AKTUELLE VERSION



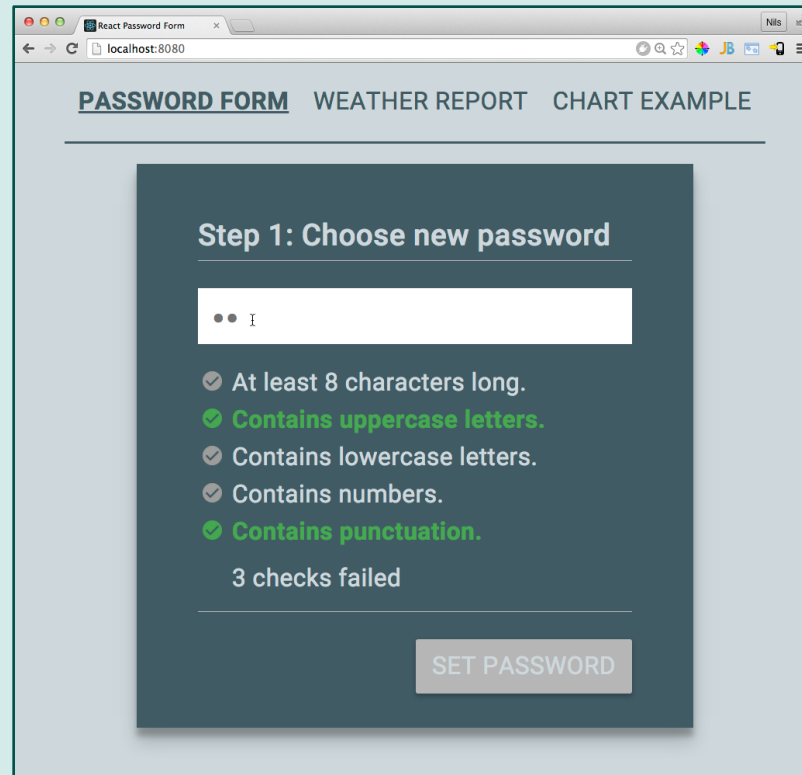
REACT IM EINSATZ

v in MVC

NUR VIEW-SCHICHT

ES6+

ECMAScript 2015



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>
```

WIEDERVERWENDBARE KOMPONENTEN

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>
```

React-Komponenten

- können eigenen, internen Zustand enthalten
- bestehen aus Logik und UI
- werden immer komplett gerendert
- Keine Templatesprache

✓ At least 8 characters long.

✓ At least 8 characters long.

✓ Contains uppercase letters.

REACT! I

✓ At least 8 characters long.

✓ Contains uppercase letters.

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 compiliert (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: AUSGANGSSITUATION

✓ At least 8 characters long.

HTML

```
<div  
  class="CheckLabel-unchecked">  
  At least 8 characters long.  
</div>
```

EINE REACT KOMPONENTE: JSX

✓ At least 8 characters long.

JSX

```
<div  
  className="CheckLabel-unchecked">  
  At least 8 characters long.  
</div>
```

EINE REACT KOMPONENTE

✓ At least 8 characters long.

Übersetzter JS Code
(z.B. mittels Babel)

```
React.createElement(  
  "div",  
  { className: "CheckLabel-unchecked" },  
  "At least 8 characters long."  
);
```

EINE REACT KOMPONENTE: ALS FUNKTION

Komponente CheckLabel

✓ At least 8 characters long.

Komponentenfunktion

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

KOMPONENTEN: PROPERTIES

✓ At least 8 characters long.

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

Properties beschreiben

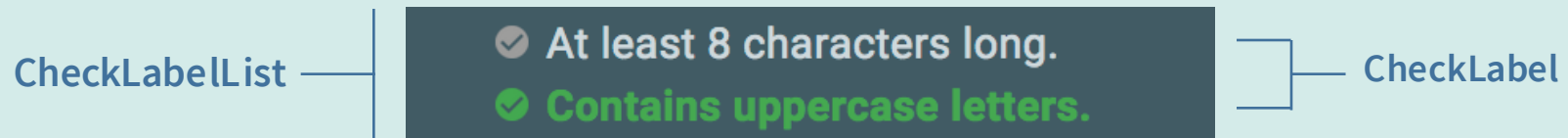
```
CheckLabel.propTypes = {  
  label: React.PropTypes.string.isRequired,  
  checked: React.PropTypes.bool  
};
```

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) {  
  // . . .  
}
```

KOMPONENTEN LISTEN

✓ 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

Lifecycle Methoden

Render-Methode (pflicht)

Properties über **props** Objekt

Property-Beschreibungen

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

ZUSTAND VON KOMPONENTEN

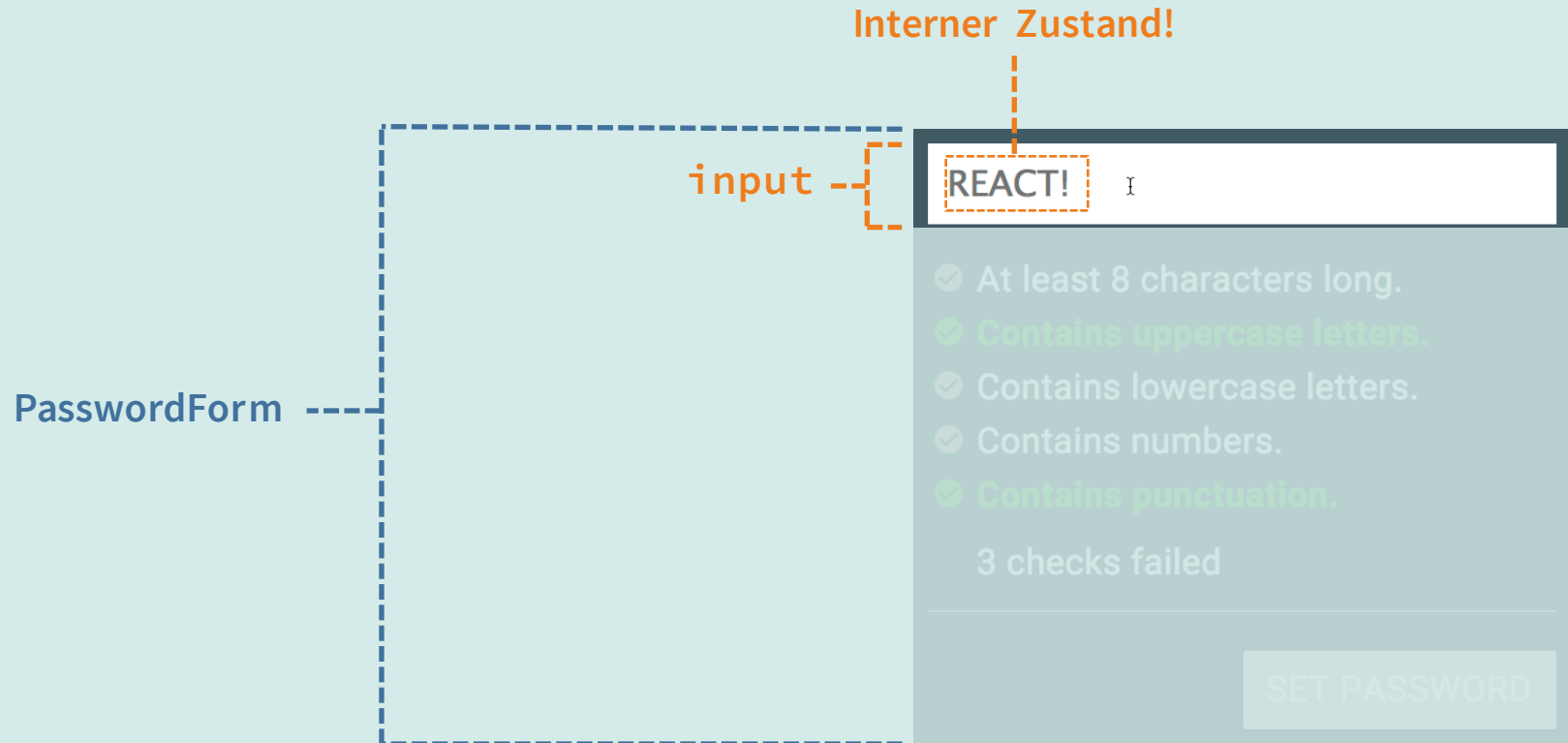
Zustand („state“): Komponenten-intern

- Beispiel: Inhalt von Eingabefeld, Antwort vom Server
- Objekt mit **Key-Value-Paaren**
- Werte üblicherweise **immutable**
- 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

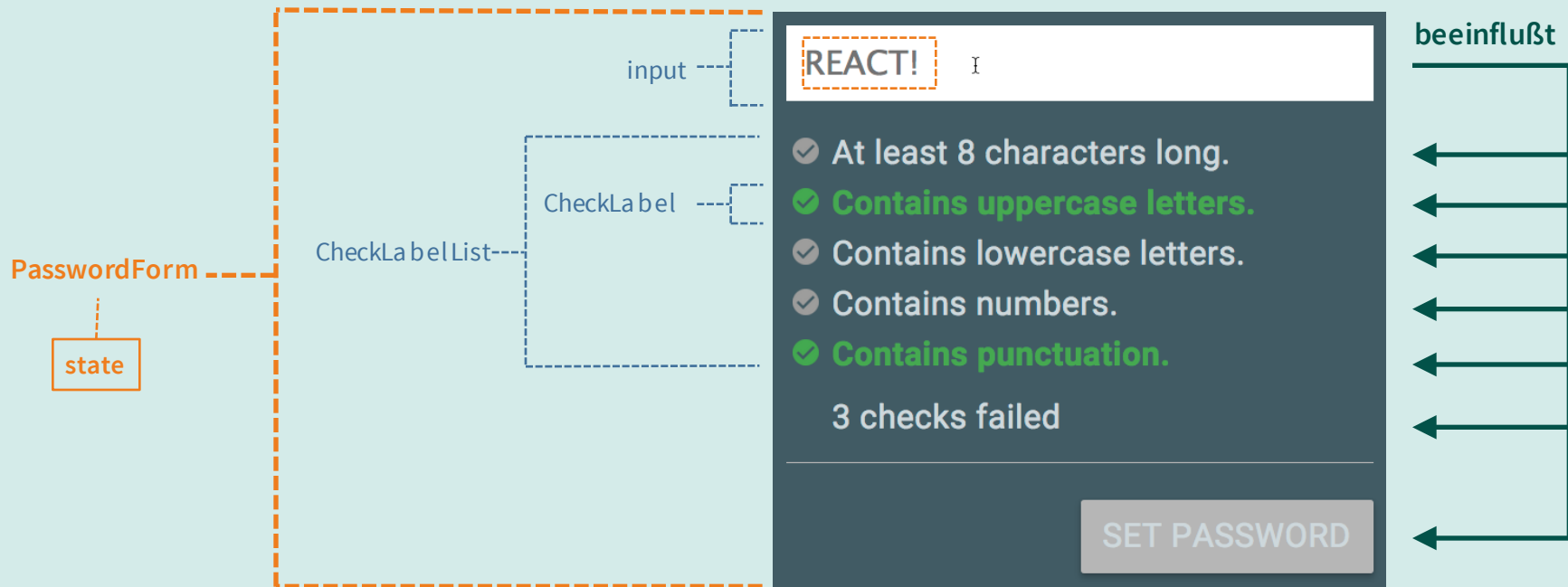
3. Zustand neu setzen

Event

Neu rendern

ZUSTAND & RENDERING

Beispiel: Password Formular

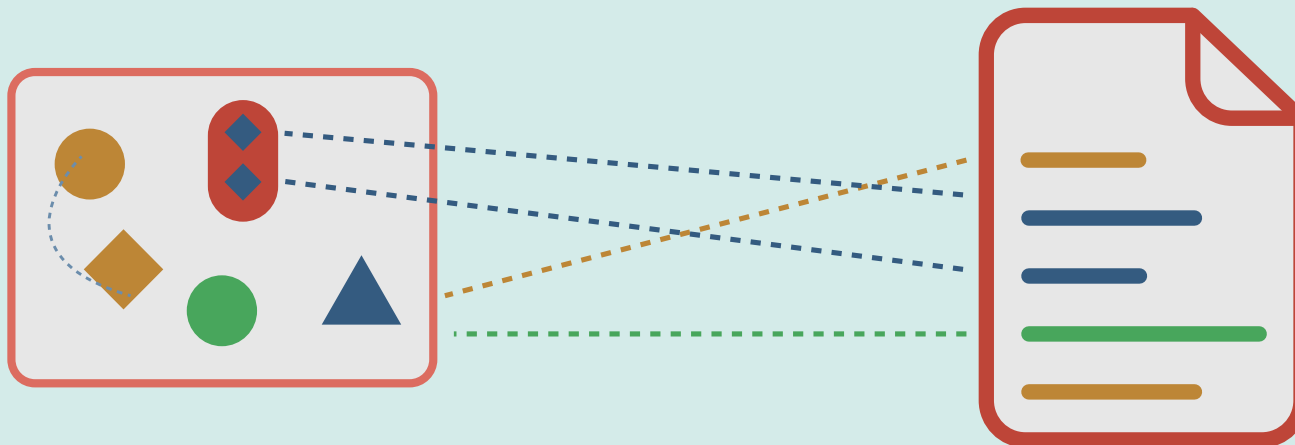


„KLASSISCHE“ OBSERVER LÖSUNG

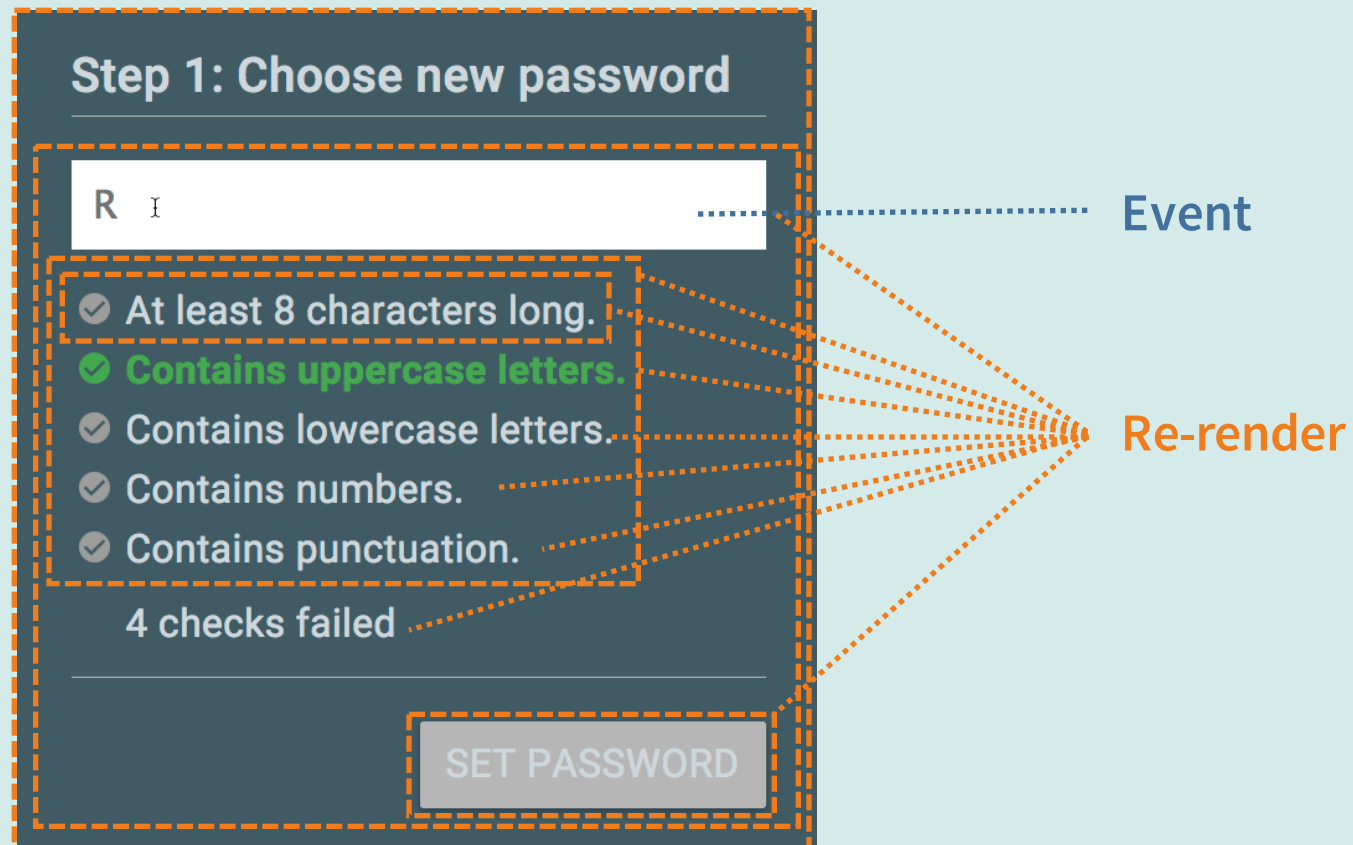
Verbinden von Model und View

- Wann wird was gebunden?
- Wie genau funktioniert das Binding?
 - Zum Beispiel: Element in Liste oder ganze Liste
- Reihenfolge von Events

Wird schnell **komplex, schwer zu durchschauen**



GANZ EINFACH: ALLES RENDERN



BEISPIEL 1: PASSWORD FORMULAR

```
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)} />

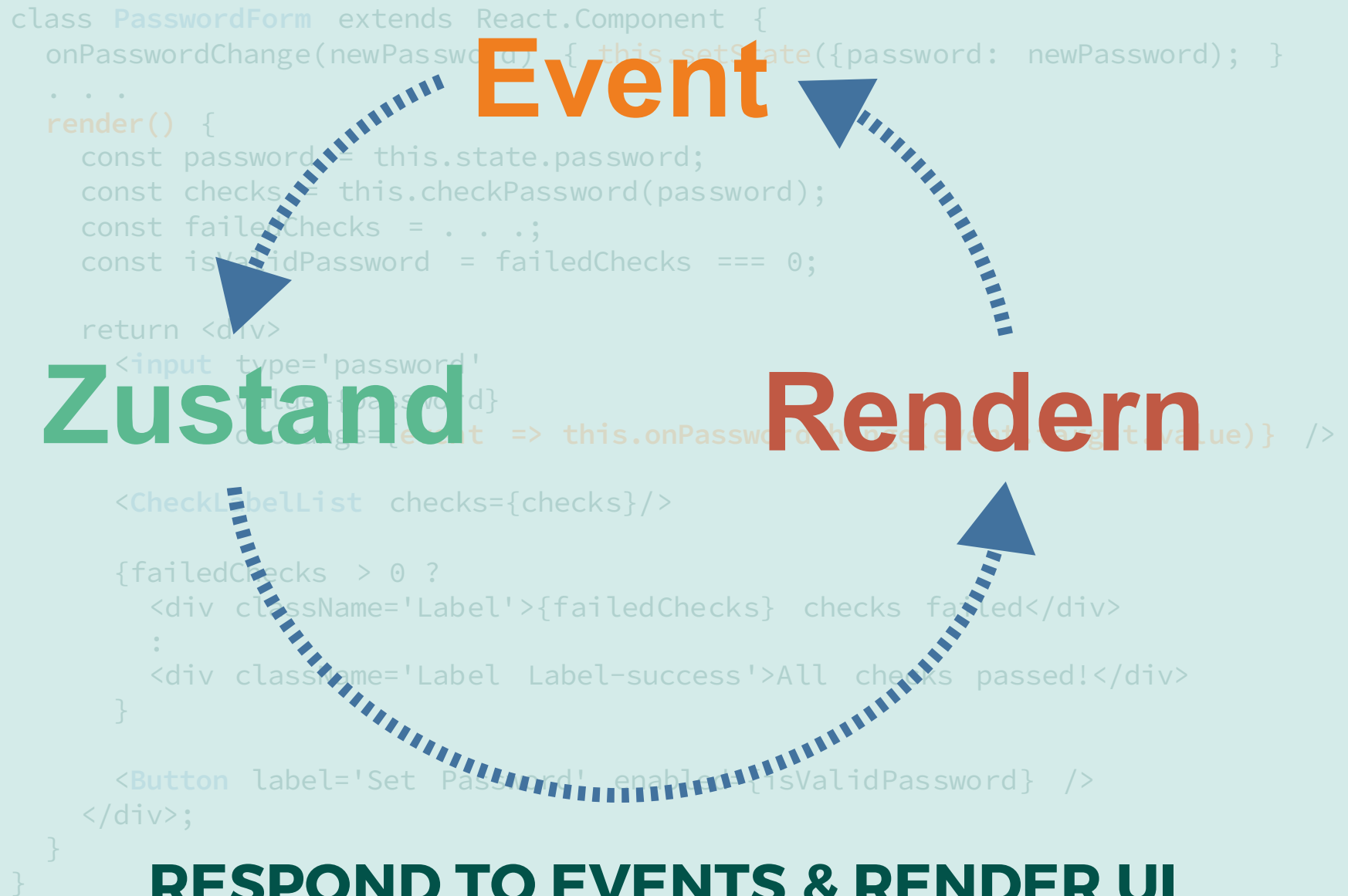
      <CheckLabelList checks={checks}/>

      {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>;
  }
}
```

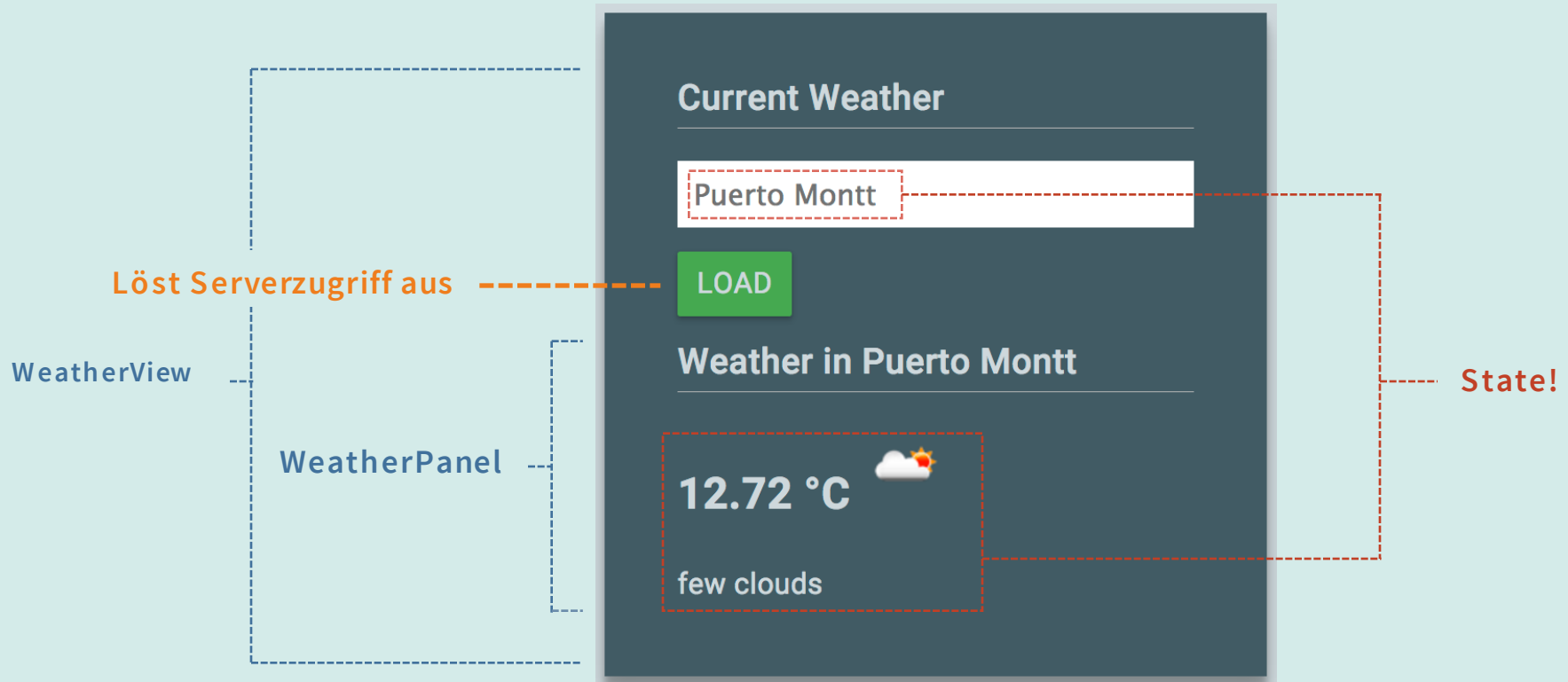
The diagram illustrates the state change and re-rendering process. A box labeled "Event" has a dashed arrow pointing up to `this.setState` in the `onPasswordChange` method. Another dashed arrow points from `this.setState` to the `render()` method, with a box labeled "Neu rendern" (New render) next to it.

REACT: UNI DIRECTIONAL DATAFLOW



BEISPIEL 2: SERVERZUGRIFFE

Gleiches Prinzip, anderes Event



- Daten werden (asynchron) vom Server geladen
- Beim Eintreffen des Ergebnisses muss neu gerendert werden

BEISPIEL 2: SERVERZUGRIFFE MIT FETCH

```
class WeatherView extends React.Component {
```

```
  fetchWeather() {  
    fetch(`http://api.w.org/${this.state.city}`)  
      .then(response => response.json())  
      .then(weather => this.setState({weather: weather}))  
    ;  
  }  
  
  render() {  
    return <div>  
      <Button label='Load' onClick={() => this.fetchWeather()} />  
      <input type='text' value={this.state.city}  
        onChange={e => this.setState({city: e.target.value})} />  
      <WeatherPanel weather={this.state.weather} />  
    </div>;  
  }  
}
```

Daten vom Server laden

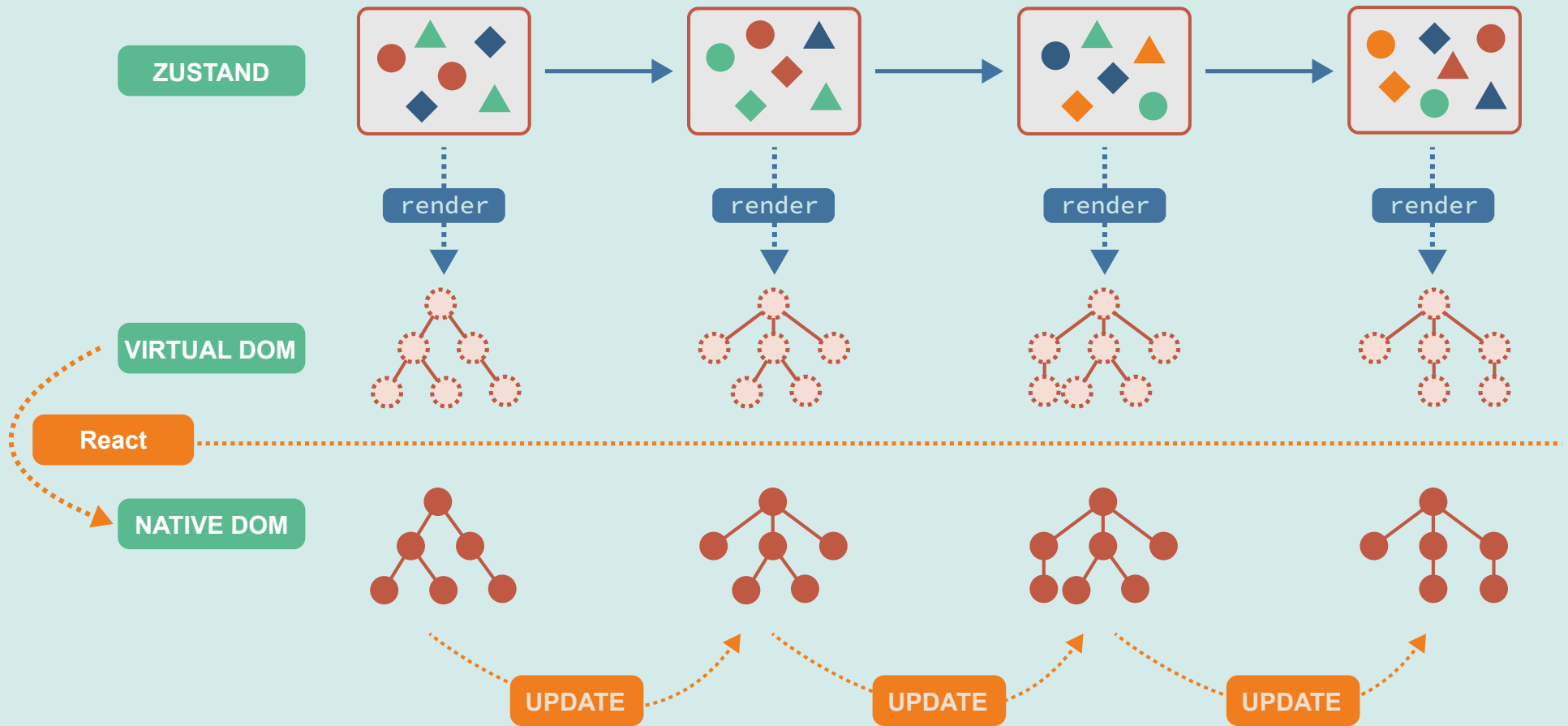
Zustand setzen
(Antwort vom Server)

Geladene Daten anzeigen

Neu rendern

Event

HINTERGRUND: VIRTUAL DOM



HINTERGRUND: VIRTUAL DOM

Virtual DOM

- `React.createElement()` liefert ein **virtuelles** DOM-Objekt zurück
- DOM **Events** sind gewrappt
- Trennung von Darstellung und Repräsentation

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

REACT: „UI AS A FUNCTION“

render()

Step 1: Choose new password

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

5 checks failed

SET PASSWORD

render(R3)

Step 1: Choose new password

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

3 checks failed

SET PASSWORD

render(R3!demo)

Step 1: Choose new password

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

All checks passed!

SET PASSWORD

render(R3)

Step 1: Choose new password

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

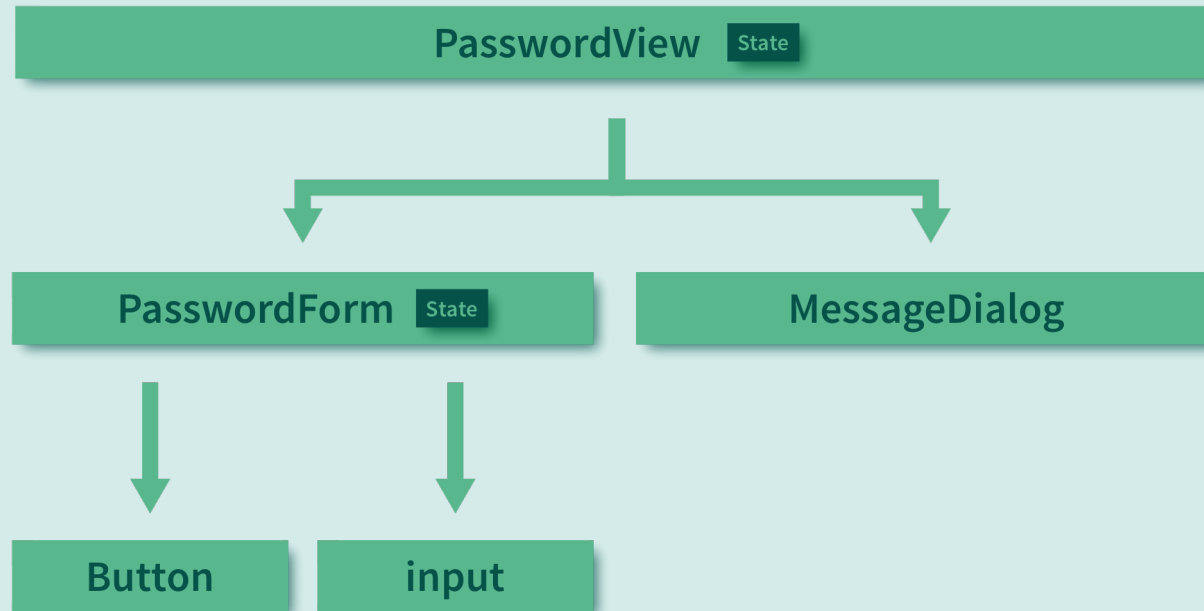
3 checks failed

SET PASSWORD

f(zustand) → UI

- Es wird genau eine UI zu genau einem Zustand gerendert
- Deklarativ, keine Seiteneffekte
- Sehr einfaches Prinzip
- Performant durch Virtual DOM

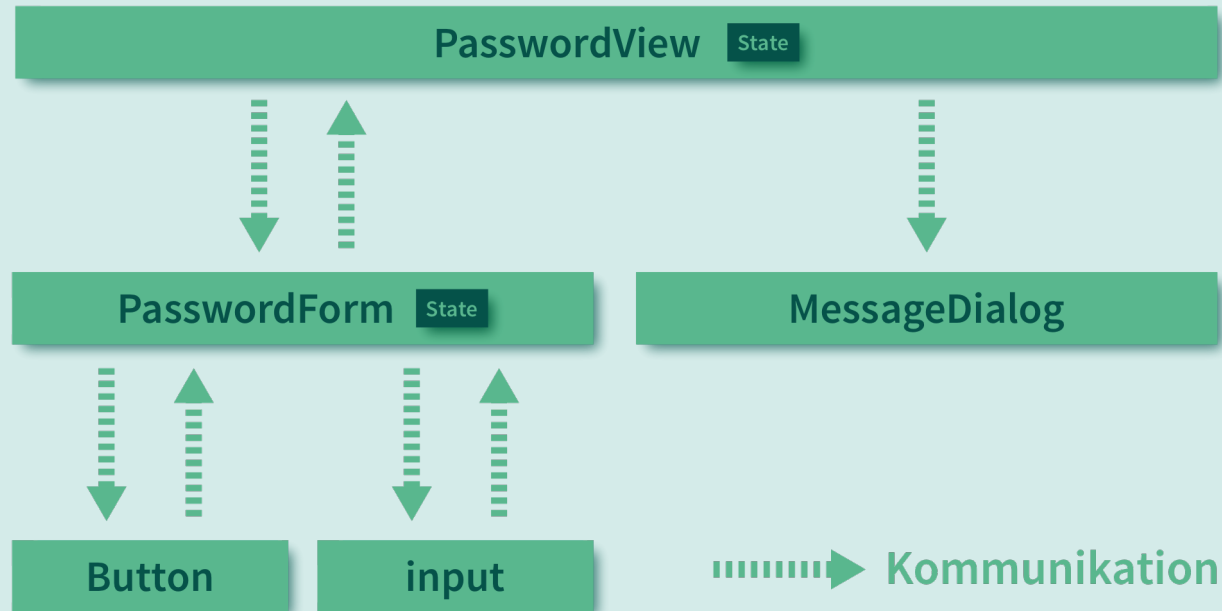
KOMPONENTENHIERARCHIEN



Typische React Anwendungen: Hierarchisch aufgebaut

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

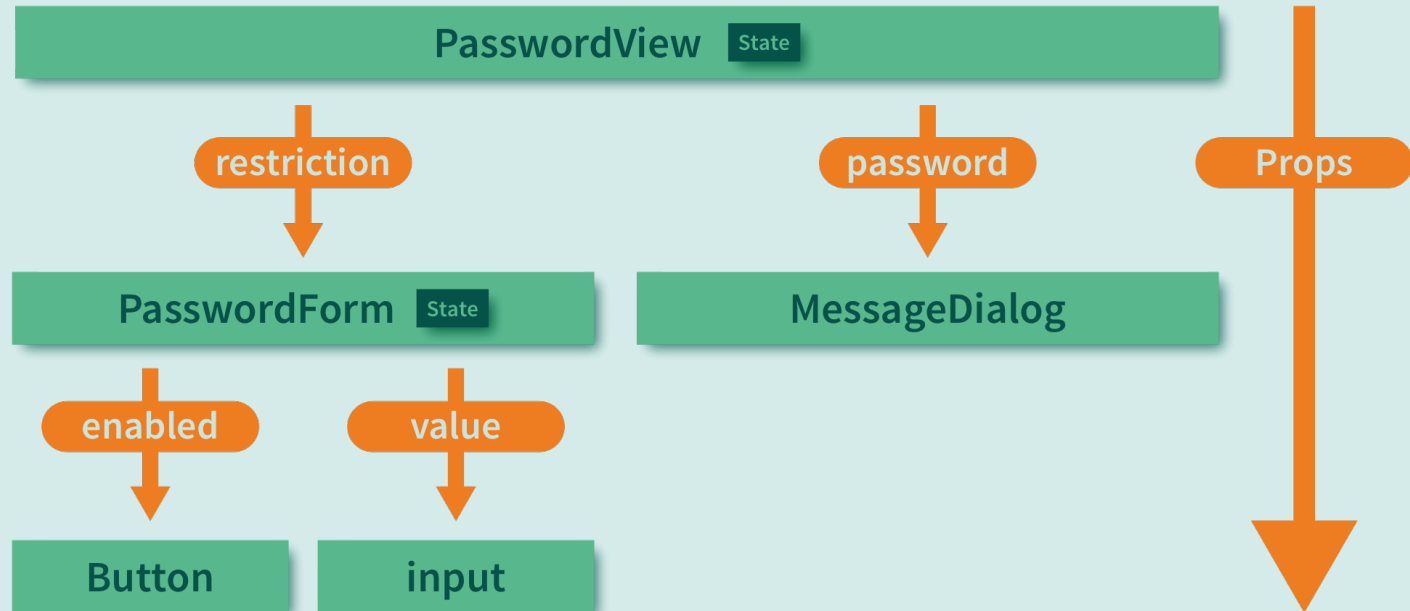
KOMMUNIKATION ZWISCHEN KOMPONENTEN



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?

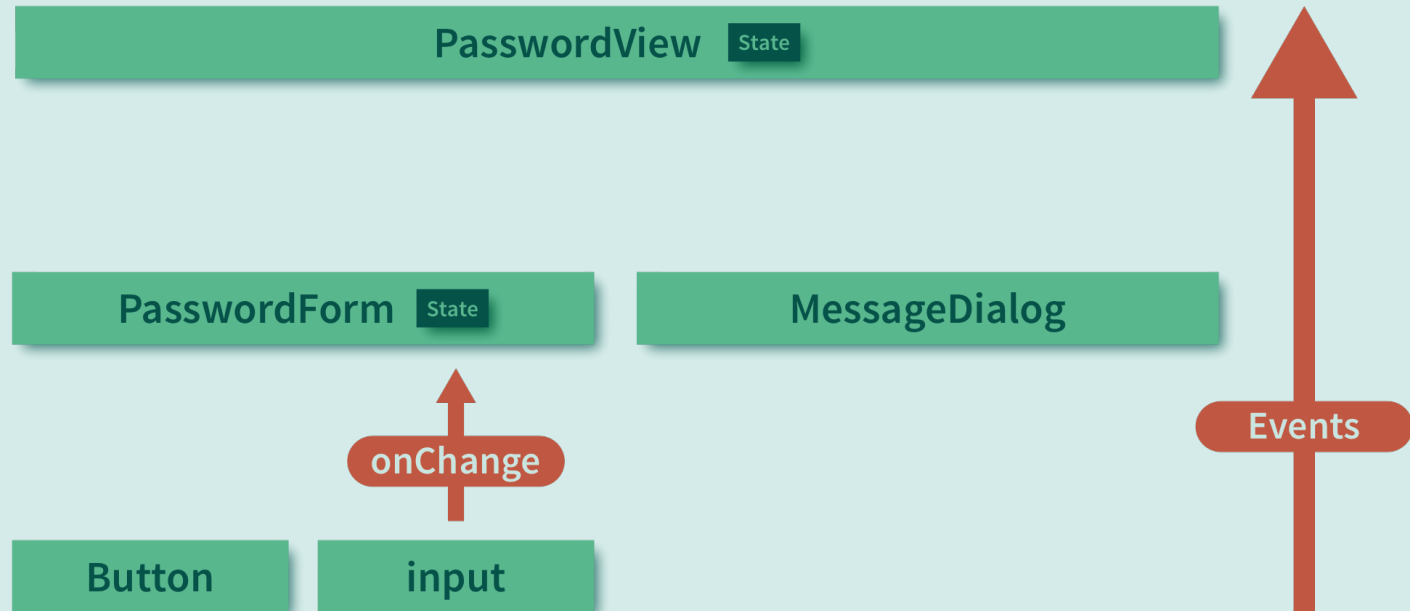
KOMMUNIKATION: PROPERTIES



Von oben nach unten: **Properties**

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

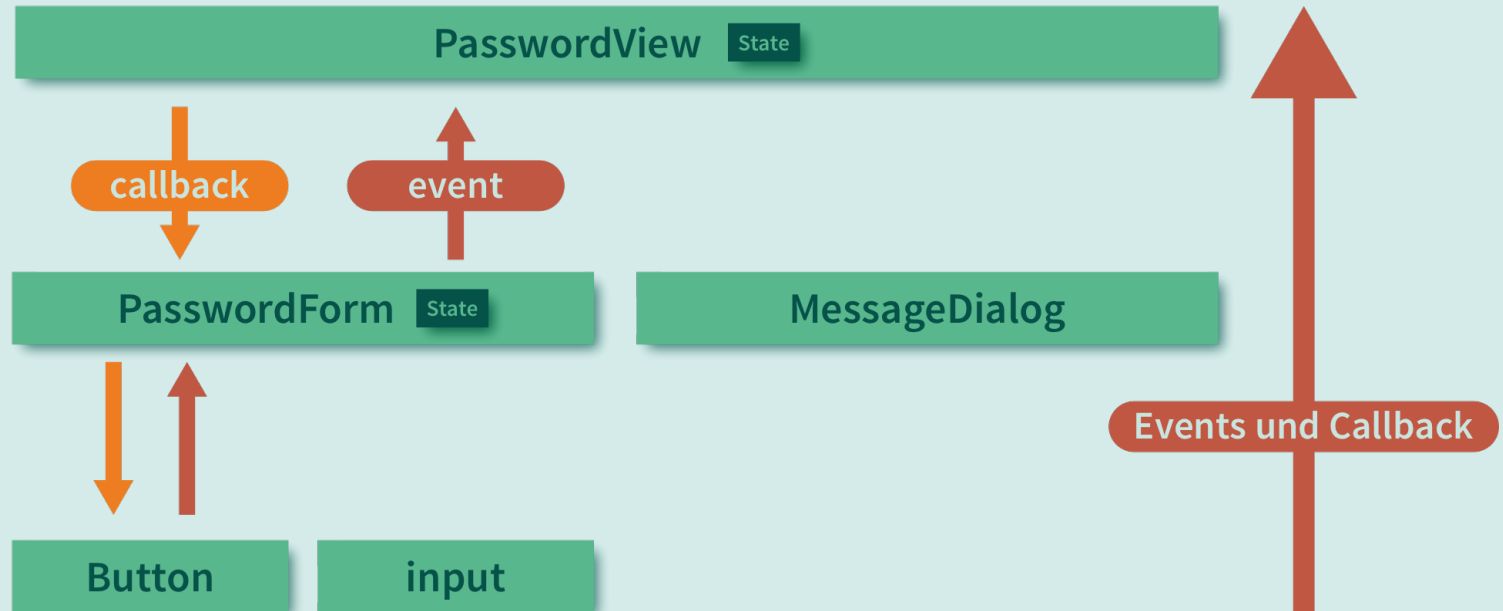
KOMMUNIKATION: DOM EVENTS



Von unten nach oben: **Events**

```
<input onChange={...} />
```

KOMMUNIKATION: EIGENE EVENTS



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

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

BEISPIEL: CALLBACK-FUNKTIONEN (1)

```
class PasswordView extends React.Component {  
  render() {  
    return . . .  
    <PasswordForm . . .  
      setPasswordHandler={p=>this.setState(newPassword: p)}  
    />;  
  }  
}
```

Callback-Funktion übergeben

BEISPIEL: CALLBACK-FUNKTIONEN (2)

```
class PasswordView extends React.Component {  
  
  render() {  
    return . . .  
    <PasswordForm . . .  
      setPasswordHandler={p=>this.setState(newPassword: p)}  
    />;  
  }  
}
```

```
class PasswordForm extends React.Component {  
  render() {  
    return . . .  
    <input value=". . ." onChange=". . ." />  
    <Button label="Set new Password"  
      onClickHandler=  
        {()=>this.props.setPasswordHandler(this.state.password)}  
    />  
  }  
}
```

Callback-Funktion aufrufen

Callback-Funktion angeben

```
PasswordForm.propTypes = {  
  setPasswordHandler: React.PropTypes.func.isRequired  
}
```


BEISPIEL: CALLBACK-FUNKTIONEN (3)

```
class PasswordView extends React.Component {
```

```
  render() {
```

```
    return . . .
```

```
    <PasswordForm . . .
```

```
      setPasswordHandler={p=>this.setState(newPassword: p)}  
    </>;
```

```
  }
```

```
}
```

Rendern

Rendern

„event“

```
class PasswordForm extends React.Component {
```

```
  render() {
```

```
    return . . .
```

```
    <input value=". . ." onChange=". . ." />
```

```
    <Button label="Set new Password"
```

```
      onClickHandler=
```

```
        {()=>this.props.setPasswordHandler(this.state.password)}
```

```
    />
```

```
  }
```

```
}
```

```
PasswordForm.propTypes = {
```

```
  setPasswordHandler: React.PropTypes.func.isRequired
```

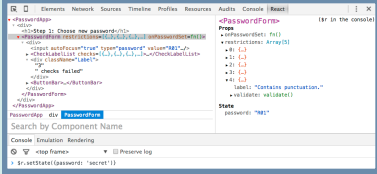
```
}
```

Callback-Funktion übergeben

Callback-Funktion aufrufen

Callback-Funktion angeben

ÖKOSYSTEM



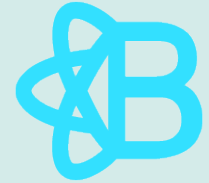
Developer Tools

material-design



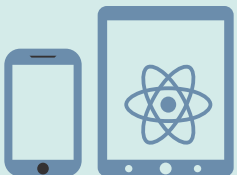
Flux Architekturpattern

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

Vielen Dank!

Fragen?

@NILSHARTMANN

AUSBLICK

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

PROJEKT

SERVERSEITIGES RENDERN (1)

Zur Erinnerung: Rendern auf dem Client

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

import PasswordView from './components/PasswordView';

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

SERVERSEITIGES RENDERN (2)

Rendern auf dem Server (vereinfacht)

```
import React from 'react';
import ReactDOMServer from 'react-dom/server';

import PasswordView from './components/PasswordView';

const html = ReactDOMServer.renderToString(<PasswordView />);

const page = `
  <head>. . . </head>
  <body><div id='mount'>${html}</div></body>
</html>`;

// page an Client senden
```

BEISPIEL: UNIT TESTS (OHNE DOM)

```
import { expect } from 'chai';
import TestUtils from 'react-addons-test-utils';
```

```
describe('CheckLabel', () => {
  it('should render a "checked" label', () => {
```

„Shallow rendering“

```
    const renderer = TestUtils.createRenderer();
    renderer.render(
      <CheckLabel label='My Label' checked={true}/>
    );

    const tree = renderer.getRenderOutput();
    expect(tree.type).to.equal('div');
    expect(tree.props.className).to.equal('CheckLabel-checked');
    expect(tree.props.children).to.equal('My Label');
  });
});
```


BEISPIEL: UNIT TESTS (MIT DOM)

```
import { expect } from 'chai';
import jsdom from 'mocha-jsdom';
import { . . . } from 'react-addons-test-utils';

describe('PasswordForm', () => {
  jsdom();

  it('updates button', () => {
    const tree = renderIntoDocument(
      <PasswordForm restrictions={ . . . } onPasswordSet={ . . . } />
    );

    expect(isCompositeComponentWithType(tree, PasswordForm)).to.be.true;
    const inputField = findRenderedDOMComponentWithTag(tree, 'input');
    const btn = findRenderedDOMComponentWithTag(tree, 'button');

    Simulate.change(inputField, {target: {value: 'xxx'}});
    expect(setPasswordButton.disabled).to.be.true;
  });
});
```

BEISPIEL: INITIALISIERUNG UND LEBENSZYKLUS

Zustand initialisieren

```
class WeatherView extends React.Component {  
  constructor() {  
    this.state = { city: 'Hamburg' };  
  }  
}
```

Initiales laden auslösen

```
  componentDidMount() { this.fetchWeather(); }
```

```
  fetchWeather() {  
    fetch(`http://api.w.org/${this.state.city}`)  
      .then(response => response.json())  
      .then(weather => this.setState({weather}))  
    ;  
  }
```

```
  render() {  
    return <div>  
      <Button label='Load' onClick={() => this.fetchWeather()} />  
      <input type='text' value={this.state.city}  
        onChange={e => this.setState({city: e.target.value})} />  
      <WeatherPanel weather={this.state.weather} />  
    </div>;  
  }  
}
```