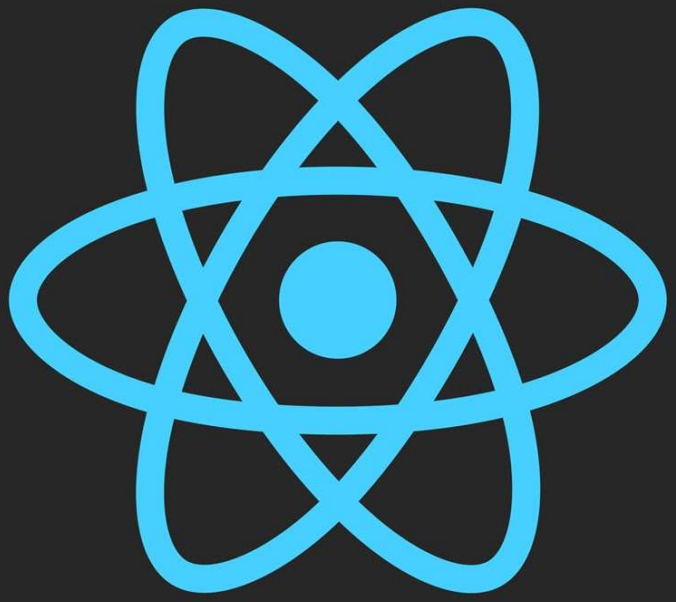
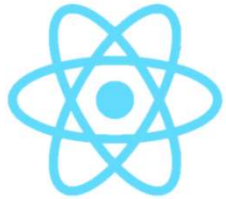


Fabian Rinner, Gregor Menz,
Patrick Ennemoser

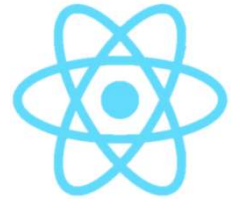
27.06.2018

Inhalt



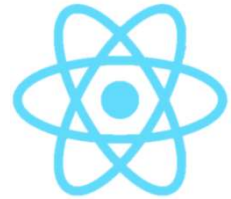
- Ursprung und Verbreitung
- Architektur
- Virtuelles DOM
- Live Demo
- Vor- und Nachteile
- Anwendungsbeispiel

Ursprung von React



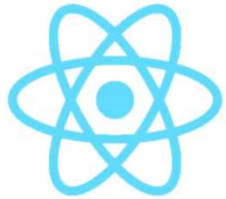
- 2011 von Facebook ins Leben gerufen
- Reine Oberflächenbibliothek
- JavaScript-Port, der XSS-Angriffe verhindern soll

Verbreitung



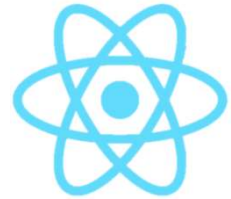
- Facebook
- Instagram
- Whatsapp Web
- AirBnB
- Netflix
- Imgur

Node.js als Umgebung



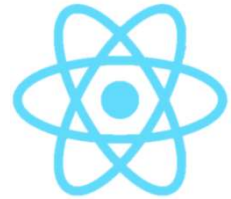
- Empfohlen von React
- JavaScript-basiertes Anwendungsframework
- Verwendet als Server und Paket Manager
- Just-In-Time-Kompilierung

Architektur



- create-react-app „Name“
- Projekt wird automatisch erstellt
  - package.json
  - index.html
  - index.css
  - index.js

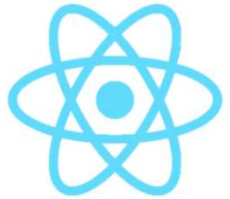
package.json



- Verwaltet dependencies für React
- Dependencies React und ReactDOM erst kürzlich geteilt
  - Grund: React Native
- ReactDOM Schnittstelle zwischen React und DOM
  - verwendet für die render Methode.
- React für alles andere
  - zum Definieren und Erstellen von Elementen

```
{
  "name": "vier",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.3.2",
    "react-dom": "^16.3.2",
    "react-scripts": "1.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

index.html und index.js



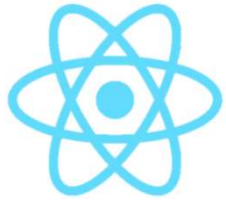
index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>React App</title>
  </head>
  <body>

    <div id="root"></div>

  </body>
</html>
```

index.js

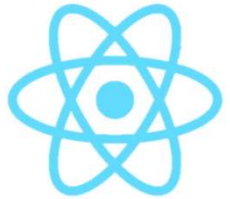
```
import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
    <h1>Hello, world!</h1>,
    document.getElementById('root')
);
```

JSX Einführung



- Syntaxerweiterung für JavaScript
- JSX erinnert an HTML, aber es kommt mit der vollen Leistungsfähigkeit von JavaScript
- JSX produziert React "Elemente“
- Warum JSX?
  - visuelle Hilfe beim Arbeiten mit der Benutzeroberfläche
  - Fehler- und Warnmeldungen
  - verhindert XSS-Angriffe (Cross-Site-Scripting).

JSX Beispiel



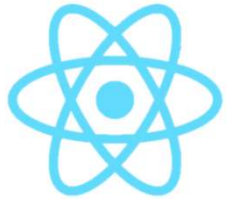
Mit JSX

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}

ReactDOM.render(
  <HelloMessage name="Taylor" />,
  mountNode
);
```

Ohne JSX

```
class HelloMessage extends React.Component {
  render() {
    return React.createElement(
      "div",
      null,
      "Hello ",
      this.props.name
    );
  }
}

ReactDOM.render(React.createElement(HelloMessage, { name: "Taylor" }), mountNode);
```

Komponenten

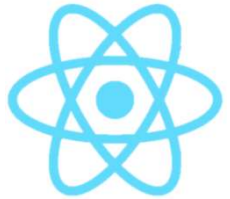


- Benutzeroberfläche in wiederverwendbare Teile aufteilen
- Functional und Class Components

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
```

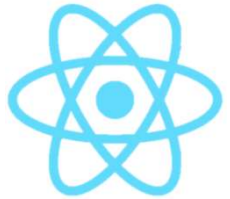
```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Virtuelles DOM



- abstrakte Kopie des tatsächlichen DOMs
- deutlich kleiner und nur das Nötigste an Informationen
- verglichen mit dem richtigen DOM

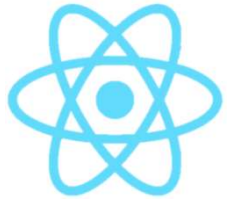
Properties/props



- read only
- props können nicht verändert werden

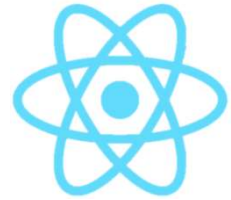
```
1  import React from 'react';
2  import ReactDOM from 'react-dom';
3
4  class HelloMessage extends React.Component {
5      render() {
6        return <h1>Hallo {this.props.name}</h1>;
7      }
8    }
9    ReactDOM.render(<HelloMessage name="Welt"/>,document.getElementById('root'));
```

States



- state ist ähnlich zu einer prop
- kontrolliert von der Komponente
- local states nur für Klassen Komponenten verfügbar
- states werden mit einem Konstruktor erstellt
- nur mit der Methode „setState“ veränderbar

lifecycle hooks

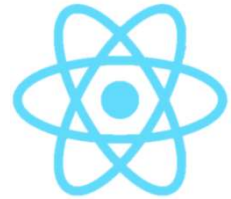


- render
- constructor
- componentDidMount
- componentDidUpdate
- componentWillUnmount
- shouldComponentUpdate

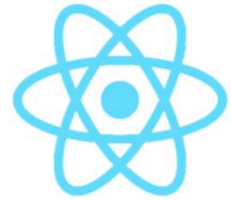
4 Zeitpunkte:

1. Beim ersten rendern
2. Erhalt einer props
3. Änderung eines states
4. Unmount

Live Demo

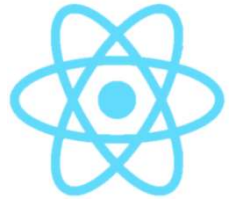


Vorteile



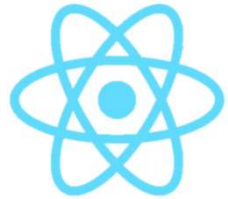
- einfach zu lernen für fortgeschrittene JavaScript Programmierer
- gute Dokumentation
- wiederverwendbare Komponenten
- The Virtual DOM
- React Developer Tools
- JSX

Nachteile



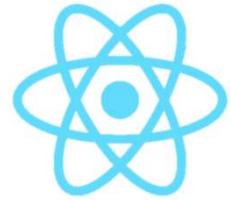
- schlechte Dokumentation für neue Tools
- Programmier-Anfänger haben eine sehr langsame Lernkurve
- nur eine Library für User Interfaces
- umständlich in ein MVC Model umzuwandeln

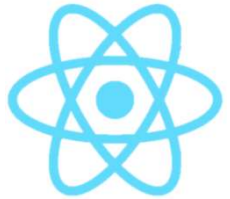
Quellen



- https://reactjs.org/
- https://www.softed.de/blog/react-naeher-angeschaut/
- https://www.infoq.com/articles/more-than-react-part-i
- https://scotch.io/@anitashah/what-problems-does-reactjs-solve-when-must-you-select-reactjs

Anwendung





Vielen Dank für eure Aufmerksamkeit