Exercises for Session 5

Goals

We're going to look at a new Browser API called Geolocation to read the current position of the user. After that we're going to play with the Google Maps API and we're setting up a "real" frontend project.

Geolocation

- We want to create a new mini app that displays a button saying "Show my location".
- When clicking the button we execute a function that:
    - calls `navigator.geolocation.getCurrentPostion(onSuccess, onError)`
    - `onSuccess` we want to read the positional coordinates: `const { latitude, longitude } = position.coords` and `console.log` them
        - if you feel like it create a `p` element to display them in the html
    - `onError` we want to `console.log` the error

Google Maps

- Follow the official Google tutorial to display a map (incl. getting your own API key): https://developers.google.com/maps/documentation/javascript/tutorial
- Instead of using the default location research the positional coordinates of Cape Town, South Africa and use it as the center of your map (`zoom: 12` gives a nice result)

Setting up a "real" project

This is mainly needed to load local data using `fetch` as it can be used from `file://` urls (like opening our HTML in the browser) but needs to be served via `http://` .

- Make sure you have node installed for example by running `npm -v` in a terminal. If nothing shows up or an error is shown head to https://nodejs.org/en/download/ and follow the instructions
- Create a new folder for this project (e.g. `maps-experiments`) and open a terminal inside that folder
- Run `npm init` to scaffold the project (you can accept all the defaults)
- Install a web server by running `npm install --save-dev http-server`
- Add a `start` script to the `package.json` file:

```
"scripts": {
    "start": "./node_modules/.bin/http-server -o"
}
```

> Info: the `-o` flag opens the browser for you

- Create a new `index.html` file
- Run `npm run start` (or short `npm start`) to boot up our web server and see the index file being loaded

Next copy over the code from the previous session to display a map centered in Cape Town in your page.

Feature 1: Display the current position

- Use the `navigator.geolocation.getCurrentPosition` API to center the map at the position of the user (`map.setCenter({ lat: YYY, lng: ZZZ })`)
- Show a marker at the position of the user

Feature 2: Load and show cities

- In your project create a new file `locations.json` with the following data:

```
const cities = [
    {
        name: 'Hamburg',
        position: {
            lat: 53.5511,
            lng: 9.9937
        }
    },
    {
        name: 'Berlin',
        position: {
            lat: 52.52,
            lng: 13.405
        }
    },
    {
        name: 'Munich',
        position: {
            lat: 48.1351,
            lng: 11.582
        }
    },
    {
        name: 'Karlsruhe',
        position: {
            lat: 49.0069,
            lng: 8.4037
        }
    },
    {
        name: 'Frankfurt',
        position: {
            lat: 50.1109,
            lng: 8.6821
        }
    },
    {
        name: 'Cologne',
        position: {
            lat: 50.9375,
            lng: 6.9603
        }
    },
    {
        name: 'Amsterdam',
        position: {
            lat: 52.3667,
            lng: 4.8945
        }
    },
    {
        name: 'Bremen',
        position: {
            lat: 53.0793,
            lng: 8.8017
        }
    }
]
```

- Loop over the locations and create a new marker for each one
  - Also ensure that the bounds of the map are extended to include each location
- After looping over the map ensure to fit the map to bounds of all locations

Feature 3: Create marker on click

- Add a `click` listener to the map. Whenever the map is click a new marker should be displayed

> Hint: `event.latLng` contains the position of the click and can be used for the creation of the marker.

Feature 3.1: Connect markers with line

- Whenever a marker is created add it an array called `allMarkers` (`allMarkers.push(marker)`)
- Draw a `Polyline` which `path` is the position of all markers (ie. connect all markers with a line). To extract the positions out of the `allMarkers` array you can use `map()`

Feature 4: Show address of click

We want to show the address of the point a user clicked on the map.

- First, enable Geocoding API in cloud console
- When a user clicks on the map create a new Geocoder and look up the address of the location (`geocoder.geocode({ location. event.latLng })`)

- Log the the result of the lookup to the console
- If you feel brave enough create a new paragraph element (`p`) and show the address on the screen