Введение в Mapbox

Технология Mapbox возникла благодаря группе веб-разработчиков, которые хотели ответить на этот вопрос: *как мы можем рисовать карту динамически в веб-браузере вместо загрузки фрагментов статической карты, отображаемых на сервере?* Они хотели встраивать динамические, интерактивные, настраиваемые карты на веб-страницы и мобильные устройства, и они объединили векторные плитки и технологию 3D-рендеринга для создания решения.

С созданием *Mapbox GL* и *спецификации стиля Mapbox с* открытым исходным *кодом* открылся мир динамических картографических приложений, и Mapbox стала компанией, занимающейся созданием инструментов, которые передают эти возможности в руки разработчиков.

Технология Mapbox

Сегодня Mapbox поддерживает карты и службы определения местоположения для широкого спектра веб-приложений, мобильных, автомобильных и игровых приложений.

Картографические данные являются основой для многих служб определения местоположения. Конвейеры обработки данных принимают новые данные с мобильных датчиков, отзывов водителей, камер с компьютерным зрением и аэрофотоснимков в наши конвейеры обработки данных и объединяют эти данные с открытыми и собственными источниками, чтобы картографические данные были актуальными, как и весь мир. Изменения.

Данные о трафике используют одни и те же конвейеры обработки для обновления профилей трафика для миллиардов участков дорог по всему миру каждые несколько минут, обеспечивая мощные возможности навигации , такие как маршрутизация с учетом трафика и интуитивно понятные пошаговые инструкции.

Технология машинного зрения использует эффективные нейронные сети для обработки изображений на уровне дорог непосредственно на мобильных или встроенных устройствах, превращая смартфоны и видеорегистраторы в датчики реального времени, интерпретирующие дорожные условия в режиме реального времени.

Инструменты для разработчиков включают мощные API-интерфейсы для сервисов Maps, Search, Navigation, Vision и Accounts. Мы предоставляем SDK, чтобы

сделать эти сервисы доступными для разработчиков веб-сайтов, мобильных устройств, игр и встроенных устройств. Инструменты разработчика включают Mapbox GL JS , наш JavaScript SDK для веб-разработчиков; Mapbox Studio , бесплатный редактор стилей карты с визуальным предварительным просмотром в реальном времени; и Mapbox Atlas , локальное приложение для клиентов с ограниченными сетевыми требованиями. Полный список инструментов см. В нашей документации .

Векторные листы для хранения и обслуживания большей части картографических данных . Формат векторных листов является компактным и предназначен для быстрого кэширования, масштабирования и обслуживания данных карты. Векторные листы содержат геометрию и метаданные, которые можно отобразить на карте. Узнайте больше о нашей *спецификации векторной плитки* ниже.

Графические библиотеки , Mapbox GL и Mapbox GL Native , сообщают Интернету и мобильным устройствам, как рисовать карты в виде визуальной графики. Mapbox GL извлекает геопространственные данные из *векторных* листов и инструкции по *стилю из документа стиля* и помогает клиенту рисовать 2D- и 3D-карты Mapbox в виде динамической визуальной графики с помощью OpenGL.

Mapbox GL опирается на два **документа со спецификациями,** которые определяют стандарты, позволяющие Mapbox GL правильно инструктировать клиентов:

- Спецификация **векторной плитки** Mapbox с открытым исходным кодом описывает, как геометрия и атрибуты в геопространственных данных должны храниться и кодироваться в векторных плитках.
- Спецификация **стиля** Mapbox с открытым исходным кодом описывает, как вы должны написать стиль карты, чтобы сообщить Mapbox GL, какие данные нужно рисовать, порядок их рисования и какие цвета, уровни непрозрачности и другие свойства применять при рисовании данных. MapBox Студия редактор стиля является визуальным инструментом для создания документа стиля , который прилипает к данной спецификации.

Как использовать Mapbox

Вы можете создать бесплатную учетную запись и получить свой первый токен доступа, чтобы начать строительство с помощью Mapbox прямо сейчас. В этом разделе описаны ресурсы, которые мы предлагаем, чтобы помочь вам в вашем пути разработчика.

Чтобы использовать любые инструменты, API или SDK Mapbox, вам понадобится токен доступа Mapbox . Mapbox использует токены доступа для связывания запросов API с вашей учетной записью. Вы можете найти свои токены доступа,

создать новые или удалить существующие на странице токенов доступа или программно с помощью API токенов Mapbox .

Каждый созданный вами токен доступа будет иметь набор разрешений, позволяющих токену выполнять определенные типы запросов к API Mapbox - они называются **областями** . В документации API перечислены области, необходимые для каждого API Mapbox. При создании токена доступа у вас будет возможность добавить к токену общедоступные или частные области .

Полный список доступных объемов и рекомендаций см. В документации по учетной записи .

Веб-приложения

Mapbox предоставляет множество инструментов для создания карт на вашем веб-сайте или в веб-приложении. Mapbox GL JS и Mapbox.js - это библиотеки JavaScript, которые вы можете использовать для отображения карт Mapbox, добавления интерактивности и настройки взаимодействия с картой в вашем приложении. Мы также предоставляем множество подключаемых модулей для расширения функциональности вашей веб-карты с помощью инструментов рисования и интерфейсов для API веб-сервисов Mapbox, таких как Mapbox Geocoding API или Mapbox Directions API.

Mapbox GL JS

Прежде чем приступить к написанию кода вашей карты Mapbox GL JS, вам необходимо включить соответствующие файлы JavaScript и CSS на свою веб-страницу. Mapbox предоставляет размещенные версии каждого из них, которые вы можете включить в `<head>`свой HTML-файл:

```
<script src='https://api.mapbox.com/mapbox-gl-js/v2.2.0/mapbox-gl.js'></script><link
href='https://api.mapbox.com/mapbox-gl-js/v2.2.0/mapbox-gl.css' rel='stylesheet' />
```

В основе каждого проекта Mapbox GL JS лежит `mapboxgl.Map`класс. Пример кода в этом разделе демонстрирует минимум, необходимый для добавления карты на вашу страницу.

```
var map = new mapboxgl.Map ( {

container: 'map',

 style: 'mapbox: // styles / mapbox / street-v 11 ',

  center: [-74.50, 40],

  zoom: 9
```

```
} );
```

- **Контейнер** : это HTML-элемент, в котором вы хотите разместить карту. В приведенном выше примере это элемент с расширением `id="map"`.
- **Стиль** : карта загружает стиль через URL-адрес `mapbox://styles/mapbox/streets-v11`. Это URL-адрес удаленного файла, который карта загрузит, чтобы определить включаемые в нее наборы элементов мозаики и их стиль для конечного пользователя. Mapbox GL JS разрешает URL-адреса вместо буквальных данных в нескольких местах, включая источники данных. Рассмотрите возможность использования стиля с оптимизированными для стиля векторными плитками для более производительных карт.
- **В центре** : если Mapbox GL JS обрабатывает координаты как массивы (здесь `[-74.50, 40]`), предполагается, что координаты указаны в `longitude, latitude`порядке (по сравнению `latitude, longitude`с Leaflet и Mapbox.js). Этот порядок соответствует порядку координат в GeoJSON и любом другом геопространственном формате, а также математическому порядку X, Y.
- **Масштаб** : уровень масштабирования, при котором карта должна быть инициализирована. В Mapbox GL JS может быть десятичным значением.

Добавление слоев на карту

Вы можете добавлять слои на карту с помощью `addLayer()`метода. `addLayer`имеет только один обязательный параметр: объект слоя стиля Mapbox. Он также принимает необязательный `before`параметр, который представляет собой идентификатор существующего слоя, перед которым вставляется новый слой. Если вы опустите этот аргумент, средство визуализации нарисует слой поверх карты. В следующих разделах описываются элементы объекта слоя стиля Mapbox.

Асинхронный

```
map.on ('load', function () {
 map.addLayer ( {
  id: 'terrain-data',
  type: 'line',
  source: {     type: 'vector',
    url: 'mapbox: //mapbox.mapbox -terrain-v2 '
   } ,
   ' исходный слой ':' контур '
  } );
} );
```

Поскольку эти ресурсы являются **удаленными** , они **асинхронны** . Таким образом, код, который подключается к Mapbox GL JS, часто использует привязку событий для изменения карты в нужное время. Например:

В приведенном выше коде код используется `map.on('load', function() {`для вызова `map.addLayer`только после загрузки ресурсов карты, включая стиль. Если бы `map.addLayer`метод был запущен немедленно, это вызвало бы ошибку, потому что стиль, к которому вы хотите добавить слой, еще не существовал бы.

Вам нужно будет определить источник, когда вы добавите новый слой. Источник принимает a `type`и a `url`(источник GeoJSON не будет иметь `url`). Есть пять типов источников, каждый со своими свойствами:

- векторные плитки
- растровые плитки
- GeoJSON
- изображение
- видео

Наборы листов могут включать в себя несколько подмножеств данных, называемых исходными слоями (набор листов Mapbox Streets содержит исходные слои для дорог, парков и т. Д.). Чтобы убедиться, что ваши слои ссылаются на правильные исходные слои, ваш объект слоя также должен включать `source-layer`(часто имя исходного файла).

addSource ()

Вы также можете добавлять источники с помощью метода Mapbox GL JS `addSource()`. При использовании этого альтернативного метода нет разницы в производительности карты, но иногда предпочтительно, чтобы код был более читабельным. Подробнее об этом методе читайте в документации Mapbox GL JS .

```
map.on('load', function() {
 map.addLayer({
  id: 'rpd_parks',
  type: 'fill',
  source: {
   type: 'vector',
   url: 'mapbox://mapbox.3o7ubwm8'
  },
  'source-layer': 'RPD_Parks'
 });
```

});

Определение свойств макета и краски

Слои имеют два специальных свойства, которые позволяют стилизовать данные: `paint`и `layout`. Они используются для определения того, как данные будут отображаться на карте. `layout`Свойства относятся к размещению и видимости, помимо других высокоуровневых предпочтений, и применяются на ранних этапах процесса визуализации. `paint`properties - это более детализированные атрибуты стиля, такие как непрозрачность, цвет и перевод. Они менее требовательны к обработке и визуализируются позже.

Следующий код добавляет на карту слой, чтобы стилизовать данные парков с зеленой заливкой.

addLayer ()

Если вы добавили свой источник с помощью альтернативного `addSource()`метода, вам нужно будет включить идентификатор источника как `source`in `addLayer()`. Подробнее об этом читайте в документации Mapbox GL JS API.

```
map.on('load', function() {
 map.addLayer({
  id: 'rpd_parks',
  type: 'fill',
  source: {
   type: 'vector',
   url: 'mapbox://mapbox.3o7ubwm8'
  },
  'source-layer': 'RPD_Parks',
  layout: {
   visibility: 'visible'
  },
  paint: {
   'fill-color': 'rgba(61,153,80,0.55)'
  }
```

```
  });
});
```

Конечный продукт: увеличенная карта Сан-Франциско со слоем парков с зеленой заливкой. Слой основан на векторном источнике данных о городских парковых угодьях.

См. Раздел руководств для получения дополнительных ресурсов Mapbox GL JS.

Добавить карту на страницу

Основная функция Mapbox.js - добавить карту на вашу HTML-страницу. Используя одну строку JavaScript, вы можете добавить карту на свою веб-страницу с базовой картой, которая панорамируется и масштабируется, устанавливается на определенное место и уровень масштабирования .

Расширьте свое веб-приложение с помощью плагинов

Mapbox GL JS и Mapbox.js поддерживают обширную экосистему плагинов, которые вы можете использовать для расширения функциональности вашей веб-карты. Существуют плагины для добавления интерактивных инструментов рисования, добавления карт-врезок, интеграции с Mapbox Geocoding API и Mapbox Directions API и многое другое! Изучите страницу плагинов Mapbox GL JS и страницу плагинов Mapbox.js для получения дополнительной информации.

Используйте Mapbox GL JS в приложении React

React - популярная библиотека JavaScript, используемая для создания пользовательских интерфейсов. Поскольку React манипулирует DOM, может сбивать с толку подключение React к другим библиотекам, которые также управляют DOM и управляют состоянием, например Mapbox GL JS.

Начиная

- **Маркер доступа Mapbox.** Ваши токены доступа к Mapbox находятся на странице вашей учетной записи .
- **Mapbox GL JS.** Mapbox GL JS - это библиотека JavaScript, используемая для создания веб-карт.
- **Текстовый редактор.** Используйте любой текстовый редактор для написания HTML, CSS и JavaScript.
- **Node.js и npm.** Чтобы запустить команды, необходимые для локального запуска вашего приложения React, установите Node.js и npm .
- **Знакомство с React.** Вам не нужно иметь большой опыт использования React, чтобы пройти это руководство, но вы должны быть знакомы с основными концепциями и рабочими процессами.

Настройте структуру приложения React

Для начала создайте новую папку с именем `use-mapbox-gl-js-with-react`.

В `use-mapbox-gl-js-with-react` папке создайте новый файл:

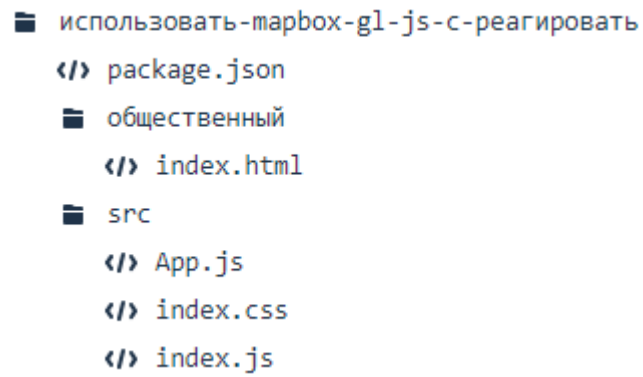
- `package.json`: Этот файл используется для указания всех пакетов Node, которые требуются вашему приложению, включая React и Mapbox GL JS.

В `use-mapbox-gl-js-with-react` папке создайте папку с именем `public` . В `public` папке создайте один файл:

- `index.html`: Этот HTML-файл будет отображать визуализированную карту Mapbox, с которой ваши пользователи смогут взаимодействовать.

В `use-mapbox-gl-js-with-reactп` апке создайте еще одну папку с именем `src`. В `src` папке создайте три файла:

- `App.js`: Этот файл JavaScript настроит приложение React.
- `index.css`: Этот файл CSS будет содержать стили для правильного форматирования карты и боковой панели.
- `index.js`: Этот файл JavaScript отобразит карту Mapbox в браузере.



Скопируйте следующий код в `package.json`:

package.json

```
{
  "name": "use-mapbox-gl-js-with-react",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "mapbox-gl": "^2.2.0",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "^4.0.3"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build"
  },
  "eslintConfig": {
    "extends": [
      "react-app"
    ]
  },
  "browserslist": [
    "defaults",
    "not ie 11"
  ],
  "author": "Mapbox",
  "license": "MIT"
}
```

В дополнение к специфичным для React пакетам Node `react` и `react-dom` этому файлу также требуется приложение, `react-scripts`также известное как Create React App, для создания приложения и `mapbox-gl` для доступа к Mapbox GL JS.

Сохраните изменения.

В командной строке перейдите в `use-mapbox-gl-js-with-react`созданную вами папку. В этой папке запустите команду `npm install`, которая установит все пакеты Node, которые вы указали в `package.json`ф айле. На этом шаге также создается `package-lock.json` файл.

Создать HTML-страницу

Откройте `public/index.html`файл и вставьте в него следующий код:

общедоступный / index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta
      name="description"
      content="Create a React web app that uses Mapbox GL JS to render a map"
    />
    <title>Use Mapbox GL JS in a React app</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Этот код создает структуру HTML-страницы, которую увидят ваши пользователи. Существует `<div>`элемент с идентификатором `root`в `<body>`страницах. Это `<div>`контейнер, в котором приложение React будет отображаться на странице.

Сохраните изменения.

Создайте приложение React

Откройте `src/index.js`файл. Добавьте следующее, чтобы импортировать две таблицы стилей и карту Mapbox GL JS, которую вы создадите:

src / index.js

```
import React from 'react';

import ReactDOM from 'react-dom';

import 'mapbox-gl/dist/mapbox-gl.css';

import './index.css';

import App from './App';


ReactDOM.render(

  <React.StrictMode>

    <App />

  </React.StrictMode>,

  document.getElementById('root')

);
```

Первая таблица стилей содержит стили Mapbox GL JS для отображения карты. Вторая таблица стилей - это `index.css`файл, который вы создали ранее, куда вы добавите свой CSS для конкретного приложения.

Добавить Mapbox GL JS

Откройте `src/App.js`файл. Добавьте следующий оператор импорта в начало файла:

src / App.js

Hooks

```
import React, { useRef, useEffect, useState } from 'react';
```

Classes

```
import React from 'react';
```

Затем импортируйте Mapbox GL и добавьте свой токен доступа. Чтобы использовать Mapbox GL с приложением Create React , вы должны добавить восклицательный знак, чтобы исключить mapbox-gl из транспиляции и отключить

правило eslint `import/no-webpack-loader-syntax`. Задайте для `mapboxgl` `accessToken`свойства значение вашего токена доступа Mapbox:

src / App.js

```
import mapboxgl from '!mapbox-gl'; // eslint-disable-line import/no-webpack-loader-syntax
mapboxgl.accessToken = 'YOUR_MAPBOX_ACCESS_TOKEN';
```

Теперь вы можете настроить приложение React! Чтобы создать структуру, в которую вы добавите код из следующих нескольких шагов, добавьте следующее внизу `App.js`:

src / App.js

Hooks

```
export default function App() {

}
```

Classes

```
export default class App extends React.PureComponent {

}
```

Сохраните изменения.

Установите состояние приложения по умолчанию

Затем вы создадите несколько значений по умолчанию для вашего приложения, которые будут использоваться для начальной широты, долготы и масштабирования карты. Добавьте внутрь следующее `App`:

src / App.js

Hooks

```
 const mapContainer = useRef(null);

 const map = useRef(null);

 const [lng, setLng] = useState(-70.9);

 const [lat, setLat] = useState(42.35);

 const [zoom, setZoom] = useState(9);
```

Classes

```
 constructor(props) {

   super(props);
```

```
  this.state = {
    lng: -70.9,
    lat: 42.35,
    zoom: 9
  };
  this.mapContainer = React.createRef();
}
```

Состояние хранит долготу, широту и масштаб карты. Все эти значения будут меняться по мере того, как ваш пользователь взаимодействует с картой.

Затем инициализируйте карту. Следующий код будет вызван сразу после того, как приложение будет вставлено в дерево DOM вашей HTML-страницы.

src / App.js

Hooks

```
useEffect(() => {
  if (map.current) return; // initialize map only once
  map.current = new mapboxgl.Map({
    container: mapContainer.current,
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [lng, lat],
    zoom: zoom
  });
});
```

Classes

```
componentDidMount() {
  const { lng, lat, zoom } = this.state;
  const map = new mapboxgl.Map({
    container: this.mapContainer.current,
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [lng, lat],
    zoom: zoom
  });
}
```

Состояние хранит долготу, широту и масштаб карты. Все эти значения будут меняться по мере того, как ваш пользователь взаимодействует с картой.

Затем инициализируйте карту. Следующий код будет вызван сразу после того, как приложение будет вставлено в дерево DOM вашей HTML-страницы.

Карта Mapbox инициализируется в ловушке React Effect или в `componentDidMount()`методе жизненного цикла, если вы используете классы. Инициализация вашей карты здесь гарантирует, что Mapbox GL JS не будет пытаться визуализировать карту до того, как React создаст элемент, содержащий карту. Вы также устанавливаете следующие параметры внутри инициализации карты:

- Эта `container`опция указывает Mapbox GL JS визуализировать карту внутри определенного элемента DOM. Здесь приложение ожидает получить `mapContainer` `useRef`или, `ref`если вы используете компоненты класса. Позже в этом руководстве вы назначите `ref`элемент HTML, который будет действовать как контейнер карты.
- `style`Параметр определяет стиль , что карта будет использовать (`mapbox://styles/mapbox/streets-v11`).
- `center`И `zoom`опции установить координаты центра и приблизить уровень карты , используя значения из `lng`, `lat`и `zoom`которые хранятся в состоянии.
- Если вы используете хуки, вы также создали файл `map` `useRef`для хранения инициализации карты. Это `ref`предотвратит перезагрузку карты при взаимодействии пользователя с картой.

Сохраните изменения.

Визуализировать карту

Теперь вам нужно отрендерить карту в вашем приложении. Точка входа для инициализации карты Mapbox в приложении React - через единственный элемент, указанный в `return`инструкции. Добавьте в приложение следующий код над закрывающей фигурной скобкой `App`:

src / App.js

```
render() {
  return (
    <div>
      <div ref={this.mapContainer} className="map-container" />
    </div>
  );
```

```
 }
```

В `mapContainer` `ref`указывает , что `App`должно быть обращено на страницу HTML в новом `<div>`элементе.

Для корректной визуализации карты необходимо соблюдать несколько правил стилизации. Добавьте в `index.css`файл следующий код :

```
.map-container {
  height: 400px;
}
```

Сохраните изменения. В командной строке запустите команду `npm start`. Это запустит локальный сервер и откроет новую страницу в вашем браузере, содержащую новую карту Mapbox.

Сохраните новые координаты

Затем вам нужно создать функцию, которая хранит новые значения широты, долготы и масштабирования, которые вы получаете, когда пользователь взаимодействует с картой. Вы напишете функцию Mapbox GL JS, `map.on('move')`которая устанавливает состояние для этих новых значений, когда пользователь перемещает карту. Если вы используете хуки, создайте дополнительный `useEffect`и добавьте следующий код. Если вы используете классы, добавьте внутрь следующий код `componentDidMount()`:

```
map.on('move', () => {
  this.setState({
    lng: map.getCenter().lng.toFixed(4),
    lat: map.getCenter().lat.toFixed(4),
    zoom: map.getZoom().toFixed(2)
  });
});
```

Эта функция используется , `useState()`если вы используете крючки, или `setState()`если вы используете классы, чтобы сбросить значения `lng`, `lat`и , `zoom`когда на карте двигается. Также используются следующие методы:

- `getCenter()`, метод Mapbox GL JS, чтобы получить новую долготу и широту точки в центре карты.

- `getZoom()`, метод Mapbox GL JS для определения уровня масштабирования карты.
- `toFixed()`, метод JavaScript, чтобы усечь полученное число с плавающей запятой до указанного количества цифр.

Показать координаты

Теперь, когда вы можете собирать и хранить эту информацию, вы можете использовать ее `return`для отображения на карте. Внутри открывающего тега созданного `<div>`вами для хранения карты добавьте новый `<div>`для отображения долготы, широты и масштаба карты. Теперь `return`инструкция будет выглядеть так:

src / App.js

```
render() {
  const { lng, lat, zoom } = this.state;
  return (
    <div>
      <div className="sidebar">
        Longitude: {lng} | Latitude: {lat} | Zoom: {zoom}
      </div>
      <div ref={this.mapContainer} className="map-container" />
    </div>
  );
}
```

Для правильного отображения боковой панели на странице требуется несколько правил стиля. Добавьте в `index.css`файл следующий CSS :

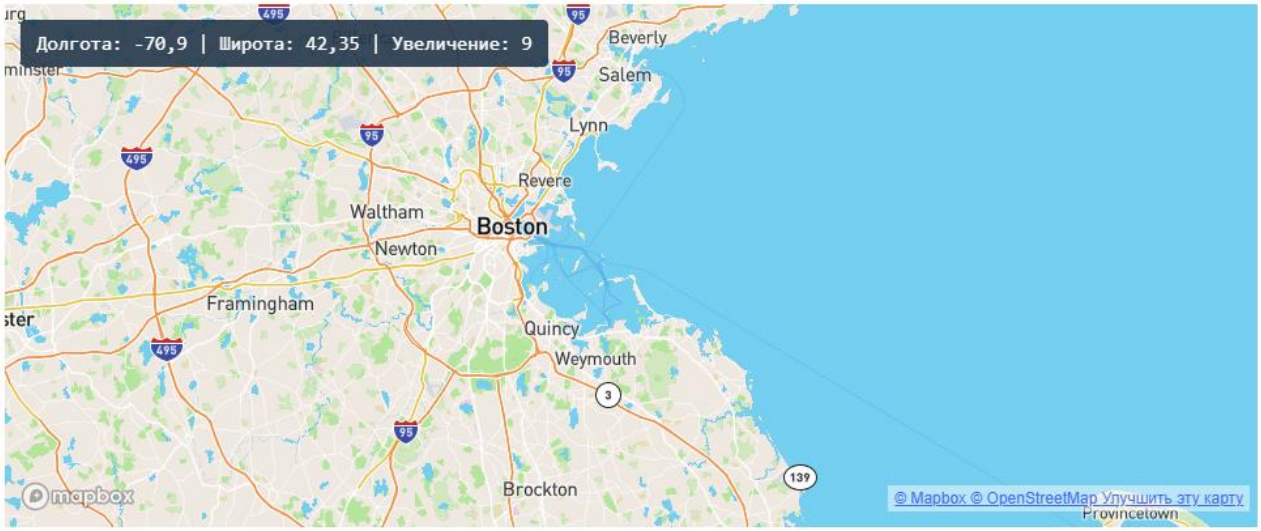
```
.sidebar {
  background-color: rgba(35, 55, 75, 0.9);
  color: #ffffff;
  padding: 6px 12px;
  font-family: monospace;
  z-index: 1;
  position: absolute;
  top: 0;
  left: 0;
```

```
  margin: 12px;
  border-radius: 4px;
}
```

Сохраните свою работу и вернитесь на страницу браузера. В левом верхнем углу карты появилась боковая панель, стилизованная в соответствии с установленными вами правилами CSS `index.css`. На боковой панели отображается текущая широта и долгота центра карты, а также уровень масштабирования. Теперь, когда вы масштабируете и перемещаете карту, содержимое боковой панели обновляется.

Конечный продукт

Приложение React, которое использует Mapbox GL JS для рендеринга карты, отображения координат центра и уровня масштабирования карты, а затем обновляет это отображение, когда пользователь взаимодействует с картой.



Последняя `index.html`страница будет выглядеть следующим образом:

общедоступный / index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta
      name="description"
      content="Create a React web app that uses Mapbox GL JS to render a map"
```

```
    />
    <title>Use Mapbox GL JS in a React app</title>
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

Окончательный `index.js`файл будет выглядеть следующим образом:

src / index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'mapbox-gl/dist/mapbox-gl.css';
import './index.css';
import App from './App';

ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
);
```

Окончательный `App.js`файл будет выглядеть следующим образом:

src / App.js

```
import React from 'react';
import mapboxgl from '!mapbox-gl'; // eslint-disable-line import/no-webpack-loader-syntax

mapboxgl.accessToken = 'YOUR_MAPBOX_ACCESS_TOKEN';

export default class App extends React.PureComponent {
  constructor(props) {
```

```
  super(props);
  this.state = {
    lng: -70.9,
    lat: 42.35,
    zoom: 9
  };
  this.mapContainer = React.createRef();
}
componentDidMount() {
  const { lng, lat, zoom } = this.state;
  const map = new mapboxgl.Map({
    container: this.mapContainer.current,
    style: 'mapbox://styles/mapbox/streets-v11',
    center: [lng, lat],
    zoom: zoom
  });

  map.on('move', () => {
    this.setState({
      lng: map.getCenter().lng.toFixed(4),
      lat: map.getCenter().lat.toFixed(4),
      zoom: map.getZoom().toFixed(2)
    });
  });
}
render() {
  const { lng, lat, zoom } = this.state;
  return (
    <div>
      <div className="sidebar">
        Longitude: {lng} | Latitude: {lat} | Zoom: {zoom}
```

```
      </div>
      <div ref={this.mapContainer} className="map-container" />
    </div>
  );
 }
}
```

Окончательный `index.css`файл будет выглядеть следующим образом:

src / index.css

```
.map-container {
 height: 400px;
}

.sidebar {
 background-color: rgba(35, 55, 75, 0.9);
 color: #ffffff;
 padding: 6px 12px;
 font-family: monospace;
 z-index: 1;
 position: absolute;
 top: 0;
 left: 0;
 margin: 12px;
 border-radius: 4px;
}
```