

Rest Countries API Function

This is a full-stack program. I used React + NodeJS + Express as the architecture of this project.

<https://restcountries.com/#rest-countries>

This URL provides all the APIs for processing country information, this contains detailed information about each country and various APIs for processing the data.

For this project, I have selected a few of these APIs. which are used to return the properties of each country, which includes the common name, the continent, the population, the area, and the Google Map geographic location.

Especially for processing the Google Map API. We input a shortUrl and we must use Backend Server to convert shortUrl into long Url.

There are only 1 file in backend and 5 files in frontend is this project. Here is the structure of this program.

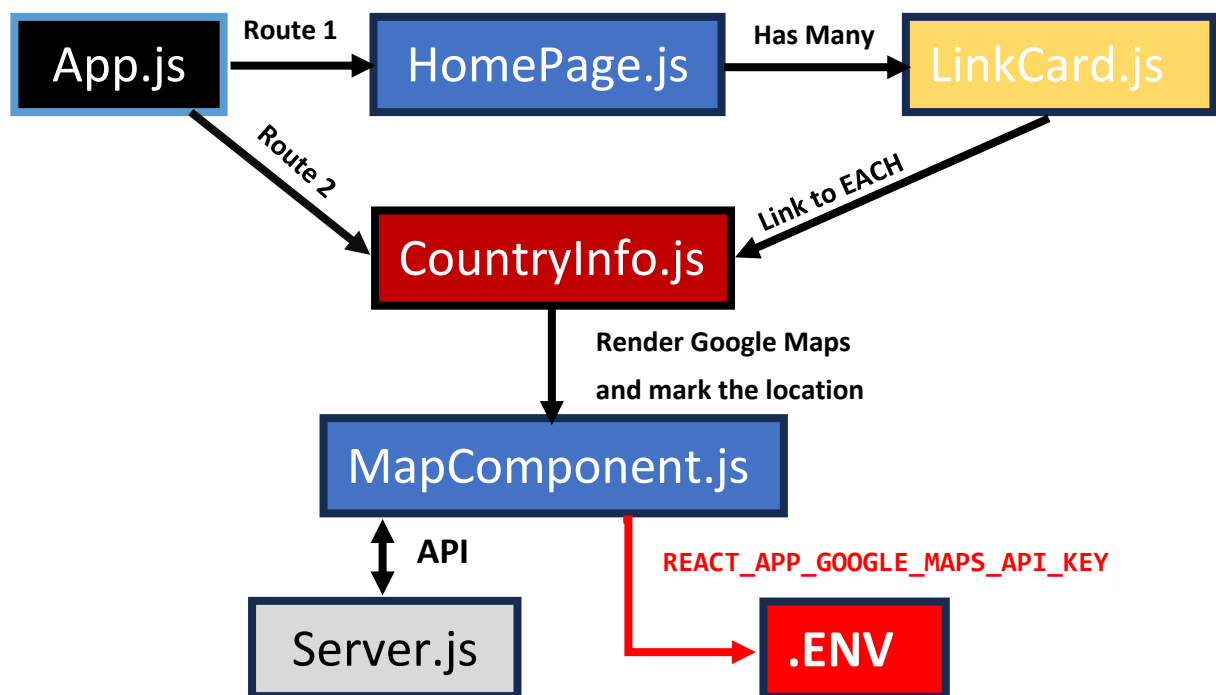
Backend:

- **Server.js**

Frontend:

- **App.js**
- **HomePage.js**
- **LinkCard.js**
- **CountryInfo.js**
- **MapComponent.js**

I draw a picture to explain the relationship between each component.



Then I will explain how we connect Google Map API with React.

There are two files that we need to check carefully –server.js and MapComponent.js.

Remember put your **google Map API Key** into **.env file** first. Then use **.gitignore** to make sure the .env file won't be uploaded to github when you deploy in the future.

If we want to get information from .env file, the variable should be starts with **"REACT_APP"**

Check Map.Component.js, line 21

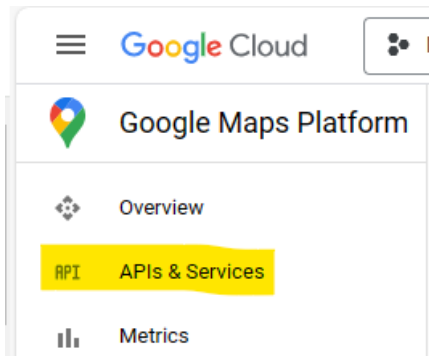
How to get Google Map API?

1) Register a Google Account

2) Go into the website

<https://console.cloud.google.com/google/maps-apis/discover>

3) Click **API and Services** on the left.



4) Select the country.

Step 1 of 2 Account Information






[SWITCH ACCOUNT](#)

Country

By using this application, you agree to the [Google Cloud Platform](#), [Supplemental Free Trial](#), and [any applicable services and APIs Terms of Service](#).

[AGREE & CONTINUE](#)

5) Account type should be individual. Type in to your personal information and bank card information, then “START FREE”

 Account type  

Individual

[START FREE](#)

6) Create a new project

Select the project



CREATE A NEW PROJECT

Search for projects and folders



7) Enable APIs and Services

[+ ENABLE APIS AND SERVICES](#)

8) Select Maps JavaScript API, then enable it.



Maps JavaScript API

Google

Maps for your website

9) That is the Google Map API key, copy and save it.


Get started with Google Maps Platform

You're set up and ready to start developing! Here are the API keys you'll need for your implementation. The API key can be found in the Credentials section.

Your API key

...yA-Zb8x...2c4



☒ Create budget alerts to help me stay informed about my spending and let me know when I'm about to exceed my \$200 Google Maps monthly credit threshold 

GO TO GOOGLE MAPS PLATFORM

That API key is confidential, remember not to show it to anyone else!!

10) Don't set any restrictions.

MapComponent.js

```
import React,{useEffect, useState} from 'react';
import { GoogleMap, LoadScriptNext, Marker } from '@react-google-maps/api';

const containerStyle = {
  width: '400px',
  height: '400px'
};

function MyMapComponent (props) {
  const [latLng, setLatLng] = useState(null);

  useEffect(() => {
    fetch(`https://rest-countries-api-backend.vercel.app/api/resolve-map-url?url=${props.mapUrl}`)
      .then(response => response.json())
      .then(data => setLatLng(data))
      .catch(error => console.error('Error:', error));
  }, [props.mapUrl]);

  return(
    <LoadScriptNext
      googleMapsApiKey={"AIzaSyAlqrv7wV0dkifj00qKrTEIAsso8yAoe0U"}>
      {latLng && (
        <GoogleMap mapContainerStyle={containerStyle} center={latLng}
          zoom={5}>
          <Marker position={latLng} />
        </GoogleMap>
      )}
    </LoadScriptNext>
  );
}

export default MyMapComponent;
```

LoadScriptNext googleMapsApiKey={...}

这是一个用于加载 Google Maps JavaScript API 的 React 组件。

LoadScriptNext 是一个高级组件，它负责加载 Google Maps 库。你需要向它提供一个有效的 API 密钥。

使用这个组件是为了确保在渲染地图相关的组件之前，Google Maps 的脚本已被加载和初始化。

LoadScriptNext vs LoadScript

LoadScript 是较早的组件，会在组件首次渲染时插入一个 `<script>` 标签到 HTML 文档的 `<head>` 部分，从而加载 Google Maps API。如果组件卸载并再次挂载，LoadScript 可能会重新加载脚本，这可能导致性能问题或不必要的网络请求。

LoadScriptNext 是 LoadScript 的改进版，在加载脚本时更加高效，特别是在组件卸载和重新挂载的情况下。它可以避免重复加载 Google Maps API 脚本，从而提高应用的性能。

`{latLng && (...}`

这是一个 JavaScript 条件渲染表达式。它检查 `latLng` 变量是否存在（或者说是“真值”）。如果 `latLng` 有值（比如一个包含纬度和经度的对象），那么括号内的组件会被渲染。如果 `latLng` 是 `null` 或 `undefined`，则不渲染任何内容。

这是一种常见的模式，**用于在数据还未准备好时避免渲染**

`<GoogleMap mapContainerStyle={containerStyle} center={latLng} zoom={5}>`

GoogleMap 组件用于在页面上显示一个 Google 地图实例。

- 1) **mapContainerStyle**: 定义地图容器的样式，比如宽度和高度。
- 2) **center**: 设置地图的中心点，这里使用 `latLng` 变量的值。它包含纬度 (`lat`) 和经度 (`lng`) 的对象。
- 3) **zoom**: 设置地图的初始缩放级别。缩放级别越高，地图显示的范围越小。

`<Marker position={latLng} />`

Marker 组件用于在地图上放置一个标记。它的 `position` 属性指定标记的位置，这里同样使用 `latLng` 变量。

server.js

```
const express = require('express');
const request = require('request');
const cors = require('cors');

const app = express();
app.use(cors());

app.get('/api/resolve-map-url/', (req, res) => {
  const shortUrl = req.query.url;

  request({ url: shortUrl, followRedirect: false }, (err, response, body) =>
  {
    if (err) {
      return res.status(500).send('Error occurred');
    }

    const longUrl = response.headers.location || shortUrl;
    const regex = /@([0-9.-]+),([0-9.-]+)/;
    const matches = longUrl.match(regex);

    if (matches && matches.length >= 3) {
      const latLng = { lat: parseFloat(matches[1]), lng:
parseFloat(matches[2]) };
      res.json(latLng);
    } else {
      res.status(404).send('Coordinates not found');
    }
  });
});

// Test the Vercel
app.get("/", (req, res) => {
  res.send("You succeeded to deploy backend to Vercel!");
});

const port = 5000;
app.listen(port, () => console.log(`Server running on port ${port}`));
```

req.query

在 Express 框架中，req.query 是一个对象，获取 query parameter 的参数。query parameter 是 URL 的一部分，位于路径之后，以 ? 开始，格式通常是 key=value 的键值对，多个键值对之间用 & 分隔。

例如 <http://example.com/api/items?color=blue&size=large>

查询字符串是 `color=blue&size=large`。

```
req.query.color = "blue"
req.query.size = "large"
```

在这里，前端传入

```
fetch(`https://rest-countries-api-backend.vercel.app/api/resolve-map-url?url=${props.mapUrl}`)
```

后端来处理

```
查询字符串是 url=${props.mapUrl}
req.query.url = ${props.mapUrl}
```

request

`request` 是一个 Node.js 中用于发起 HTTP 请求的函数。这里调用了一个外部 API，目的是将短 URL 转换为长 URL。

在这里输入 `shortUrl` 请求，就能收到回复 `response.headers.location`

followRedirect: false

禁用 HTTP 重定向。

HTTP 重定向是一种网络请求的响应方式，其中服务器告诉客户端（如浏览器、API 调用者）去访问另一个 URL 而非原始请求的 URL。重定向在 Web 开发中非常常见，用于多种场景，比如网页搬迁、短链接服务、负载均衡等。

如果你访问一个网页，服务器可能返回一个 301 状态码和一个新的 URL，在这种情况下，你的浏览器会自动跳转到这个新的 URL。

`Request` 函数在遇到 HTTP 重定向（常用于短链接服务）时不要自动跟随到新的 URL。这意味着即使 `shortUrl` 指向另一个地址（即长 URL），请求也不会自动转向那个地址。

shortUrl -> longUrl

转换网址：<https://urlex.org/>

shortUrl: <https://goo.gl/maps/p9qC6vgiFRRXzvGi7>

longUrl:

<https://www.google.com/maps/place/China/@34.4137724,86.0453361,4z/data=!3m1!4b1!4m5!3m4!1s0x31508e64e5c642c1:0x951daa7c349f366f!8m2!3d35.86166!4d104.195397?ucbc=1>

REGEX

```
const regex = /@([0-9.-]+),([0-9.-]+),/;
```

这是一个正则表达式

- 1) 以 **@** 开头，因为 google Map longUrl 的格式就是**@维度，经度，缩放大小**
- 2) **([0-9.-]+)** 至少有 1 个，①数字 0 到 9 / ②点. / ③负号 -
- 3) **捕获组**，上述代码有 2 个捕获组**([0-9.-]+)**，捕获符合要求的字符串。

语法: **/ (pattern1) (pattern2) /**

说明: pattern1 和 pattern2 是你想要捕获的正则表达式的一部分。

第一个捕获组捕获 pattern1，第二个捕获组捕获 pattern2

- 4) **匹配 match**

```
string.match(regex);
```

string: 原始字符串

regex: 正则表达式对象

如果匹配 **match** 和**捕获组 ()** 一起使用，那么会返回一个的**数组**。

例如项目中的

```
const matches = longUrl.match(regex);
```

match 方法返回一个**数组**

matches[0] : 整个匹配的字符串，包括 **@** 符号和两个坐标。

matches[1]: **第一个捕获组**的匹配结果，即**纬度**。

matches[2] **第二个捕获组**的匹配结果，即**经度**。

parseFloat

把字符串转化为对应的浮点数。如果匹配 **match** 和**捕获组 ()** 一起使用