

Table of Contents

Introduction	0
Basics	1
React	1.1
ES6	1.2
npm	1.3
First steps	2
Base React application	2.1
Development notes	2.2
Include react-geo dependency	2.3
Include a react-geo component	2.4
A taste of react-geo components	3
MapComponent	3.1
Titlebar	3.2
NominatimSearch	3.3
MeasureButton	3.4
LayerTree	3.5
Higher order components / Provider	4
MapProvider / `mappify`	4.1
VisibleComponent / `isVisibleComponent`	4.2



Workshop *react-geo - mapping mit React*

Welcome to the **react-geo - mapping mit React** workshop. This workshop is designed to give you a comprehensive overview of [react-geo](#) as a library of geo-related application components available in combination with [React](#), [antd](#) and [OpenLayers](#).

If you want to visit this page on your own device or to print the PDF version, you can download the workshop materials [here](#).

Setup

The following instructions and exercises assume that you have some requirements fulfilled on your local machine. Please check if you have the consequent packages installed:

- A suitable text editor, e.g. the lightweight [Atom](#) editor.
- [NodeJS](#) in version 8 or higher.

All set? Then, lets' go!

Overview

This workshop is presented as a set of modules. In each module you will perform tasks designed to achieve a specific goal for that module. Each module builds upon lessons learned in previous modules and is designed to iteratively build up your knowledge base.

- [Basics](#) - Dive into the basics of EcmaScript 6, React and npm.
- [First steps](#) - Learn how to create your own React app and how to include react-geo in it.
- [A taste of react-geo components](#) - Extend your application with some react-geo components.
- [Higher order components / Provider](#) - Have a look at more advanced components.

Authors

- André Henn (henn@terrestris.de)
- Daniel Koch (koch@terrestris.de)
- Kai Volland (volland@terrestris.de)

Basics

React

- von facebook

JSX

- Geschwungene Klammern

First steps

Now that we have set up our development setup and learned the basics about React and EcmaScript 6, we will start by creating a simple React based webapplication by the use of [create-react-app](#), that will include a simple react-geo component. This application will be extended towards a fully functional mapping application little by little later on.

- [Base React application](#)
- [Development notes](#)
- [Include react-geo dependency](#)
- [Include a react-geo component](#)

First steps

As a matter of course we could start this workshop by creating a React based webapplication by hand, but as you could imagine this would be a tough job for starters. So we want to dive into react-geo directly without the need to stick together all development tools to get a webapp running. Thankfully there is a project available, that we can use to generate an application for us (even without any configuration!): [create-react-app](#).

Creating a new application is easy. Just navigate to a folder of your choice and create a new app named *my-app* inside this directory with:

```
npx create-react-app my-app
```

This will take while, but finally you will see a list of commands you can run inside the created folder. Now switch to the project's folder with:

```
cd my-app
```

Finally we can start the development server with:

```
npm start
```

To view the application in your browser please open <http://localhost:3000/>.

Webpack erklären

- App.js editieren --> hotreload



```
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" />
          <h1 className="App-title">Hallo Welt!</h1>
        </header>
        <p className="App-intro">
          Hallo Welt!
        </p>
      </div>
    );
  }
}

export default App;
```

Include react-geo dependency

`react-geo` is published at <https://www.npmjs.com/> and can be integrated and installed in your *my-app* application via basic `npm` commands.

Add react-geo dependency

To add the dependency `react-geo` please navigate to your project's folder (if not already done) and execute:


```
npm i @terrestris/react-geo
```

This will add the latest version of `react-geo` to your local `package.json` file (into the `dependencies` section) and download the distributed version of the library to the `node_modules` directory.

Add Ant Design und OpenLayers dependencies

You may have noticed that the step from above has produced some warnings, which include `react-geo` :

```
npm WARN @terrestris/react-geo@5.6.0 requires a peer of antd@~3.0 but none
npm WARN @terrestris/react-geo@5.6.0 requires a peer of ol@~4.0 but none
```

A screenshot of a terminal window showing the output of the npm install command. It displays two warning messages: 'npm WARN @terrestris/react-geo@5.6.0 requires a peer of antd@~3.0 but none' and 'npm WARN @terrestris/react-geo@5.6.0 requires a peer of ol@~4.0 but none'. The terminal has a light gray background and a dark gray border.

`npm` provides several techniques to express the type of a dependency of a project. While `dependencies` are used to directly specify packages needed to *run* your application's code (e.g. a front-end library like `Bootstrap`), `devDependencies` are reserved to specify packages needed to *build* your application's code (e.g. test harnesses like `Jest` or transpilers like `Babel`). However, under some conditions, one wants to express the *compatibility* of a certain package with the host package and npm calls this dependency a `peerDependencies`. Usually this is used to express the dependency of a plugin inside this host package or similar. In `react-geo` we need to have `antd`, `ol` and `react` defined as peer dependencies due to scope issues, because all of them were usually referenced by the host package/the application itself in a certain version. As `npm` handles dependencies hierarchically including those packages in `react-geo` twice would lead to two different dependencies available in your application at runtime. To share the dependencies between your host application and `react-geo`, we advice `react-geo` to use the dependencies given by the host package.

To met these requirements we have to install the requested peer dependencies by ourselves with:

```
npm i antd ol
```

Now we're ready to make use of all `react-geo` components and utilities inside our *my-app* application.

Erste Komponente

SimpleButton

Ein einfacher Button. [Beispiele](#)

```
<SimpleButton onClick={() => {alert('huhu')}}/>
```

Zur Verwendung des `SimpleButton` muss die entsprechende Klasse importiert werden.

Um die Komponenten korrekt darzustellen müssen die stylesheets von Ant Design und react-geo importiert werden.

Hinweis auf import syntax (css, svg, ...).



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

import './App.css';
import 'antd/dist/antd.css';
import './react-geo.css';

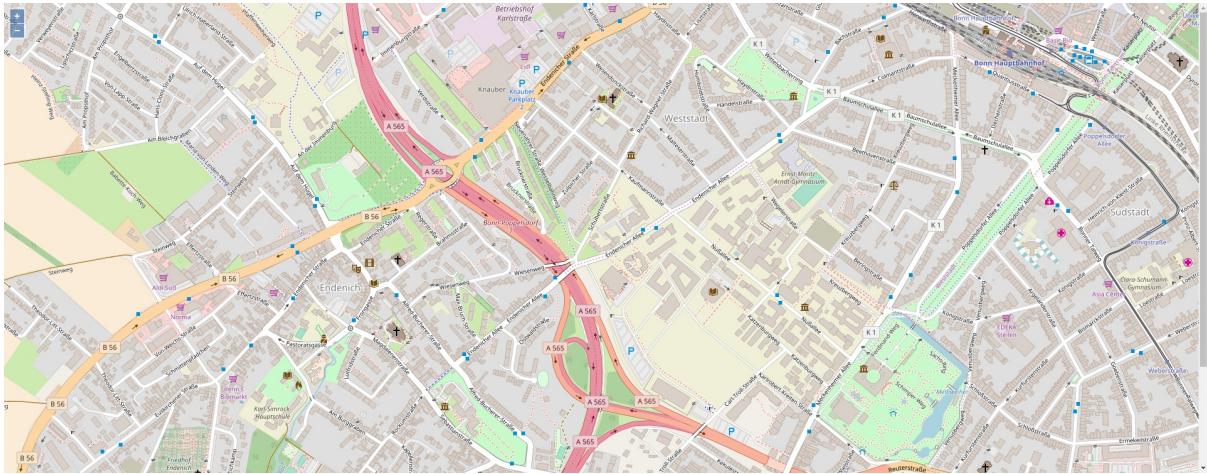
import {
  SimpleButton
} from '@terrestris/react-geo';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Hallo Welt!</h1>
        </header>
      </div>
    );
  }
}
```

```
        <p className="App-intro">
          <SimpleButton
            onClick={() => {alert('huhu')}}
          >
            Hallo Welt!
          </SimpleButton>
        </p>
      </div>
    );
  }
}
```

Kartenapplikation mit react-geo Komponenten

MapComponent



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

import './App.css';
import 'ol/ol.css';
import 'antd/dist/antd.css';
import './react-geo.css';

import OlMap from 'ol/map';
import OlView from 'ol/view';
import OlLayerTile from 'ol/layer/tile';
import OlSourceOsm from 'ol/source/osm';

import {
  MapComponent
} from '@terrestris/react-geo';

const layer = new OlLayerTile({
  source: new OlSourceOsm()
});

const center = [ 788453.4890155146, 6573085.729161344 ];

const map = new OlMap({
  view: new OlView({
    center: center,
    zoom: 16,
  }),
  layers: [layer]
});

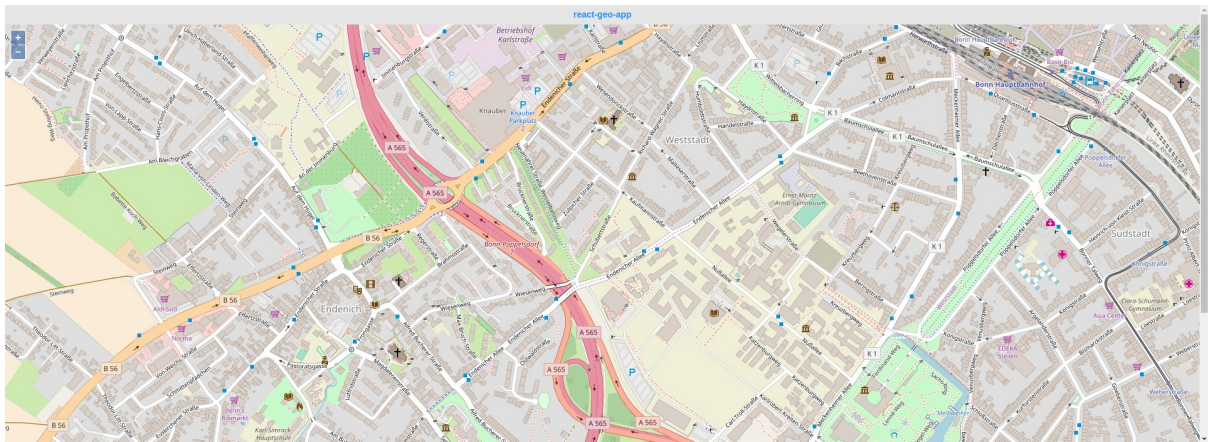
class App extends Component {
  render() {
    return (
```



```
    <div className="App">
      <MapComponent
        map={map}
      />
    </div>
  );
}
}

export default App;
```

Titlebar



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

import './App.css';
import 'ol/ol.css';
import 'antd/dist/antd.css';
import './react-geo.css';

import OlMap from 'ol/map';
import OlView from 'ol/view';
import OlLayerTile from 'ol/layer/tile';
import OlSourceOsm from 'ol/source/osm';

import {
  MapComponent,
  Titlebar
} from '@terrestris/react-geo';

const layer = new OlLayerTile({
  source: new OlSourceOsm()
});

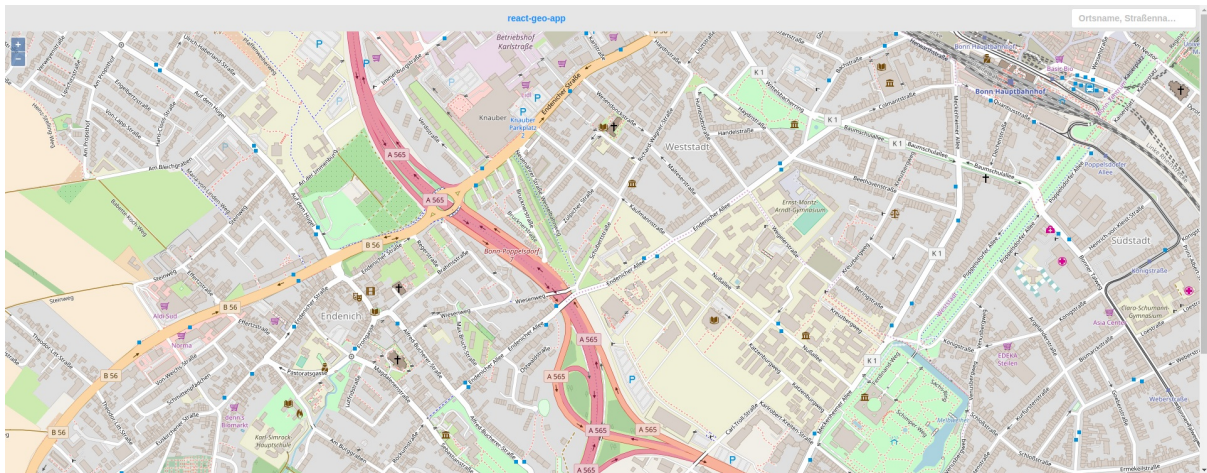
const center = [ 788453.4890155146, 6573085.729161344 ];

const map = new OlMap({
  view: new OlView({
    center: center,
    zoom: 16,
  }),
  layers: [layer]
});

class App extends Component {
```

```
render() {  
  return (  
    <div className="App">  
      <Titlebar className="titlebar">  
        react-geo-app  
      </Titlebar>  
      <MapComponent  
        map={map}  
      />  
    </div>  
  );  
}  
}  
  
export default App;
```

NominatimSearch



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

import './App.css';
import 'ol/ol.css';
import 'antd/dist/antd.css';
import './react-geo.css';

import OlMap from 'ol/map';
import OlView from 'ol/view';
import OlLayerTile from 'ol/layer/tile';
import OlSourceOsm from 'ol/source/osm';

import {
  MapComponent,
  NominatimSearch,
  MeasureButton,
  Titlebar
} from '@terrestris/react-geo';

const layer = new OlLayerTile({
  source: new OlSourceOsm()
});

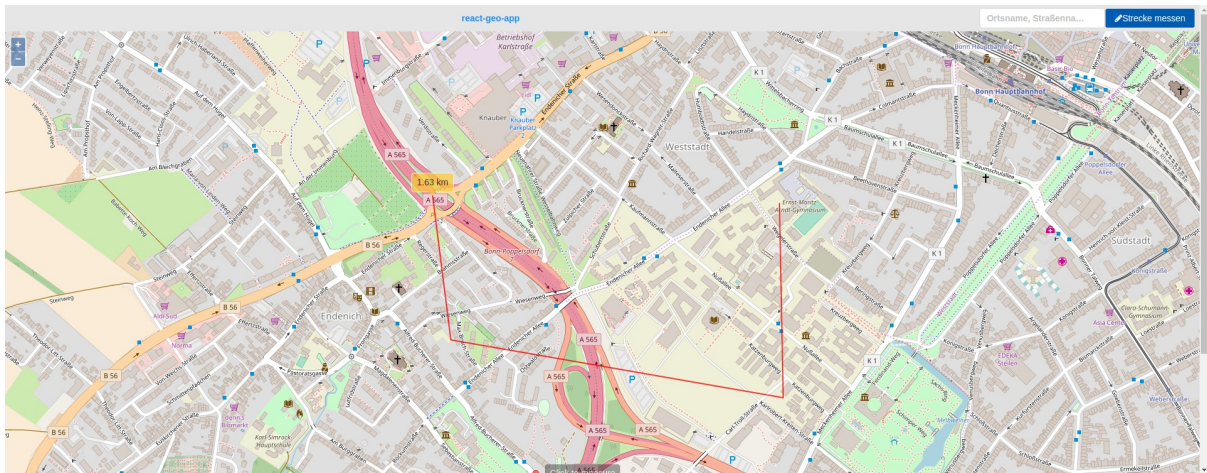
const center = [ 788453.4890155146, 6573085.729161344 ];

const map = new OlMap({
  view: new OlView({
    center: center,
    zoom: 16,
  }),
  layers: [layer]
});
```

```
class App extends Component {
  render() {
    return (
      <div className="App">
        <Titlebar className="titlebar" tools={[
          <NominatimSearch
            key="search"
            map={map}
          />,
          <MeasureButton
            key="measureButton"
            name="line"
            map={map}
            measureType="line"
          >
        ]}>
          react-geo-app
        </Titlebar>
        <MapComponent
          map={map}
        />
      </div>
    );
  }
}

export default App;
```

MeasureButton



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

import './App.css';
import 'ol/ol.css';
import 'antd/dist/antd.css';
import './react-geo.css';

import OlMap from 'ol/map';
import OlView from 'ol/view';
import OlLayerTile from 'ol/layer/tile';
import OlSourceOsm from 'ol/source/osm';

import {
  MapComponent,
  NominatimSearch,
  MeasureButton,
  Titlebar
} from '@terrestris/react-geo';

const layer = new OlLayerTile({
  source: new OlSourceOsm()
});

const center = [ 788453.4890155146, 6573085.729161344 ];

const map = new OlMap({
  view: new OlView({
    center: center,
    zoom: 16,
  }),
  layers: [layer]
});
```

```
class App extends Component {
  render() {
    return (
      <div className="App">
        <Titlebar className="titlebar" tools={[
          <NominatimSearch
            key="search"
            map={map}
          />,
          <MeasureButton
            key="measureButton"
            name="line"
            map={map}
            measureType="line"
            icon="pencil"
          >
            Strecke messen
          </MeasureButton>
        ]}>
          react-geo-app
        </Titlebar>
        <MapComponent
          map={map}
        />
      </div>
    );
  }
}

export default App;
```

```
<LayerTree
  map={map}
/>
```

Verwendung einer [OpenLayers LayerGroup](#)

```
const layerGroup = new OlLayerGroup({
  name: 'Layergroup',
  layers: [
    new OlLayerTile({
      name: 'Food insecurity layer',
      minResolution: 200,
      maxResolution: 2000,
      source: new OlSourceTileJson({
        url: 'https://api.tiles.mapbox.com/v3/mapbox.20110804-hoa-foodinsecurity-3mon',
        crossOrigin: 'anonymous'
      })
    }),
    new OlLayerTile({
      name: 'World borders layer',
      minResolution: 2000,
      maxResolution: 20000,
      source: new OlSourceTileJson({
        url: 'https://api.tiles.mapbox.com/v3/mapbox.world-borders-light.json?secure',
        crossOrigin: 'anonymous'
      })
    })
  ]
});

<LayerTree
  layerGroup={layerGroup}
  map={map}
/>
```

Weitere Beispiele [hier](#)

Higher order components

- siehe facebook.

MapProvider / mappify

- erläuterungen....

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

import './App.css';
import 'ol/ol.css';
import 'antd/dist/antd.css';
import './react-geo.css';

import OlMap from 'ol/map';
import OlView from 'ol/view';
import OlLayerTile from 'ol/layer/tile';
import OlSourceOsm from 'ol/source/osm';

import {
  MapComponent,
  NominatimSearch,
  MeasureButton,
  Titlebar,
  MapProvider,
  mappify
} from '@terrestris/react-geo';

const MappifiedNominatimSearch = mappify(NominatimSearch);
const MappifiedMeasureButton = mappify(MeasureButton);
const Map = mappify(MapComponent);

const layer = new OlLayerTile({
  source: new OlSourceOsm()
});

const center = [ 788453.4890155146, 6573085.729161344 ];

const map = new OlMap({
  view: new OlView({
    center: center,
    zoom: 16,
  }),
  layers: [layer]
});

class App extends Component {
  render() {
    return (
      <div className="App">
        <MapProvider map={map}>
```

```
    <Titlebar className="titlebar" tools={[
      <MappifiedNominatimSearch
        key="search"
      />,
      <MappifiedMeasureButton
        key="measureButton"
        name="line"
        measureType="line"
        icon="pencil"
      >
        Strecke messen
      </MappifiedMeasureButton>
    ]}>
    react-geo-app
  </Titlebar>
  <Map/>
</MapProvider>
</div>
);
}
}

export default App;
```

VisibleComponent / isVisibleComponent

```
import React from 'react';
import { render } from 'react-dom';
import { Button } from 'antd';
import { isVisibleComponent } from '../index.js';

// Enhance (any) Component by wrapping it using isVisibleComponent().
const VisibleButton = isVisibleComponent(Button);

// The activeModules is a whitelist of components (identified by it's names) to
// render.
const activeModules = [{
  name: 'visibleButtonName'
}, {
  name: 'anotherVisibleButtonName'
}];

render(
  <div>
    <VisibleButton
      name="visibleButtonName"
      activeModules={activeModules}
      type="primary"
      shape="circle"
      icon="search"
    />
    <VisibleButton
      name="notVisibleButtonName"
      activeModules={activeModules}
      type="primary"
      shape="circle"
      icon="search"
    />
    <VisibleButton
      name="anotherVisibleButtonName"
      activeModules={activeModules}
      type="primary"
      shape="circle"
      icon="poweroff"
    />
  </div>,
  document.getElementById('exampleContainer')
);
```

