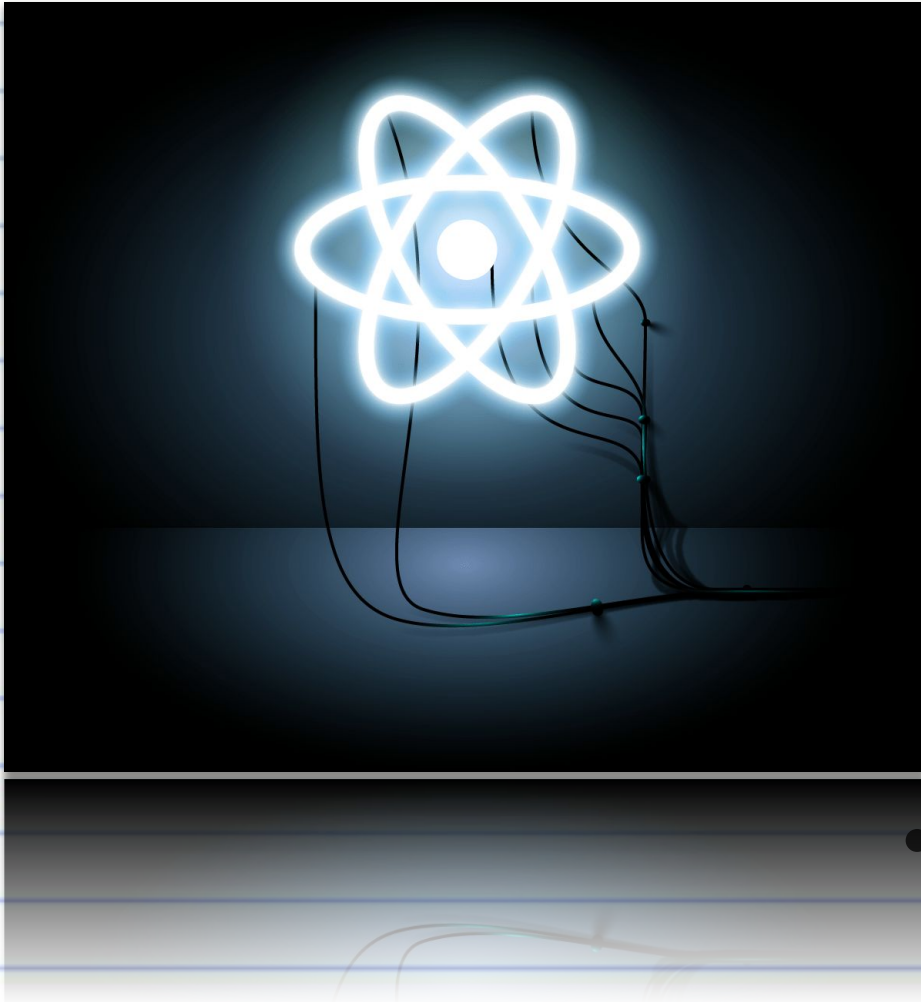


React: Forms revisited

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Forms revisited



- Wir wollen 3 Formularfelder kreieren, die automatisch upgedatet werden

AddNinja.js als
Klassenbasierte Komponente

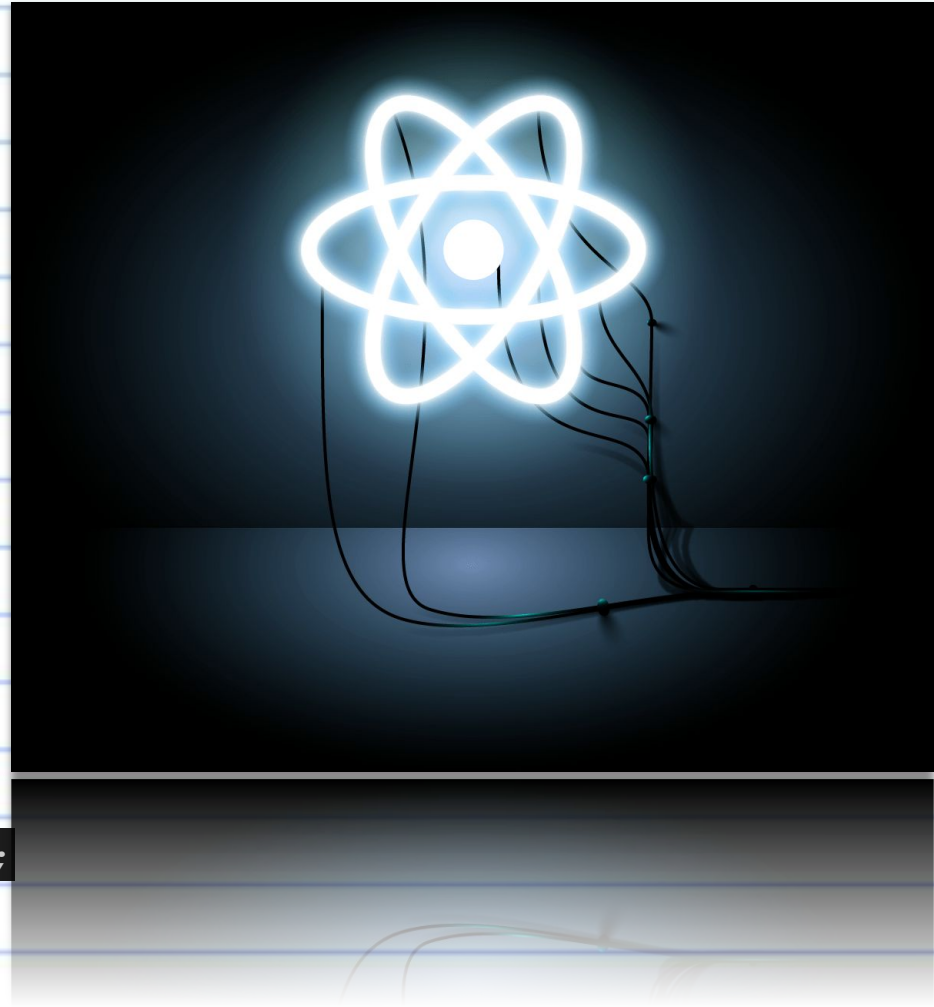
Form enthält kein Action
Attribut, weil wir direkt im
JS Submit Event "fangen"

Labels brauchen anstatt for
htmlFor

- Input Elemente haben
`onChange={this.handleChange}`

React: Forms revisited

```
class AddNinja extends Component {  
  state = {  
    name: null,  
    age: null,  
    belt: null  
  }  
  
  handleChange = (e) => {  
    this.setState({  
      [e.target.id]: e.target.value  
    });  
  }  
  
  handleSubmit = (e) => {  
    e.preventDefault();  
    this.props.addNinja(this.state);  
  }  
}
```



React: Functions as Props

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Funktionen als Properties

- Wir wollen den neuen Ninja in den State hinzufügen
- Wir können den State nicht direkt im AddNinja.js bearbeiten (Stateless)
- Die Funktion addNinja muss also im App.js kreiert werden (State Container)
- Akzeptiert einen Ninja als Input
- Und wird als Property in die Komponente Übergeben:

```
<AddNinja addNinja={this.addNinja}>
```

- Im AddNinja.js können wir mit `this.props.addNinja(this.state)` den neuen Ninja hinzufügen
- Wir benutzen dabei den Spread-Operator um den State zu kopieren und den neuen Ninja hinzuzufügen
- Neuer Ninja erscheint sofort nach Abschicken in der Ninja Komponente (wird automatisch neu gerendert)



JS Exkurs: ... "Spread Operator"

Die Spread-Syntax erlaubt es, einen einzelnen Array-Ausdruck oder String an Stellen zu expandieren, an denen Null oder mehr Argumente (für Funktionsaufrufe) oder Elemente (für Array-Literale) erwartet werden.

```
function sum(x, y, z) {  
    return x + y + z;  
}  
  
const numbers = [1, 2, 3];
```

```
console.log(sum(...numbers));  
// expected output: 6
```

Mehr Infos unter

https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Operators/Spread_syntax

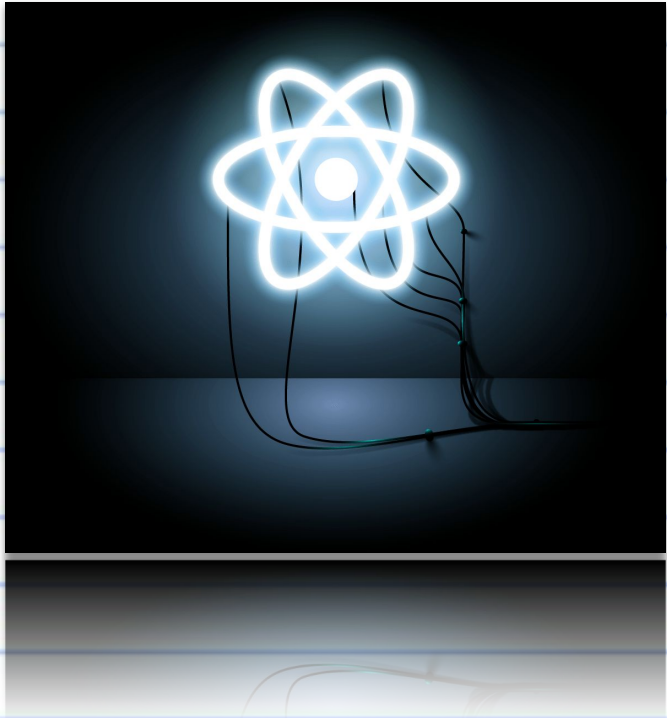


React: Daten löschen

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Daten Löschen

- Funktion `deleteNinja()` kommt auch in die `App.js` als Props



- Jeder Ninja bekommt einen Button, der die Löschung triggert
- Damit nicht automatisch beim Rendern gelöscht wird, müssen wir eine Arrow-Funktion benutzen =>
- Via ID und nicht-destruktives Filtern unterscheiden wir die verschiedenen Ninjas und können den Richtigen löschen

JS Exkurs: Array.prototype.filter

`filter()` erstellt ein neues Array mit allen Elementen, die den von der bereitgestellten Funktion implementierten Test bestehen.

```
const words = ['spray', 'limit', 'elite',  
               'exuberant', 'destruction', 'present'];
```

```
const result = words.filter(word =>  
word.length > 6);
```

```
console.log(result);
```

```
// expected output: Array ["exuberant",  
                           "destruction", "present"]
```

Mehr Infos unter

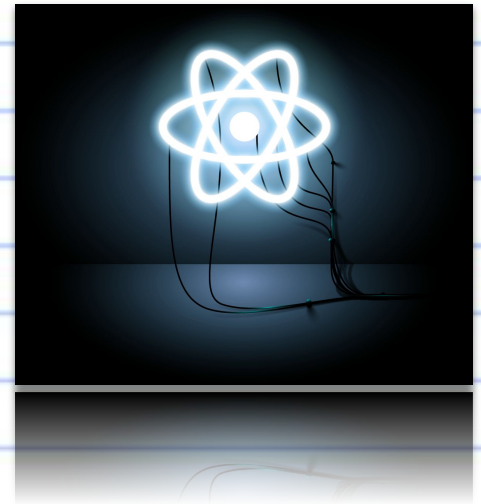
https://developer.mozilla.org/de/docs/Web/JavaScript/Reference/Global_Objects/Array/filter



React: Daten Löschen

```
deleteNinja = (id) => {  
  // console.log(id);  
  let ninjas = this.state.ninjas.filter(ninja => {  
    return ninja.id !== id  
  });  
  this.setState({  
    ninjas: ninjas  
  });  
}
```

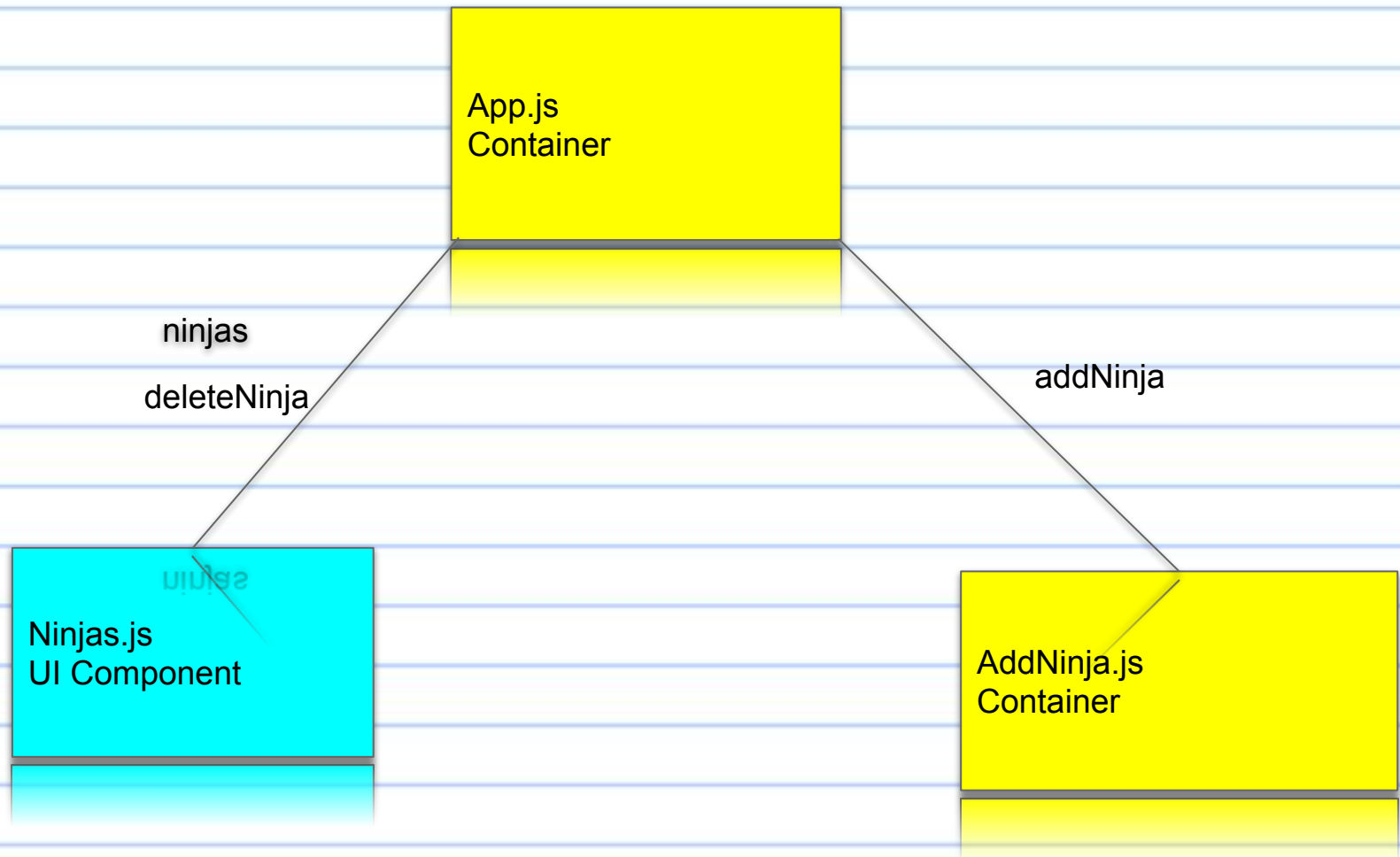
```
<Ninjas ninjas={this.state.ninjas}  
deleteNinja={this.deleteNinja} />
```



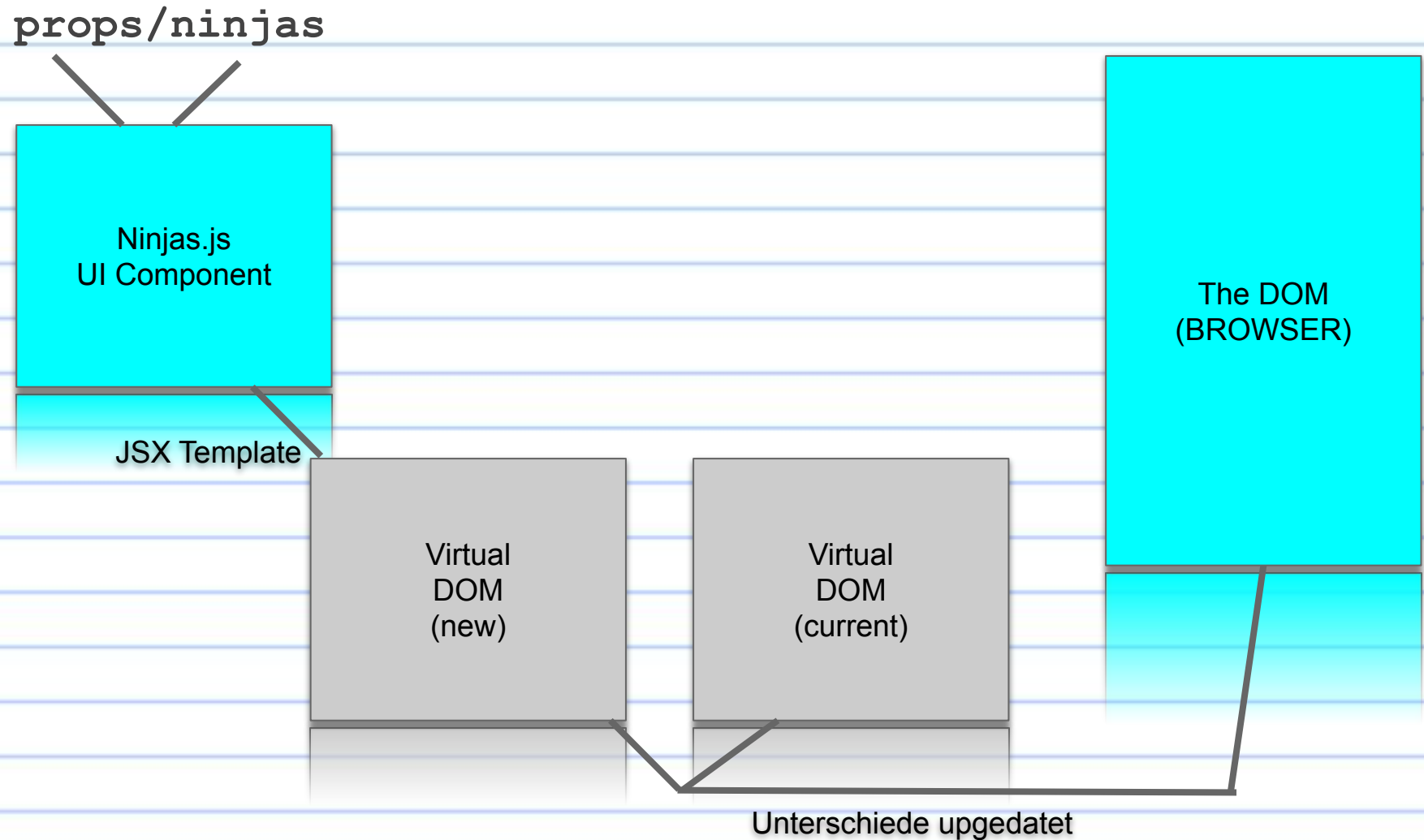
React: Recap & Virtual DOM

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Recap & Virtual DOM



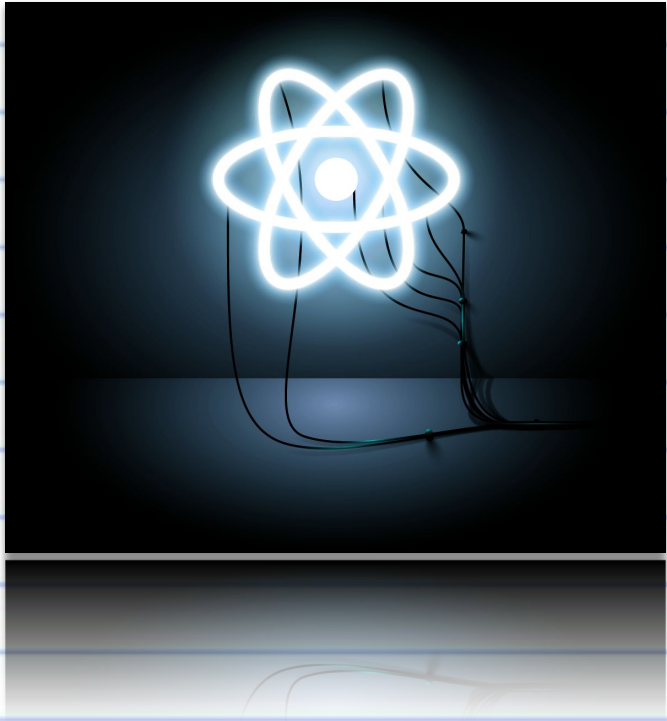
React: Recap & Virtual DOM



React: CSS Files

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: CSS Files

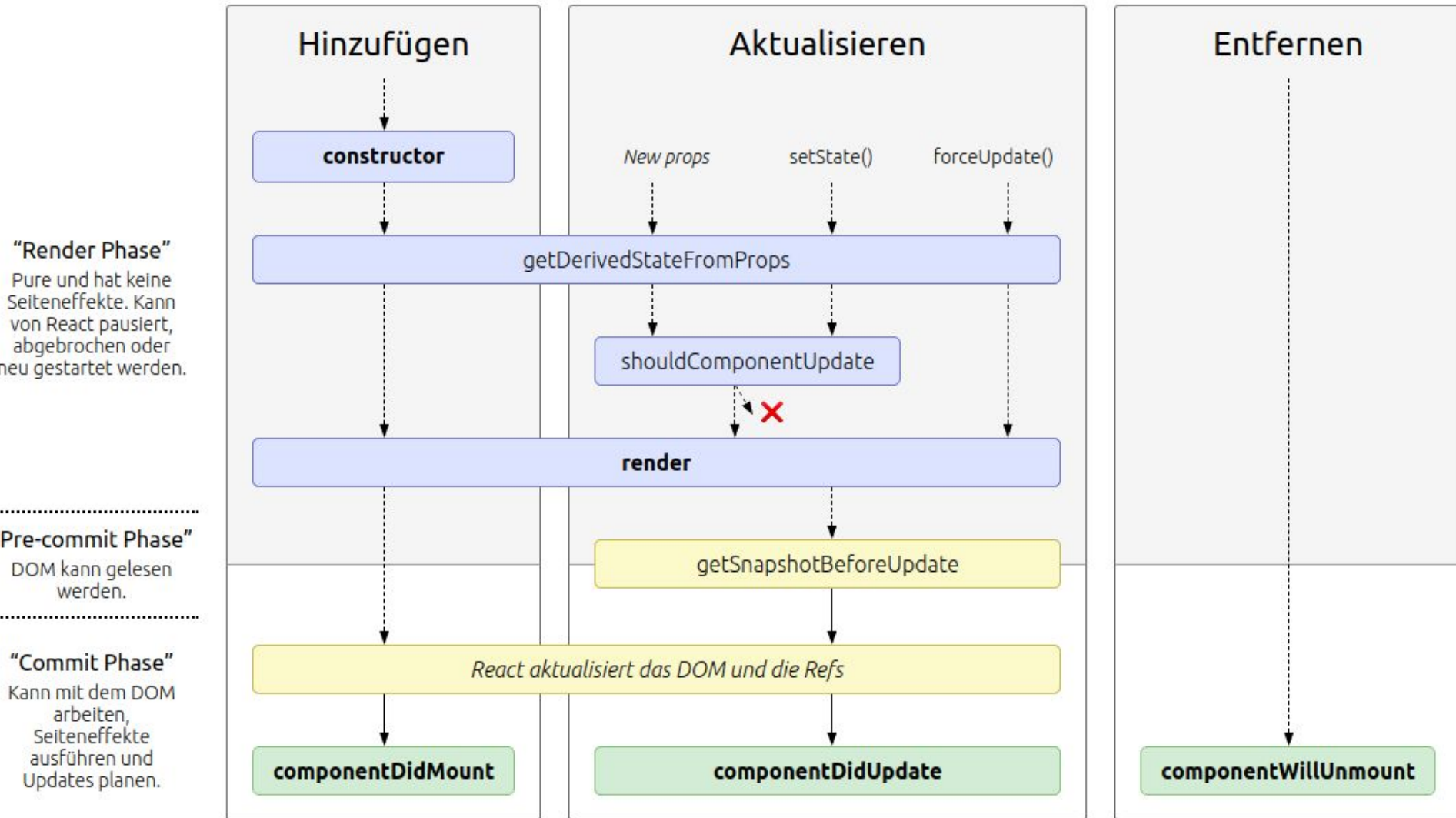


- Es gibt verschiedene Arten CSS im React einzubinden
- Man kann pro Komponente ein CSS File erstellen und importieren
- z.B. `import './Ninjas.css'`
- Achtung, das CSS wird nicht nur für die Komponente gerendert, sondern für die ganze Seite!
- Scope des CSS ist wichtig!
- Vendor-Prefixes kann man ignorieren, macht React automatisch, z.B. `transition: all 1s;`
- Oder einfach alles ins `index.css` ;-)

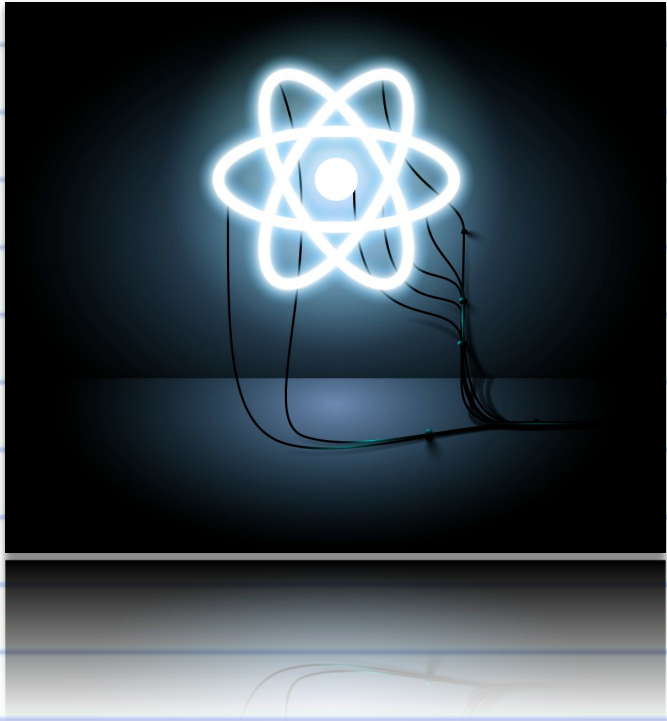
React:Lifecycle Methods

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React Lifecycle



React: Lifecycle Methoden



- On Refresh: `ComponentDidMount()`
- Wie `onLoad` / `DOMContentLoaded` in JS
- Wird nur einmal gefeuert beim ersten Rendern!
- On Change: `ComponentDidUpdate()`
- Kein entsprechendes Event im JS
- Wird immer wieder gefeuert, jedes mal wenn es ein Update gab. VORSICHT! Hier kann man sich schnell Endlosschleifen einhandeln
- Kennt den vorherigen Props und den vorherigen State
- Kann ein Update verhindern!

React: Todo App (1)

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Alles in einer neuen App: Todo App

```
npx create-react-app todoapp (--use-npm)
```

index.html wird mit Materialize CSS gefüllt:

<https://materializecss.com/getting-started.html>

CDN Links nur für CSS hinzufügen, kein JS nötig!

```
<!-- Compiled and minified CSS -->  
<link rel="stylesheet"  
href="https://cdnjs.cloudflare.com/  
ajax/libs/materialize/1.0.0/css/  
materialize.min.css">
```

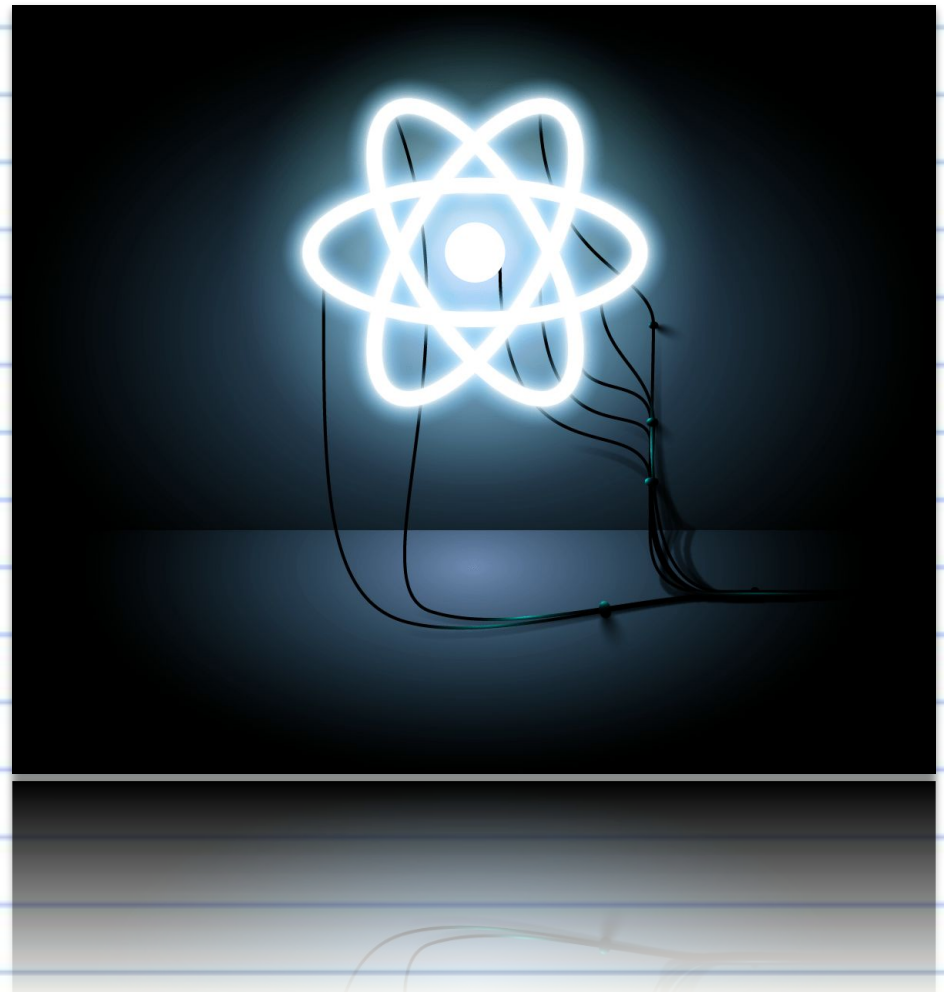


Applikation aufräumen (Logos, Tests und CSS löschen!)

Todo App: Wieder Files aufräumen

my-app

- |— README.md
- |— node_modules
- |— package.json
- |— .gitignore
- |— public
 - |— favicon.ico
 - |— index.html
 - |— manifest.json
- |— src
 - |— App.css
 - |— App.js
 - |— App.test.js
 - |— index.css
 - |— index.js
 - |— logo.svg
 - |— serviceWorker.js
 - |— setupTests.js



Todo App: State definieren

Todos werden im State der App.js gespeichert!

```
import Todos from './Todos'
```

```
state = {  
  todos: [  
    { id: 1, content: 'Milch einkaufen' },  
    { id: 2, content: 'Spiele Mario Kart' }  
  ]  
}
```

```
return ...
```

```
<Todos todos={this.state.todos} />
```



Todo App: Komponente Todos.js

```
import React from 'react';

const Todos = ({todos, deleteTodo}) => {

  const todoList = todos.length ? (

    todos.map(todo => {

      return (<div className="collection-item" key={todo.id}>

        <span onClick={() => {deleteTodo(todo.id)}}>{todo.content}</span>

        </div>)))) :

    (<p className="center">You have no todo's left, yay!</p>);

  return (

    <div className="todos collection">

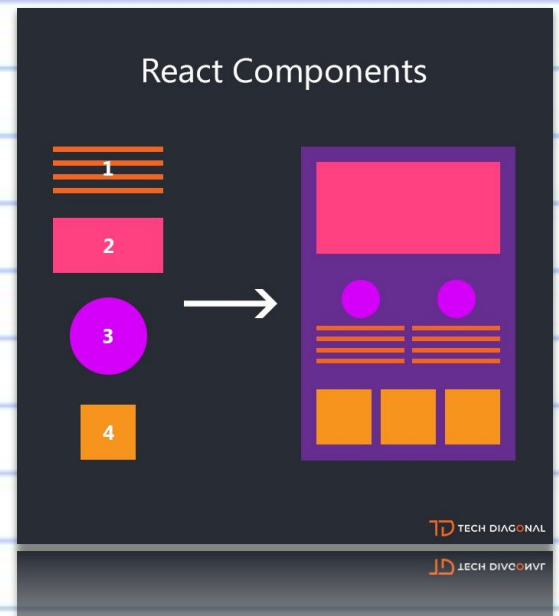
      {todoList}

    </div>

  )

}

export default Todos;
```



JS Exkurs: Object Property Short

Neu ist, dass in ES6/EcmaScript 2015, wenn man ein Objekt definiert und Key und Value Name identisch sind, dass es ausreicht, nur einmal den Identifier zu schreiben, z.B.

```
const cat = 'Miaow';
```

```
const dog = 'Woof';
```

```
const bird = 'Peet peet';
```

```
const someObject = {
```

```
  cat,
```

```
  dog,
```

```
  bird
```

```
};
```

Mehr Infos unter

<https://alligator.io/js/object-property-shorthand-es6/#:~:text=Electron.js-,Object%20Property%20Value%20Shorthand%20in%20JavaScript%20with%20ES6,simply%20pass%20the%20key%20name.>



Todo App: Todo Löschen (im App.js)

```
deleteTodo = (id) => {  
  const todos = this.state.todos.filter(todo => {  
    return todo.id !== id  
  });  
  this.setState({  
    todos  
  });  
}
```



```
<Todos todos={this.state.todos} deleteTodo={this.deleteTodo} />
```

React:

Todo App (2)

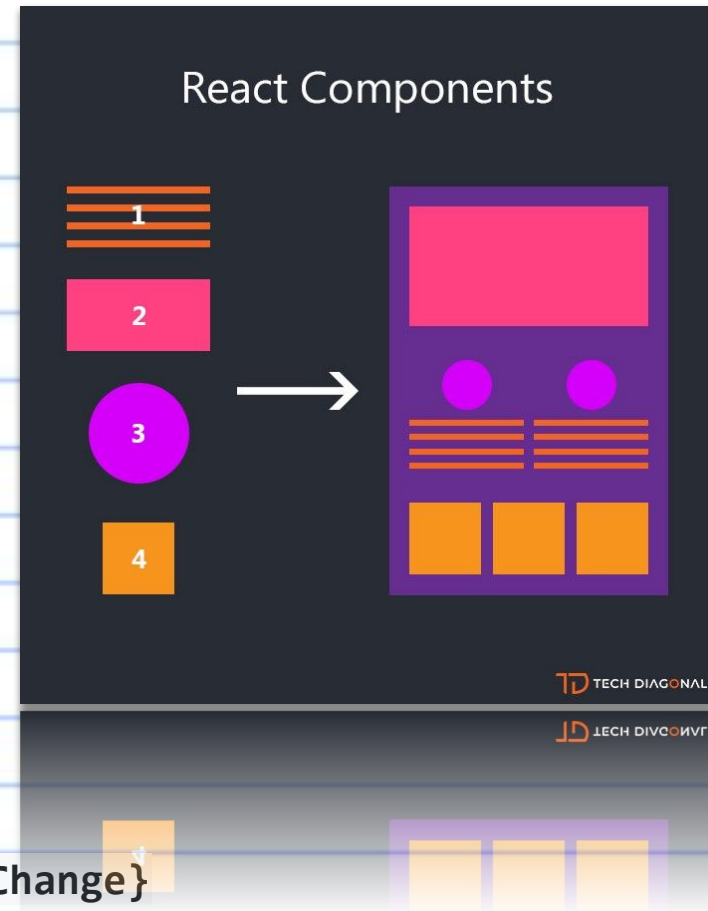
Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin

Todo App: Neue Komponente AddTodo

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

class AddTodo extends Component {
  state = {content: ''}
  handleChange = (e) => {
    this.setState({content: e.target.value});
  }
  handleSubmit = (e) => {
    e.preventDefault();
    // call function to add a todo
    this.props.addTodo(this.state);
    this.setState({content: ''})
  }

  render() {
    return (
      <form onSubmit={this.handleSubmit}>
        <label>Add a new todo:</label>
        <input type="text" onChange={this.handleChange}
value={this.state.content} />
      </form>
    )
  }
}
```



Todo App: AddTodo in App.js adden

```
import AddTodo from './AddTodo'
```

```
addTodo = (todo) => {
```

```
  todo.id = Math.random();
```

```
  let todos = [...this.state.todos, todo];
```

```
  this.setState({
```

```
    todos
```

```
  });
```

```
}
```

```
<AddTodo addTodo={this.addTodo} />
```

React Components



1



2



3



4

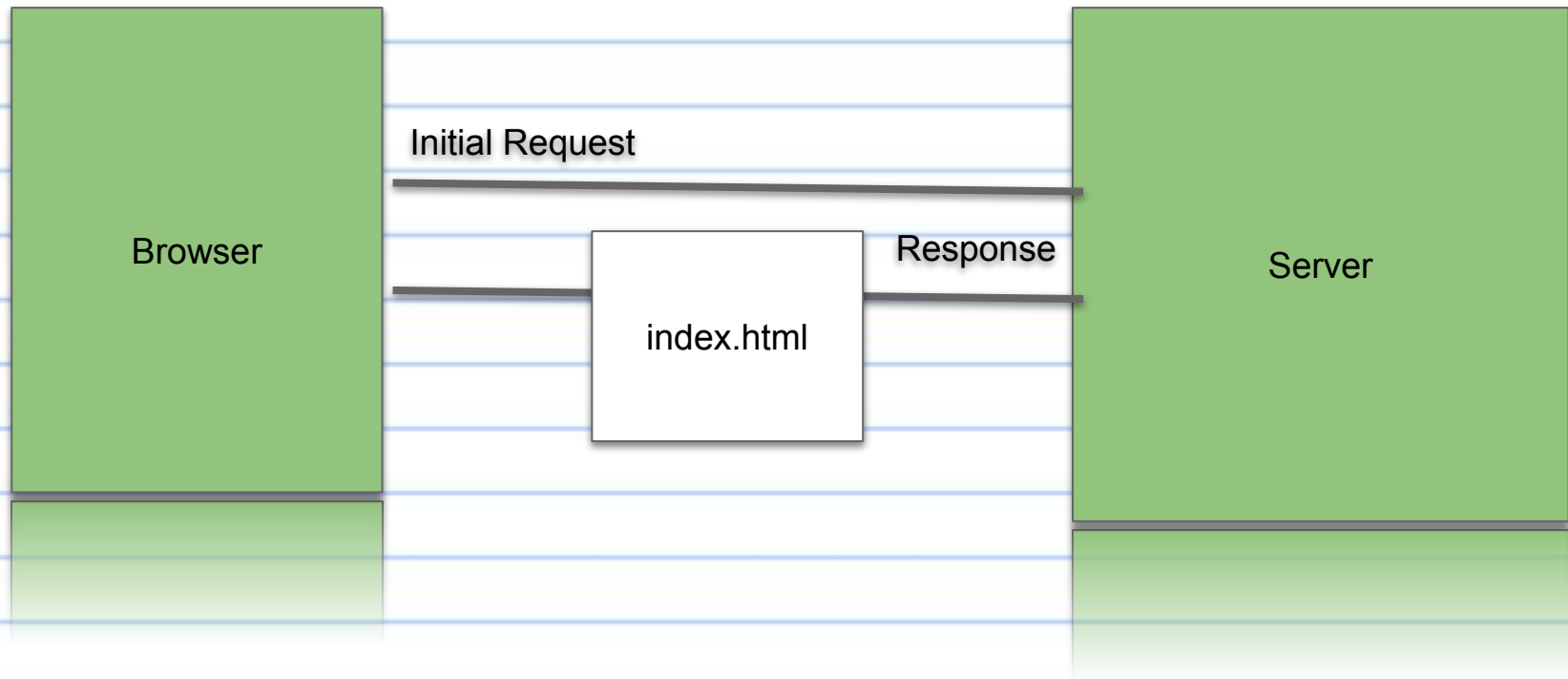


React: Router

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

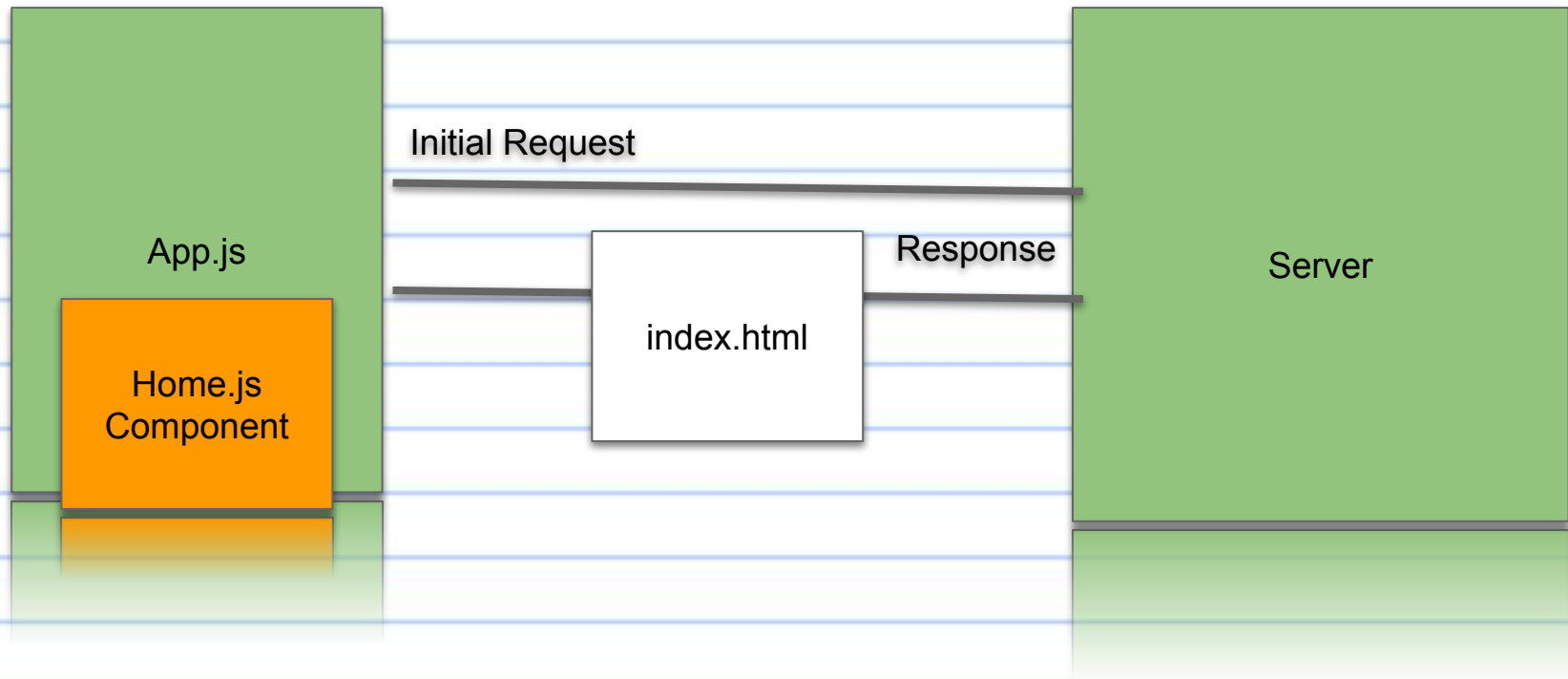
React: Router

/home



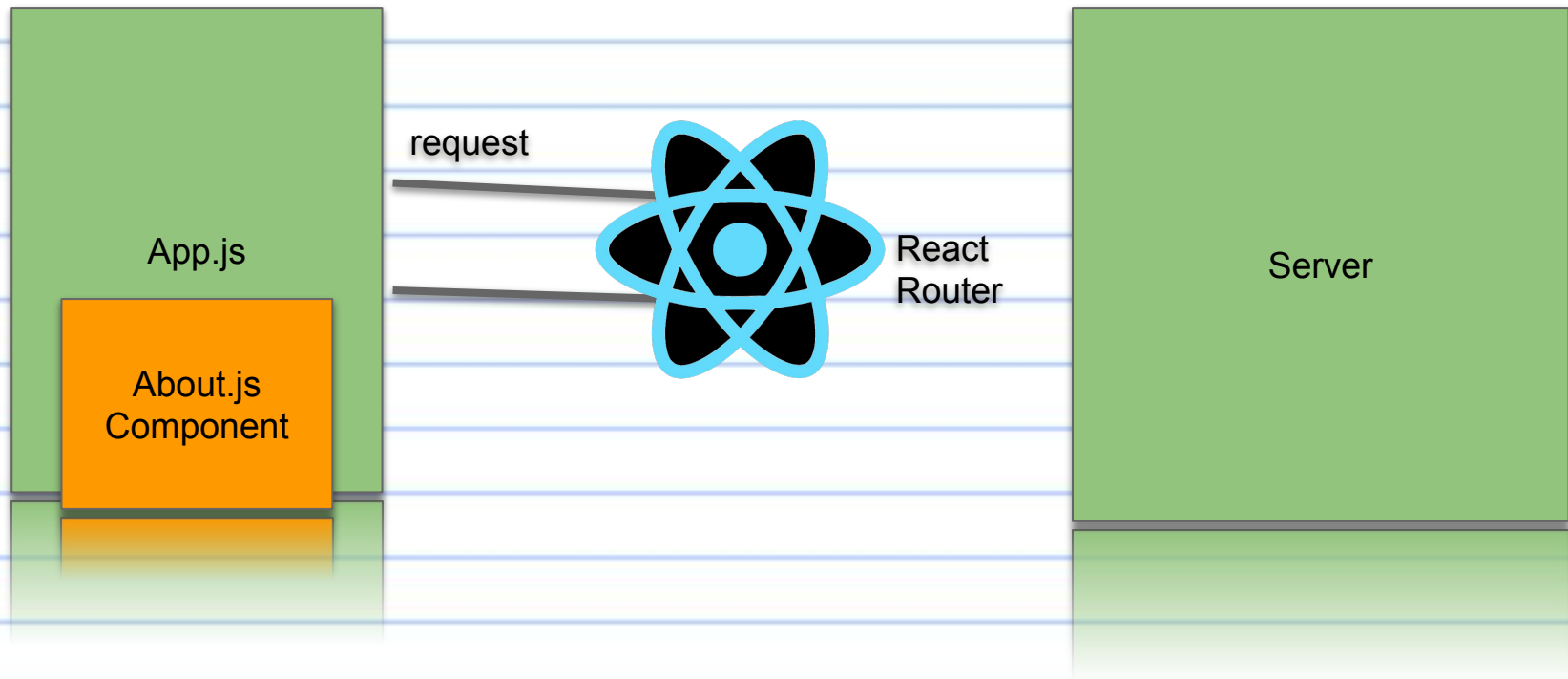
React: Router

/home



React: Router

/about



Poketimes: Pokemon Referenz

```
npx create-react-app poketimes (--use-npm)
```

index.html wird mit Materialize CSS gefüllt:

<https://materializecss.com/getting-started.html>

CDN Links nur für CSS hinzufügen, kein JS nötig!

```
<!-- Compiled and minified CSS -->  
<link rel="stylesheet"  
href="https://cdnjs.cloudflare.com/  
ajax/libs/materialize/1.0.0/css/  
materialize.min.css">
```



Applikation aufräumen (Logos, Tests und CSS löschen!)

Neuer Ordner Components // Home.js

```
import React from 'react'
```

```
const Home = () => {
```

```
  return (
```

```
    <div>
```

```
      <div className="container">
```

```
        <h4 className="center">Home</h4>
```

```
        <p>Lorem ipsum dolor sit amet consectetur
```

```
adipiscing elit. Recusandae repudiandae repellat
```

```
illo magni eligendi cupiditate voluptates eius nam
```

```
voluptate. Incidunt nihil ullam quae quia officia
```

```
quaerat, deserunt eligendi explicabo totam?</p>
```

```
      </div>
```

```
    </div>
```

```
  )
```

```
}
```

```
export default Home
```



Kopiere in About.js & Contact.js

```
import React from 'react'
```

```
const About = () => {
```

```
  return (
```

```
    <div>
```

```
      <div className="container">
```

```
        <h4 className="center">About</h4>
```

```
        <p>Lorem ipsum dolor sit amet
```

```
consectetur adipisicing elit. Recusandae
```

```
repudiandae repellat illo magni eligendi
```

```
cupiditate voluptates eius nam voluptate.
```

```
Incidunt nihil ullam quae quia officia
```

```
quaerat, deserunt eligendi explicabo
```

```
totam?</p>
```

```
      </div>
```

```
    </div>
```

```
  )
```

```
}
```

```
export default About
```

```
import React from 'react'
```

```
const Contact = () => {
```

```
  return (
```

```
    <div>
```

```
      <div className="container">
```

```
        <h4 className="center">Contact</h4>
```

```
        <p>Lorem ipsum dolor sit amet
```

```
consectetur adipisicing elit. Recusandae
```

```
repudiandae repellat illo magni eligendi
```

```
cupiditate voluptates eius nam voluptate.
```

```
Incidunt nihil ullam quae quia officia
```

```
quaerat, deserunt eligendi explicabo
```

```
totam?</p>
```

```
      </div>
```

```
    </div>
```

```
  )
```

```
}
```

```
export default Contact
```

Navigation in Navbar.js (UI)

```
import React from 'react';
```

```
const Navbar = () => {
```

```
  return (
```

```
    <nav className="nav-wrapper red darken-3">
```

```
      <div className="container">
```

```
        <a className="brand-logo">Poke' Times</a>
```

```
        <ul className="right">
```

```
          <li><a href="/">Home</a></li>
```

```
          <li><a href="/about">About</a></li>
```

```
          <li><a href="/contact">Contact</a></li>
```

```
        </ul>
```

```
      </div>
```

```
    </nav>
```

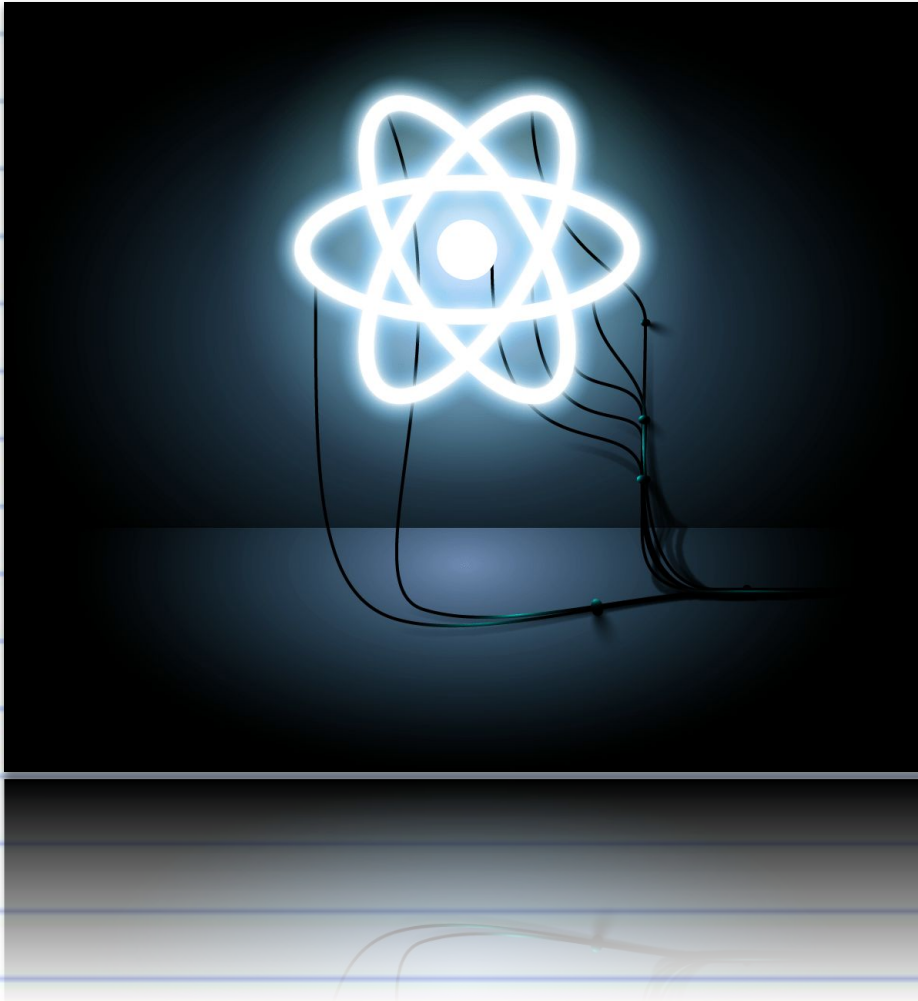
```
  )
```

```
}
```

```
export default Navbar
```



React Exkurs: "react-router-dom"



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

Mehr Infos unter

<https://www.npmjs.com/package/react-router-dom>

Navbar in die App.js importieren

```
import Navbar from './components/Navbar'  
import { Route, BrowserRouter } from  
'react-router-dom'  
import Home from './components/Home'  
import About from './components/About'  
import Contact from './components/Contact'
```

```
class App extends Component {  
  render() {  
    return (  
      <BrowserRouter>  
        <div className="App">  
          <Navbar />  
          <Route exact path="/" component={Home} />  
          <Route path="/about" component={About} />  
          <Route path="/contact" component={Contact} />  
        </div>  
      </BrowserRouter>  
    )  
  }  
}
```



Wie funktioniert Routing?

Achtung! Falls wir auf "/" ohne Exact lauschen, dann wird es ÜBERALL angezeigt!

Da der Slash ja immer noch Teil der URL ist, egal wohin wir gehen.

Mit diesem Code gibt es allerdings immer noch einen kompletten Seiten Reload jedes Mal wenn wir auf einen Link in der NavBar klicken.

Eigentlich ist ja die Idee, dass React dann dazwischenfunkt und die Serveranfrage lokal beantwortet, statt einen Reload zu machen. Das funktioniert wie folgt...

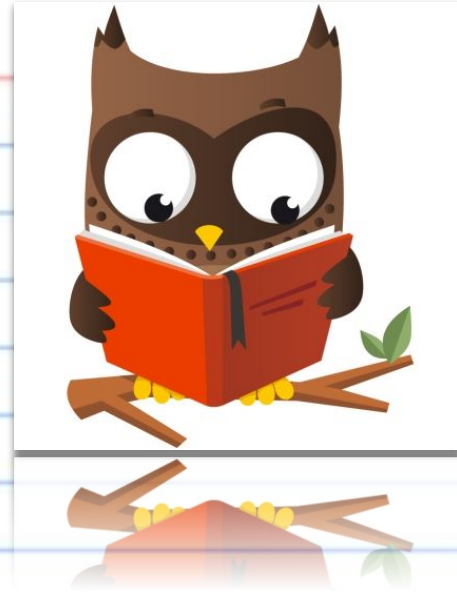
React: Links & NavLinks

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Import Link & NavLink von Router

```
import React from 'react';
import { Link, NavLink } from 'react-router-dom'

const Navbar = () => {
  return (
    <nav className="nav-wrapper red darken-3">
      <div className="container">
        <Link className="brand-logo" to="/">Poke' Times</Link>
        <ul className="right">
          <li><NavLink exact to="/">Home</NavLink></li>
          <li><NavLink to='/about'>About</NavLink></li>
          <li><NavLink to='/contact'>Contact</NavLink></li>
        </ul>
      </div>
    </nav>
  )
}
```



Wie funktioniert Linking?

Das Link-Tag verhindert in React den automatischen Reload, wenn wir auf einen Link klicken!

Jetzt wird der Request wie gewünscht nicht mehr zum Server gesendet, sondern React "intercepted".

Der NavLink-Tag fügt automatisch eine aktive Klasse zum zuletzt geklickten bzw. Zu der Seite, auf der wir uns aktuell befinden, so dass wir diese möglichst einfach stylen können.

Also je nachdem ob wir einen normalen Link brauchen oder einen Navigations-Link, immer die passenden React-Tags Link bzw. NavLink benutzen, um den vollen Vorteil von React auszunutzen.

React: Redirects

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Wie funktioniert Browser Router?

Auslesen z.B. mit `console.log(props)` ;

Der Browser Router fügt extra Informationen zu den Properties hinzu, sowas wie

- die Browser-History,
- die aktuelle Browser-Location,
- der Match zu den Pfaden

Ausprobieren z.B. `props.history.push('/about')`

Geht nur bei `<Route>` automatisch

In NavBar müssen wir noch was hinzufügen...



Wie funktioniert withRouter ?

```
import { Link, NavLink, withRouter }  
from 'react-router-dom'
```

```
const Navbar = (props) => {  
  setTimeout(() => {  
    props.history.push('/about');  
  }, 2000);  
}
```

```
export default withRouter(Navbar)
```

React: Axios benutzen

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Unsere Test API: JSON Placeholder

<https://jsonplaceholder.typicode.com/posts>

```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat pro
occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae
consequuntur expedita et cum\nreprehenderit molestiae
ut ut quas totam\nnostrum rerum est autem sunt rem
eveniet architecto"
  },
  {
```



npm install axios => statt fetch?

Home wird zu einer Klassenbasierten Komponente:

```
import axios from 'axios'

class Home extends Component {
  state = {
    posts: []
  }

  componentDidMount () {
    axios.get('https://jsonplaceholder.typicode.com/posts/')
      .then(res => {
        console.log(res);
        this.setState({
          posts: res.data.slice(0,10)
        });
      })
  }
}
```


Nur rendern falls es Posts gibt...



```
const { posts } = this.state
const postList = posts.length ? (
  posts.map(post => {
    return (
      <div className="post card" key={post.id}>
        <div className="card-content">
          <span className="card-title">{post.title}</span>
          <p>{post.body}</p>
        </div>
      </div>
    )
  })
) : (
  <div className="center">No posts to show</div>
);
return (
  <div>
    <div className="container">
      <h4 className="center">Home</h4>
      {postList}
    </div>
```

React: Route Parameter (1)

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Route Parameter

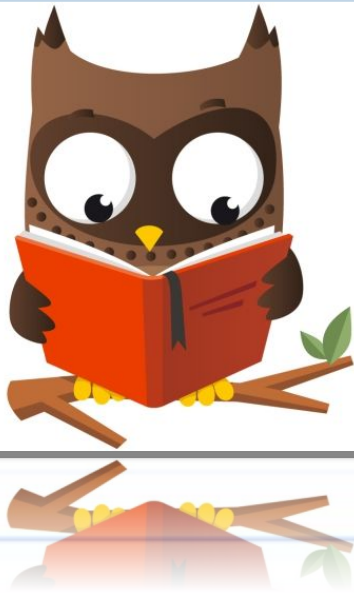
- Teile einer URL, die sich ändern können
- Normalerweise sieht man das in URLs für individuelle Daten
- z.B. User Profil Route:
 - `mysite.com/users/yoshi2k1`
 - `mysite.com/users/mariotheman`
- Oder Kochrezeptseite:
 - `mycookingsite.com/recipes/12345`
 - `mycookingsite.com/recipes/98765`



Neue Route für URL Parameter /:post_id

ID PER POST = ROUTE PARAMETER

```
import Post from './components/Post'
```



```
<BrowserRouter>
```

```
  <div className="App">
```

```
    <Navbar />
```

```
    <Route exact path="/" component={Home} />
```

```
    <Route path="/about" component={About} />
```

```
    <Route path="/contact" component={Contact}
```

```
  />
```

```
    <Route path="/:post_id" component={Post} />
```

```
  </div>
```

```
</BrowserRouter>
```

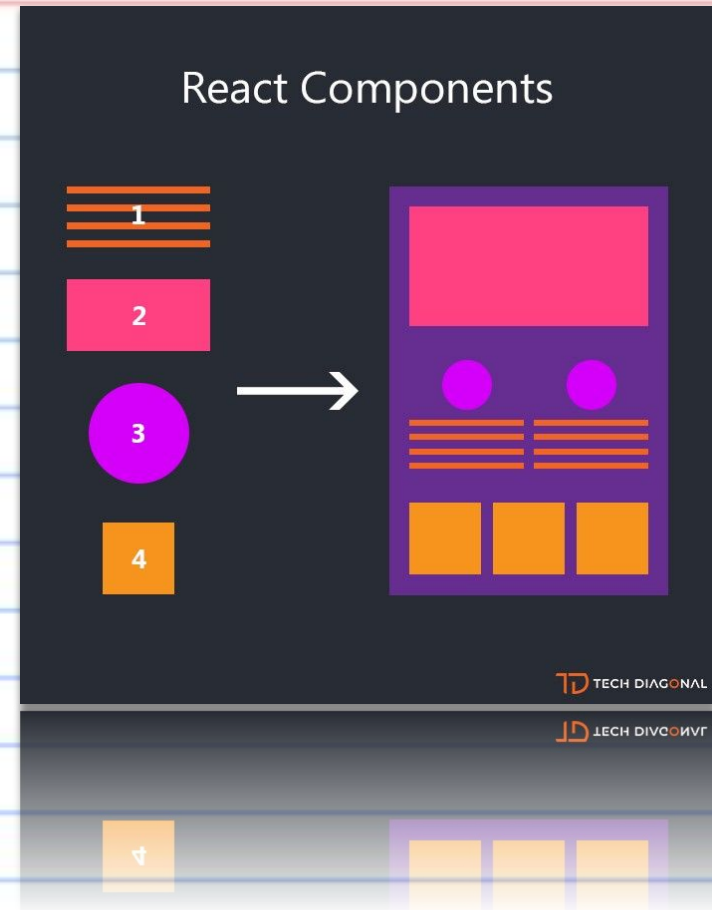
Neue Komponente: Post.js mit match

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

class Post extends Component {
  state = {
    id: null
  }

  componentDidMount() {
    let id = this.props.match.params.post_id;
    this.setState({
      id
    })
  }

  render() {
    return (
      <div className="container">
        <h4>{this.state.id}</h4>
      </div>
    )
  }
}
```



React: Route Parameter (2)

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Verlinken der Posts mit `<Link>` und Route-Parameter ID

```
import { Link } from 'react-router-dom'
```

```
<Link to={'/' + post.id}>
```

```
  <span className="card-title red-text">{post.title}</span>
```

```
</Link>
```

Und in der `Post.js` ändern wir den API Link je nach ID

```
import axios from 'axios'
```

```
state = {
```

```
  post: null
```

```
}
```

```
componentDidMount() {
```

```
  let id = this.props.match.params.post_id;
```

```
  axios.get('https://jsonplaceholder.typicode.com/posts/' + id)
```

```
    .then(res => {
```

```
      this.setState({
```

```
        post: res.data
```

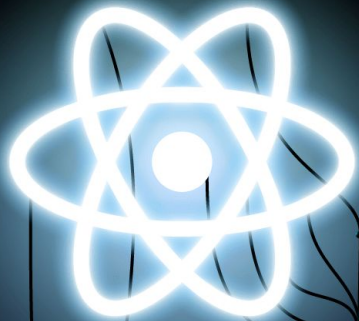
```
      });
```

```
      //console.log(res.data);
```

```
    })
```



Falls der API Post zurück kommt...



```
const post = this.state.post ? (  
  <div className="post">  
    <h4  
      className="center">{this.state.post.title}</h4>  
    <p>{this.state.post.body}</p>  
  </div>  
  ) : (  
    <div className="center">Loading  
      post...</div>  
  );  
  
return (  
  <div className="container">  
    {post}  
  </div>  
)
```


React: Switch Parameter

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Bug: Problem mit dem Routing

Es gibt ein Problem mit dem Router, da jetzt ALLES nach dem / als :post-id interpretiert wird, Auch wenn wir z.B. nur auf /contact gehen wollen.

Ein einfacher Quick-Fix wäre, wenn wir
Einen Parameter immer vor die Id stellen

z.B.

```
<Link to={` /posts/` + post.id}>
```

```
<Route path="/posts/:post_id component={Post} />
```

Zweite Möglichkeit: Switch Tag

```
<Switch>
```

```
  <Route exact path="/" component={Home} />
```

```
  <Route path="/about" component={About} />
```

```
  <Route path="/contact" component={Contact} />
```

```
  <Route path="/:post_id" component={Post} />
```

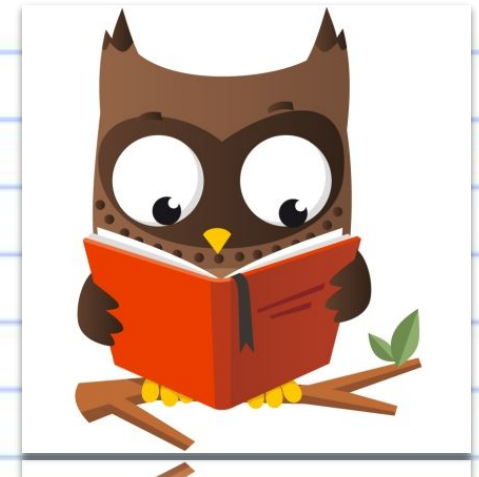
```
</Switch>
```

Damit kann nur ein Path kann gleichzeitig richtig sein

Und der erste Hit wird genommen

Danach sucht er nicht weiter nach

Passenden Routen!

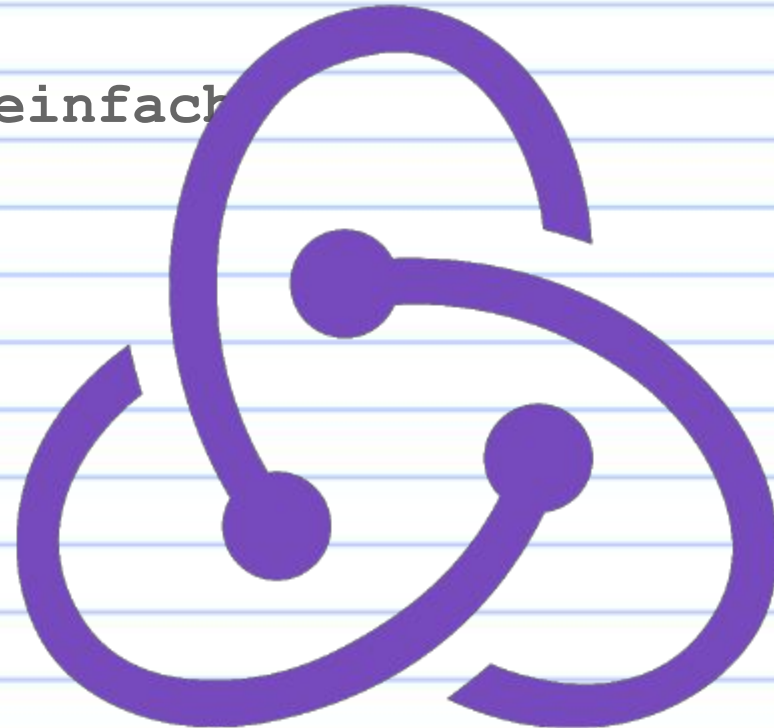


React: Redux

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

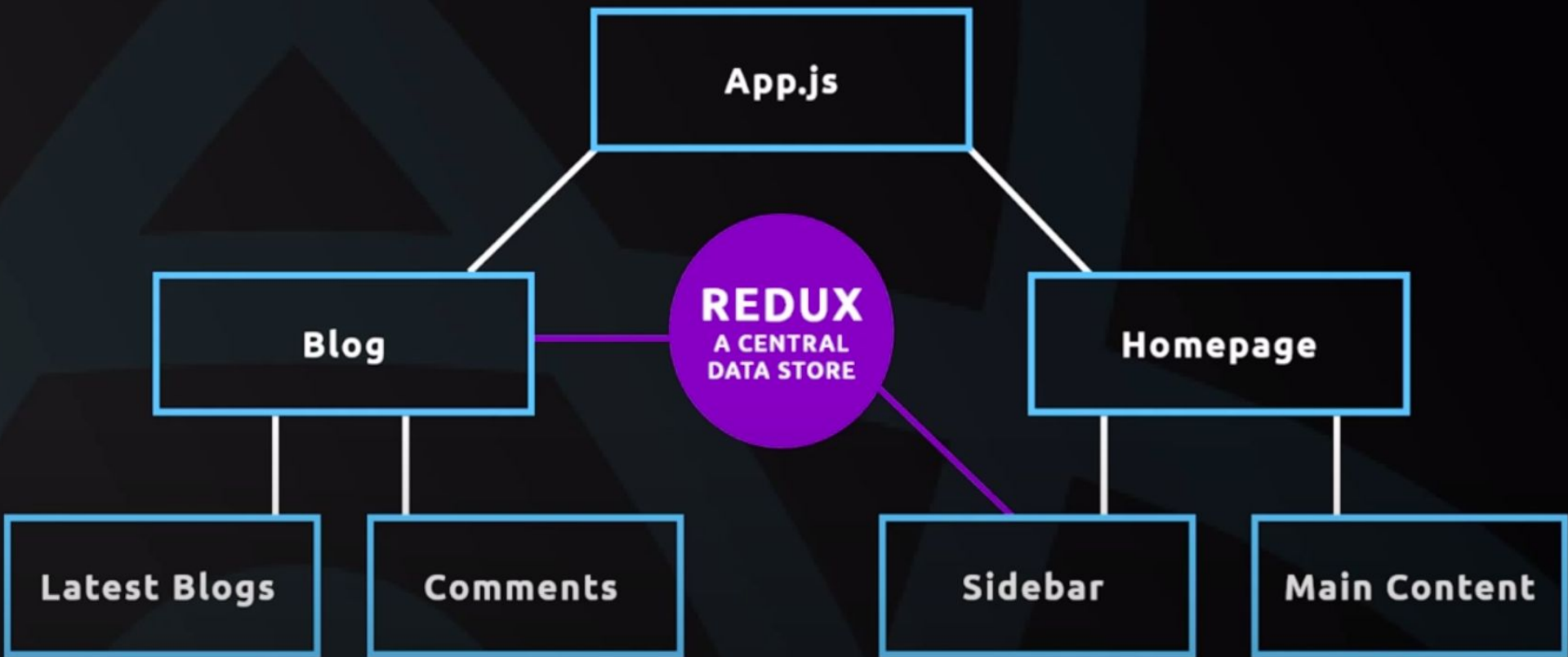
Was ist Redux? Wofür ist es gut?

- Zentraler Datenspeicher für all unsere Applikationsdaten
- Jede Komponente hat Zugriff auf diese Daten
- Macht das State Management einfach

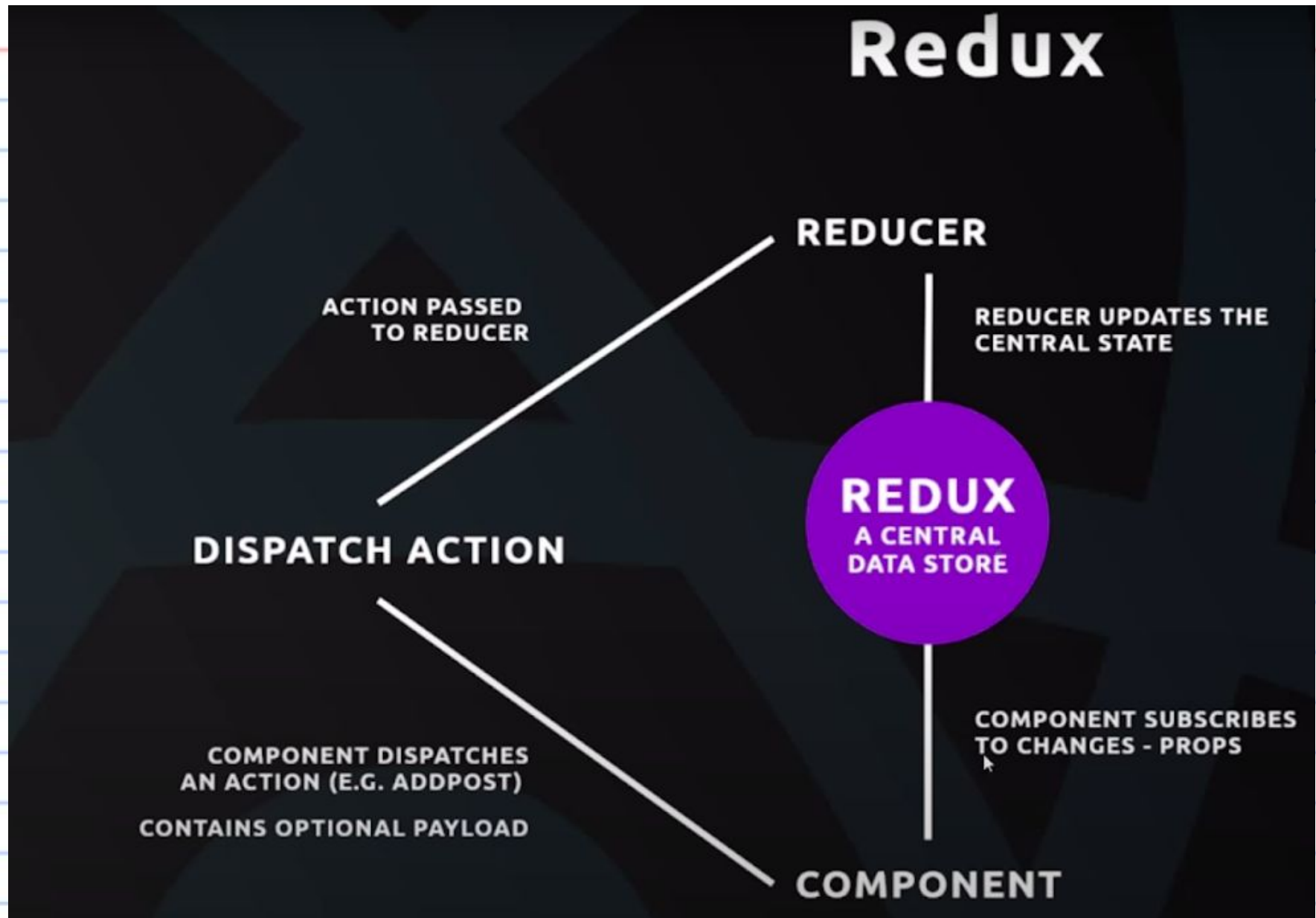


Zentraler Datenspeicher für alle

Redux



React: Redux Lifecycle Route



React: Redux Store

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

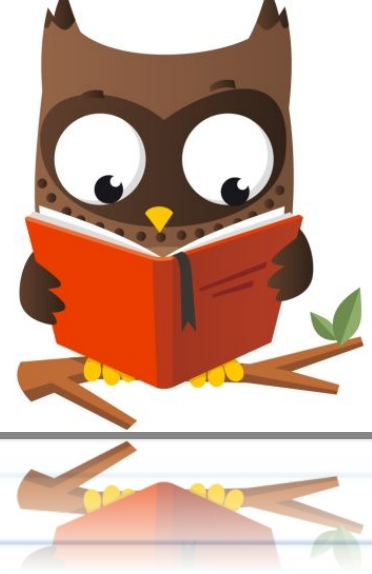
Codepen Redux Store Basics

```
const { createStore } = Redux;
```

```
const initState = {  
  todos: [],  
  posts: []  
}
```

```
function myreducer(state = initState, action){  
  
}
```

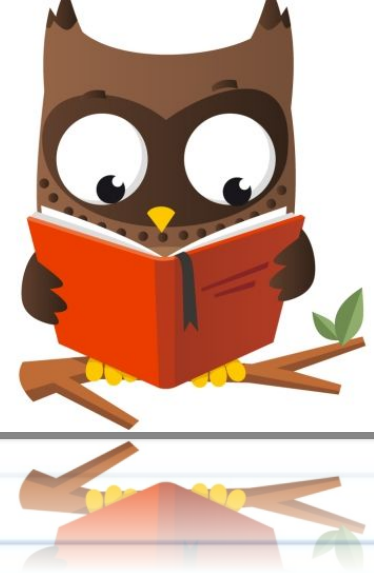
```
const store = createStore(myreducer);
```



React: Redux Action

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Codepen Redux Action Basics



```
const { createStore } = Redux;
```

```
const initState = {  
  todos: [],  
  posts: []  
};
```

```
function myreducer(state = initState, action) {  
  console.log(state, action);  
}
```

```
const store = createStore(myreducer);  
const todoAction = { type: "ADD_TODO", todo: "buy milk" };  
store.dispatch(todoAction);
```

React: Redux Reducers

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Codepen Redux Reducers

```
function myreducer(state = initState, action) {  
  if (action.type === "ADD_TODO") {  
    return {  
      ...state,  
      todos: [...state.todos, action.todo]  
    };  
  }  
}
```

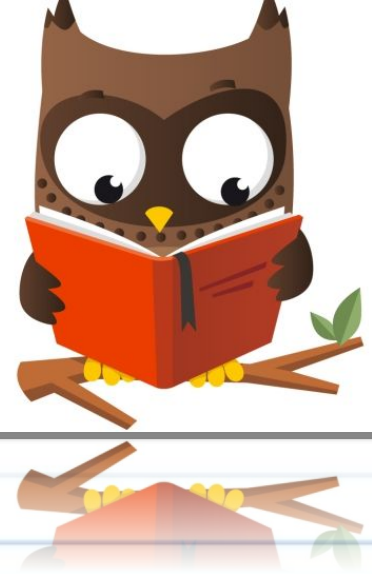
```
const store = createStore(myreducer);  
const todoAction = { type: "ADD_TODO", todo: "buy milk" };  
store.dispatch(todoAction);
```



Redux: Store Subscriptions

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Codepen Store Subscriptions



```
else if (action.type === "ADD_POST") {  
  return {  
    ...state,  
    posts: [...state.posts, action.post]  
  };  
}
```

```
}  
  
store.subscribe(() => {  
  console.log('state updated');  
  console.log(store.getState());  
});  
  
store.dispatch({ type: "ADD_TODO", todo: "buy milk" });  
store.dispatch({ type: "ADD_TODO", todo: "sleep some more" });  
store.dispatch({ type: "ADD_POST", todo: "Egg hunt with yoshi"  
});
```

React: Redux installieren

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

React: Redux installieren

- `npm install redux react-redux`
- Store in `index.js` generieren:

```
import { createStore } from 'redux'  
import { Provider } from 'react-redux'  
import rootReducer from './reducers/rootReducer'
```

```
const store = createStore (rootReducer);
```

```
ReactDOM.render(<Provider store={store}><App /></Provider>,  
document.getElementById('root'));
```

- Ordner `"reducers"` & `rootReducer.js` generieren

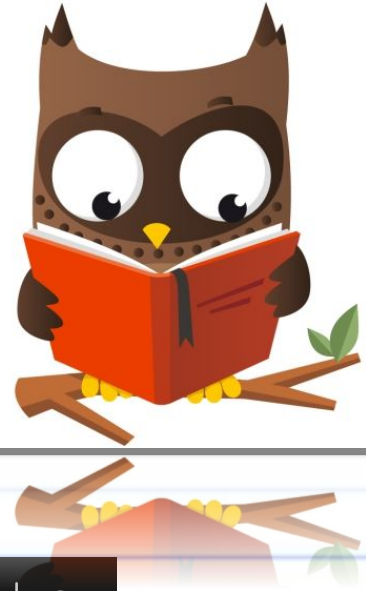


Aufsetzen des rootReducer.js

```
const initState = {  
  posts: []  
}
```

```
const rootReducer = (state = initState,  
  action) => {  
  return state;  
}
```

```
export default rootReducer
```




Redux: Zuordnen State zu Props

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

RootReducers initialer State

```
const initState = {  
  posts: [  
    {id: '1', title: 'Squirtle Laid an Egg', body: 'Lorem ipsum, dolor sit amet  
consectetur adipisicing elit. Consequatur voluptate laborum perferendis, enim  
repellendus ipsam sunt autem at odit dolorum, voluptatum suscipit iste harum cum  
magni itaque animi laudantium fugiat' },  
    {id: '2', title: 'Charmander Laid an Egg', body: 'Lorem ipsum, dolor sit amet  
consectetur adipisicing elit. Consequatur voluptate laborum perferendis, enim  
repellendus ipsam sunt autem at odit dolorum, voluptatum suscipit iste harum cum  
magni itaque animi laudantium fugiat' },  
    {id: '3', title: 'a Helix Fossil was Found', body: 'Lorem ipsum, dolor sit amet  
consectetur adipisicing elit. Consequatur voluptate laborum perferendis, enim  
repellendus ipsam sunt autem at odit dolorum, voluptatum suscipit iste harum cum  
magni itaque animi laudantium fugiat' }  
  ]  
}  
  
const rootReducer = (state = initState, action) => {  
  return state;  
}  
  
export default rootReducer
```



Home.js mit Higher Order connect!

```
import { connect } from 'react-redux'
```

```
const { posts } = this.props
```

```
const mapStateToProps = (state) => {
```

```
  return {
```

```
    posts: state.posts
```

```
  }
```

```
}
```

```
export default connect(mapStateToProps)(Home)
```



Redux: Blog Detail Seite

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Blog Post.js verändern mit Redux

```
import { connect } from 'react-redux'

class Post extends Component {
  render() {
    const post = this.props.post ? (
      <div className="post">
        <h4 className="center">{this.props.post.title}</h4>
        <p>{this.props.post.body}</p>

```

```
const mapStateToProps = (state, ownProps) => {
  let id = ownProps.match.params.post_id;
  return {
    post: state.posts.find(post => post.id === id)
  }
}
```

```
export default connect(mapStateToProps)(Post)
```



React & Redux: Action Delete

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Home.js mit Delete & Dispatch

```
handleClick = () => {  
  this.props.deletePost(this.props.post.id);  
  this.props.history.push('/');  
}
```

```
<button className="btn grey" onClick={this.handleClick}>  
  Delete Post  
</button>
```

```
const mapDispatchToProps = (dispatch) => {  
  return {  
    deletePost: (id) => dispatch({type: 'DELETE_POST', id: id})  
  }  
}
```

```
export default connect(mapStateToProps, mapDispatchToProps)(Post)
```



RootReducers delete Action

```
const rootReducer = (state = initState, action) => {  
  console.log(action);  
  if(action.type === 'DELETE_POST') {  
    let newPosts = state.posts.filter(post => {  
      return post.id !== action.id  
    });  
    return {  
      ...state,  
      posts: newPosts  
    }  
  }  
  return state;  
}
```



React & Redux: Action Creators

*Martina Freundorfer
Hochschule für Technik &
Wirtschaft (HTW) Berlin*

Neuer Ordner: "actions"

- Um besser lesbarer und wiederverwendbarer zu sein nehmen wir eine Abstraktion von Aktionen vor
- Hilft mit asynchronem Code später
- Und asynchronen Actions ebenfalls!
- Neues File:
 - `postActions.js` in diesem Ordner
- Kann am Anfang wie ein Overkill wirken, Ist es aber nicht!!!

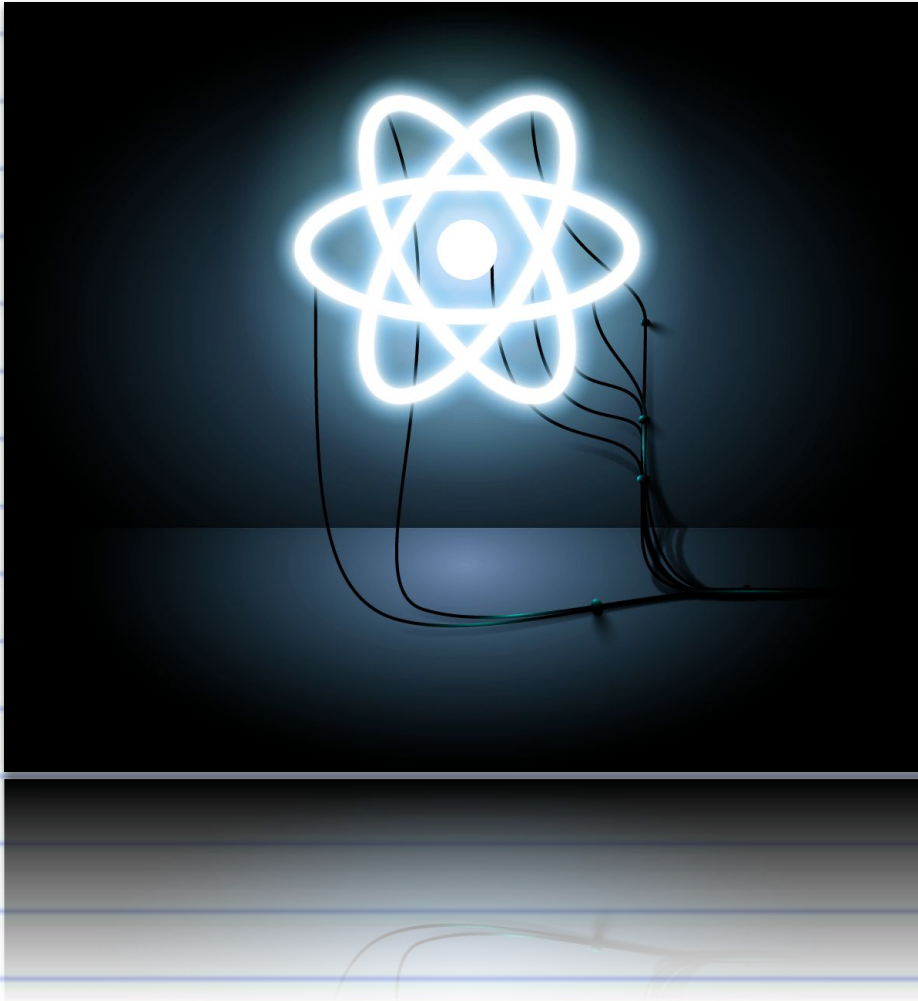


postActions.js

```
export const deletePost = (id) => {  
  return {  
    type: 'DELETE_POST',  
    id  
  }  
}
```



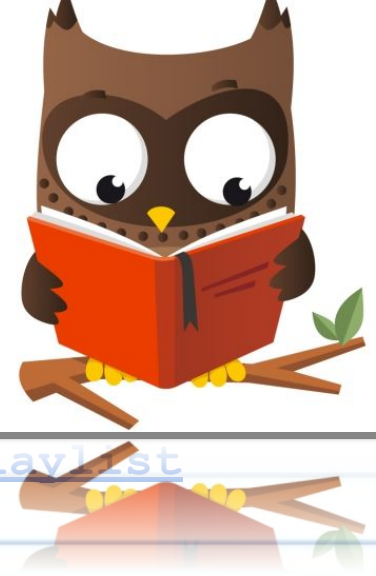
Benutze die Abstraktion im post.js



```
import { deletePost } from  
'../actions/postActions'
```

```
const mapDispatchToProps =  
(dispatch) => {  
  return {  
    deletePost: (id) =>  
      dispatch(deletePost(id))  
  }  
}
```

Credits & Mehr Lesestoff



Die Kursdateien für das React & Redux Tutorial

<https://github.com/htw-web/react-redux-complete-playlist>

Shaun "The Net Ninja" Pelling (Github: iamshaunjp)

<https://www.youtube.com/channel/UCW5YeuERMmlnqo4oq8vwUpq>

<https://github.com/iamshaunjp>

React lifecycle methods diagram (Github: wojtekmaaj)

<https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram>

Vielen Dank für Ihre Aufmerksamkeit!