Webbasierte und interaktive Darstellung von Kartendaten

webbasierte und interaktive Darstellung von Kartendaten

[expand title="mehr lesen…"]

Für eine webbasierte und interaktive Darstellung von Kartendaten eignet sich die Kombination aus SVG-Standard und React-Framework.

-tract

- Um eine Karte interaktiv zu gestalten, sind mehrere Schritte nötig. Das Open-Source-Webframework React dient als Grundlage für das Erstellen komponentenbasierter Webanwendungen.
- Das Tool overpass turbo des freien Geodatenprojekts OpenStreetMap hilft, die Kartendaten zu extrahieren. gpx2svg wandelt sie in SVG um.
- Die Bibliothek Emotion bringt CSS an die React-Komponenten.

Berlin als Großstadt überzeugt mit vielfältigem Leben in den Bezirken. Um dieses Spektrum im Web abzubilden, bieten sich interaktive Karten an. Die geografischen Umrisse der Stadtteile lassen sich so mit Statistiken und Metriken verbinden. Dies nutzen vor allem Immobilienfirmen in Analysedashboards oder stadteigene Anwendungen.

Die Beispiele des Artikels sind mit dem Webframework React erstellt, das sich für das Programmieren interaktiver

Webanwendungen eignet. Es nutzt HTML direkt in JavaScript mit JSX (JavaScript XML).

Neben React kommt der Grafikstandard SVG zum Einsatz, der sich für Kartenanwendungen gut eignet. 1998 von der W3C--Arbeitsgruppe als offener Standard entwickelt, beschreibt er als Vektoren dargestellte Grafiken. Eine Grafik lässt sich damit ohne Darstellungsverluste vergrößern. SVG ist ein von Maschinen und Menschen lesbares Format, das in HTML eingefügt und so ein integraler Bestandteil einer Webseite wird. Die SVG-Grafik ist zudem animierbar.

Die Umrissdaten der Bezirke von Berlin liefert das freie Geodatenprojekt OpenStreetMap. 2004 von Steve Coast initiiert, verfügt es über umfangreiches globales Kartenmaterial. Alle Daten aus OpenStreetMap stehen unter der Open Database License – ODbL frei zur Verfügung. Mit dem OpenStreetMap-Werkzeug overpass turbo können Anwender die Overpass-API abfragen – eine schreibgeschützte API, die benutzerdefinierte ausgewählte Teile der OpenStreetMap-Kartendaten bereitstellt. So extrahieren und filtern sie gewünschte Daten mit einer Abfragesprache. Die gewonnenen Daten lassen sich in verschiedenen Dateiformaten exportieren.

Die Abfragesprache von overpass turbo nennt sich Overpass Query Language, kurz Overpass QL. Jede Angabe schließen Anwender mit einem Semikolon ab. Listing 1 zeigt die Overpass-QL-Abfrage für eine Umrisszeichnung der Bezirke von Berlin. Am einfachsten generieren Anwender das Listing mit dem Wizard, indem sie boundary=administrative and admin_level=9 eingeben und ausführen. Die Variable admin_level mit dem Wert 9 bezeichnet in deutschen Städten die Bezirksebene.

Listing 1: Overpass-QL-Abfrage der Berliner Bezirke

```
[out:json][timeout:25];
{{geocodeArea:Berlin}}->.searchArea;
```

```
    (
      // query part for: "boundary=administrative and
admin_level=9"
node["boundary"="administrative"]["admin_level"="9"](area.sear
chArea);
way["boundary"="administrative"]["admin_level"="9"](area.searc
hArea);
relation["boundary"="administrative"]["admin_level"="9"](area.
searchArea);
);
// print results
out body;
>;
out skel qt;
```

Leider steht im Exporter von overpass turbo das benötigte SVG-Format nicht zur Verfügung. Deshalb sind zunächst die Daten als GPS Exchange Format, GPX, zu exportieren und herunterzuladen. Zum Umwandeln der Daten von GPX in SVG bietet sich das Werkzeug gpx2svg von Tobias Leupold an (siehe ix.de/zxex). Zum Ausführen benötigt man Python 3 und die GPX-Datei. Das folgende Kommando wandelt die Bezirksumrissdatei von GPX in SVG um:

```
python3 gpx2svg -i berlin.gpx -o berlin.svg
```

Nun kann die SVG-Karte schon mit gängigen Bildbearbeitungsprogrammen wie Inkscape oder Adobe Illustrator angesehen und wenn gewünscht weiterverarbeitet werden. Für das Web braucht es aber ein wenig Vorarbeit.

Vorbereitungen treffen

Zunächst gilt es, Node.js und npm auf dem Rechner zu installieren. Mit npm können Entwickler benötigte JavaScript-Softwarebibliotheken verwalten. Es empfiehlt sich, Node.js und npm mit einem Node Version Manager zu installieren. Versionsupdates von Node.js sind damit durchführbar. Auch zum Testen der Anwendung unter verschiedenen Node.js-Versionen eignet sich ein Versionsmanager gut: für Linux und Mac OS X

etwa nvm, für Windows nvm-windows. Das verwendete Beispiel hier nutzt die neueste LTS-Version von Node.js, 12.18.3. LTS steht für Long Term Support und ist die stabile und empfohlene Version von Node.js, die auch für den produktiven Einsatz geeignet ist.

Als Nächstes ist der Paketmanager Yarn in Version 1 zu installieren, der ähnlich wie NPM arbeitet und damit kompatibel ist. Man muss nur aufpassen, NPM und Yarn nicht in einem Projekt zusammen zu verwenden.

Eine gute Möglichkeit, schnell eine neue lauffähige React-Anwendung zu erstellen, bietet das Paket create-react-app. Es fertigt automatisch ein gutes Grundgerüst der neuen Anwendung an. Darauf aufbauend lässt sich die Anwendung um eine neue Komponente erweitern. Der folgende Befehl erstellt eine neue React-Anwendung mit dem Namen react-district-map-berlin:

```
npx create-react-app react-district-map-berlin
```

npx führt das angegebene Paket create-react-app automatisch aus, in diesem Fall mit dem Parameter react-district-map-berlin, dem Namen der Anwendung. Die Bibliothek Emotion setzt das Styled-Components-Muster in React um. Damit fügen Entwickler an React-Komponenten den CSS-Stil hinzu. Die benötigten Pakete der Bibliothek, @emotion/styled und @emotion/core, installieren Entwickler mit Yarn:

```
yarn add @emotion/styled @emotion/core --save
```

Die nun erzeugte Anwendung hat den Eintrittspunkt in der Datei src/index.js, die Hauptkomponente App.js ist erstellt. Es gilt jetzt, eine neue Komponente BerlinMap im Verzeichnis src/components anzufertigen. Diese Komponente wird später in die Datei App.js importiert, um sie im Programm nutzen zu können. Listing 2 zeigt einen Ausschnitt der Berlin-Kartenkomponente.

Listing 2: Die Anwendung mit Funktionskomponente und eingefügter SVG-Datei

```
import React from "react";
import styled from "@emotion/styled";

function BerlinMap(props) {
    const handleMouseOver = (evt) => {
      for (const attr of evt.target.attributes) {
        if (attr.name === "name") {
          props.callback(attr.value);
        }
      }
    };
    return (
      <>
        <svg
          xmlns="http://www.w3.org/2000/svg"
          id="svg626"
          viewBox="0 0 3001.18 2470"
          version="1.1"
          style={{ width: "50%" }}
        >
          <defs id="defs630" />
          <g id="g624">
            <Borough
              id="path598"
                 d="m 575.41788,646.71475 -4.2765,12.94153
-2.30212,3.5527 -2.4876,3.44405          -0.83693,3.14533
0.15517,3.36692  1.00458,2.55306  1.38493,1.90799  1.83438,
2.20371  4.82182,4.28924  2.10949,3.35063  1.67163,3.41958
1.00726,3.61913   d-1.2846,9.32649   -0.22294,8.49046
3.96037,10.2116  2.79393,3.71342 4.05623,5.07077
d...
  d3.29064,6.03348  1.75457,7.10791  -1.68679,6.8105
-3.12522,10.42868 z"
              name="Reinickendorf"
              onMouseOver={handleMouseOver}
            />
[...]
</g>
```

```
        </svg>
      </>
    );
}

const Borough = styled.path`
    fill: #000080;
    stroke: yellow;
    :hover {
      fill: red;
    }
`;
```

Die Datei besteht aus einem Importteil, der Funktionskomponente BerlinMap und einer CSS-Stildefinition als Styled Component (mit Emotion). Entwickler fügen die generierte SVG-Datei in die React-Komponente ein und passen sie an die Bedürfnisse der Anwendung an. Dabei bauen sie den Inhalt der SVG-Datei zwischen einem Fragment (<></>) ein. Um das Fragment mit dem SVG-Code in der Anwendung nutzen zu können, müssen sie manuelle Änderungen daran vornehmen. Der SVG-Code besteht aus dem Tag svg und einem Containerelement g. Letzteres dient dazu, weitere gleichartige Elemente zu verbinden. Solche Elemente sind im vorliegenden Fall die Pfadelemente path, die als verbundene Punkte die Form eines Bezirksumrisses beschreiben.

Als Nächstes fügen Entwickler im Element svg das Attribut viewBox hinzu und entfernen die ursprünglichen Attribute width und height. Das Attribut viewBox setzt das initiale Koordinatensystem fest und ist wichtig, weil es eine skalierte Darstellung der Karte ermöglicht. Die einzelnen path-Elemente definieren jeweils einen Bezirk. Um diese Bezirke richtig zu stylen, kommen die Styled Elements aus der Emotion-Bibliothek zum Einsatz. Sie lassen sich wie React-Komponenten verwenden.

Bezirke einfärben

Ein JavaScript-Tagged-Template-Literal beschreibt den Stil als

CSS und ermöglicht Entwicklern CSS in JavaScript zu schreiben, ohne andere Methoden wie das Setzen eines Inline-Styles oder CSS-Klassen verwenden zu müssen. In der Komponente Borough (englisch für Bezirk) ist das CSS-Attribut fill die Füllfarbe des Bezirks, stroke bezeichnet die Farbe der Umrisslinien. Der Pseudoselektor :hover tritt in Aktion, wenn der Nutzer mit dem Mauscursor über dem jeweiligen Bezirk liegt. Dann färbt sich der Bezirk rot.

Alle path-Elemente können Entwickler nun durch die React-Styled-Komponente Borough ersetzen. Mit dem Attribut name weisen sie der Komponente den richtigen Namen der Bezirke zu. Bewegt sich der Mauscursor über den Bezirk, stellt das ein Ereignis dar, auf das reagiert werden soll. Bei einem Mouse-over ruft der Browser die angegebene Funktion auf, hier handleMouseOver. Darin iteriert man über alle Attribute der jeweiligen Bezirkskomponenten, bis das name-Attribut gefunden ist. Eine aufgerufene Funktion callback gibt das Elternelement als Prop mit. Der Parameter der callback-Funktion ist der Inhalt des Attributs name, also der Bezirksname.

Logik aus Komponenten recyceln

Im Elternelement lässt sich die callback-Funktion bearbeiten, indem Entwickler mit setBorough den Status der Variablen borough setzen (Listing 3). React aktualisiert automatisch die Variable borough im DOM (Document Object Model) und der jeweilige Bezirksname erscheint.

Listing 3: Status der Variablen borough festlegen

```
import React, { useState } from "react";
import BerlinMap from "./components/BerlinMap/BerlinMap";
import styled from "@emotion/styled";

function App() {
    const [borough, setBorough] = useState("");

    const handleBorough = (name) => {
```

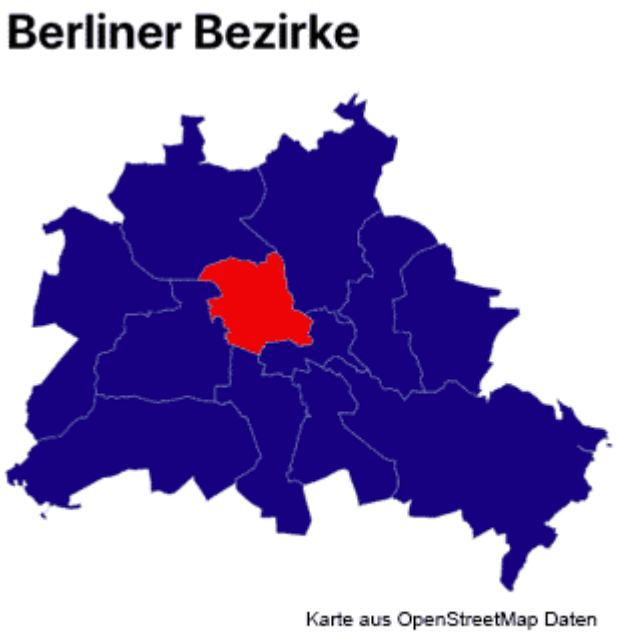
```
      setBorough(name);
    };

    return (
      <>
        <Container>
          <h1>Berliner Bezirke</h1>
          <BerlinMap callback={handleBorough} />
          <h1>{borough}</h1>
        </Container>
      </>
    );
}

const Container = styled.div`
    margin-left: auto;
    margin-right: auto;
    width: 800px;
`;

export default App;
```

Listing 3 verwendet den React-Hook useState(""). Mit React-Hooks lassen sich Statusfunktionen in Funktionskomponenten nutzen. Entwickler können eine statusbezogene Logik aus Komponenten extrahieren und wiederverwenden. Der Parameter von useState("") ist der Initialwert der Variablen. Er liefert als Ergebnis ein Array zurück, dessen erstes Element die Variable zum Lesen des Wertes ist (hier borough). Das zweite Element des Arrays ist eine Funktion zum Setzen des Wertes, hier setBorough. Beide werden durch Array Destructuring extrahiert. Ruft man nun die Anwendung mit yarn start auf, sieht man im Browser die Anwendung wie in der Abbildung.



Der Name des Bezirks unter der Karte aktualisiert sich dynamisch, je nachdem wo sich der Mauscursor befindet. *OpenStreetMap*
Die Kartenanwendung lässt sich mit Statistiken und Metriken zu den Bezirken erweitern. Alternativ sind Bezirke je nach Höhe einer Metrik anders einfärbbar. Die fertige Anwendung findet sich unter ix.de/zxex. (nb@ix.de)

1. Quellen
2. Informationen zu gpx2svg, Listings und die fertige Anwendung: ix.de/zxex

Thomas Derflinger

ist freiberuflicher Softwareentwickler mit Schwerpunkt auf der Frontend-Entwicklung von modernen Webanwendungen.

[/expand]