

NEA - Untitled Mapping Project

Code available at: <https://github.com/PeterWarrington/NEA-mapping>

I am working to create a project that is capable of parsing, representing and displaying real-world map data, which can then be used to calculate the most efficient paths between points. This would work in conjunction with a backend API that returns map data to a client, where the backend is responsible for calculating paths and caching these so that they can be returned to clients more quickly.

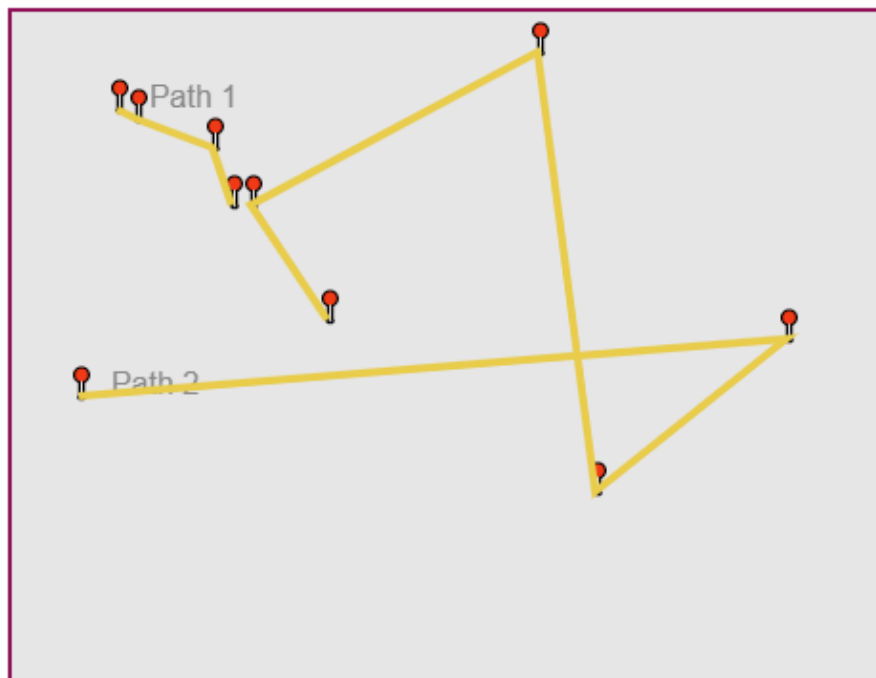
I believe that a web app built using standard web technology (HTML5, CSS, JS) would be the most effective way to achieve this, as it enables use of the service using one client codebase regardless of device specifics and is well supported by mobiles in particular. This is what I have started to create as part of my mini-project, leaving behind the backend for now so I can build the data structures and build a more simple base from which the rest of the project can be effectively built on. The codebase I have created as it stands is capable of displaying map data to the user given a set of connecting points, where this map is scalable and translatable.

The backend I have built at the moment simply serves the static site that acts as the client, but is and will be built using Node.JS, a backend JavaScript runtime environment that is suited to building APIs and is well used in professional environments. It means that in addition I can reuse the same data structures for both client and server, making transfer and development between the two faster and more efficient such that I do not have to make two separate implementations of two data structures for example.

I will be using OpenStreetMap data as part of my project, which provides comprehensive, worldwide community built map-data under a freely-usable license given attribution.

Some screenshots are included below:

Map:



Controls:


Zoom: 

Figure 1: Screenshot of web app

```

1  class CanvasState {
2      /** The {CanvasRenderingContext2D} that is used on the canvas */
3      ctx
4      /** The canvas element that displays the map */
5      canvas
6      /** Indicates whether the mouse button is down */
7      canvasMouseDown = false
8      /** Indicates the last recorded mouse position relative to the page in X */
9      lastPageX = -1
10     /** Indicates the last recorded mouse position relative to the page in Y */
11     lastPageY = -1
12     /** The array of {Path}s to be drawn to screen */
13     paths = []
14     /** A {int} multiplier to represent the zoom level */
15     zoomLevel = 1
16     /** {int} representing how the map has been translated in x */
17     xTranslation = 0
18     /** {int} representing how the map has been translated in y */
19     yTranslation = 0
20
21     constructor () {
22         this.canvas = document.getElementById("mapCanvas");
23         this.ctx = this.canvas.getContext('2d');
24
25         // Zoom functionality
26         document.getElementById("zoom-in").onclick = (event) => {
27             this.zoom(1);
28         };
29
30         document.getElementById("zoom-out").onclick = (event) => {
31             this.zoom(-1);
32         };
33
34         // Translation functionality
35         this.canvas.onmousedown = (event) => {

```

Figure 2: Screenshot of some of the code, showing the CanvasState class