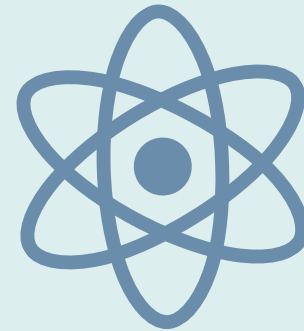


EINSTIEG IN



React

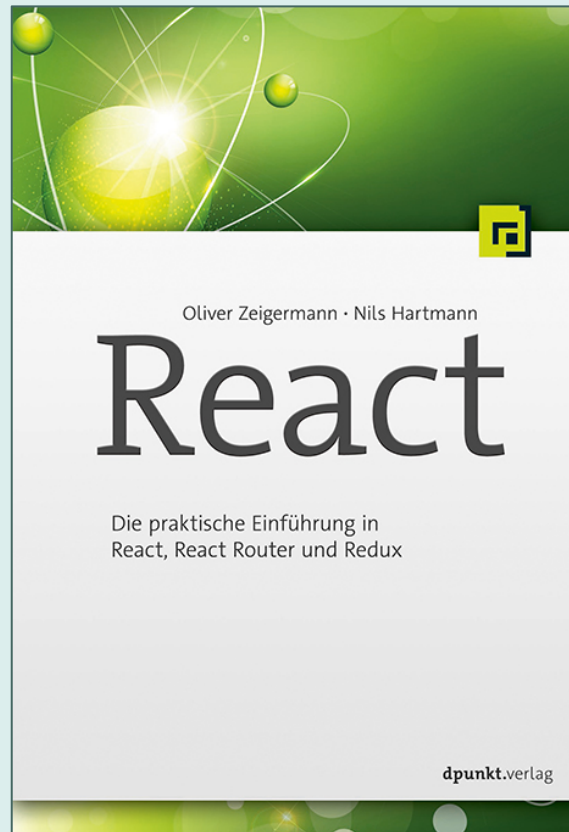
OLIVER ZEIGERMANN & NILS HARTMANN | ENTERJS | JUNI 2016

NILS HARTMANN

@NILSHARTMANN

OLIVER ZEIGERMANN

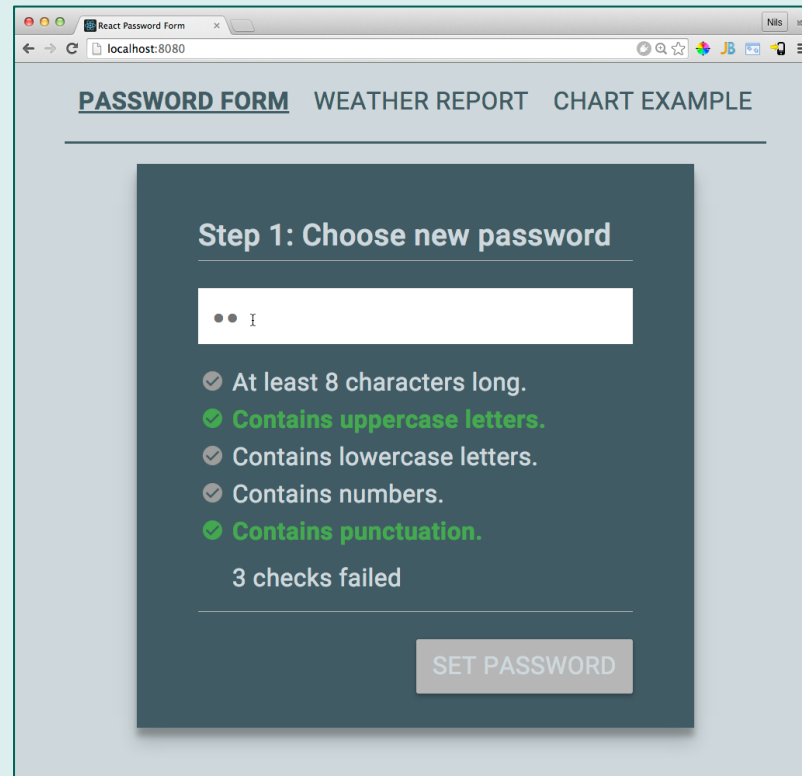
@DJCORDHOSE



[HTTP://REACTBUCH.DE](http://reactbuch.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>
```

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>
```

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

✓ At least 8 characters long.

✓ At least 8 characters long.

✓ Contains uppercase letters.

REACT! 

✓ 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 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: ALS FUNKTION

Komponente CheckLabel

✓ At least 8 characters long.

Komponentenfunktion

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

JSX

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.

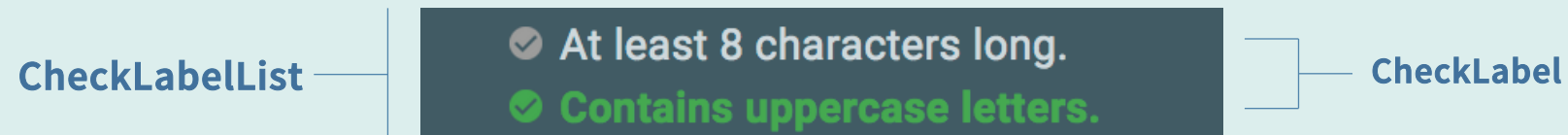
```
{  
  checked: false,  
  label: 'At least 8 characters long.'  
}
```



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

KOMPONENTEN 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 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

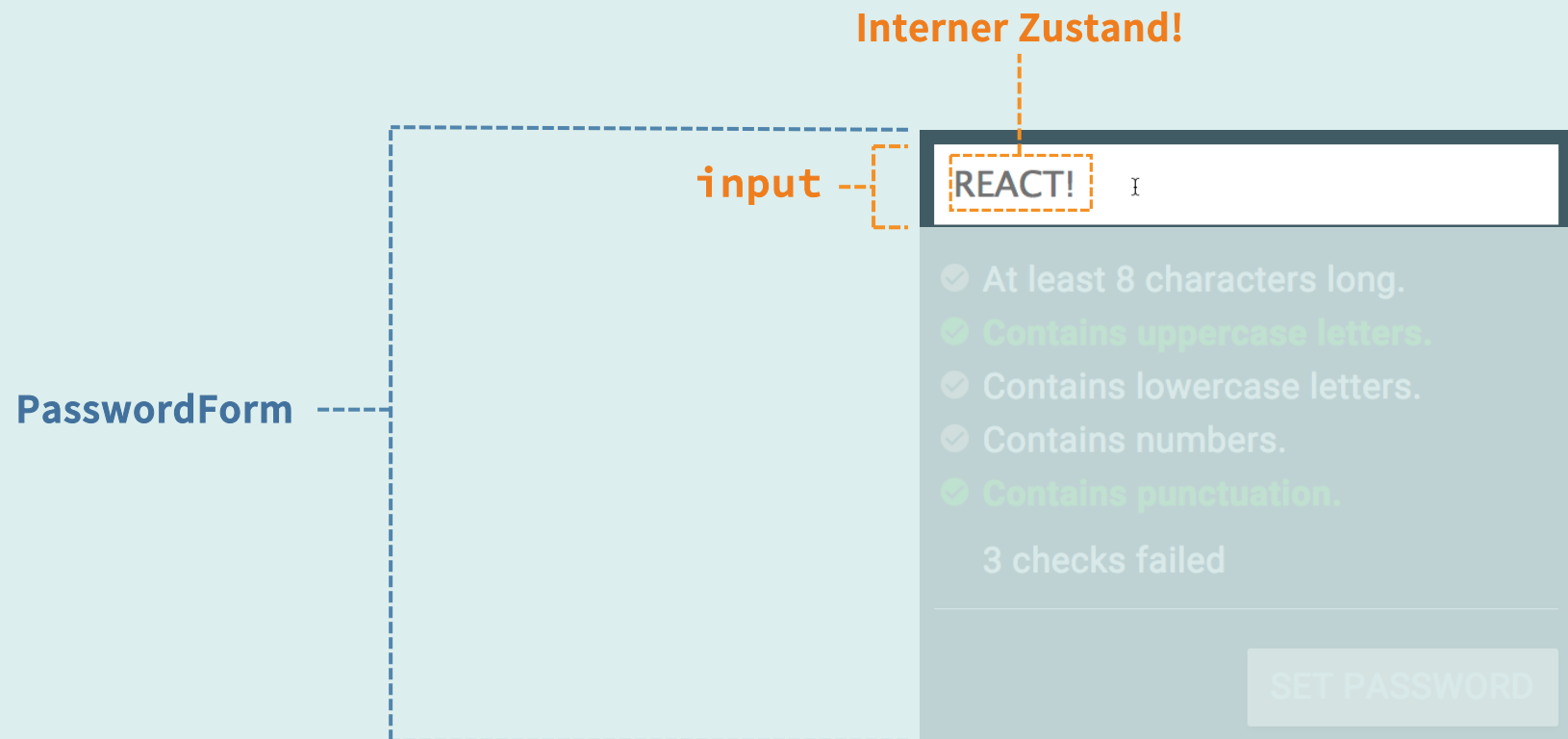
Zustand („state“): Komponenten-intern

- Beispiel: Inhalt von Eingabefeld, Antwort vom Server
- Objekt mit Key-Value-Paaren
- 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



1. Input mit Wert aus State befüllen

2. Event Listener

```
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});
  }
}
```

ZUSTAND: EINGABEFELD



1. Input mit Wert aus State befüllen

2. Event Listener

3. Zustand neu setzen

```
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});  
  }  
}
```

Diagram illustrating the state management flow:

- An **Event** (represented by a box) triggers the `onPasswordChange` method.
- The `onPasswordChange` method calls `this.setState({password: newPassword});`.
- This leads to **Neu rendern** (New render), which triggers a re-render of the component.
- The re-render updates the `value` prop of the `<input>` component in the `render()` method.

ZUSTAND & RENDERING

Beispiel: Password Formular

The diagram illustrates a password form with a text input field containing the text "REACT!". Below the input field are five validation rules, each preceded by a checkmark icon. The first rule, "At least 8 characters long.", has a grey checkmark. The second rule, "Contains uppercase letters.", has a green checkmark. The third rule, "Contains lowercase letters.", has a grey checkmark. The fourth rule, "Contains numbers.", has a grey checkmark. The fifth rule, "Contains punctuation.", has a green checkmark. Below the 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., whether a character is uppercase) influences the rendering of these elements.

REACT! |

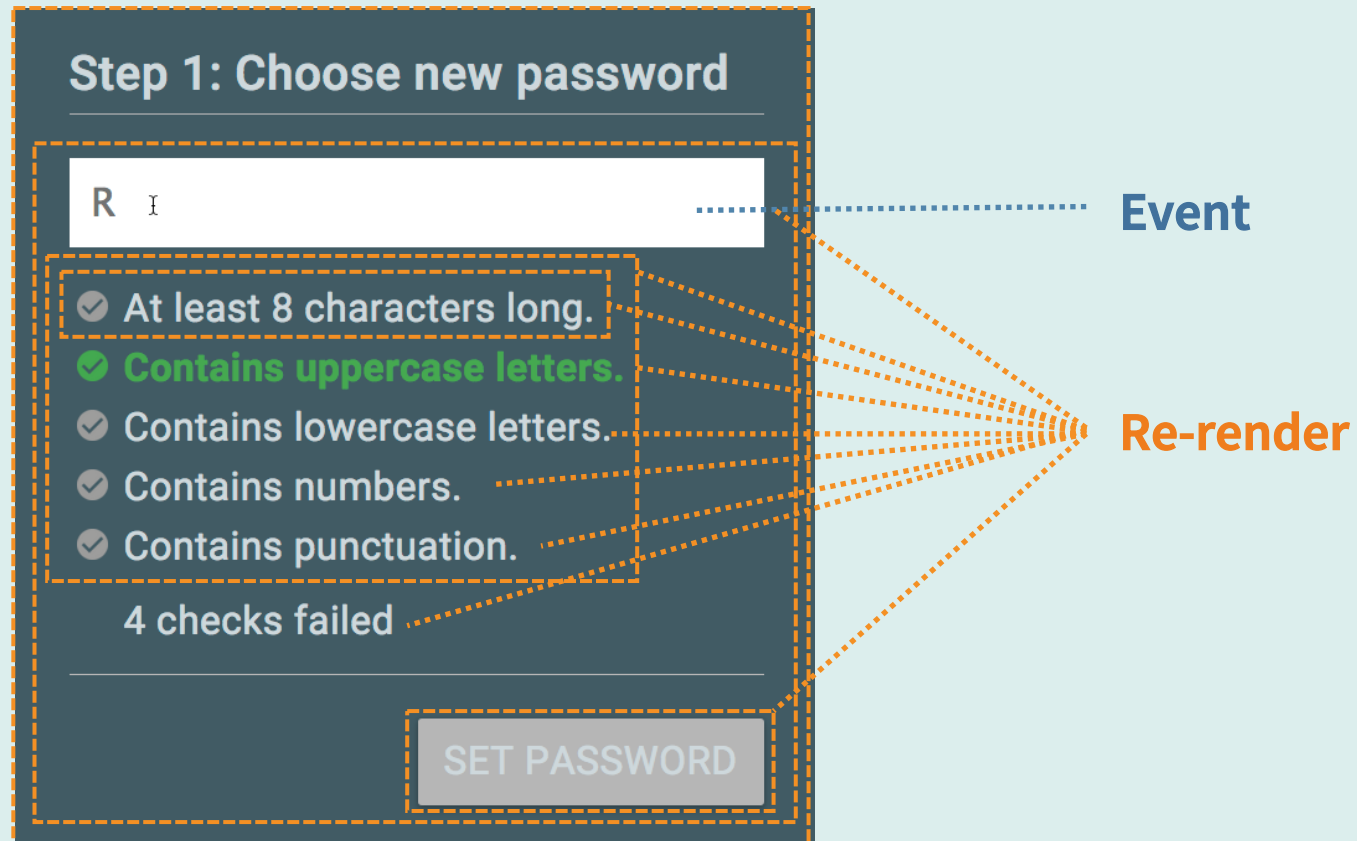
- ✓ At least 8 characters long.
- ✓ Contains uppercase letters.
- ✓ Contains lowercase letters.
- ✓ Contains numbers.
- ✓ Contains punctuation.

3 checks failed

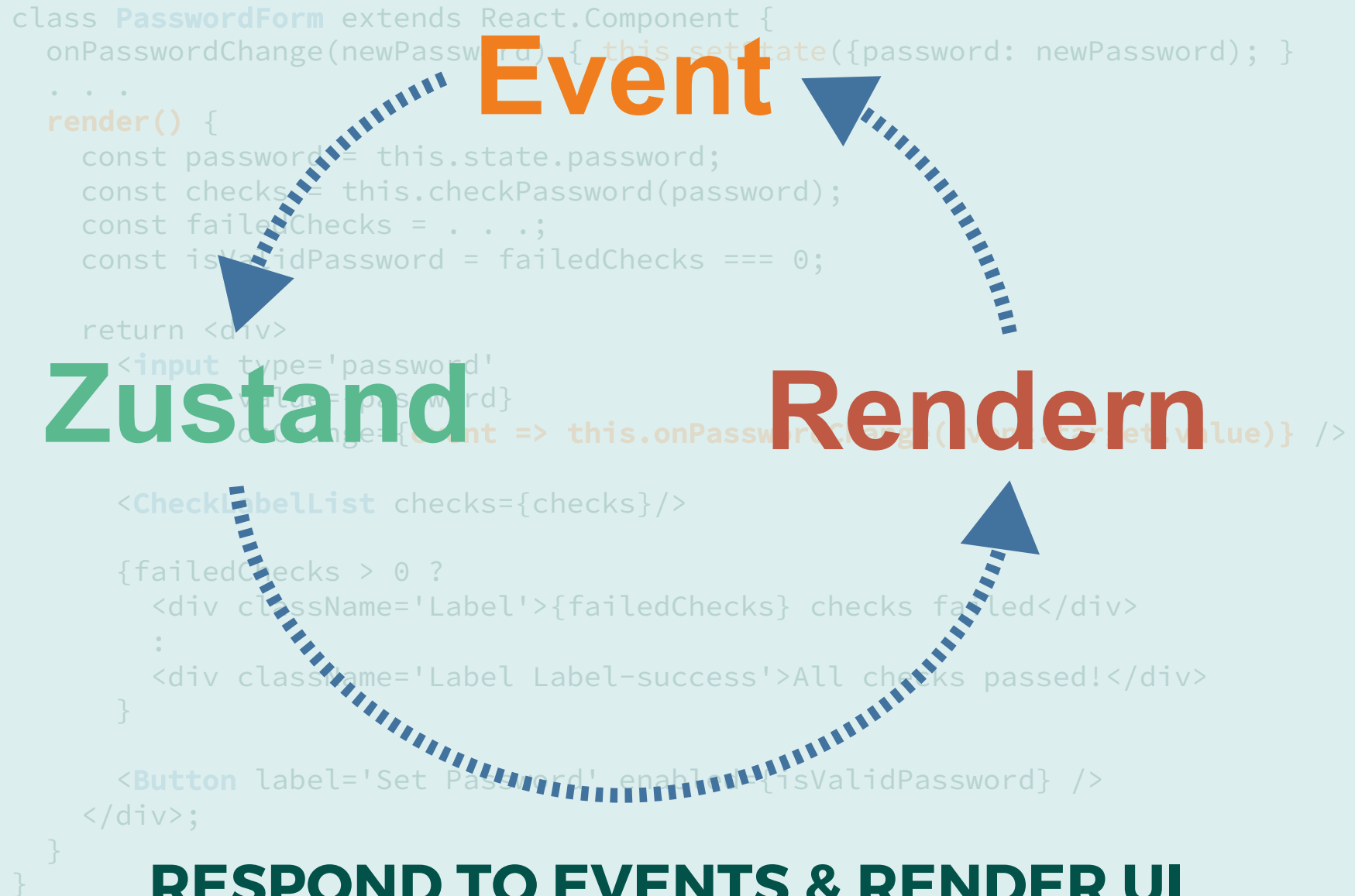
SET PASSWORD

beeinflusst

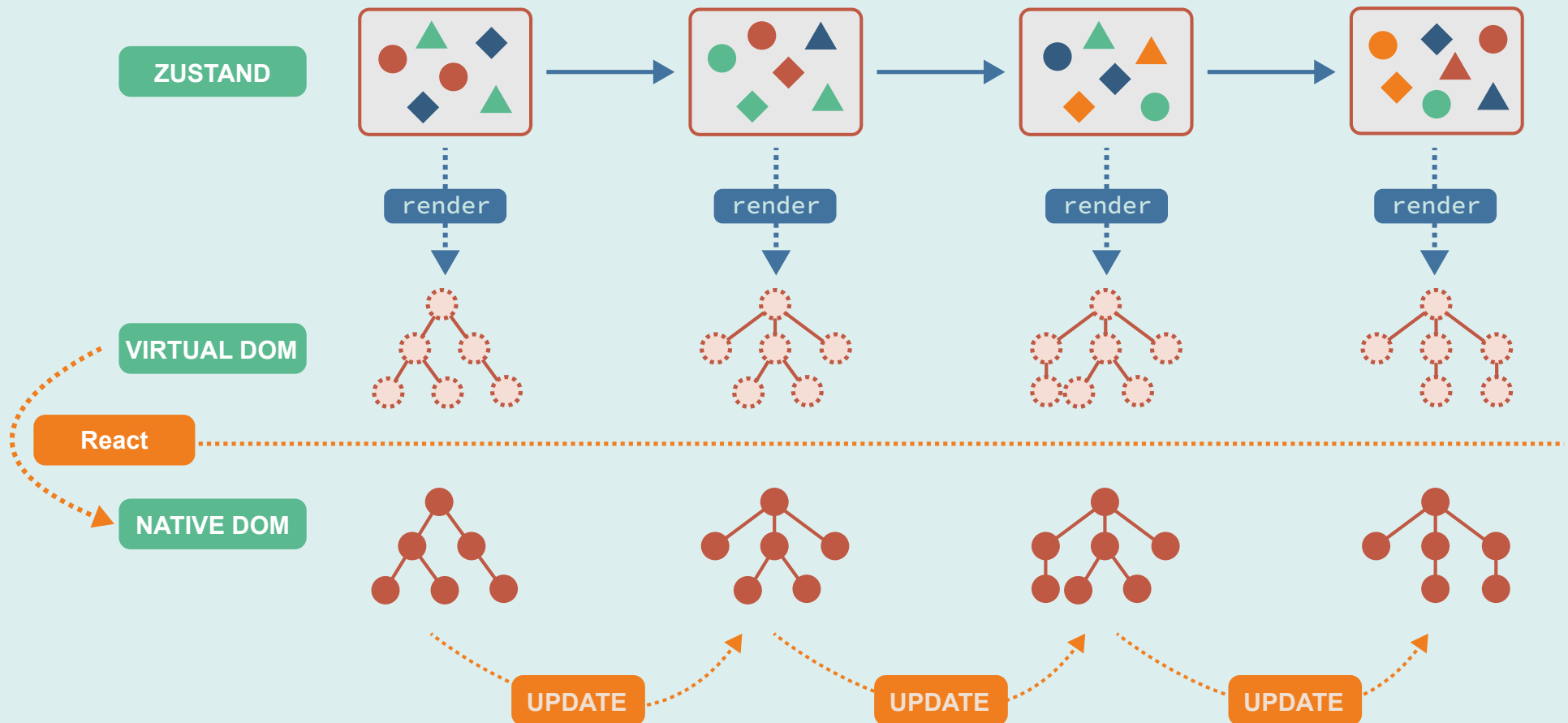
GANZ EINFACH: ALLES RENDERN



REACT: UNI DIRECTIONAL DATAFLOW



HINTERGRUND: VIRTUAL DOM



HINTERGRUND: VIRTUAL DOM

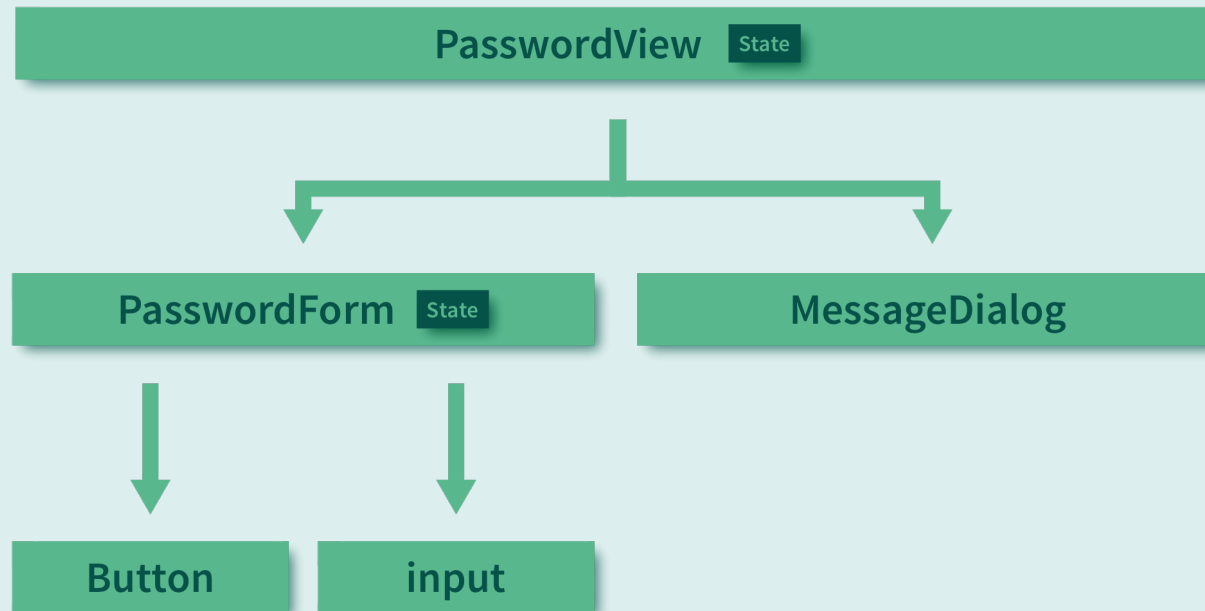
Virtual DOM

- 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

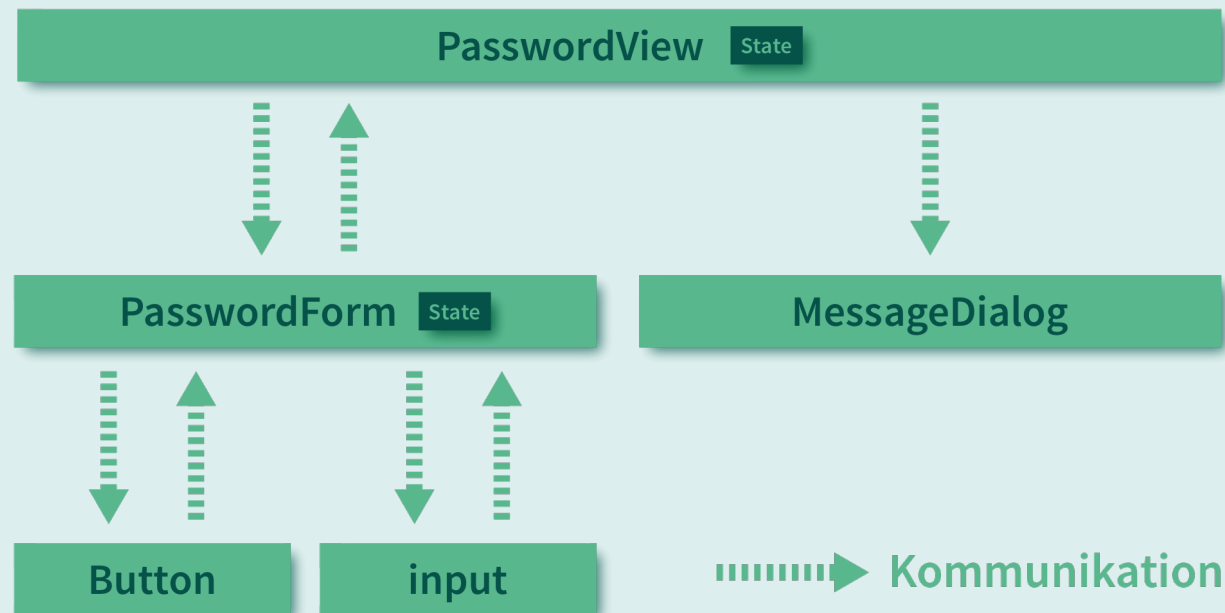
KOMPONENTENHIERARCHIEN



Typische React Anwendungen: Hierarchisch aufgebaut

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

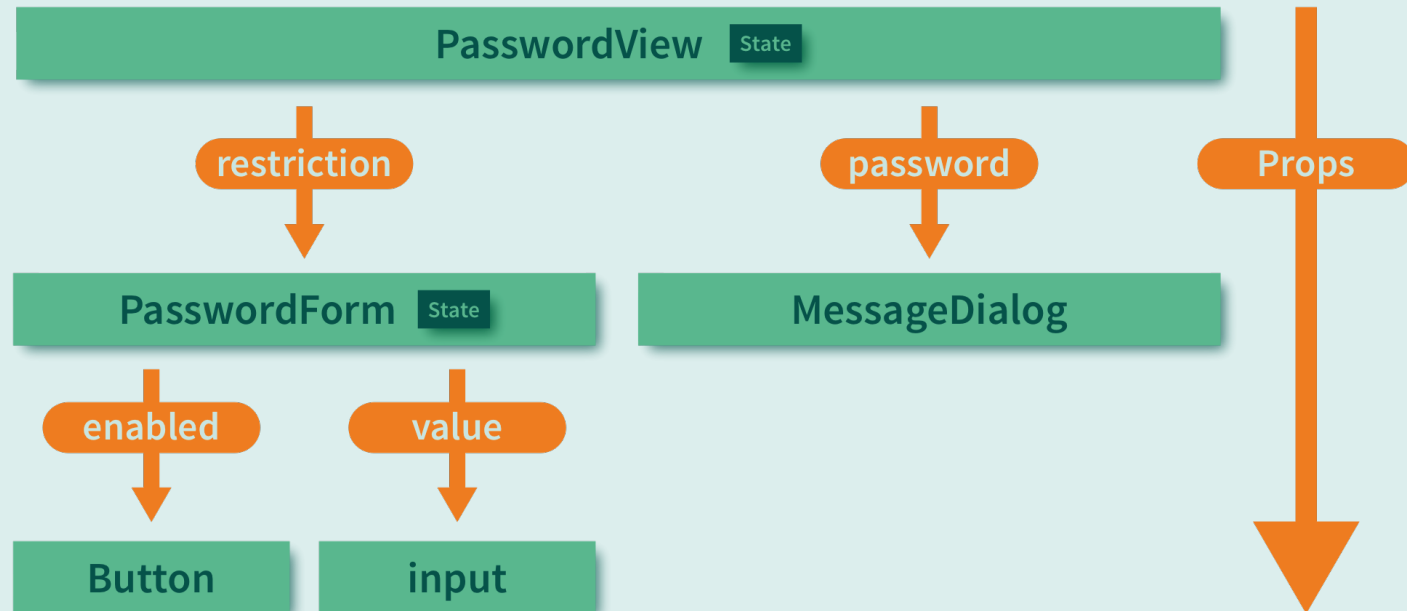
KOMMUNIKATION ZWISCHEN KOMPONENTEN



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?

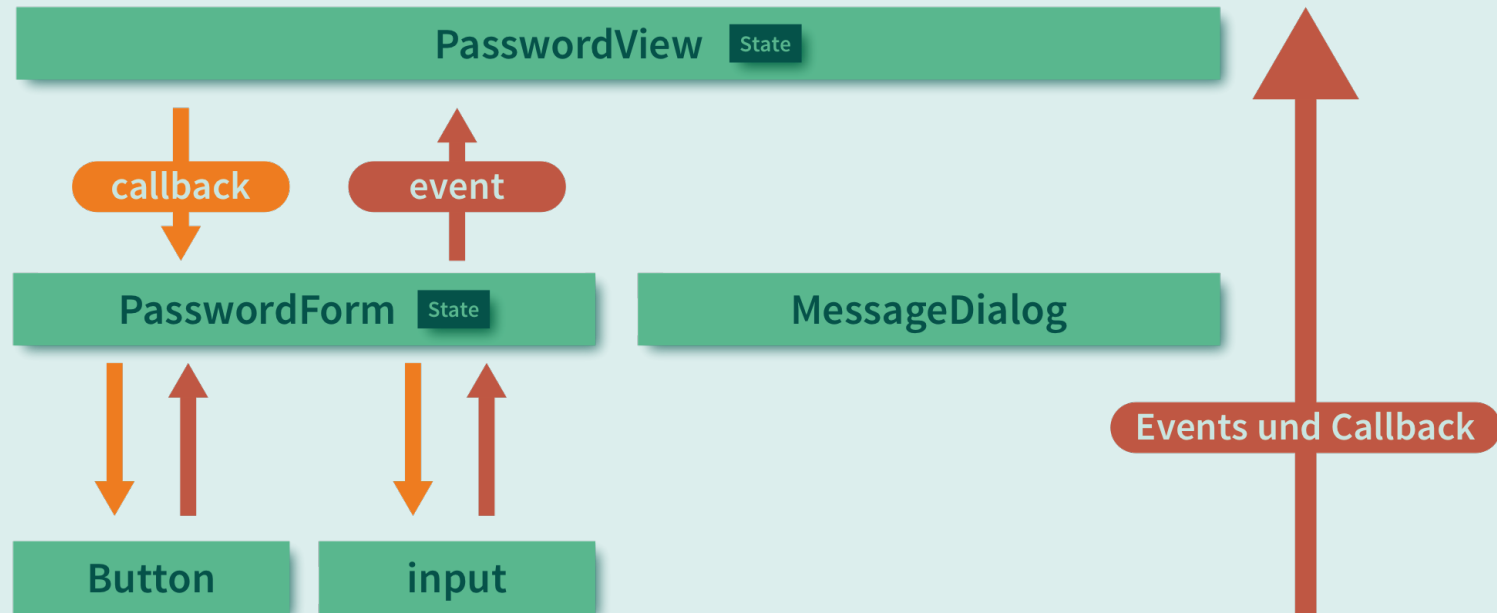
KOMMUNIKATION: PROPERTIES



Von oben nach unten: **Properties**

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

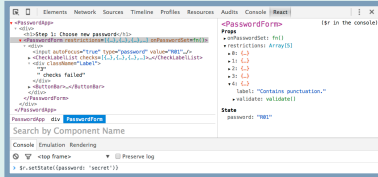
KOMMUNIKATION: EVENTS



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

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

ÖKOSYSTEM



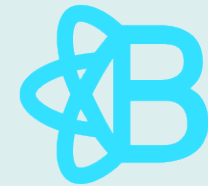
Developer Tools

material-design



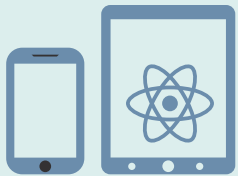
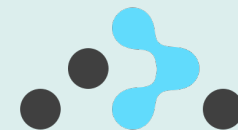
Flux Architekturpattern

Bootstrap



GraphQL & Relay

React Router



React Native

Fertige Komponenten



ZUSAMMENFASSUNG

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!

Fragen?

@NILSHARTMANN | @DJCORDHOSE