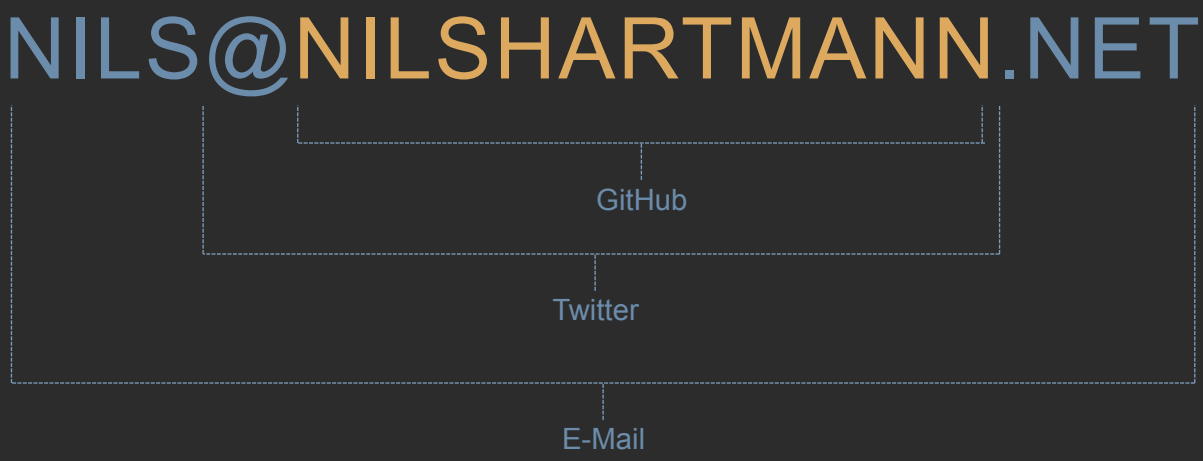
MIT REACT

UI-KOMPONENTEN ENTWICKELN

NILS HARTMANN | W-JAX 2015


NILS@NILSHARTMANN.NET
GitHub
Twitter
E-Mail

A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES

React

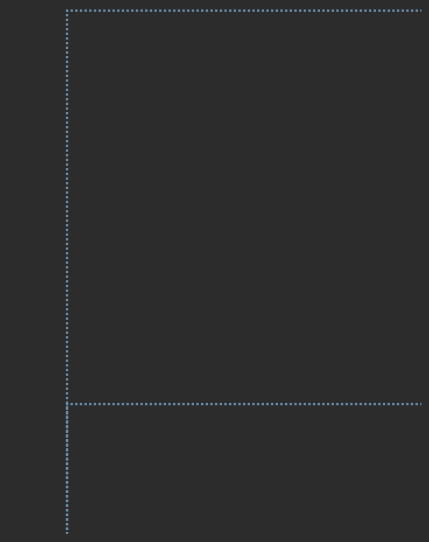
SINGLE PAGE APPLICATIONS

React

OPEN SOURCE VON FACEBOOK

HTTPS://FACEBOOK.GITHUB.IO/REACT/

React



„Major“-Releases

0.3

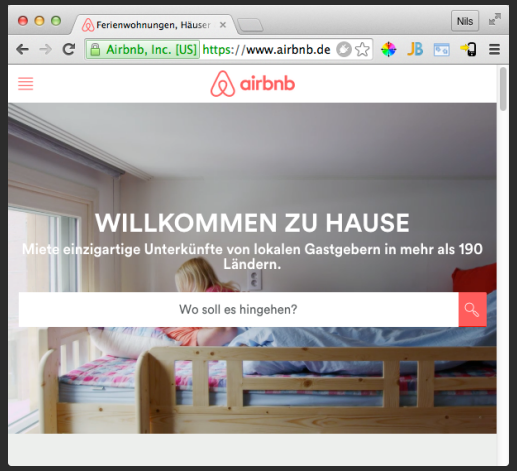
05 | 2013 Open Source

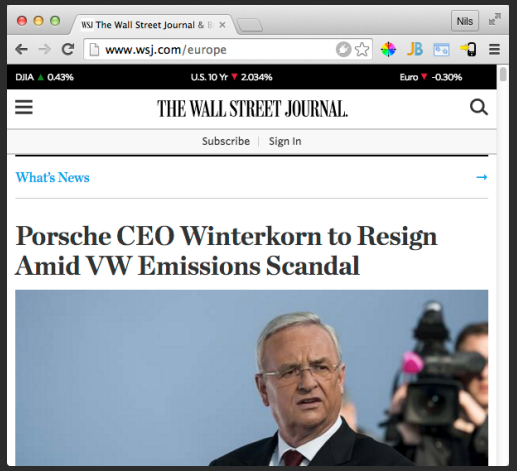
0.14.1

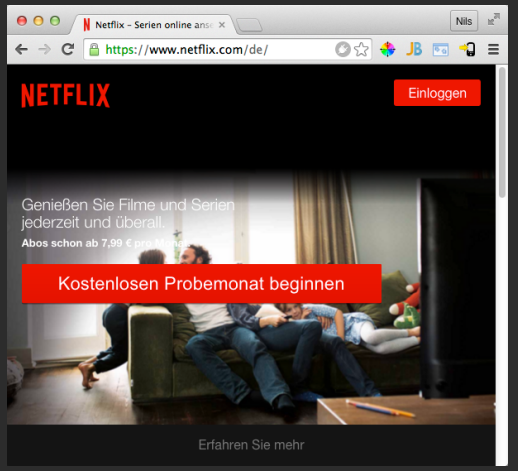
rc
beta
alpha

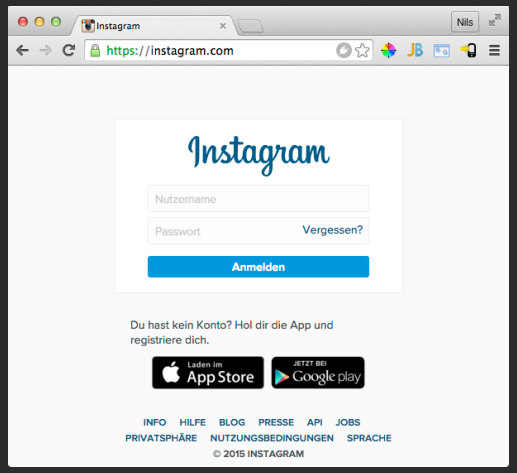
10 | 2015 Aktuelles Release

BUILT WITH REACT



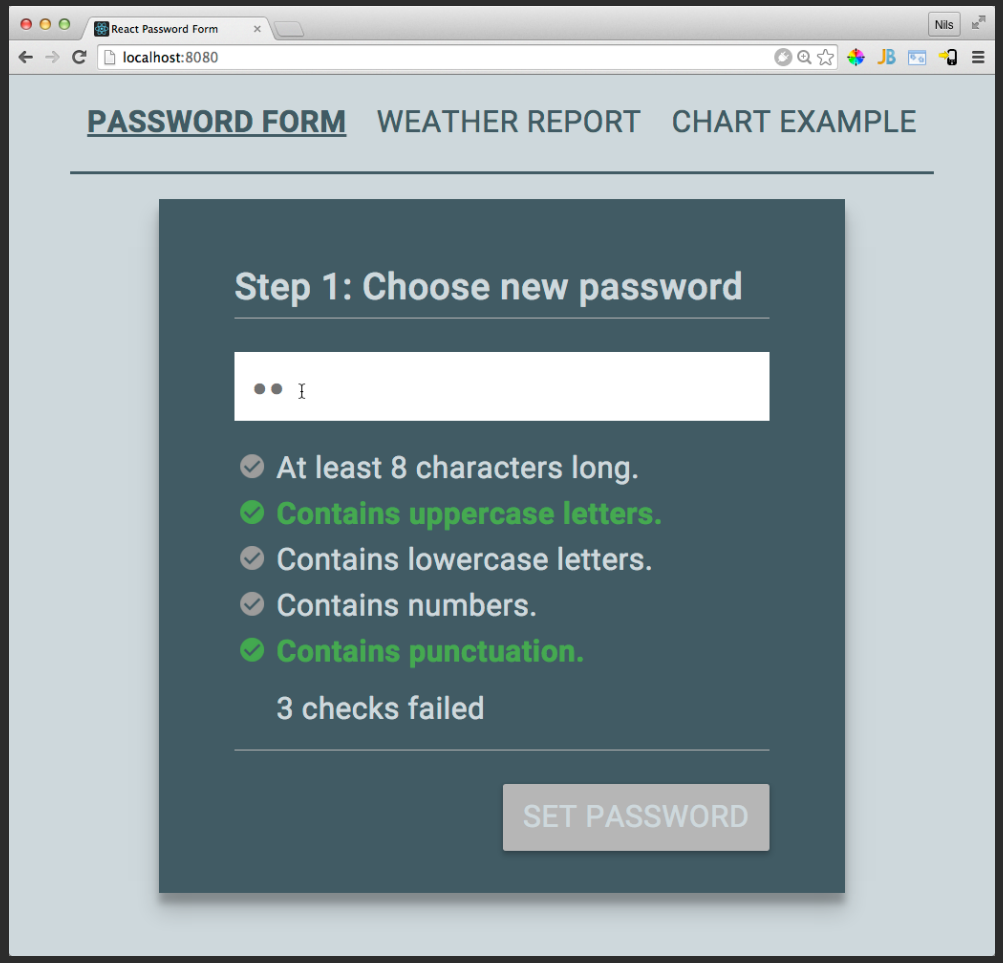






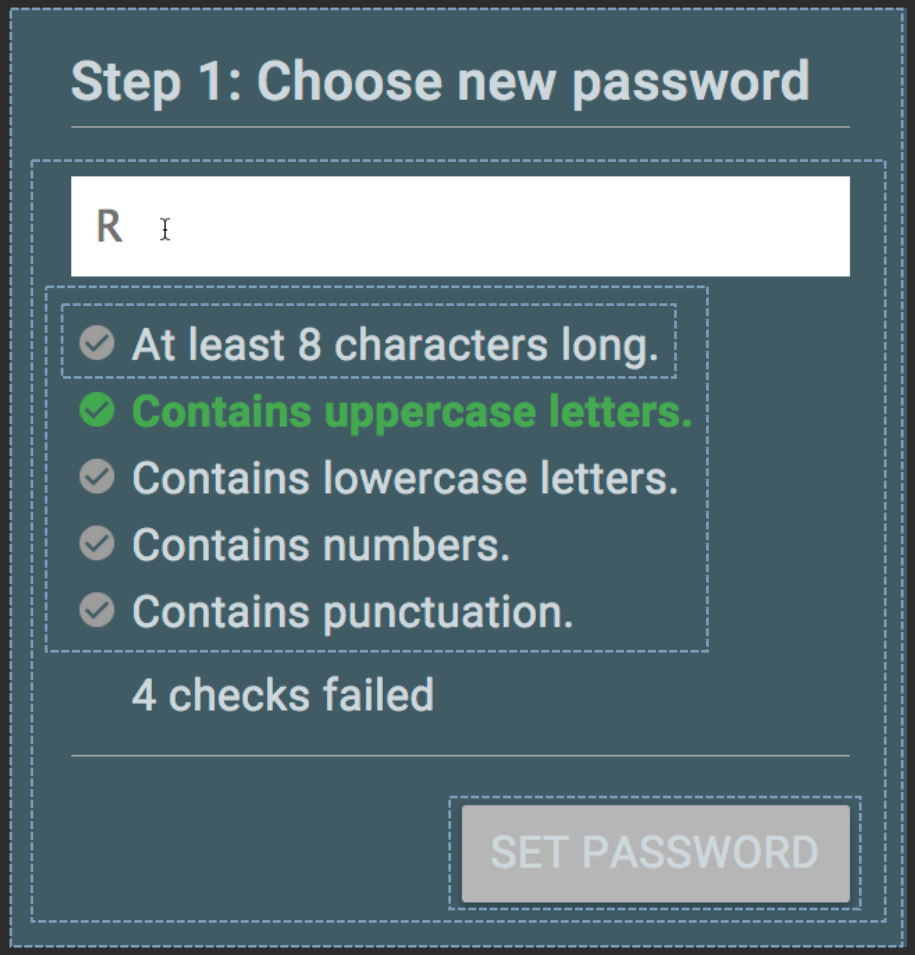
V in MVC

W-JAX DEMO ANWENDUNG



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

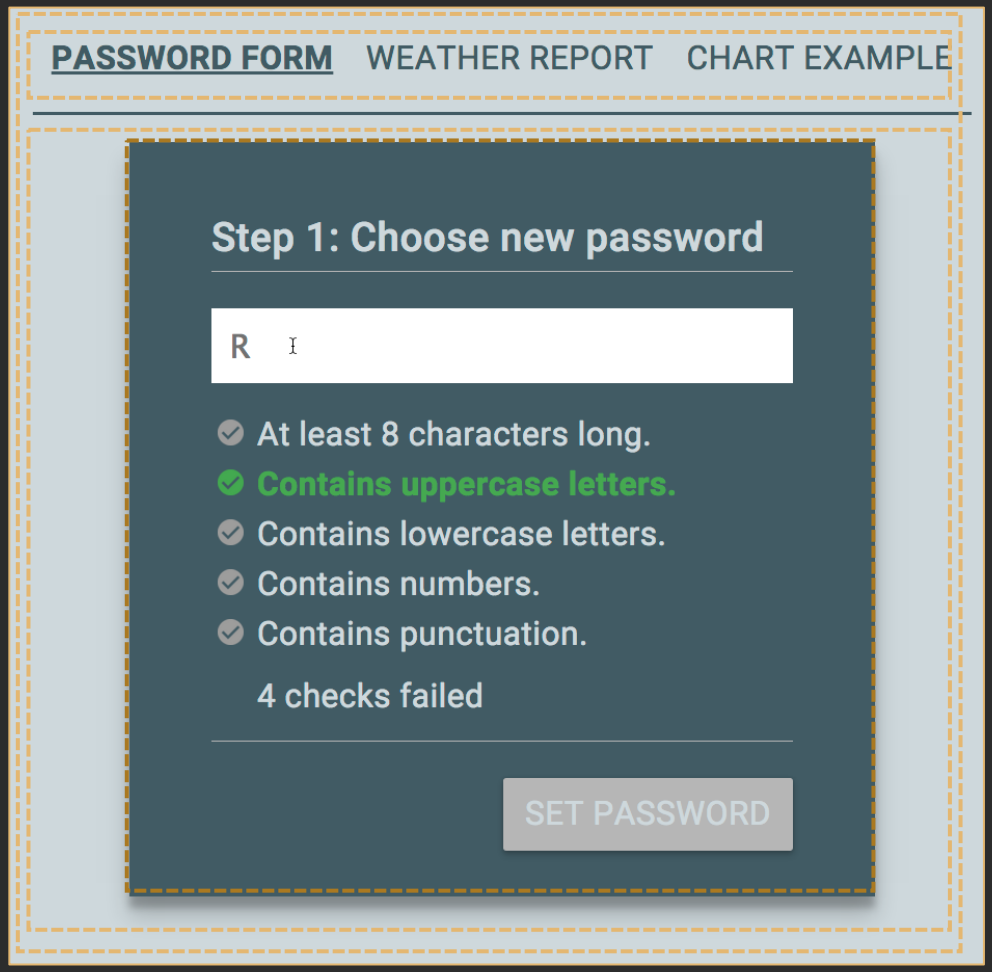
Demo: https://nilshartmann.github.io/react-example-app/




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

Wiederverwendbar
Hierarchisch
Logik und UI

KOMPONENTEN



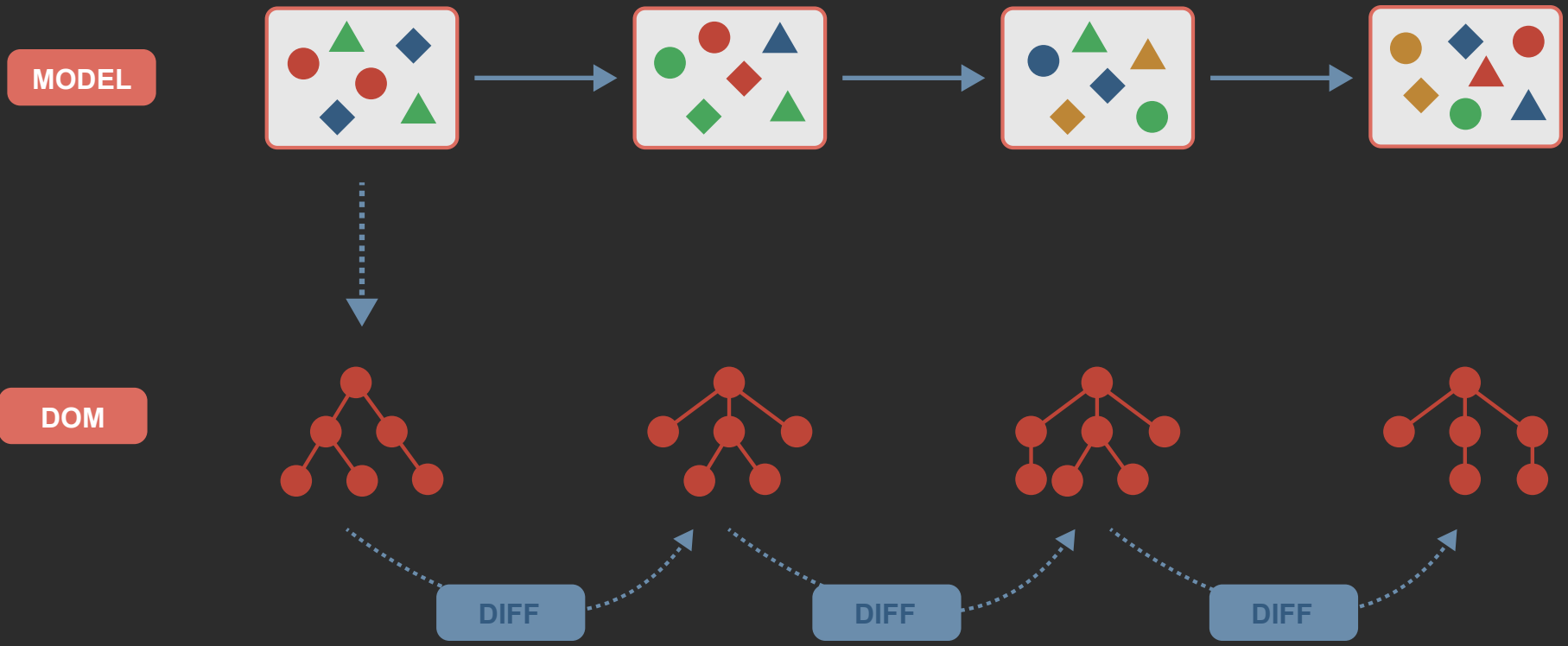

```
<Application>
  <Navigation />
  <ViewContainer>
    <PasswordView>
      . . .
    </PasswordView>
  </ViewContainer>
</Application>
```

Aus Komponenten aggregiert

ANWENDUNGEN

Hintergrund

REACT

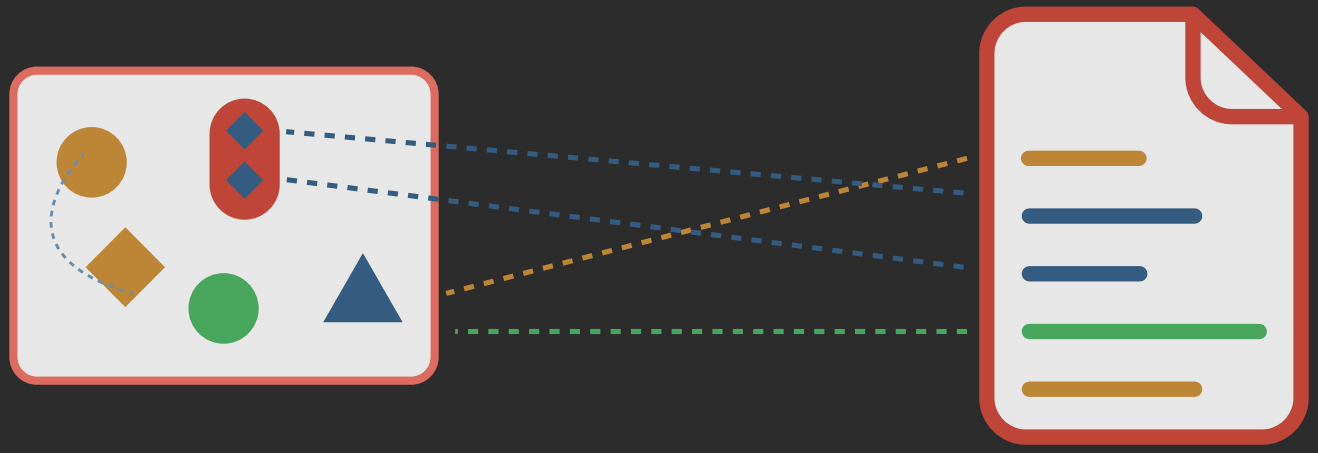


Manuelle DOM-Manipulationen

Umständliche API

Fehleranfällig

Performance-kritisch



Verbinden von Model und View

Wann wird was gebunden?

Wie funktioniert das Binding?

Reihenfolge von Events?

respond to events & render UI

Einfachheit

REACT

Einfachheit

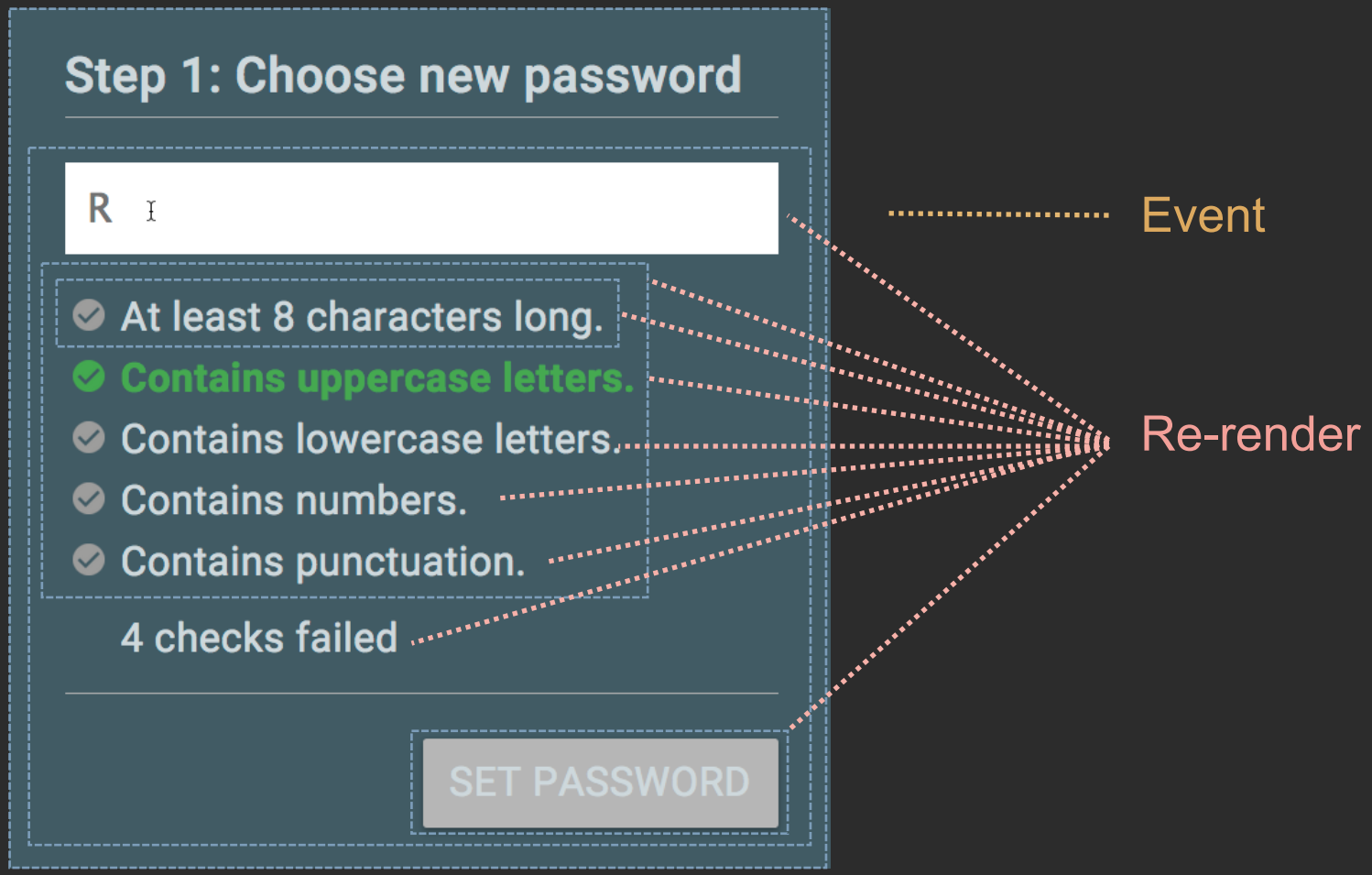
respond to events & and render UI

Immer ganze Komponente rendern

Kein 2-Way-Databinding
Kein dirty checking

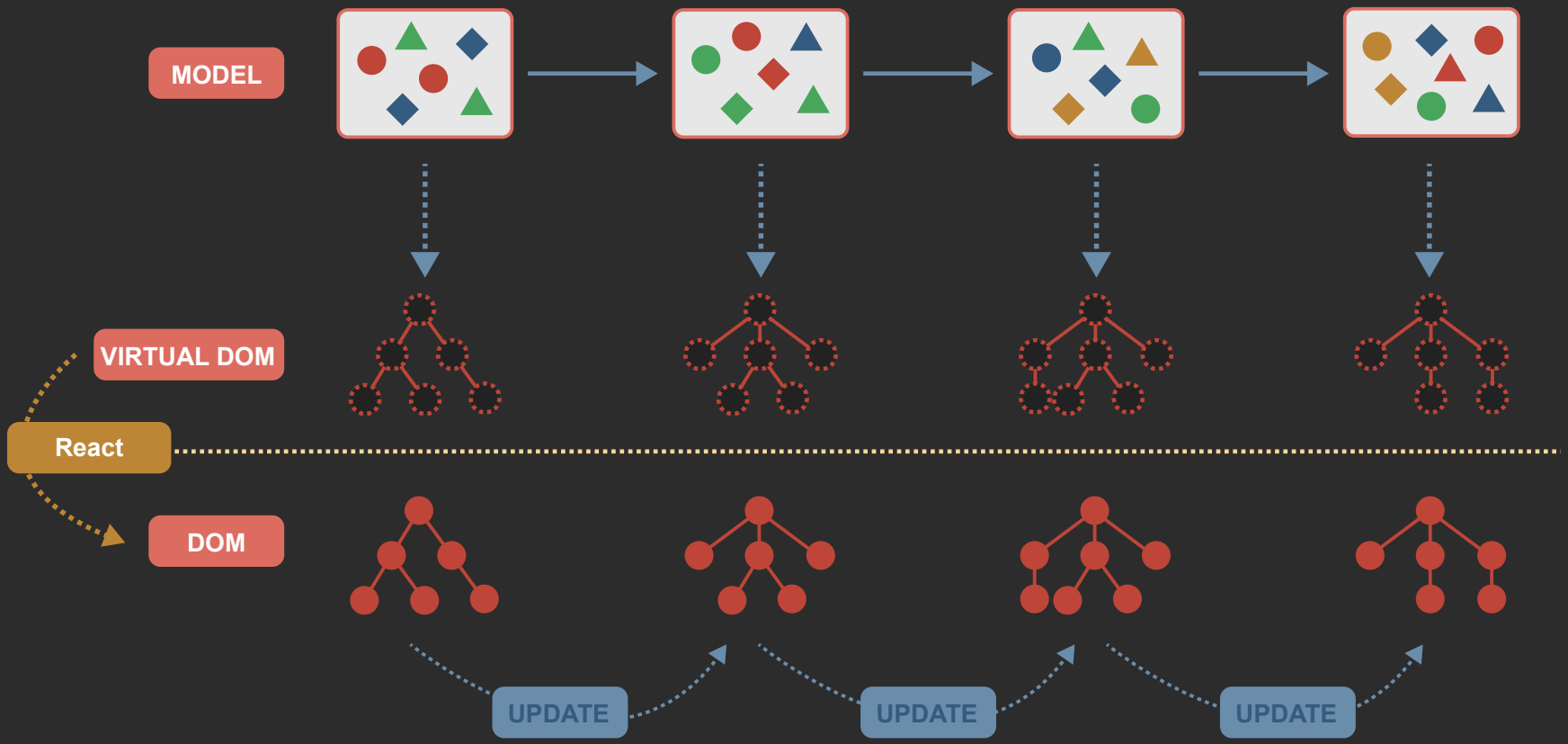
Einfachheit

respond to events & and render UI


Step 1: Choose new password
R
At least 8 characters long.
Contains uppercase letters.
Contains lowercase letters.
Contains numbers.
Contains punctuation.
4 checks failed
SET PASSWORD
Event
Re-render

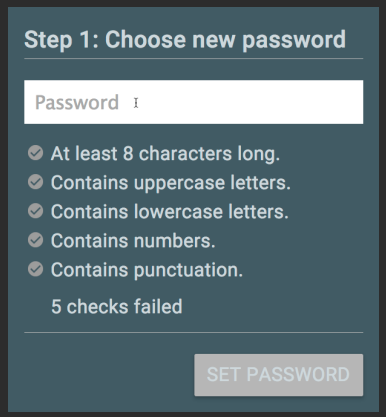
Einfachheit

respond to events & and render UI

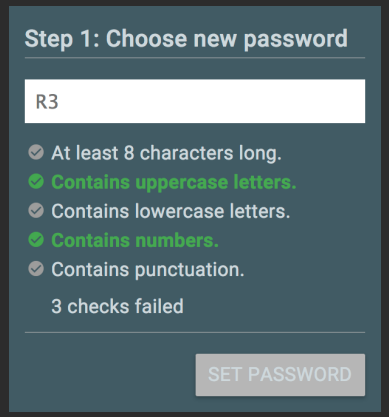


VIRTUELLER DOM

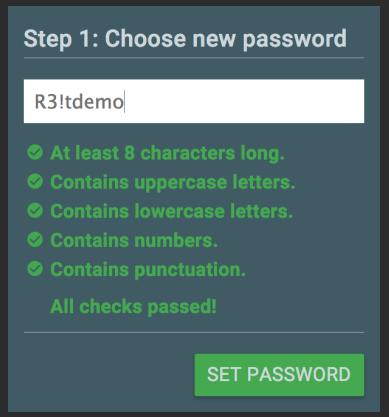
`render()`



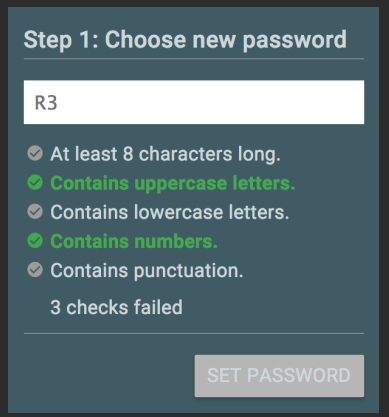
`render(R3)`



`render(R3!tdemo)`



`render(R3)`



∫ (model) → UI

Model mit allen Zuständen (Textfelder, Auswahllisten etc)

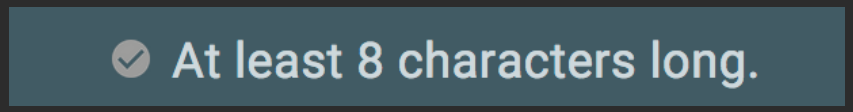
Immer ein Zeitpunkt

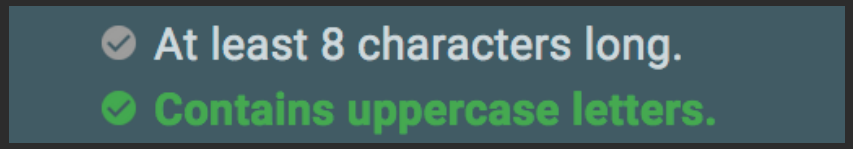
Keine Dynamik

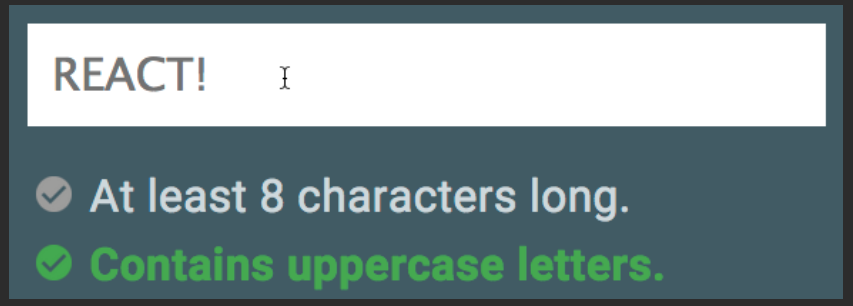
UI AS A FUNCTION

React

PRAXIS



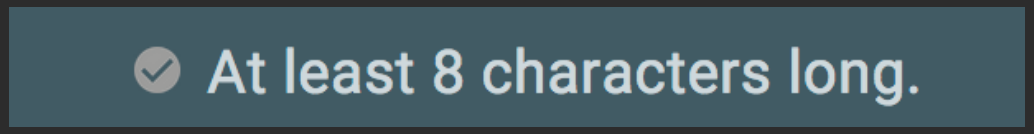




React

SCHRITT FÜR SCHRITT

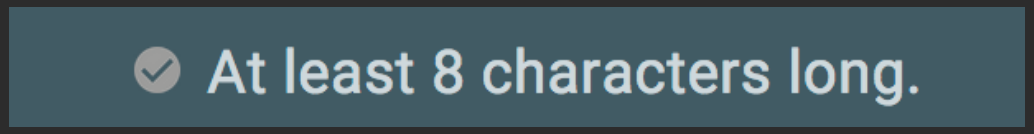
EINE REACT KOMPONENTE 1




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

HTML

EINE REACT-KOMPONENTE 2



Komponente CheckLabel

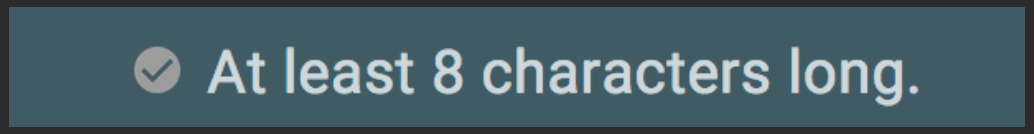
Komponentenfunktion
(seit 0.14)

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

JSX: Statt Template-Sprache

CheckLabel.js

EINE REACT-KOMPONENTE 3

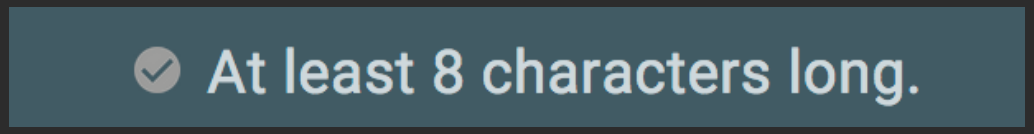


Erzeugt „virtuelles“ DOM-Element

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

Übersetzter JavaScript Code

KOMPONENTE RENDERN




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

import CheckLabel from './CheckLabel';


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

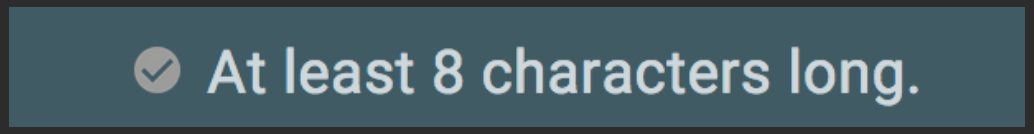
app.js

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

Webpack

index.html

PROPERTIES

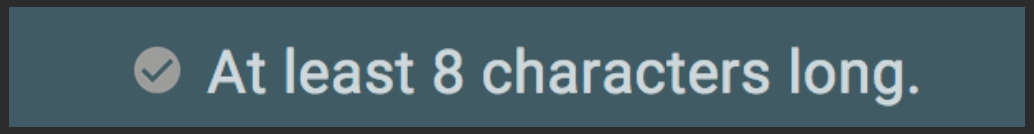



```
{
  checked: false,
  label: 'At least 8 characters long.'
}
```

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

Properties (destrukturiert)

PROPERTIES BESCHREIBEN



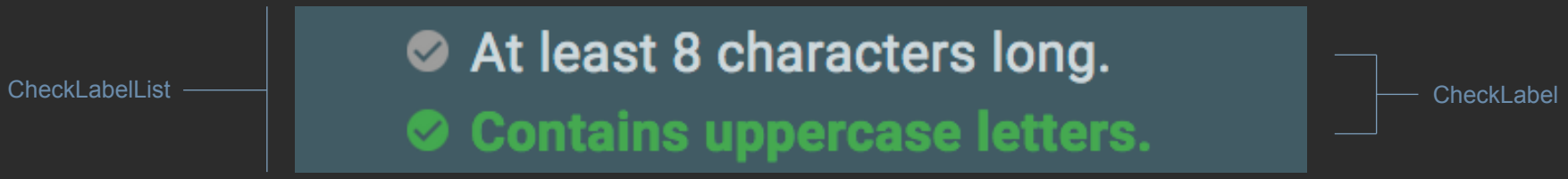

```
function CheckLabel({checked, label}) {
  // . . .
}

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

Beschreibung der Properties

Überprüfung zur Laufzeit

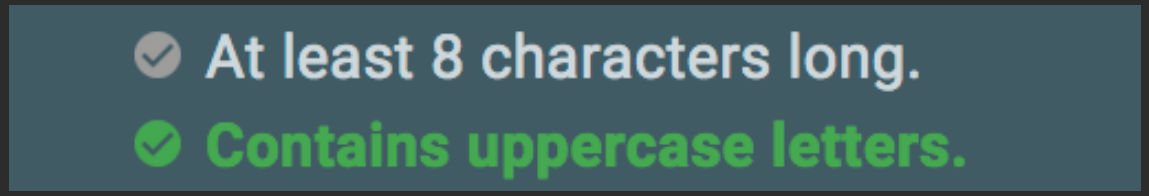
```
▶Warning: Failed propType: Required prop `label` was not specified   main.js:12889
in `CheckLabel`. Check the render method of `CheckLabelList`.
```

KOMPONENTEN VERWENDEN




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

function CheckLabel({checked, label}) {
  // . . .
}
```

LISTEN



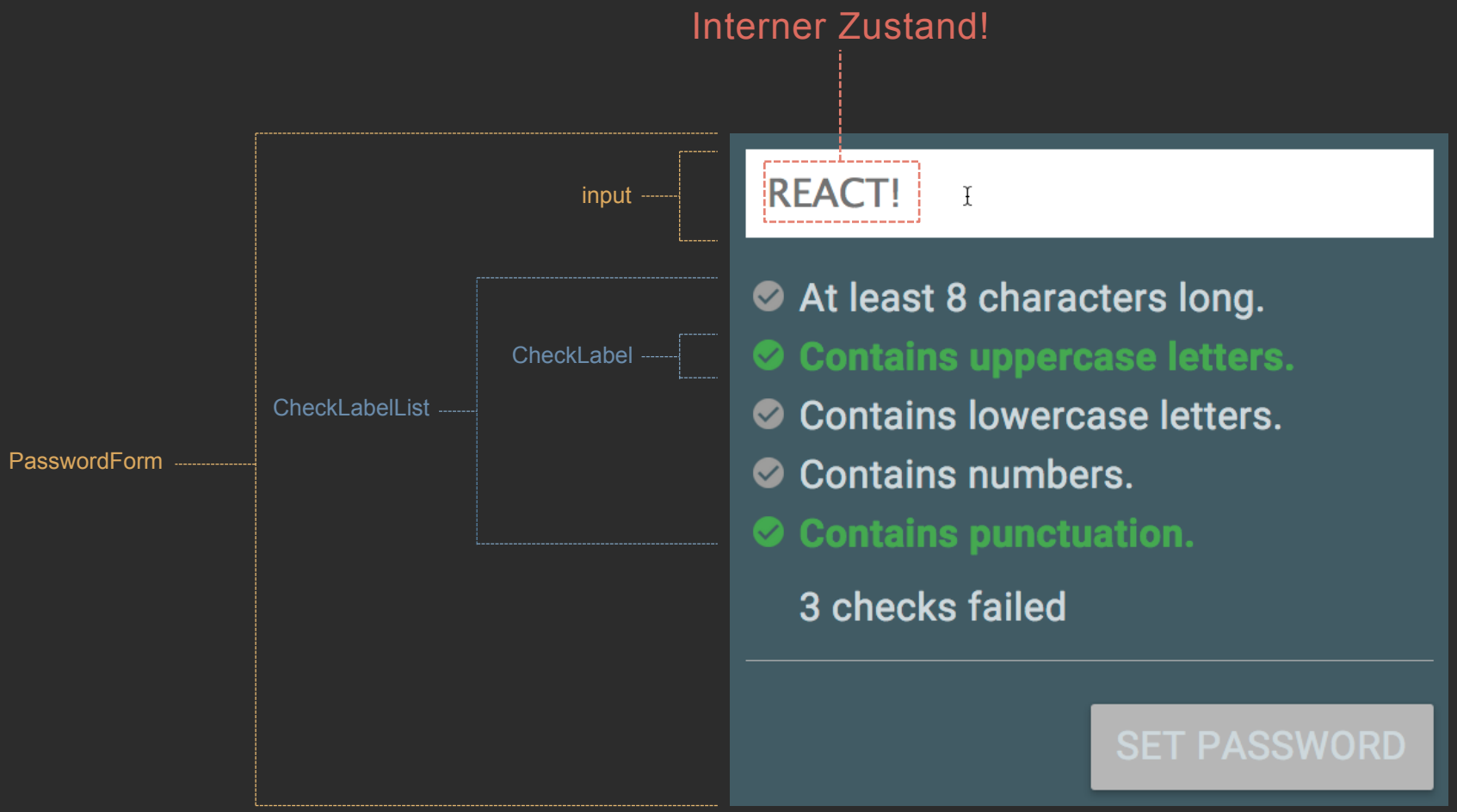

```
[
  { checked: false, label: 'At least 8 characters long.' },
  { checked: true,  label: 'Contains uppercase letters' }
];
```

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

ES5 Array.prototype.map()

Eindeutiger Schlüssel

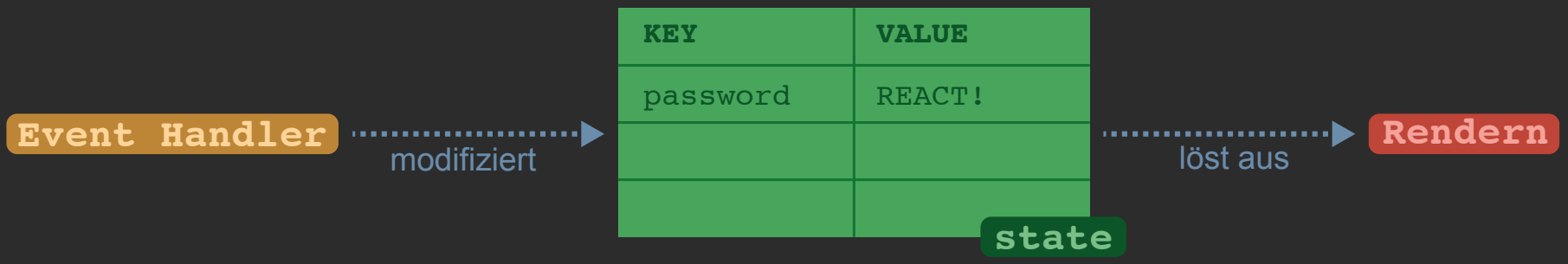
ZUSTANDSBEHAFTETE KOMPONENTEN



ZUSTAND

Event → **Zustand** → **Rendern**

- Textfeld
- Auswahl in Liste
- Checkbox
- Serverantwort
- . . .



KOMPONENTEN KLASSEN

ECMAScript 2015 Klasse

Properties über Konstruktor

React Lifecycle Methoden

Render Methode

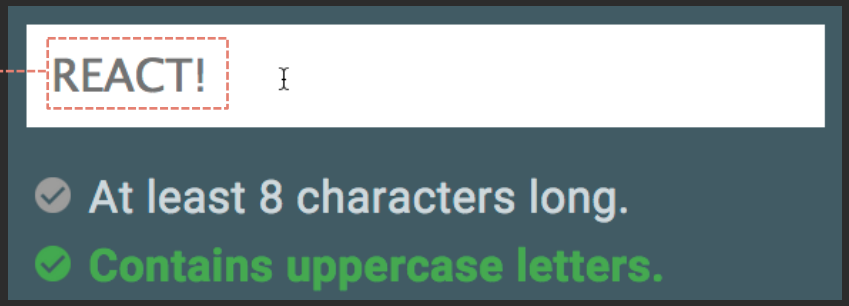
Properties über props-Objekt

Property-Beschreibungen

```
class PasswordForm extends React.Component {
  constructor(props) {
    super(props);
  }

  componentDidMount() { . . . }
  componentWillReceiveProps() { . . . }
  shouldComponentUpdate() { . . . }
  . . .

  render() {
    return <div>{this.props.label}</div>;
  }
}

PasswordForm.propTypes = {
  . . .
};
```

ZUSTAND UND RENDERING

Zustand!

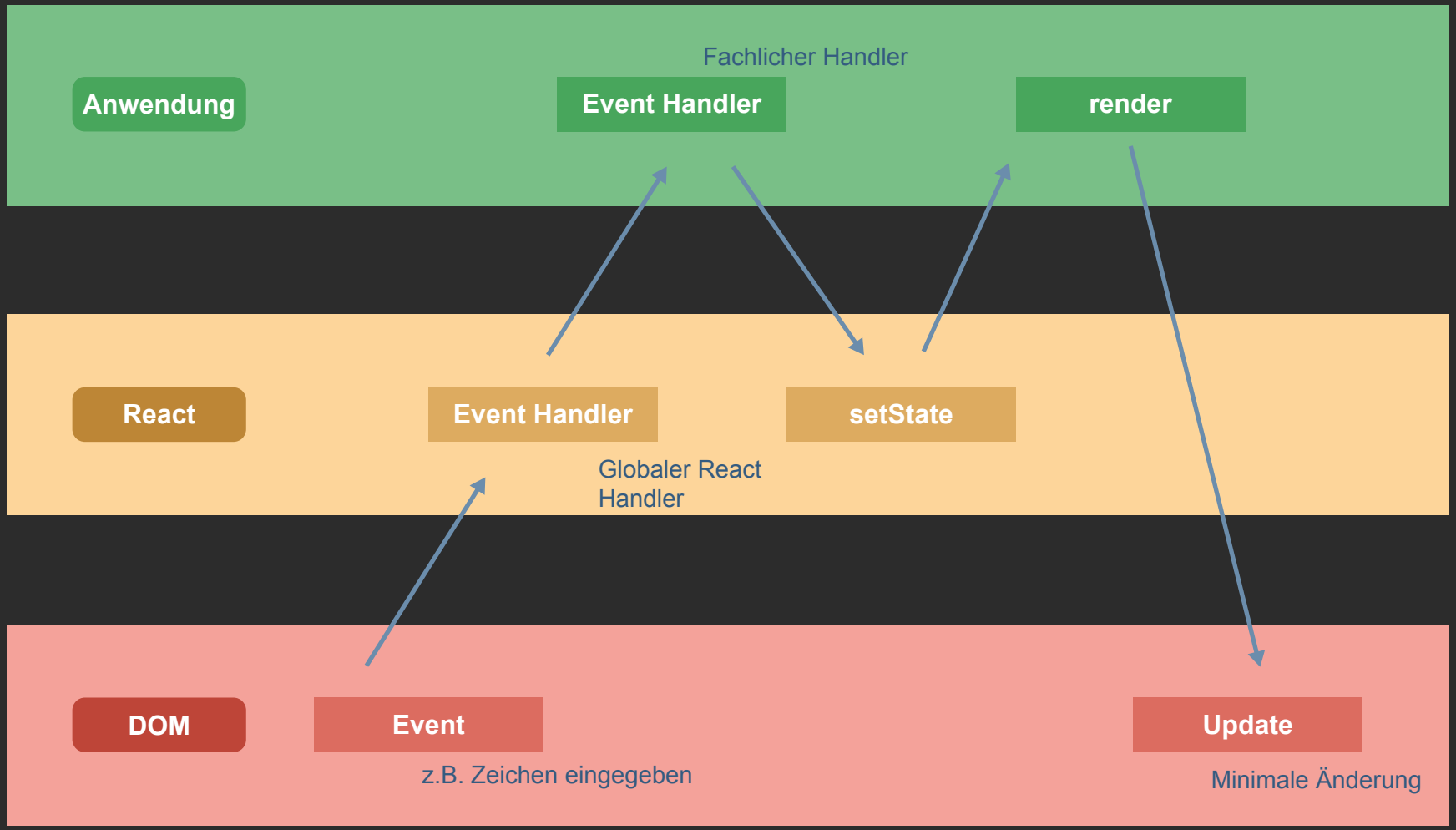



```
class PasswordForm extends React.Component {
 checkPassword(password) { return [ . . . ]; }

 onPasswordChange(input) {
  this.setState({password: input});
 }

 render() {
  const checks = this.checkPassword(this.state.password);

  return . . .
   <input value={this.state.password}
          onChange={e=>this.onPasswordChange(e.target.value)}
   />
   <CheckLabelList checks={checks} />
   <Button enabled={passwordValid} />
 }
}
```

2. Zustand neu setzen

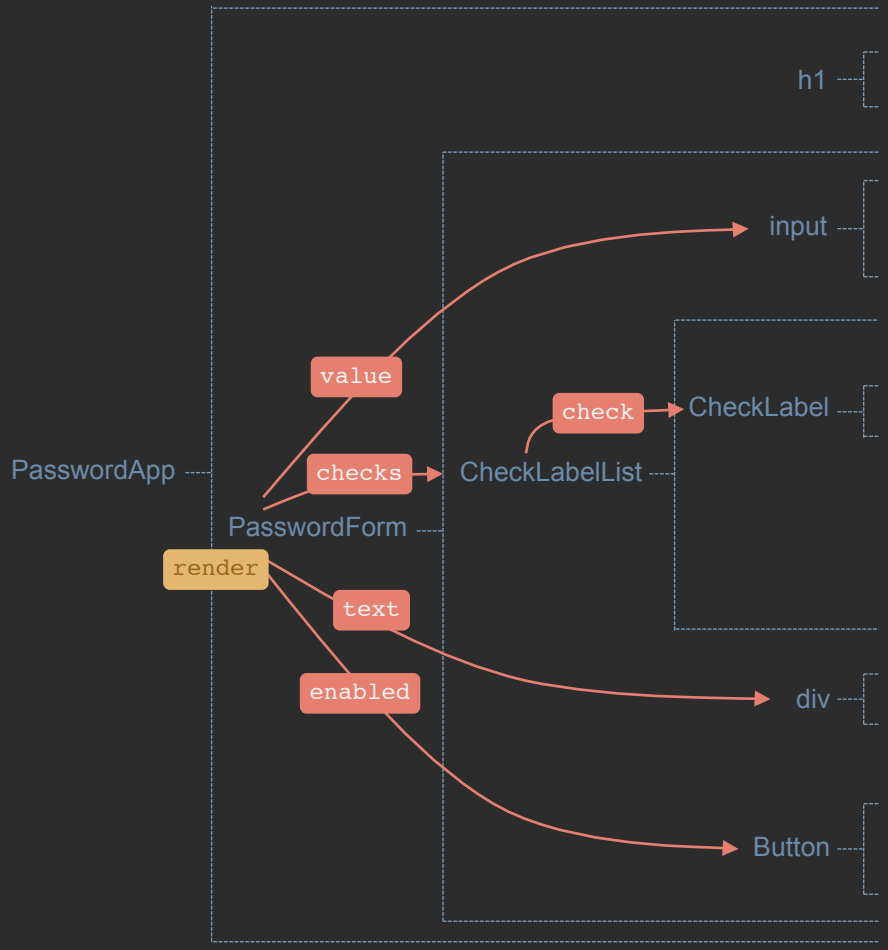
3. löst rendern der gesamten Komponente aus

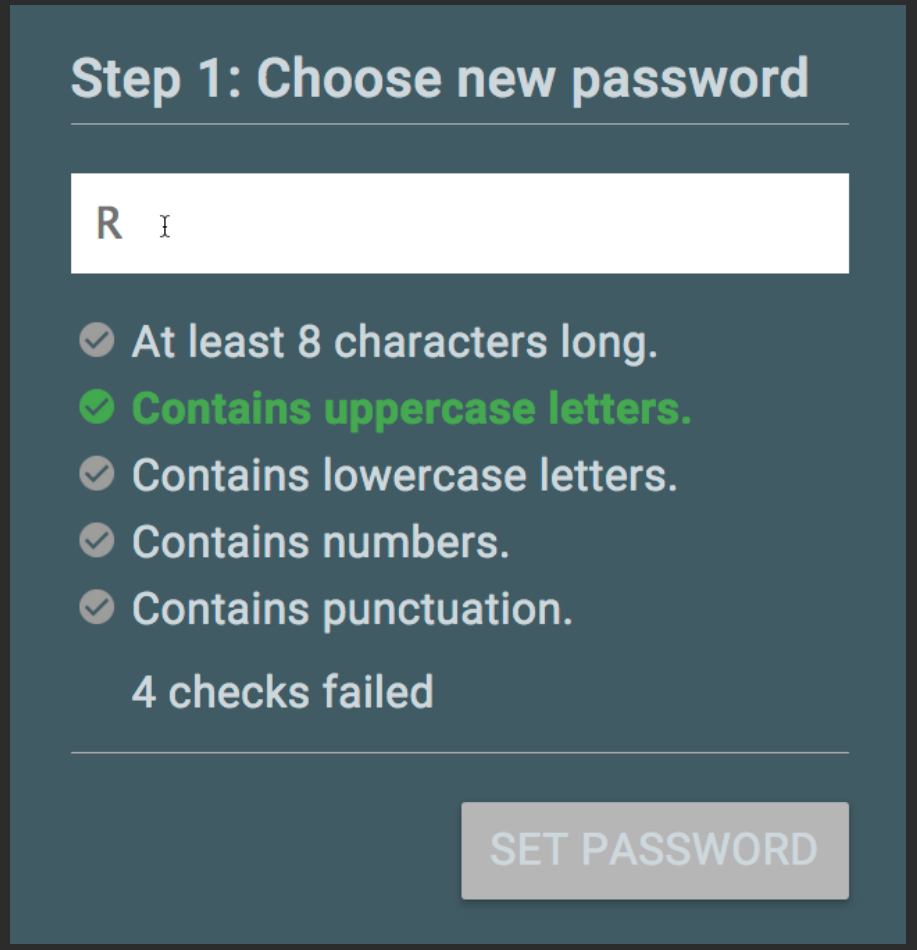
1. Event tritt ein



DOM UPDATES - BIG PICTURE

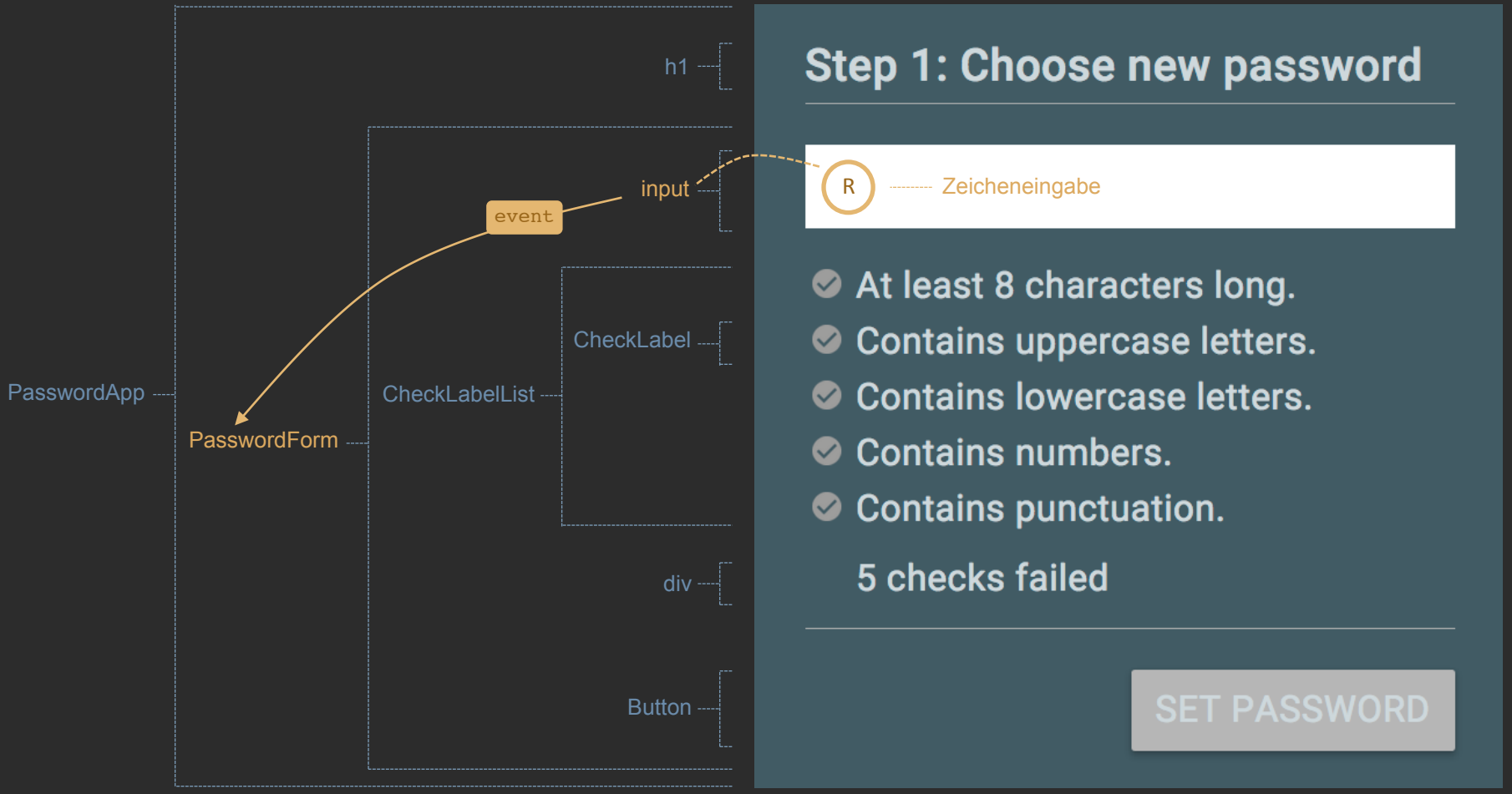
KOMMUNIKATION: PROPERTIES





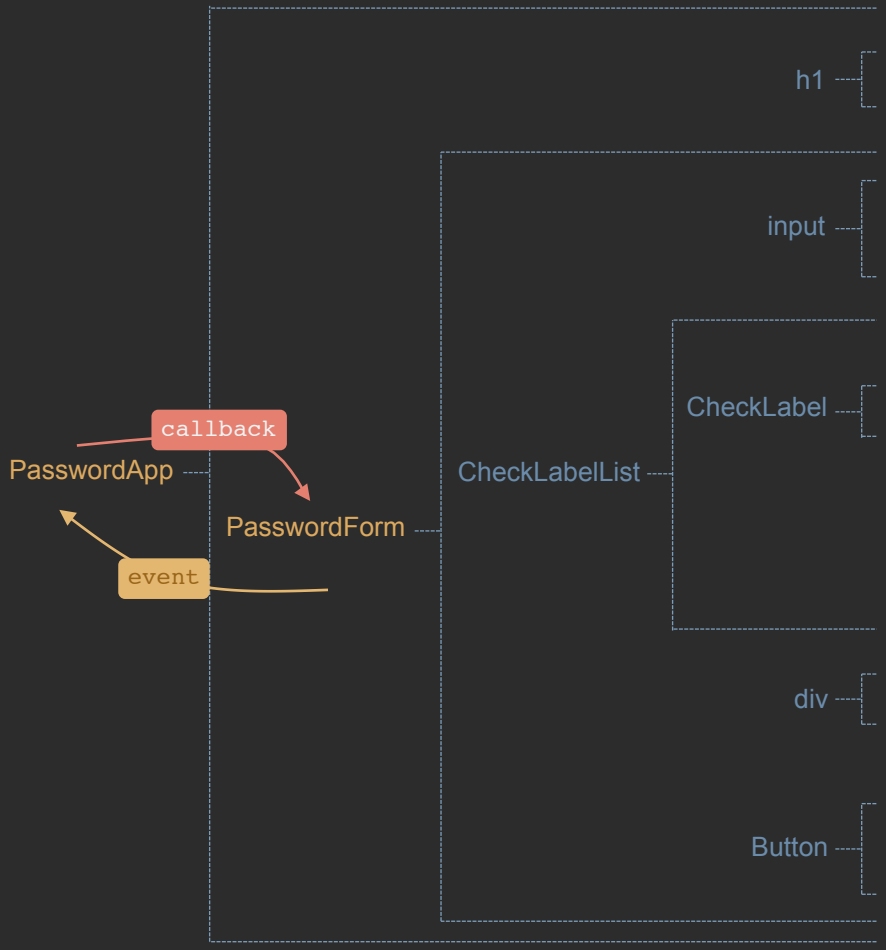
Von oben nach unten: **props**

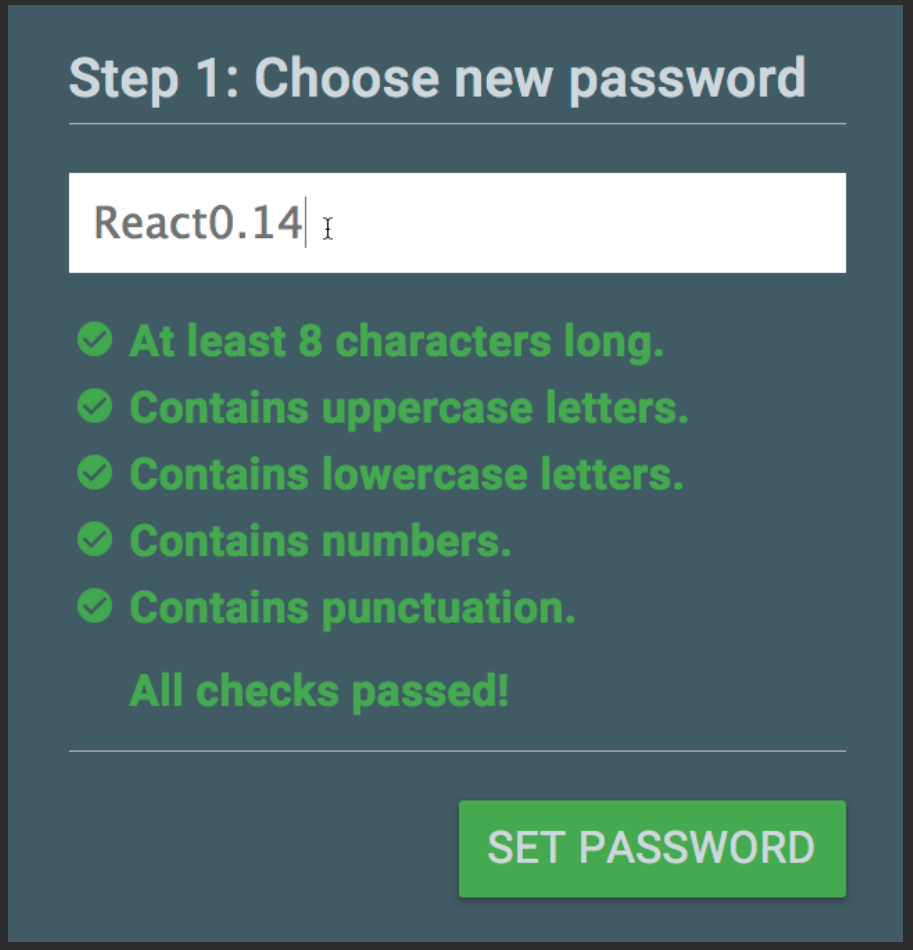
KOMMUNIKATION: EVENTS 1



Von unten nach oben: **events**

KOMMUNIKATION: EVENTS 2





Von unten nach oben: **events** und **callbacks**

KOMMUNIKATION: CALLBACK

```
class PasswordApp extends React.Component {
 onSetPassword(password) { . . . }

 render() {
  return . . .
   <PasswordForm . . .
     onSetPasswordHandler={p=>this.onSetPassword(p)}
   />;
 }
}


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

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

3. Event verarbeiten (ggf. Zustand setzen)

1. Callback übergeben

2. Callback aufrufen

UNIT-TESTS (OHNE DOM)

```
import TestUtils from 'react-addons-test-utils';

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

  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');
 });
});
```

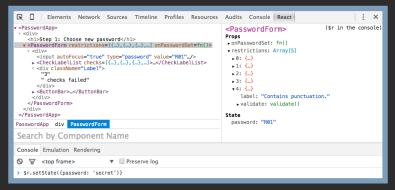
„Shallow rendering“

EINE „ANWENDUNG“ (BEISPIEL)

```
class App extends React.Component {
 handleItemSelected(item) {
  this.setState({component: item.component});
 }

 render() {
  return <div className='App'>
   <NavigationBar
     onItemSelected={item=>this.handleItemSelected(item)}
     items={[
      { label: 'Change password', component: <PasswordApp />},
      { label: 'Show weather',    component: <WeatherApp /> }
     ]}
    />

   <MainView>
    {this.state.component}
   </MainView>
  </div>;
 }
}
```

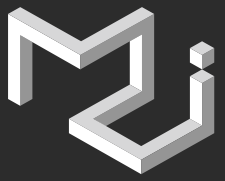
Navigation (Komponentenauswahl)

Ausgewählte Komponente einfügen



dev tools

material-design

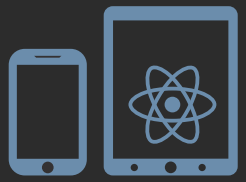


flux

bootstrap

graphql & relay

router



native

fertige Komponenten

Ökosystem

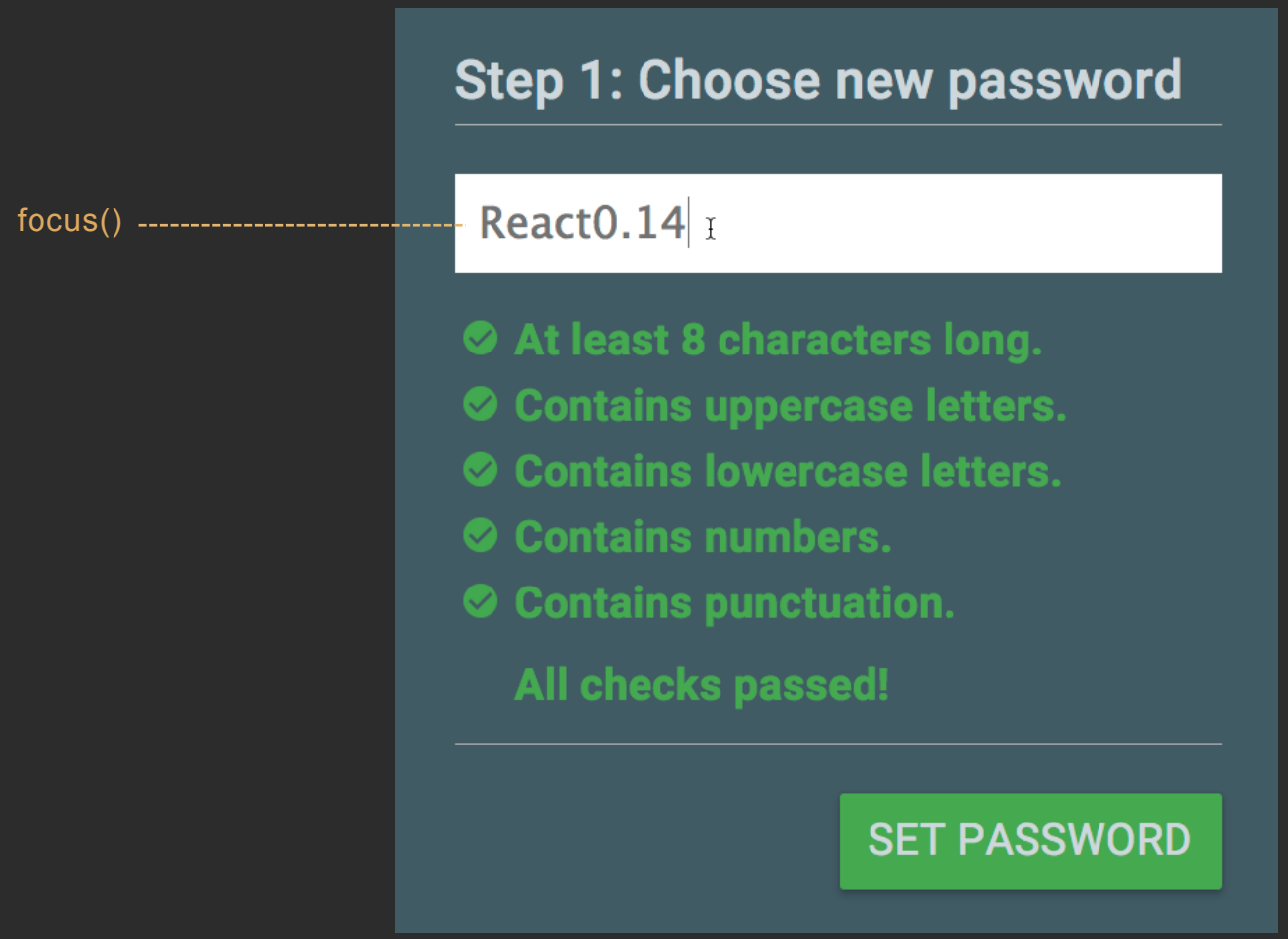
AUSBLICK

Mittagspause (wohlverdient!)

React Router
Serverzugriffe
Integration von Dritt-Bibliotheken
Build-Prozess

Zugabe

ZUGRIFF AUF NATIVEN DOM 1



Beispiel: `focus()` auf input-Feld aufrufen

ZUGRIFF AUF NATIVEN DOM 2

```
class PasswordForm extends React.Component {

  componentDidMount() {
    this.refs.passwordField.focus();
  }

  render() {
    const password = this.state.password;

    return <div>
      <input ref='passwordField' value={password} />
      . . .
    </div>;
  }
}
```

React-Callback:
Komponente wurde in den nativen DOM gehängt

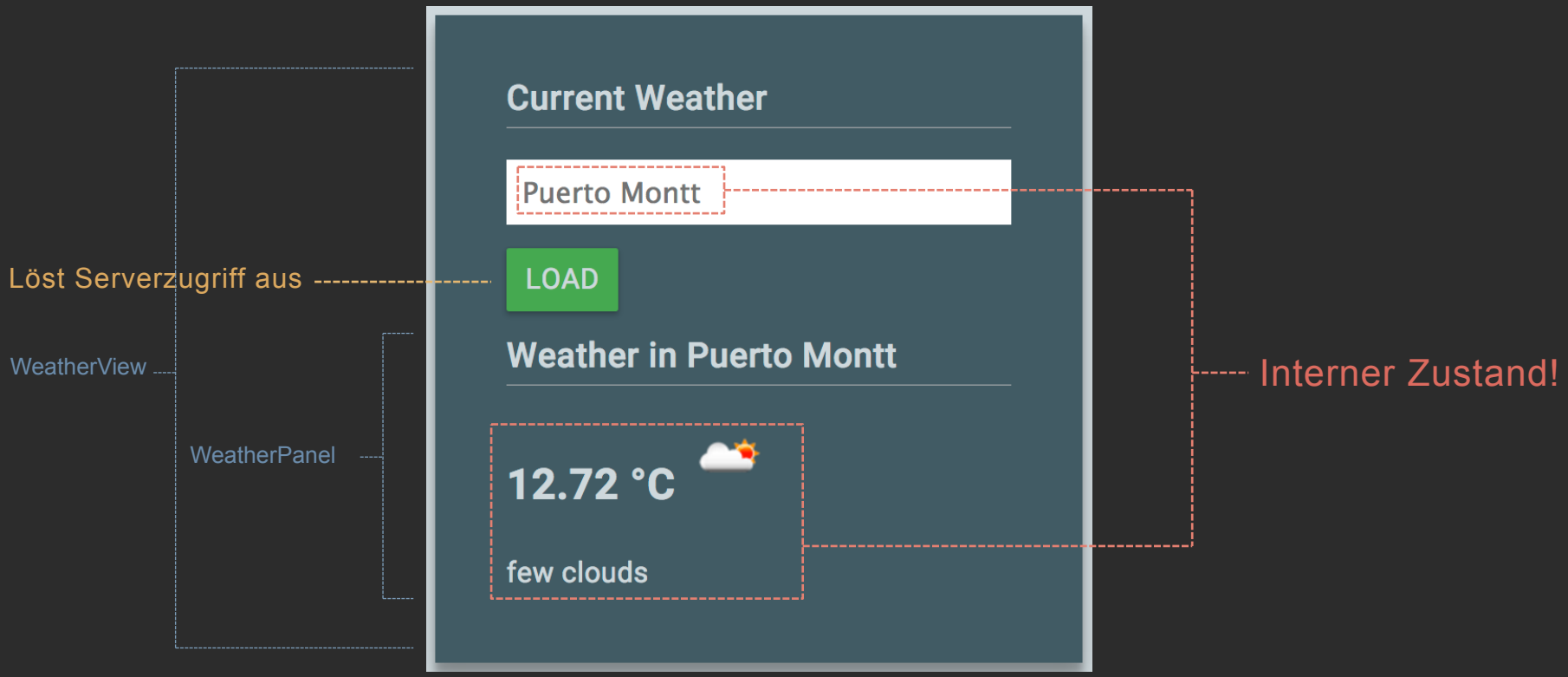
Natives DOM-Element

Virtuelles DOM-Element

Referenz anlegen

`this.refs` enthält native DOM-Elemente, die mit `ref` ausgezeichnet wurden

SERVERZUGRIFF



SERVERZUGRIFF 1

fetch-Bibliothek: https://fetch.spec.whatwg.org/

```
import WeatherPanel from ‘./WeatherPanel’;

class WeatherView extends React.Component {
 constructor() {
  super();
}

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

 render() {
  const { city, weather } = this.state;
  <input type=‘text’ value={city} onChange={. . .} />
  <Button label=‘Load’ onClick={() => this.fetchWeather()}
  <WeatherPanel weather={weather} />
 }
}
```

Daten vom Server laden

Zustand neu setzen (triggert Rendering)

Geladene Daten anzeigen

SERVERZUGRIFF 2

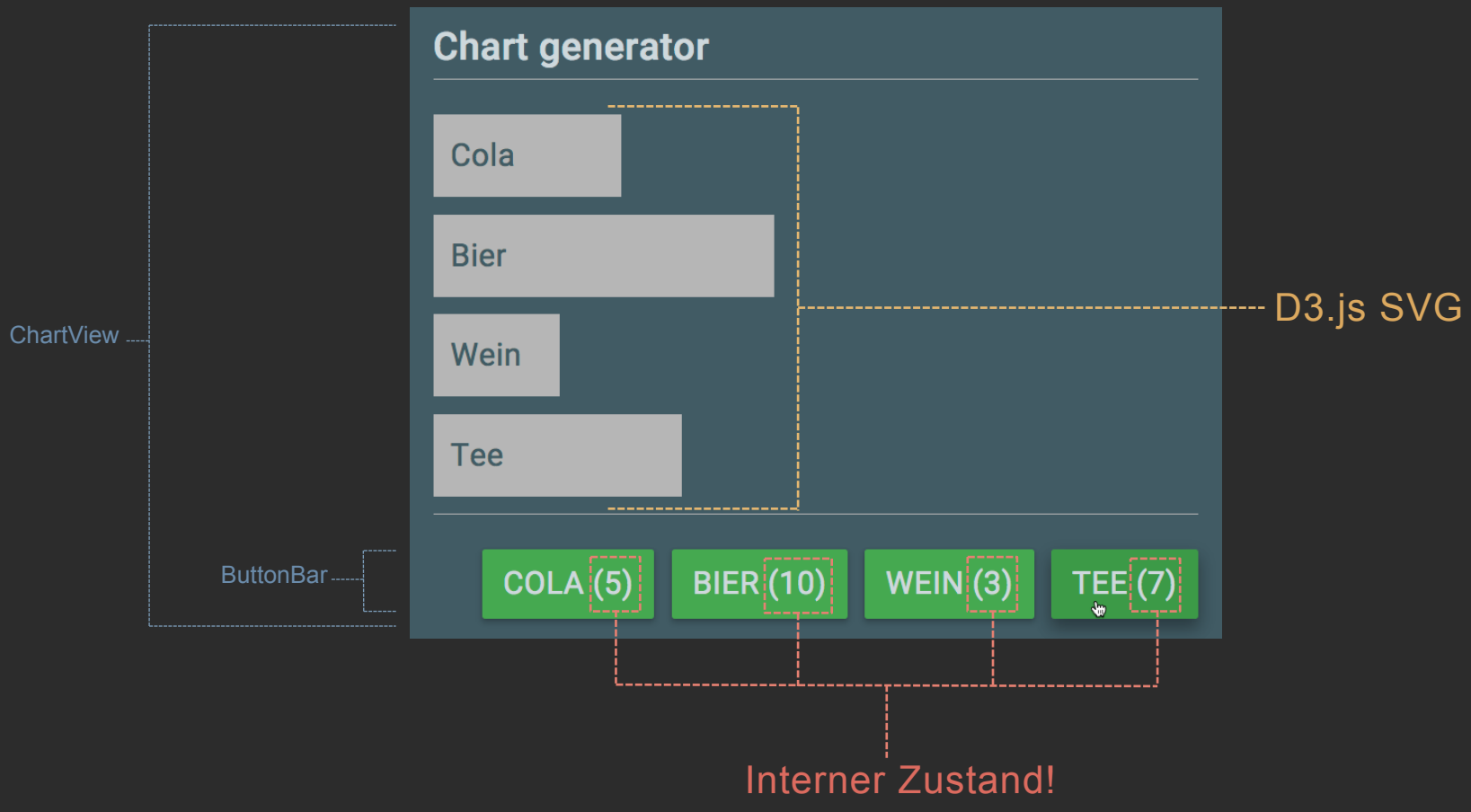
State initialisieren

Wetterdaten laden, sobald Komponente in DOM gehängt wurde

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

 componentDidMount() {
  this.fetchWeather();
 }

 fetchWeather() { . . . }

 render() { . . . }
}
```

INTEGRATION THIRD-PARTY-LIBS

Beispiel: D3.js



D3.JS

```
import d3 from 'd3';

class ChartView extends React.Component {
 constructor() { this.state = { . . .}; }

 increaseDrink(drink) { this.setState({ . . .}); }

 componentDidMount() { this.renderChart(); }
 componentDidUpdate() { this.renderChart(); }

 renderChart() {
  d3.select(this.refs.chart)
   .data(this.state.drinks)
   .enter().append(. . .);
 }
 render() {
  return . . .
   <div ref='chart'></div>
   <ButtonBar>
     { drinks.map(d => <Button
         label={ . . .}
         onClickHandler={()=>this.increaseDrink(d)} />)
     }
   </ButtonBar>
 }
}
```

Diagramm (neu) zeichnen, sobald Komponente in DOM gehängt bzw dort aktualisiert wurde

Natives DOM-Element

D3 Diagramm wird nicht in render() gezeichnet, weil hier kein natives DOM-Element

Viele ECMAScript 2015 Features

Compiler notwendig

Alias: ES6

ES2015

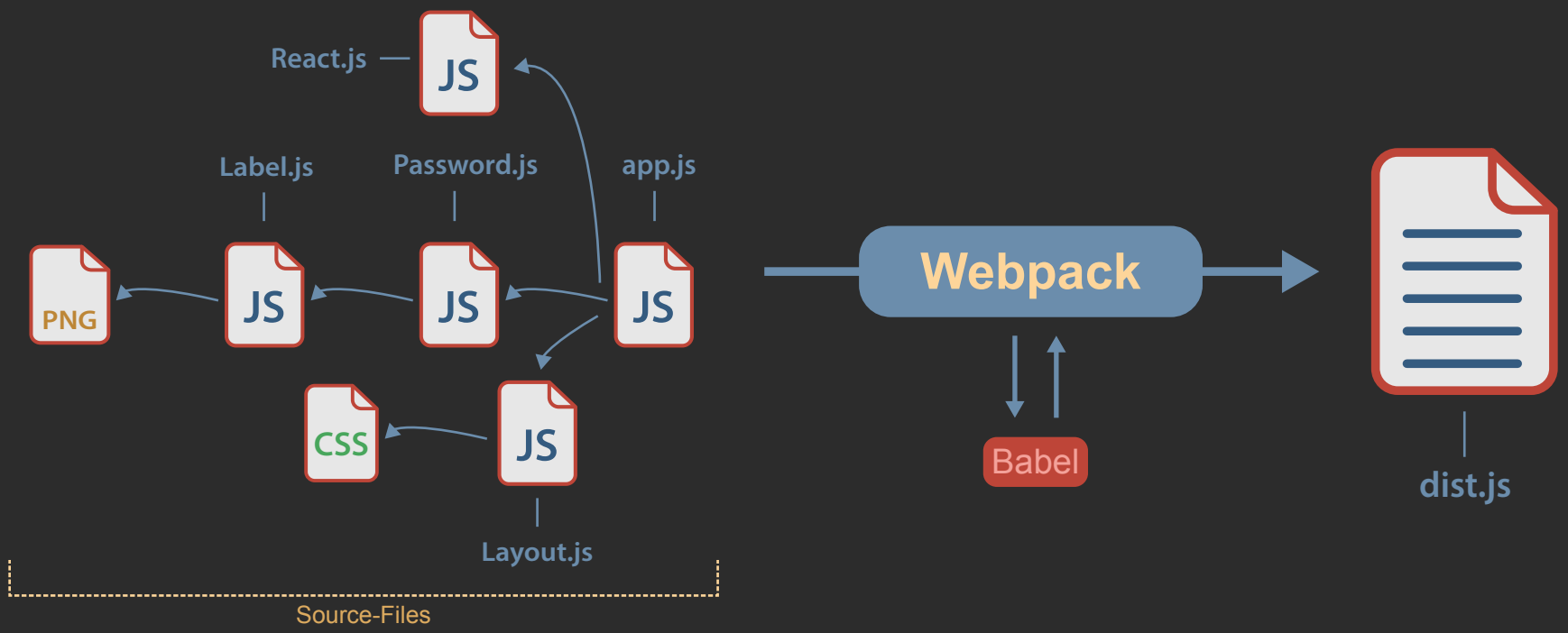
ES2015
„ES7“
...

ES5

Babel is a JavaScript compiler

http://babeljs.io

Babel

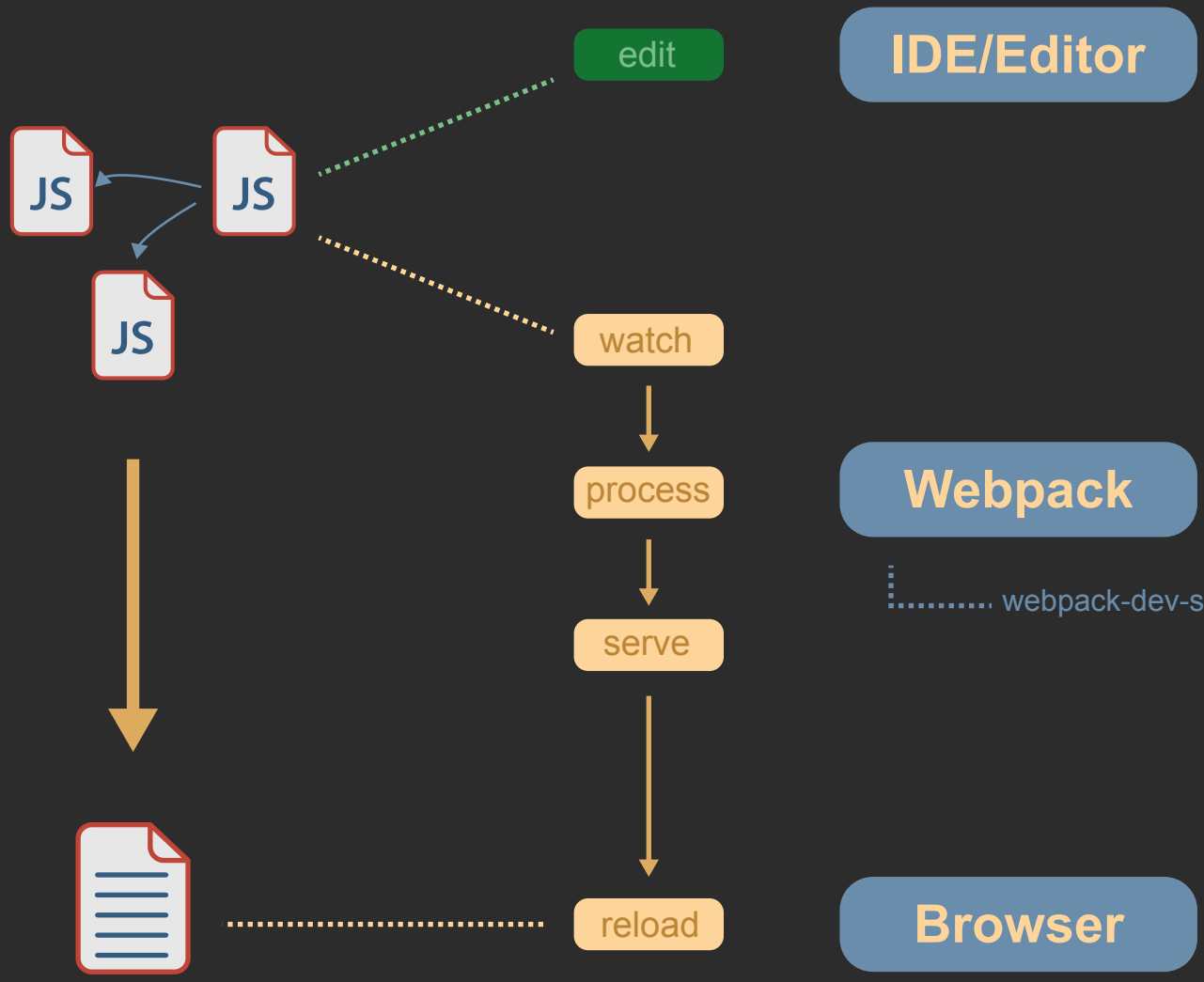


Module bundler - Erzeugt zentrales JavaScriptFile

Webpack

https://webpack.github.io

BUILDPROZESS



webpack-dev-server

BUILDPROZESS

Vielen Dank!

Fragen?

NILS@NILSHARTMANN.NET