Seminar:

Odabrane tehnologije za prikaz točaka na geografskoj karti i kazaljki pokazivača u React-u

Uvod

U ovom seminaru će biti demonstirana dva načina prikazivanja točaka na geografskoj karti te jedan način prikazivanja vrijednosti na mjeraču u React-u.

GeoJSON

GeoJSON je često korišten format za zapisivanje geografskih podataka. U 2016. je specificiran unutar RFC 7946. U ovom seminaru će se koristiti za set točki koje će se prikazati na mapi.

Mapbox

Mapbox je izeđu ostlog platforma za prikazivanje karata na web stranicama. Nudi različite značajke kao što su prilagodljivi stilovi karata, navigacija i pretvaranje adresa u geografske koordinate. Popularan je zbog svoje fleksibilnosti, jednostavnosti korištenja i opsežne dokumentacije.

React map gl

react-map-gl je React „wrapper” za Mapbox-ove karte. Omogućava korištenje svih pogodnosti Mapbox-a iz React-a, za razliku od „čistog” javascripta.

React d3 speedometer

react-d3-speedometer je React biblioteka za prikazivanje mjerača koji izgleda poput brzinomjera koja „ispod haube” koristi D3.js. D3.js (Data-Driven Documents) je JavaScript biblioteka za manipuliranje DOM-a baziranom na podacima.

Početni programski kod

Sam React projekt je stvoren pomoću create-react-app alata. Taj alat u jednoj naredbi postavi osnovni React projekt koji „funkcionira”, tj. prikazuje jednostavnu stranicu s logom React-a. Nažalost, create-react-app je zastario u 2023. te se više ne održava – no unatoč tome još uvijek radi dovoljno dobro za potrebe ovog seminara.

Sve datoteke koje nije stvorio create-react-app se nalaze u src direktoriju. Zajedno sa slikama za marker i avion.

Datoteka index.js je ostala onakva kakvu ju je create-react-app stvorio. Dakle samo poziva App komponentu unutar React.StrictMode:

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(

<React.StrictMode>

<App />

</React.StrictMode>

);

U datoteci index.html je potrebna samo jedna promjena, dodavanje Mapbox-ovog css-a u <head>. Ta promjena je nužna jer bez nje se markeri na karti ne prikazuju ispravno.

<link href='https://api.mapbox.com/mapbox-gl-js/v3.1.2/mapbox-gl.css' rel='stylesheet' />

Datoteka index.css je skoro u potpunosti promijenjena i sada sadrži:

body {

margin: 0;

}

.mapContainer {

position: absolute;

height: 100%;

width: 100%;

z-index: 1;

}

.gaugeContainer {

position: absolute;

height: 50%;

width: 25%;

z-index: 2;

}

Uklanjanje margina na <body> elementu je jedino što je ostalo od create-react-app, dok su ostale dvije klase ovdje da „poslože” kartu i kazaljku na ekranu. Karta će biti rastegnuta preko cijelog ekrana i kazaljka će biti nacrtana na ekranu preko karte u gornjem-desnom kutu.

Konačno, datoteka App.js sada samo import-a kartu i kazaljku te ih vraća unutar fragment-a:

import MyMap from "./MyMap.js";

import Gauge from "./Gauge.js";

export default function App() {

return (

<>

<MyMap />

<Gauge />

</>

);

}

Prikaz karte

Sav programski kod potreban za prikaz karte je sadržan u datoteci MyMap.js. Ta datoteka izgleda ovako:

import Map, { Marker, Source, Layer } from "react-map-gl";

import marker from "./marker.png";

import plane from "./plane.png";

import { useRef } from "react";

const geojson = {

type: "FeatureCollection",

features: [

{

type: "Feature",

geometry: {

type: "Point",

coordinates: [14.57203549703318, 45.220035573837016],

},

},

],

};

export default function MyMap() {

const mapRef = useRef(null);

return (

<div className="mapContainer">

<Map

ref={mapRef}

mapboxAccessToken="pk.eyJ1IjoidHJpbS..."

initialViewState={{

longitude: 14.42579,

latitude: 45.337461,

zoom: 10,

}}

mapStyle="mapbox://styles/mapbox/streets-v9"

onLoad={() => {

const map = mapRef.current.getMap();

map.loadImage(plane, (error, image) => {

if (error) return console.log(error);

if (!map.hasImage("plane")) map.addImage("plane", image);

});

}}

>

<Marker longitude={14.42579} latitude={45.337461} anchor="bottom">

<img src={marker} alt="marker" />

</Marker>

<Source id="my-data" type="geojson" data={geojson}>

<Layer

id="point"

type="symbol"

layout={{

"icon-image": "plane",

"icon-size": 0.5,

}}

/>

</Source>

</Map>

</div>

);

}

Na sam prikaz mape odnose se samo <div> s klasom mapContainer i sama Map komponenta s parametrima: mapboxAccessToken, initialViewState i mapStyle.

<div> je ovdje samo radi css-a. Da bi se mapa prikazala mora biti smještena u container-u koji ima definirane dimenzije (širinu i visinu).

Map komponenta predstavlja samu kartu. Ima mnogo parametera koji joj se mogu proslijediti međutim da bi se karta prikazala potrebno ih je samo dva:

- mapboxAccessToken koji se može dobiti kada se napravi besplatni korisnički račun na Mapbox-ovoj web stranici ([www.mapbox.com](http://www.mapbox.com/)). U gornjem primjeru je otprilike dvije trećine token-a zamijenjeno sa tri točke.

- mapStyle – URI datoteke koja sadrži sve što je Mapbox-u potrebno da prikaže kartu. Može biti jedan od Mapbox-ovih unaprijed definiranih stilova ili stil koji je korisnički napravljen preko njihovog web-editora. U ovom slučaju se koristi unaprijed definirani stil streets-v9.

Parametar initialViewState nije nužno potreban ali je ovdje postavljen kako bi mapa bila centrirana na RiTeh - gdje je i postavljen marker.

Prikazivanje markera na karti

Prikazivanje običnih markera je jako jednostavno tako da je skoro sav kod potreban za postavljanje markera na kartu sadržan unutar <Marker> komponente. <Marker> komponenta je postavljena unutar <Map> komponente (dakle dodan na kartu) i kao parametre su mu dane geografske koordinate na kojima se prikazuje (u ovom slučaju geografke koordinate RiTeh-a) te anchor parametar koji određuje koji dio slike (tj. html-a sadržanog unutar <Marker> komponente) će dodirivati te koordinate. Default-na vrijednost je "center", ali u ovom slučaju je postavljena na "bottom" jer korištena slika marker-a ima „iglu” na dnu.

Unutar <Marker> elementa je postavljen samo <img> element koji prikazuje sliku marker-a.

Prikazivanje ikonica na karti

Iako se ikonice mogu prikazivati na karti pomoću <Marker> komponenti taj pristup se ne skalira dobro pa u slučaju da je potrebno prikazati stotine marker-a/ikonica bolje je tom problemu pristupiti drugačije.

Grupe ikonica je najbolje prikazivati unutar <Layer> komponenti tipa symbol. <Layer> komponente prikazuju na mapi ono što im proslijedi <Source> komponenta. Dakle <Source> definira što će se prikazati na karti, a <Layer> to prikazuje.

U ovom primjeru je korišten <Source> tipa geojson. On čita GeoJSON objekt koji mu je proslijeđen kroz data parametar te šalje feature-e <Layer> komponenti unutar sebe za prikaz. <Layer> komponenta tipa symbol je namijenjena za prikaz ikonica te joj u parametar layout proslijeđujemo objekt koji je ekvivalentan Mapbox-ovom layout objektu i u njemu definiramo ikonicu (sliku) koja će se koristiti te veličinu u kojoj će se prikazati.

Međutim, da bi se slika mogla prikazati na mapi ona mora biti u njenom style dokumentu. Tako da ju je potrebno dodati. Nažalost, react-map-gl ne nudi jednostavan način za to napraviti pa je zbog toga <Map> komponenti dodana funkcija koja će se izvršiti kada se karta učita (to je najraniji trenutak kada smo sigurni da je useRef postavljen i da možemo od <Map> komponente tražiti referencu na originalni Mapbox-ov map objekt). Ta funkcija će samo uzeti referencu na Mapbox-ov originalni map objekt i dodati sliku na tu kartu tako da će prvo pozvati loadImage funkciju a nakon toga uz provjeru (za svaki slučaj) da slika nije već tamo pozvati addImage.

Dakle, za prikazivanje grupa ikonica na mapi na ovaj način potrebno je definirati GeoJSON objekt (u ovom slučaju ima samo jedan feature – zračnu luku Rijeka), dodati željenu ikonicu na mapu kada se mapa učita te kreirati <Source> i <Layer> komponente unutar <Map>-e.

Prikazivanje kazaljke i „brzinomjera”

Sav programski kod potreban za prikaz kazaljke je sadržan u datoteci Gauge.js. Ta datoteka izgleda ovako:

import ReactSpeedometer from "react-d3-speedometer";

import { useInterval } from "./useInterval.js";

import { useState } from "react";

export default function Gauge() {

const [value, setValue] = useState(0);

useInterval(() => {

setValue(Math.round(Math.random() \* 100));

}, 3000);

return (

<div className="gaugeContainer">

<ReactSpeedometer

value={value}

minValue={0}

maxValue={100}

segmentColors={["#5BE12C", "#f0f059", "#EA4228"]}

textColor="#000"

customSegmentStops={[0, 85, 95, 100]}

valueTextFontSize="20px"

fluidWidth={true}

paddingVertical={10}

needleHeightRatio={0.75}

ringWidth={20}

/>

</div>

);

}

Ovaj <div> je postavljen ovdje radi urednosti i preglednosti koda, tj. da bi css bio „riješen” jednako kao i kod karte. Ali za razliku od karte, brzinomjer ima svoju default-nu visinu i širinu te bi se prikazao i bez specificiranja css-a.

Osim toga, komponenta je vrlo jednostavna. Čuva vrijednost koju će brzinomjer prikazivati u useState i mijenja ju u slučajni broj svakih 3 sekunde (3000ms) te poziva ReactSpeedometer komponentu.

ReactSpeedometer komponenta je jako prilagodljiva pa prima puno parametara, ali ovdje su korišteni samo neki osnovni:

value – vrijednost na koju će igla pokazivati

minValue – vrijednost na lijevoj strani skale

maxValue – vrijednost na desnoj strani skale

segmentColors – niz koji definira boje, ali i broj obojanih segmenata na skali

textColor – boja brojeva i ostalog texta koji se prikazuje

customSegmentStops – definira vrijednosti između kojih se segmenti protežu

valueTextFontSize – veličina slova kojom se prikazuje vrijednost ispod brzinomjera

fluidWidth – kada se postavi na true brzinomjer će poprimiti širinu container-a u kojem se nalazi i njoj primjerenu visinu

paddingVertical – padding iznad i ispod brzinomjera, ponekad je potreban zbog labela koje se tamo nalaze

needleHeightRatio – broj koji označava koliko će igla biti dugačka u odnosu na radijus skale

ringWidth – širina skale

Pomočni hook useInerval poziva funkciju koja mu je proslijeđena u pravilnim razmacime. Ovdje je njen kod:

import { useRef, useEffect } from "react";

export function useInterval(callback, delay) {

const savedCallback = useRef();

useEffect(() => {

savedCallback.current = callback;

}, [callback]);

useEffect(() => {

function tick() {

savedCallback.current();

}

if (delay !== null) {

let id = setInterval(tick, delay);

return () => clearInterval(id);

}

}, [delay]);

}

Zaključak

Iako nisu savršene, react-map-gl i react-d3-speedometer su vrlo korisne biblioteke koje nisu teše za korištenje – a opet: „služe svrsi”. Bilo koji programer koji iz nekog razloga treba ili želi implementirati jednu od njihovih funkcionalnosti će biti zadovoljan s njihovom jednostavnosti korištenja i moći „ispod haube”.