HOCHSCHULE HAMM-LIPPSTADT

Softwareprojekt 2

Einstieg in React

Prof. Dr. Darius Schippritt
Büro L4.2-E02-140
darius.schippritt@hshl.de

Überblick

- **Das letzte Mal… und Lernziele**
- Einstieg in React
- Zum Schluss…

Das letzte Mal...

express

static

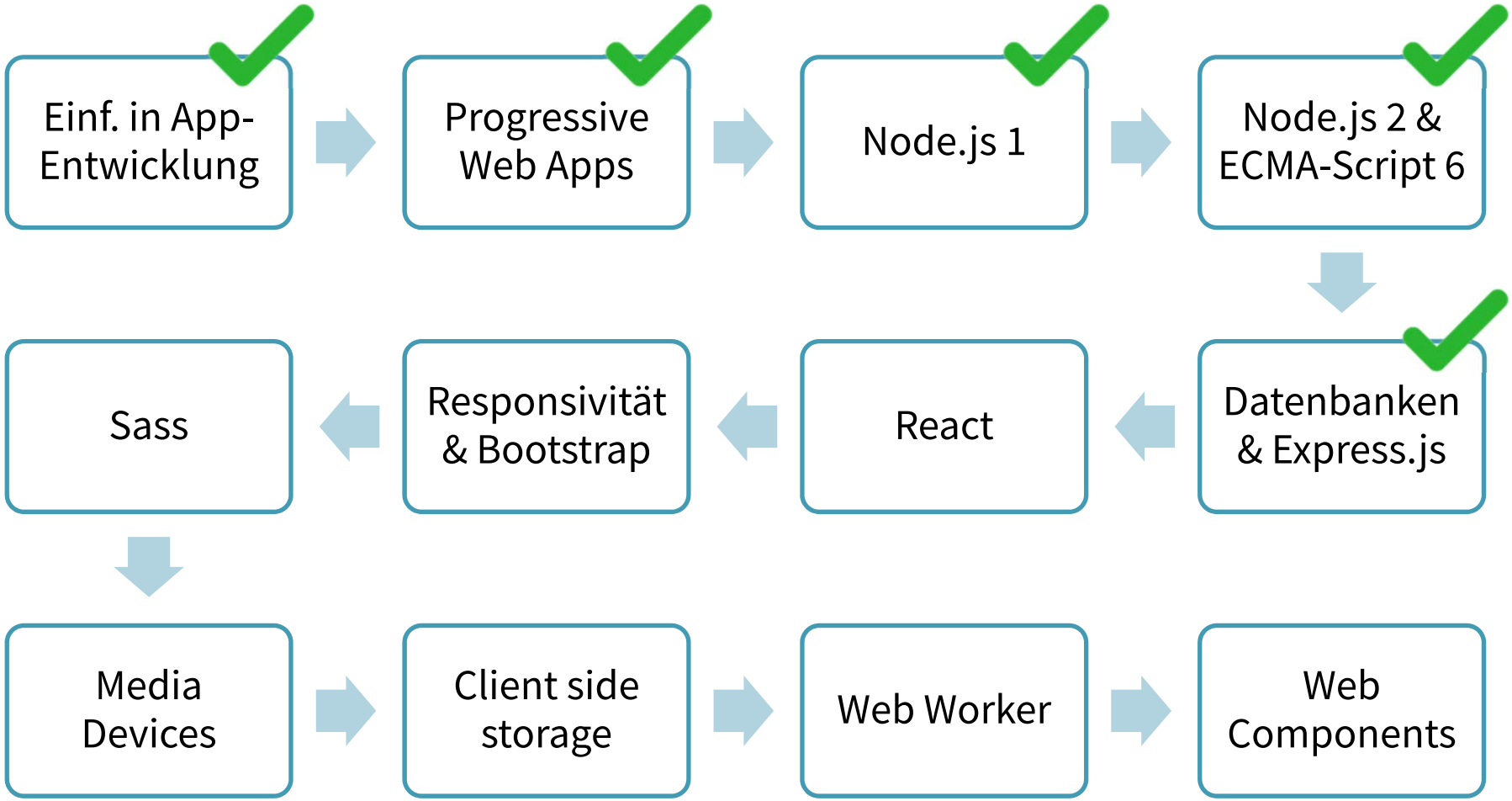
GET und POST

Upload

Cookies

REST

Ziele und Inhalte



HOCHSCHULE HAMM-LIPPSTADT

Lernziele

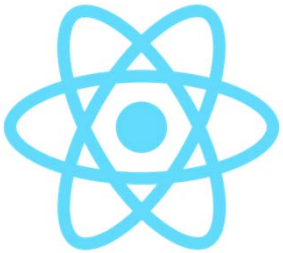
- Sie können ein React-Projekt unter Node.js erstellen und starten.
- Sie können JSX-Quellcode schreiben und unter React auf DOM-Elemente zugreifen.
- Sie können erläutern, wie das Routing in React funktioniert.
- Sie können auf State und Props zugreifen.
- Sie können erklären, wie das Eventhandling funktioniert und können dieses in einem Projekt anwenden.

Überblick

- Das letzte Mal… und Lernziele
- **Einstieg in React**
- Zum Schluss…

Einstieg in React

Überblick



- React (ReactJS) ist eine JavaScript-Bibliothek, welches primär für die Entwicklung von Single Page Applications entwickelt wurde
- Aktuelle Version 16 vom 18.09.2018
- Ursprünglich von Facebook entwickelt
- Ist unter der MIT-Lizenz frei verwendbar
- Unidirektionaler Datenfluss und Virtual DOM erlauben performante Anwendungen

Einstieg in React

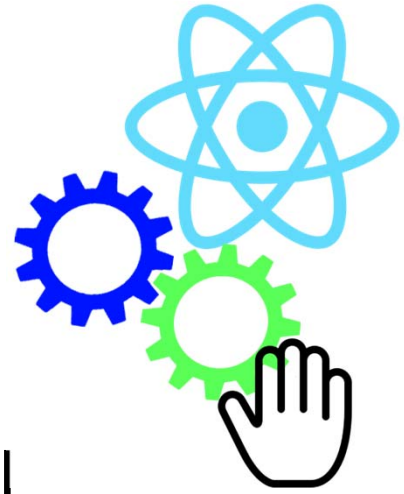
Überblick

- Virtual DOM (JavaScript-Objekt) sorgt dafür, dass nur geänderte DOM-Teile aktualisiert werden
- Eigene Template-Sprache JSX (JavaScript XML) erlaubt Kapselung von Web-Komponenten
- Häufig mit Node.js verwendet
- Aber auch Integration mit PHP möglich

vs.

Einstieg in React

Installation und Anlage eines Projektes



- React-Projekt kann manuell oder „automatisch“ erstellt werden
- Manuelle Anlage verwendet Erweiterungen Webpack und Babel
- Installation von React, Webpack und Babel
- Anpassung der Projektkonfiguration
- Detaillierte Anleitung z.B. unter https://www.tutorialspoint.com/reactjs/reactjs_environment_setup.htm verfügbar

Einstieg in React

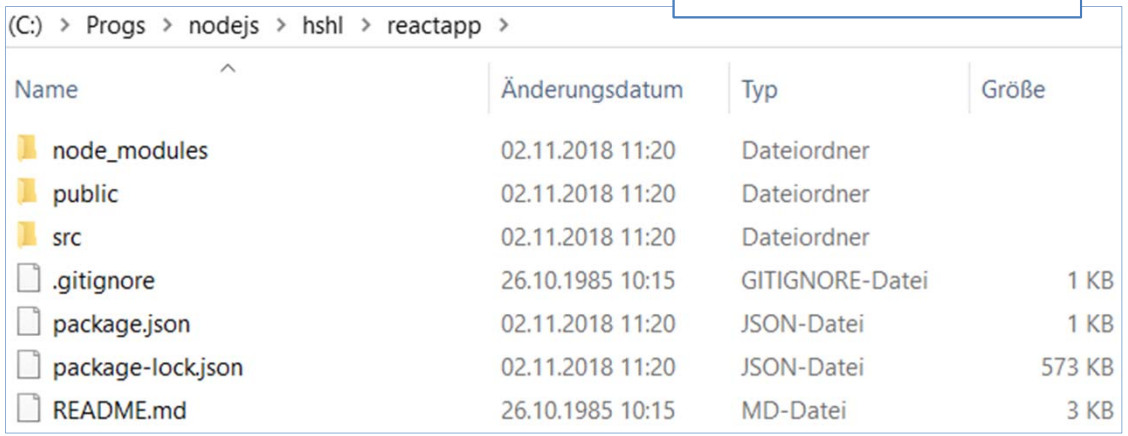
Installation und Anlage eines Projektes

- Installation des React-Moduls und Erstellung einer React-App

```
npx create-react-app reactapp
```

- Installation aller Module
- komplette Verzeichnisstruktur inkl. Dateien und Ressourcen wird angelegt

Größe ca. 210 MB



Einstieg in React

Installation und Anlage eines Projektes

```
c:\Progs\nodejs\hshl>npx create-react-app reactapp
npx: installed 63 in 3.153s

Creating a new React app in c:\Progs\nodejs\hshl\reactapp.

Installing packages. This might take a couple of minutes.
Installing react, react-dom, and react-scripts...

+ react-dom@16.6.0
+ react@16.6.0
+ react-scripts@2.1.1
added 1716 packages from 661 contributors and audited 35649 packages in 69.04s
found 0 vulnerabilities


Success! Created reactapp at c:\Progs\nodejs\hshl\reactapp
Inside that directory, you can run several commands:

  npm start
    Starts the development server.

  npm run build
    Bundles the app into static files for production.

  npm test
    Starts the test runner.

  npm run eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can't go back!

We suggest that you begin by typing:

  cd reactapp
  npm start

Happy hacking!
```

Einstieg in React

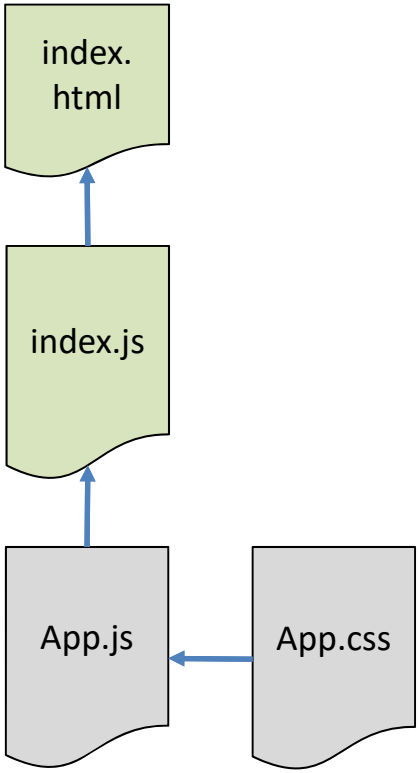
Verzeichnisstruktur des React-Projektes

```
- public
    - favicon.ico
    - index.html
    - manifest.json
- src
    - App.css
    - App.js
    - App.test.js
    - index.css
    - index.js
    - logo.svg
    - serviceWorker.js
```

- **index.html** → Startseite
- **App.js** → erste Komponente
- **index.css** → globale CSS-Definitionen
- **favicon.ico** → Bookmark-Icon
- **index.js** → Startskript; importiert notwendige Ressourcen und Komponenten
- **serviceWorker.js** → Service Worker stellt „Offline-Funktion“ zur Verfügung

Einstieg in React

Zusammenhang zwischen den Dateien



index.htm

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1,
        shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json">
    <title>React App</title>
  </head>
  <body>
    <noscript>
      You need t
    </noscript>
    <div id="root"></div>
  </body>
</html>
```

„Platzhalter“ für Inhalt aus ./app/app.component.ts

index.js

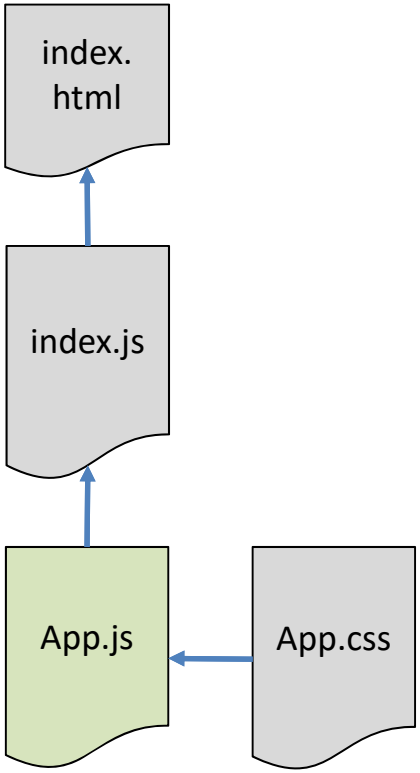
```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

serviceWorker.unregister();
```

Einstieg in React

Zusammenhang zwischen den Dateien



App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <p>
            Edit <code>src/App.js</code> and save to reload.
          </p>
          <a
            className="App-link"
            href="https://reactjs.org
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React
          </a>
        </header>
      </div>
    );
  }
}

export default App;
```

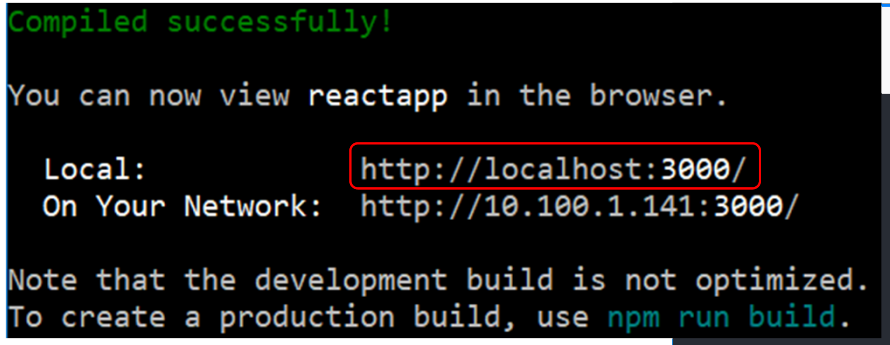
JSX muss ein übergeordnetes Element besitzen

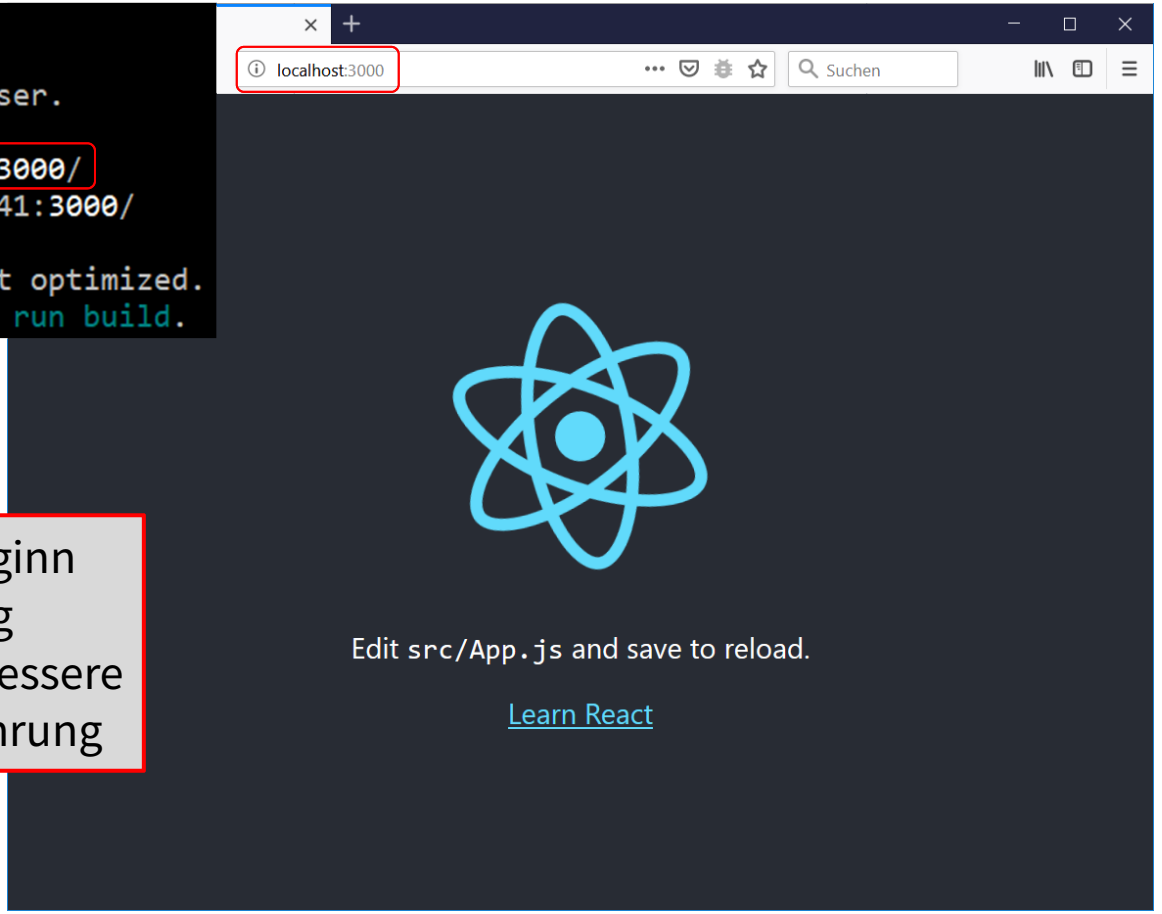
JavaScript-Ausdrücke {} werden beim Kompilieren interpretiert

Einstieg in React

Start des Projektes

- Start der App per **`npm start`**





Ressourcen werden zu Beginn und bei jeder Änderung automatisch kompiliert → bessere Performance bei der Ausführung

Einstieg in React

JSX: JavaScript-Ausdrücke

- JavaScript-Ausdrücke werden in **{}** gekappselt
- Werden beim Kompilieren interpretiert
- Ermöglichen z.B. Rechnen und bedingte Anweisungen

```
class App extends Component {
    render() {

        var myVar = 2;

        return (
            <div className="App">
                3 * 8 = {3*8} <br></br>

                {myVar > 1 ? 'myVar > 1' : 'myVar <= 1'}
            </div>
        );
    }
}
```

3 * 8 = 24

In XML müssen ALLE Tags geschlossen werden!

myVar > 1

Einstieg in React

JSX: inline CSS-Angaben

- CSS-Anweisungen können inline als Variablen angegeben werden
- Definition einer Variablen → CSS-Eigenschaften in camelCase-Schreibweise

```
var myCSS = {color: '#00FFAA', backgroundColor: 'blue'};
```

```
<p style={myCSS}>
```

- Kommentare werden wie JavaScript-Anweisungen in **{ }** gekappselt

```
{/*Mehrzeiliger Kommentar*/}
```

Einstieg in React

Zugriff auf DOM

- Auswahl der DOM-Nodes über bekannte Selektor-Funktionen **`document.getElement…`** und **`document.querySelector()`**
- Manipulation des Virtual DOM erfordert aber React-Funktion **`.findDOMNode()`**
- Import von ‘react-dom‘

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

```
changeColor(selector, color) {
    var elem = document.querySelector(selector);
    ReactDOM.findDOMNode(elem).style.color = color;
}
```

```
<button onClick={() => this.changeColor("#test", "blue")}>
    change font color</button>
```

Einstieg in React

Erstellung neuer Komponenten

- Auslagerung des Headers in eigene Komponente
- Erstellung von Header.js

```
import React, { Component } from 'react';

class Header extends Component {
    render() {
        return (
            ...
        );
    }
}

export default Header;
```

- Import und Verwendung von Header in App.js

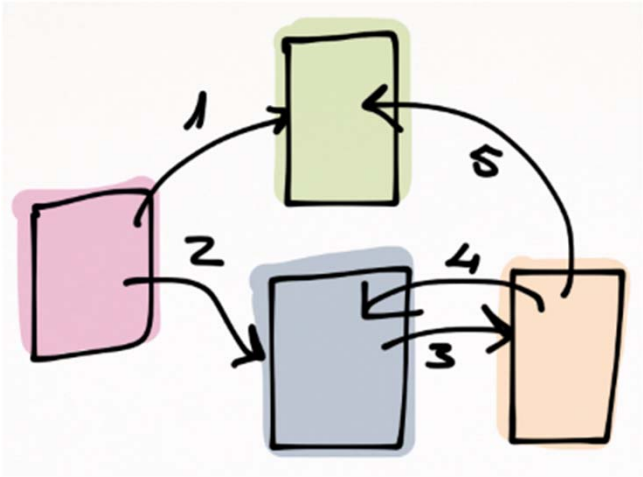
```
import Header from './Header.js';
```

```
return (
    <div className="App">
        <Header />
        ...
    </div>
);
```

Einstieg in React

Routing

- Single Page Applications bestehen aus **EINER** Seite → Erfordert Routing
- „normale“ Navigation → Laden anderer Seite
- Routing → dynamisches Laden geänderter DOM-Bestandteile
- Router muss URL erkennen und richtige Inhalte anzeigen



Einstieg in React

Routing

- React-Router besteht aus drei Node-Paketen
  - **react-router**: Kern-Komponenten für das Routing
  - **react-router-dom**: API für Webbrowser
  - **react-router-native**: API für mobile Apps
- Installation des React-Routers per Kommandozeile

```
npm install --save react-router-dom
```

- Befehl installiert automatisch **react-router-dom** und **react-router**

Einstieg in React

Routing

- React-Routing bietet zwei Formen von Routern
  - BrowserRouter: URLs ohne #
  - HashRouter: URLs mit #, z.B. http://hshl.webpraktikum.net/student.html#!listStudents
- Folgende Beispiele mit BrowserRouter für Webbrowser

Einstieg in React

Routing

- Erweiterung von index.js
- Import von BrowserRouter
- **<Router>**-Tag umgibt **<App />** → erlaubt Router Inhalt von **<Router>** dynamisch zu rendern

```
import React from 'react';
import ReactDOM from 'react-dom';
import {BrowserRouter as Router, Route, Link} from 'react-router-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<Router><App /></Router>, document.getElementById('root'));
...
```

Einstieg in React

Routing

- Definition der Routen erfolgt in App.js
- Import von react-router-dom
- Definition der Inhalte als Konstanten → JSX

```
import {Route, Link} from 'react-router-dom';
import FormExample from './FormExample.js';
```

```
const JSX = () => (
    <div class="bordered">
        <h2>JSX</h2>
        <p style={myCSS}>
            3 * 8 = {3*8}
            ...
    </div>
);

const Form = () => (<FormExample />);
```

Einstieg in React

Routing

- Aufbau der Navigation in JSX
- **<Route path=““** zur Festlegung der URL
- **component={}** legt dynamischen Inhalt fest
- Attribut **exact** notwendig, damit **“/“** nicht immer interpretiert wird (**“%/%“** = like)

```
<div class="topnavigation">
    <ul>
        <li><Link to="/">JSX</Link></li>
        <li><Link to="/form">Form Example</Link></li>
    </ul>
</div>

<Route path="/" exact component={JSX}/>
<Route path="/form" component={Form}/>
```

Einstieg in React

Routing: Vollständiges Beispiel

```
import React, { Component } from 'react';
import {Route, Link} from 'react-router-dom';
import './App.css';
import FormExample from './FormExample.js';

const JSX = () => (
    <div class="bordered">
        <h2>JSX</h2>
        ...
    </div>
);
const Form = () => (<FormExample />);

class App extends Component {
    render() {
        return (
            <div className="App">
                <div class="topnavigation">
                    <ul>
                        <li><Link to="/">JSX</Link></li>
                        <li><Link to="/form">Form Example</Link></li>
                    </ul>
                </div>

                <Route path="/" exact component={JSX}/>
                <Route path="/form" component={Form}/>
            </div>
        );
    }
}

export default App;
```

Einstieg in React

Routing

- Routing von definierten Komponenten (const) ist teilweise umständlich
- React erlaubt Routing kleiner JSX-Snippets
- Direkte Definition des JSX-Snippets in der **<Route**-Anweisung
- Sollte nur bei sehr kleinen Elementen genutzt werden

```
<Route path="/renderProp" render={() => (
    <div class="bordered">
        <h2>Render prop example</h2>
        <p>Beispiel</p>
    </div>
)}/>
```

Einstieg in React

Routing: Handling nicht-existenter Routen / URLs

- Anstatt einer Server-Fehlermeldung soll eine kontrollierte Meldung erfolgen
- Erweiterung des Imports und JSX um Switch
- Festlegung einer Default-Route

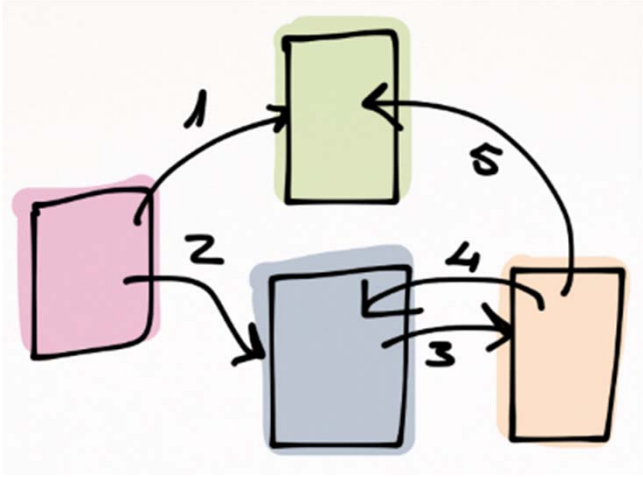
```
import {Route, Link, Switch} from 'react-router-dom';
```

```
<Switch>
    <Route path="/" exact component={JSX}/>
    <Route path="/renderProp" render={() => (
        <div class="bordered">
            <h2>Render prop example</h2>
            <p>Beispiel</p>
        </div>
    )}/>
    <Route path="/form" component={Form}/>
    <Route render={() => (
        <div>404: Seite existiert nicht</div>
    )}/>
</Switch>
```

Einstieg in React

Routing

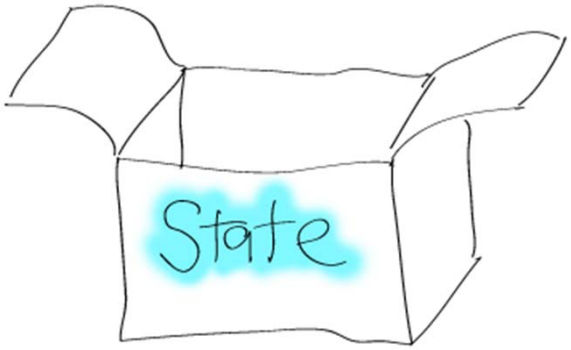
- React-Routing bietet viele weitere Funktionen, z.B.
  - Auslesen und Verarbeiten von GET-Parametern
  - Geschützte Routen mit Login
  - Aktualisierung mehrerer Bereiche (z.B. Content und Sidebar)
  - Tutorial dazu unter: https://auth0.com/blog/react-router-4-practical-tutorial/



Einstieg in React

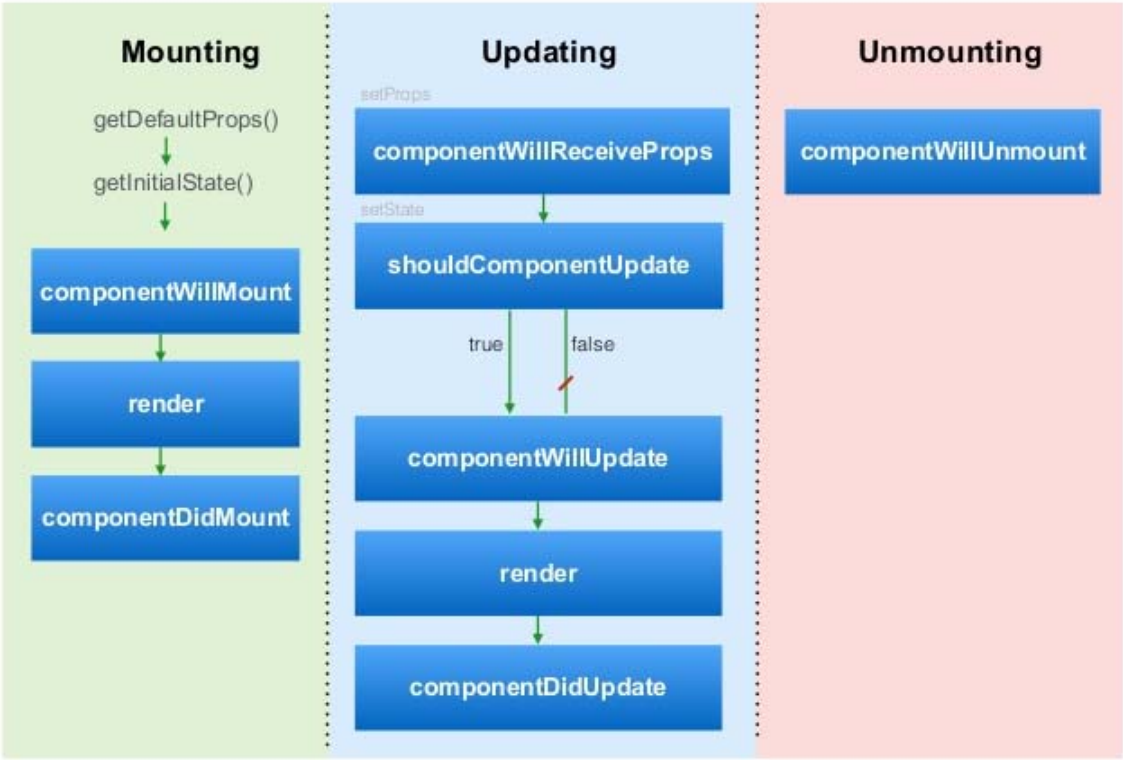
State: Zustandsvariablen

- Bisherige React-Komponenten „stateless“
- Dynamische Anwendungen benötigen zustandsfähige Variablen
- State-Daten können nur von eigener Komponente verwaltet werden („private“)
- State-Daten können als sog. Props an andere Komponenten übergeben werden
- Änderung des States führt automatisch zur Aktualisierung aller Props und Aufruf der render-Methoden



Einstieg in React

State: Lifecycle



Referenz: https://www.slideshare.net/kejun/react-48512723

Einstieg in React

State: Zugriff auf State-Daten

- Deklaration des States im Konstruktor
- Veränderung durch **`.setState()`** → autom. Aktualisierung des Renderings

```
class StateButton extends Component {
    constructor() {
        super();
        this.state = {count: 0};
    }

    render() {
        return (
            <div>
                <button onClick={() =>
                    this.setState({count: this.state.count + 1})}>+</button>
                <p>Count = {this.state.count}</p>
            </div>
        );
    }
}
```

Einstieg in React

State: Verwendung von Props

- Mapping von **`data`**-Listenelementen zu **`person`** und Übergabe an Output-Klasse

```
class Student extends Component {
    constructor() {
        super();
        this.state = {
            data: [
                {"id":1, "name":"Schmidt", "vorname":"Thomas", "MatrNr":"2141234"},
                {"id":2, "name":"Kühn", "vorname":"Maximilian", "MatrNr":"2141235"},
                {"id":3, "name":"Adler", "vorname":"Sandra", "MatrNr":"2141443"}
            ]
        }
    }

    render() {
        return (
            this.state.data.map((person, i) =>
                <Output key = {i} data = {person} />)
        );
    }
}
```

Einstieg in React

State: Verwendung von Props

- **`Student.state`** wird zu **`Output.props`**
- Probs sind nur lesbar
- Änderungen an **`Student.state`** führen autom. zur Aktualisierung von **`Output.props`**

```
class Output extends Component {
    render() {
        return (
            <p>
                <span>{this.props.data.id}: </span>
                <span>{this.props.data.name}, </span>
                <span>{this.props.data.vorname} -> </span>
                <span>{this.props.data.MatrNr}</span>
            </p>
        );
    }
}
```

1: Schmidt, Thomas -> 2141234

2: Kühn, Maximilian -> 2141235

3: Adler, Sandra -> 2141443

Einstieg in React

Event Handling

- EventBinding erfolgt leicht abgewandelt per **`event={eventHandler}`**

```
<button onClick={this.updateData}>add student</button>
```

- EventHandler muss in Konstruktor registriert werden

```
constructor() {
    super();
    ...
    this.updateData = this.updateData.bind(this);
}

updateData() {
    var newItem = {"id":4, "name":"Mustermann", "vorname":"Max", "MatrNr":"2141999"};

    this.setState(state => {
        const data = state.data.concat(newItem);
        return {data, newItem};
    });
}
```

Einstieg in React

Event Handling

- Alternativ erfolgt das Binding über folgende Syntax:
- `… onClick={() => this.updateData()} …`
- `… onClick={this.updateData.bind()}…`
- Übergabe von Argumenten erfolgt, wie gewohnt, z.B.
- `… onClick={() => this.updateData(this, 2)}…`

Einstieg in React

Formulare

- Verwendung „normaler“ Formulare → Formular-State wird nicht von React verwaltet
- Verwaltung durch React → Verwendung von „controlled components“
- Wert von Inputs wird in State gespeichert
- Erfordert Binding von EventHandlern zum Aufruf von **`setState()`**

Einstieg in React

Formulare

- Registrierung der EventHandler
- Verknüpfung mit Formular

```
this.updateData = this.updateData.bind(this);
this.handleSubmit = this.handleSubmit.bind(this);
```

```
<form onSubmit={this.handleSubmit}>
    Name: <input type="text" value={this.state.name} onChange={this.updateData} />
    <input type="submit" value="Send" />
    <p>this.state.name: {this.state.name}</p>
    <p>this.state.sendForm: {this.state.sendForm}</p>
</form>
```

```
updateData(event) {
    this.setState({name: event.target.value});
}

handleSubmit(event) {
    this.setState({sendForm: this.state.name});
    event.preventDefault();
}
```

- Update des State bei Auslösen der registrierten Events

Einstieg in React

Formulare

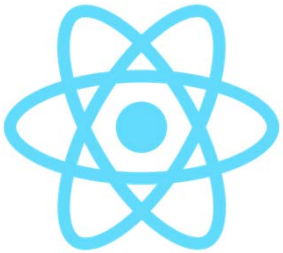
- Anders, als bei HTML wird auch bei Textareas der Inhalt per **`value`** definiert

```
<textarea value={this.state.name} onChange={this.updateData} />
```

- Bei select-Elementen wird Auswahl nicht per **`<option selected`**, sondern per **`<select value=`** festgelegt
- So muss nur ein Element verändert werden

```
<select value={this.state.color} onChange={this.updateColor}>
    <option value="green">green</option>
    <option value="blue">blue</option>
    <option value="red">red</option>
</select>
<div style={{backgroundColor:this.state.color}}> </div>
```

Einstieg in React



Fazit

- React ermöglicht die einfache und schnelle Entwicklung von Single Page Applications
- Unterstützung für modulare Architektur erlaubt Entwicklung kleiner, flexible einsetzbarer Komponenten
- JSX bietet Kombination von HTML und JavaScript (ECMAScript 6+)
- Kombination aus State und Props erlaubt autom. Aktualisierung dargestellter Inhalte in allen Views

HOCHSCHULE HAMM-LIPPSTADT

Überblick

- Das letzte Mal… und Lernziele
- Einstieg in React
- **Zum Schluss…**

Folgendes sollten Sie nun (beantworten) können:

- Wofür steht JSX? Was ist in Bezug auf HTML-Tags zu beachten?
- Was ist der Unterschied zwischen React-Routing und der „normalen“ Navigation?
- Erläutern Sie den Zusammenhang zwischen State und Props?
- Wie kann man in JSX auf interne Funktionen zugreifen?
- Wie erfolgt der Zugriff auf DOM-Elemente?

Zum Schluss…

Weiterführende Links und Literatur

[Node.js] https://nodejs.org/en/
[React] https://reactjs.org
[Dokumentation] https://reactjs.org/docs/
[Tutorial für React] https://reactjs.org/tutorial/tutorial.html
[Tutorial für React Routing] https://auth0.com/blog/react-router-4-practical-tutorial/

Golo Roden: „Node.js & Co: Skalierbare, hochperformante und echtzeitfähige Webanwendungen professionell in JavaScript entwickeln“, dpunkt.Verlag GmbH, 1. Auflage, 2012.
Sebastian Springer: „Node.js: Das Praxisbuch“, Rheinwerk Computing, 2. Auflage, 2016.

Zum Schluss…

Quellen

[sonstige Abbildungen] https://pixabay.com und https://icon-icons.com

HOCHSCHULE
HAMM-LIPPSTADT

Vielen Dank für Ihr Interesse!