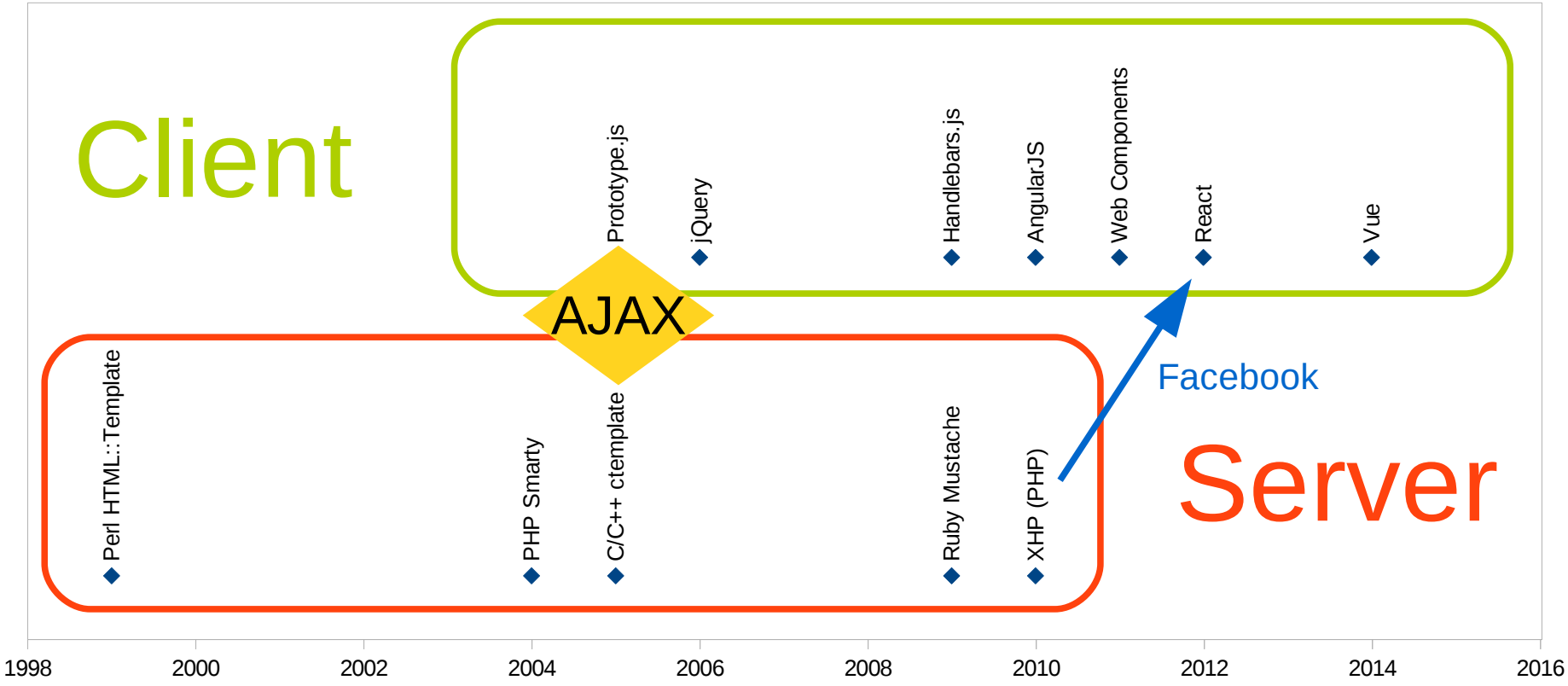
React Hooks

Funktional am Haken

Dipl.-Ing. Franz Knipp

Linuxwochen Eisenstadt, 27. April 2019

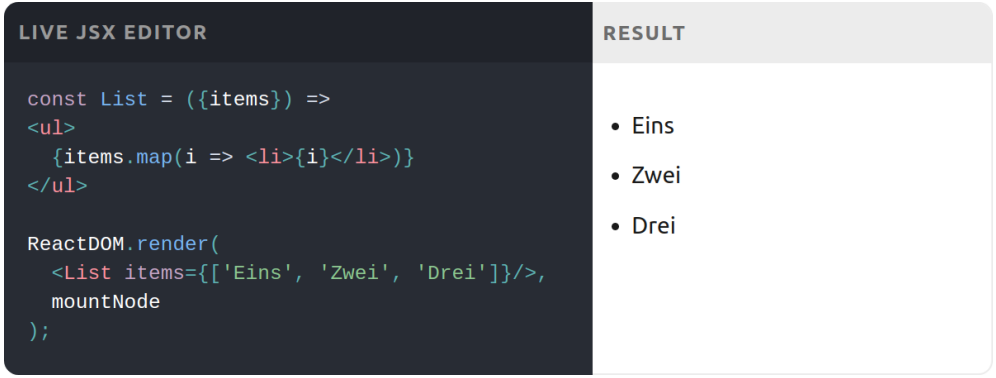
Wie kommen Inhalte auf die Webseite?



React

- Komponentenbasiertes Webframework
- HTML in JavaScript (JSX)
- Virtual DOM

- Facebook, Open Source seit 2013
- Aktuelle Version: 16.8.6
- > 127.000 Github stars
- https://reactjs.org/



Vor- und Nachteile

- Stabile Entwicklung
- Viele 3rd Party Komponenten
- React Native

- Übersetzung von JSX am Server erforderlich
- Performance und Dateigröße

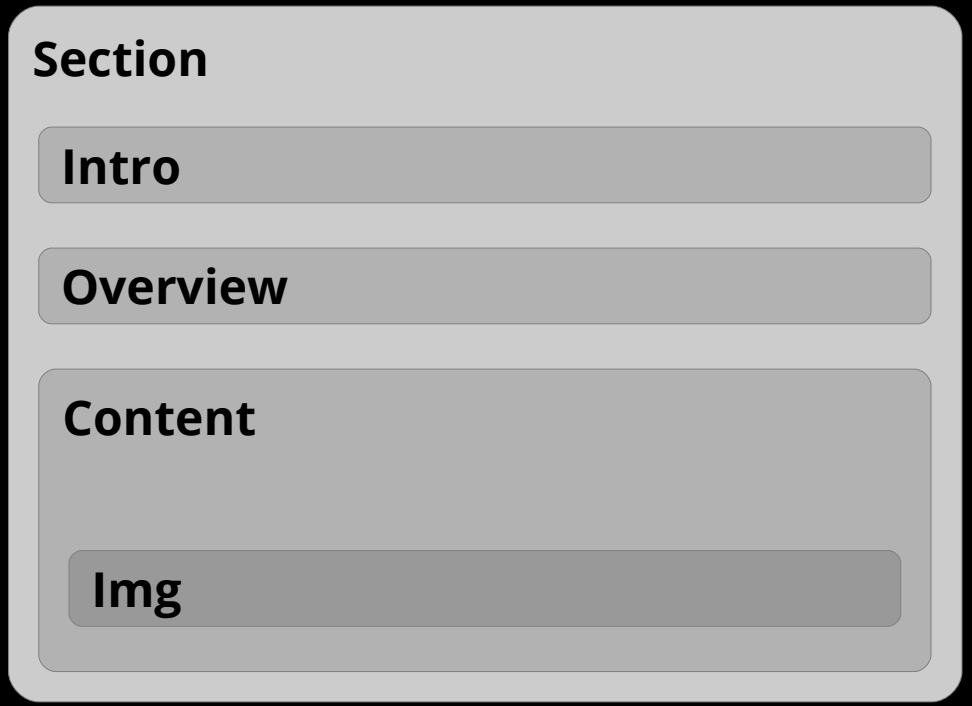
JSX

- Keine eigene Templating-Sprache erforderlich

- Code und Markup gemischt

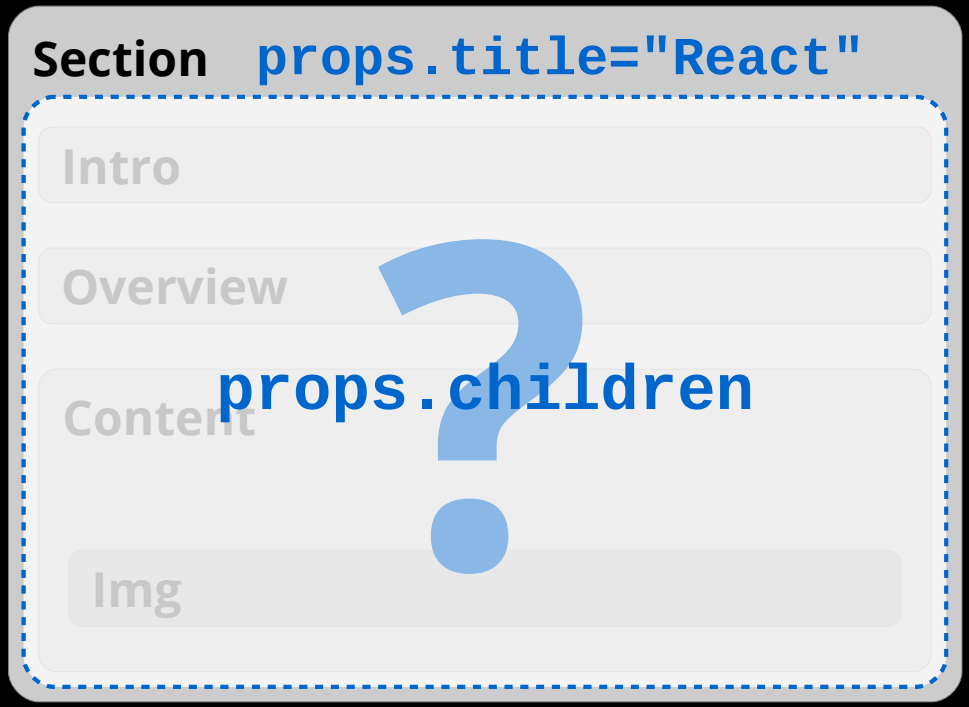
Aufbau mit Komponenten

```
<Section title="React">
  <Intro>React is fun!</Intro>
  <Overview />
  <Content>
    <p>Our contentful page.</p>
    <Img src="logo.png" />
  </Content>
</Section>
```



Der Datenfluss

```
<Section title="React">
  <Intro>React is fun!</Intro>
  <Overview />
  <Content>
    <p>Our contentful page.</p>
    <Img src="logo.png" />
  </Content>
</Section>
```



Wir bauen eine Komponente

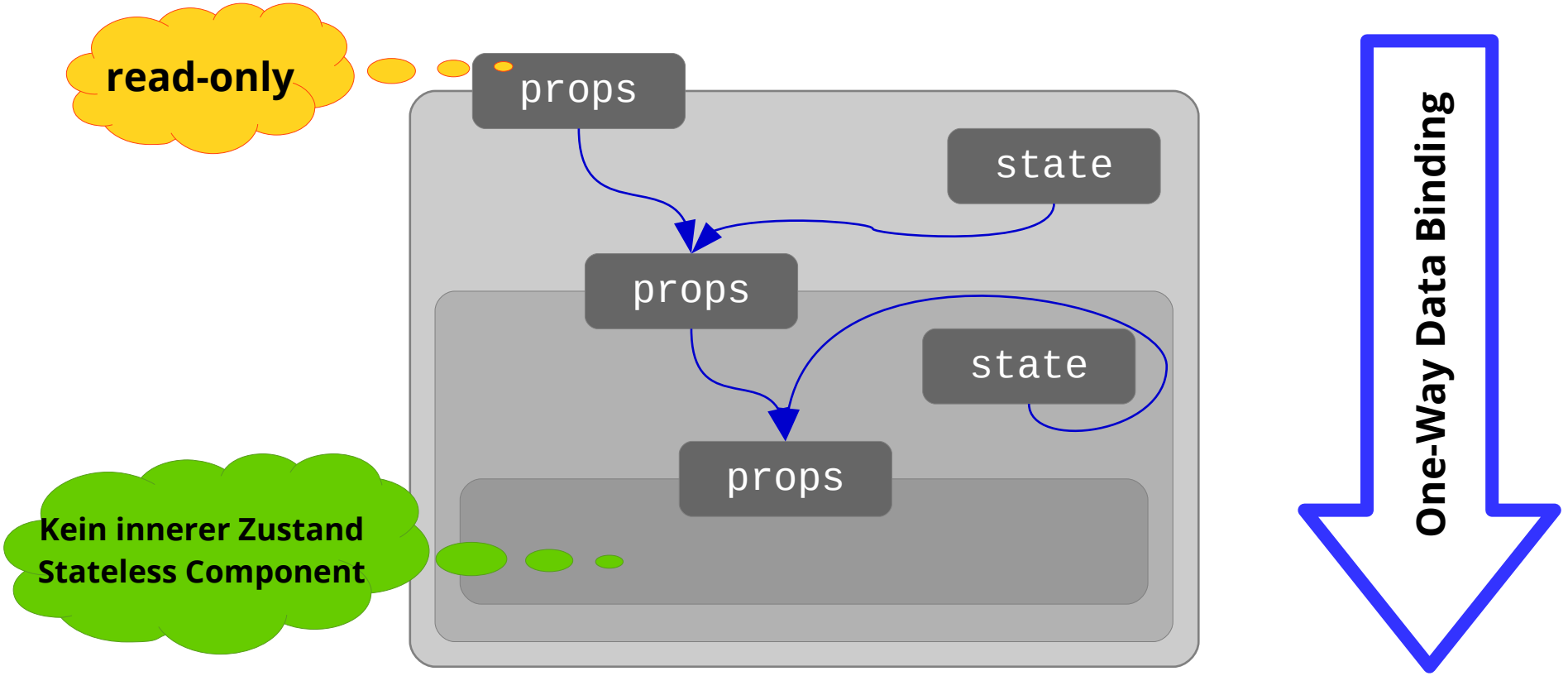
```
class Section extends React.Component {
  render() {
    return (
      <section>
        <h1>{this.props.title}</h1>
        {this.props.children}
      </section>
    );
  }
}
```

… oder als Funktion (ab Version 0.14.0, Oktober 2015)

```
const Section = (props) => (
  <section>
    <h1>{     props.title}</h1>
    {     props.children}
  </section>
);
```

```
const Section = ({ title, children }) => (
  <section>
    <h1>{title}</h1>
    {children}
  </section>
);
```

Woher kommen die Daten?



Eine Komponente mit Zustand

```
class ClickCounter extends React.Component {

  constructor() {

    super();

    this.state = { count: 0 };

  }


  handleClick(e) {

    this.setState(state => ({ count: state.count + 1}))

  }


  render() {

    return (<button onClick={this.handleClick.bind(this)}>Clicked {this.state.count} times</button>)

  }

}
```

Initialisieren des Zustands

Änderung mit `this.setState`

Abfrage mit `this.state`

https://react.qnipp.com/click-counter/

Separation of Concerns

```
const ClickButton = ({ count, onClick }) =>
  (<button onClick={onClick}>Clicked {count} times</button>)


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

  handleClick(e) {
    this.setState(state => ({ count: state.count + 1}))
  }

  render() {
    return (<ClickButton count={this.state.count} onClick={this.handleClick.bind(this)}/>);
  }
}
```

Presentational Component

Container Component

https://react.qnipp.com/click-counter-soc/

Und jetzt funktional! (ab Version 16.8.0, Februar 2019)

```
const ClickButton = ({ count, onClick }) =>
  (<button onClick={onClick}>Clicked {count} times</button>)


const ClickCounter = () => {
  const [ count, setCount ] = React.useState(0);


  const handleClick = (e)


  return (<ClickButton count={count} onClick={handleClick}/>);
}
```

Initialisieren des Zustands mittels `useState` Hook

Änderung

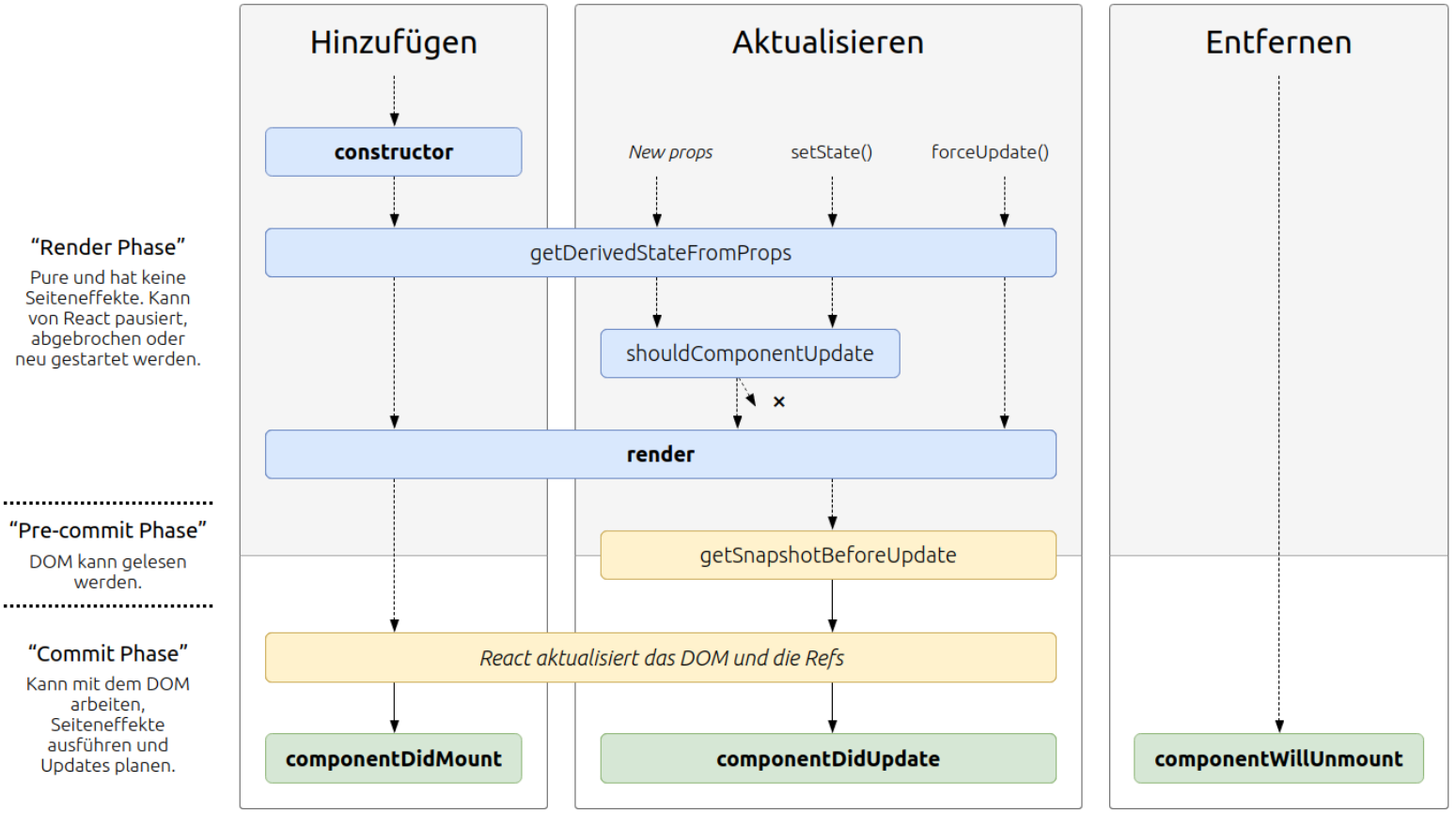
Abfrage

https://react.qnipp.com/click-counter-hook/

React Hooks bringen Vorteile

- Weniger Tipparbeit
- Bessere Wiederverwendbarkeit statt Container
- Übersichtlicher als Verteilung in den Lifecycle-Methoden
- Keine Verwendung mehr von Klassen für viele React-Features erforderlich

Der Lebenszyklus



Quelle: http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

Welche Hooks gibt es?

- **useState:** Zugriff auf den Zustand
- **useEffect:** Durchführung von Side Effects nach dem DOM-Update
  - componentDidMount, componentDidUpdate
  - componentWillUnmount
- **useContext:** Verwendung eines Context
  - React.Context: Informationsaustausch über Komponenten hinweg
- Eigene Hooks
  - starten mit **use**

Beispiel mit `setInterval`

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

  componentDidMount() {
    this.interval = setInterval(() => {
      this.setState(state => ({ count: state.count + 1 }))
    }, 1000)
  }

  componentWillUnmount() {
    clearInterval(this.interval)
  }

  render() {
    return (
      <center>
        <h3>
          {this.state.count}
        </h3>
      </center>
    )
  }
}
```

https://react.qnipp.com/set-interval

… und jetzt mit `useEffect`

```
const Counter = () => {

  const [ count, setCount ] = React.useState(0);

  React.useEffect(() => {
    const interval = setInterval(() => setCount(c => c + 1), 1000);
    console.log('New interval');
    return () => { clearInterval(interval); console.log('Delete interval') }
  }, []);

  return (
    <center>
      <h3>
        {count}
      </h3>
    </center>
  )
}
```

https://react.qnipp.com/set-interval-hook/

Data Fetching

```
const SearchSWAPI = () => {

  const [ searchWord, setSearchWord ] = React.useState("r2");
  const [ data, setData ] = React.useState([]);

  async function loadData() {
    console.log('Loading data');
    const res = await fetch(`https://swapi.co/api/people/?search=${searchWord}`);
    const data = await res.json();
    console.log(data);
    setData(data.results || []);
  }

  React.useEffect(() => {
    loadData();
  }, [ searchWord ]);

  return (
    <>
      <input value={searchWord} onChange={e => setSearchWord(e.target.value)}/>
      <ul>
        {data.map(entry => <li key={entry.url}>{entry.name}</li>)}
      </ul>
    </>
  )
}
```

https://react.qnipp.com/data-fetching-swapi/

React: Let's act!

- Website: https://reactjs.org/
- Hooks: https://reactjs.org/docs/hooks-intro.html

- create-react-app: https://github.com/facebook/create-react-app
- Next.js: https://nextjs.org/
- Gatsby (für statische Seiten): https://www.gatsbyjs.org/
- Meteor (komplettes Framework): https://www.meteor.com/

Mehr Interesse?

franz@qnipp.com

qnipp