实验：使用 useFetch 获取数据

Skills Network

预计所需时间： 40分钟

你将学到什么

在这个实验中，你将学习如何使用自定义钩子（UseFetch）和相应的组件（FetchData）在 React 中获取数据。通过将数据获取逻辑封装到可重用的钩子中，你可以提高代码的可维护性。你将掌握 React 的 useState 和 useEffect 钩子，以高效地管理状态和异步操作。此外，你还将学习如何使用 JSX 动态渲染获取的数据，从而创建动态和交互式组件。

学习目标

完成本实验后，你将能够：

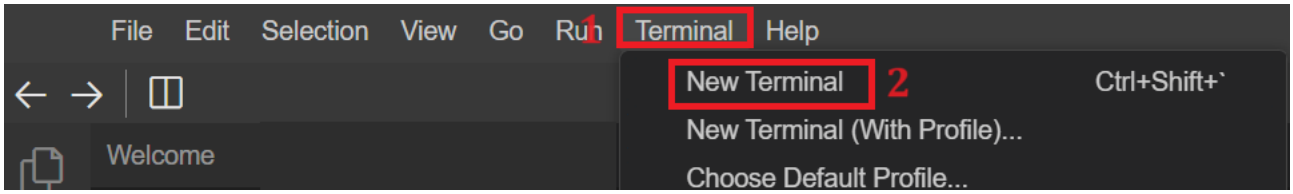
- 在 React 应用程序中使用 fetch API 方法从外部 API 获取数据
- 实现一个自定义 React 钩子（UseFetch），以封装获取数据的逻辑
- 将从 API 获取的数据渲染到 UI 上，并显示每个项目的相关信息，如名称、重要性、好处以及最佳摄入时间

先决条件

- 基础的 HTML 知识
- 中级的 JavaScript 知识
- 基础的 React 钩子和自定义钩子知识

设置环境

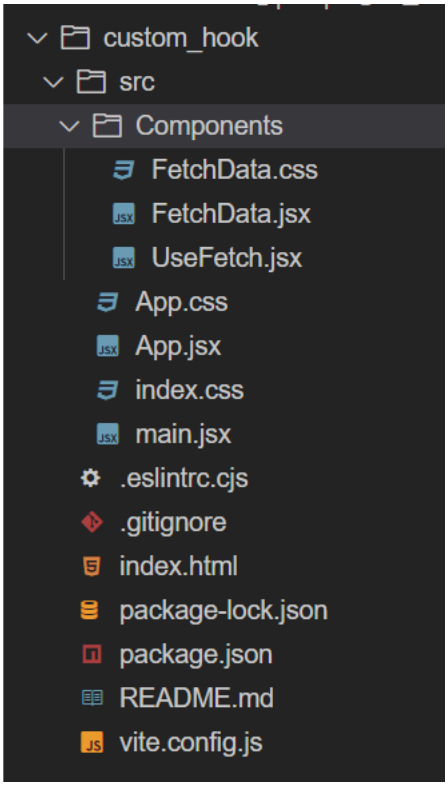
1. 从实验室顶部菜单中，点击窗口右上角的 **Terminal** 选项卡，如给定截图中的数字 1 所示，然后点击 **New Terminal**，如数字 2 所示。



2. 在终端中输入以下命令以克隆此 React 应用程序的模板，并按 Enter 键。custom_hook 应用程序包含名为 **UseFetch.jsx** 和 **FetchData.jsx** 的类组件，以及一个名为 **FetchData.css** 的 css 文件。

```
git clone https://github.com/ibm-developer-skills-network/custom_hook.git
```

3. 这将在 **project** 文件夹下创建一个名为 **custom_hook** 的文件夹，结构如给定截图所示。



4. 在终端中输入命令以进入 **custom_hook** 文件夹。使用以下命令在终端中导航到 **custom_hook** 文件夹。

    ```
    cd custom_hook
    ```

5. 为了确保你克隆的代码能够正常工作，你需要按照以下步骤操作：

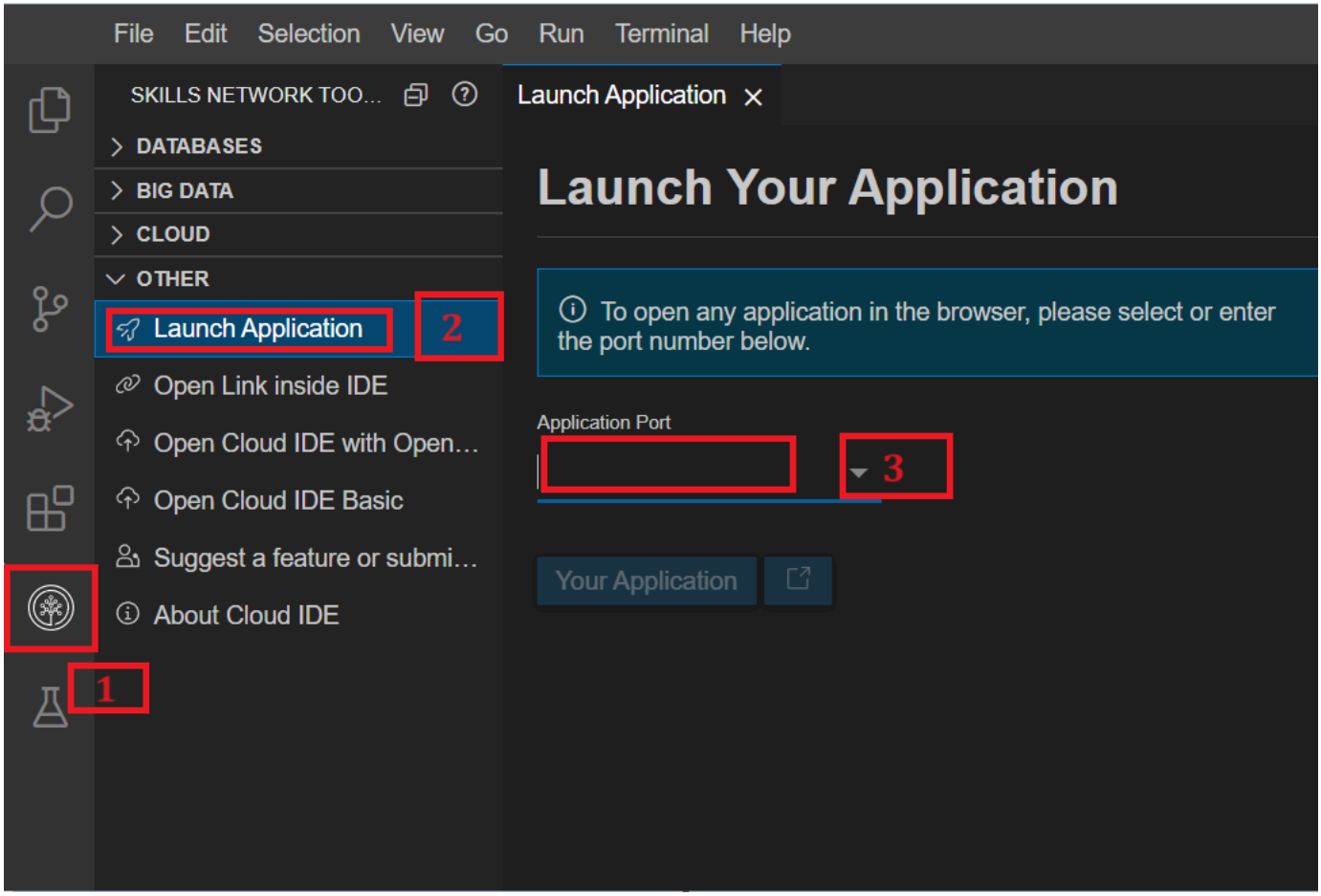
    - 在终端中输入以下命令并按 Enter 键，以安装执行应用程序所需的所有必要包：

        ```
        npm install
        ```

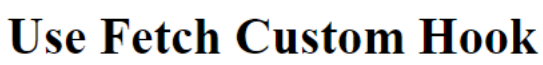
    - 然后执行以下命令以运行应用程序，这将为你提供端口号 4173：

        ```
        npm run preview
        ```

6. 要查看你的 React 应用程序，请点击左侧面板上的技能网络图标（参见数字 1）。此操作将打开 **SKILLS NETWORK TOOLBOX**。接下来，点击 **Launch Application**（参见数字 2）。在 **Application Port** 中输入端口号 **4173**（参见数字 3），然后点击 。



7. 输出将如给定截图所示。



创建自定义 Hook

1. 接下来，导航到位于克隆的 **custom_hook** 文件夹中的 **src** 目录下的 **Components** 文件夹内的 `FetchData.jsx` 组件。

2. `FetchData.jsx` 组件的基本结构已经提供，包含一个默认的函数组件，里面有一个 `<h1>` 标签作为标题和一个类名为 **list_data_main** 的 `<ul>` 标签。

```
import React from 'react'
const FetchData = () => {
  return (
    <>
     <ul className='list_data_main'>
        <h1 className='usefetch_heading'>Use Fetch Custom Hook</h1>
     </ul>
    </>
  )
}
export default FetchData
```

3. 现在你需要从 API 数据链接 获取数据。获取数据有两种方法：

   - 首先，在每个需要获取数据的组件中使用 fetch API 方法实现逻辑。假设你有 5 个组件需要从外部源加载数据。在这种情况下，你需要为每个组件编写类似的逻辑来获取详细信息。
   - 其次，创建一个可重用的自定义钩子。这个自定义钩子封装了数据获取逻辑，使其可以在多个组件中重用，而无需在每个组件中重写整个获取详细信息的代码。

4. 要创建自定义钩子，你需要导航到 **src** 文件夹下 **Components** 文件夹中的 `useFetch.jsx` 组件。组件的基本布局已提供给你，如下所示：

```
const UseFetch = (url) => {
}
export default UseFetch
```

   - 在上面的代码中，声明了一个带有一个参数 `url` 的箭头函数。
   - 此方法已作为默认导出。

5. 在基本布局中，你需要实现 `useEffect` 钩子，以创建一个获取数据的模板。

6. 使用 `useState` 钩子创建一个名为 `data` 的变量，并将 `setData` 作为函数。将此代码包含在箭头函数组件中。

```
 const[data,setData]=useState();
```

7. 还要在顶部包含给定的语句 `useFetch.jsx` 组件，以确保 **useState** 钩子的正常工作。

```
import { useState } from "react";
```

6. 现在在箭头函数中实现 `useEffect` 钩子，在其中创建从给定 URL 使用 fetch API 方法获取数据的逻辑，并将数据作为数组返回。将此代码包含在使用 **useState** 钩子声明变量之后。

```
 useEffect(()=>{
        fetch(url).then((res)=>res.json())
        .then((data)=>setData(data))
    },[])
return [data]
```

   - `useEffect`：这是一个 React 钩子，用于在函数组件中执行副作用。通常用于获取数据、订阅事件或任何其他不涉及渲染的副作用。
   - `fetch(url)`：这会向指定的 URL 发起 HTTP 请求。
   - `.then((res) => res.json())`：这将服务器的响应转换为 JSON 格式。
   - `.then((data) => setData(data))`：这将检索到的数据设置为状态变量 data。setData 是一个用于更新 React 函数组件中状态的函数。

7. 将给定代码包含在 `useFetch.jsx` 组件的顶部，以确保 **useEffect** 钩子的正常工作。

```
import {useState, useEffect } from "react";
```

8. 整个代码应具有与以下相同的结构：

```
import React, { useEffect, useState } from 'react'
 const UseFetch = (url) => {
    const[data,setData]=useState();
    useEffect(()=>{
        fetch(url).then((res)=>res.json())
        .then((data)=>setData(data))
    },[])
  return [data]
}
export default UseFetch
```

`url` 变量作为参数传递给 `UseFetch` 箭头函数，以指定从特定网站获取数据的确切 URL。此函数允许使用自定义钩子的组件动态提供 URL，从而实现无缝的数据获取，避免冗余代码。

使用自定义 Hook 获取数据

1. 现在再次导航到 **custom_hook** 文件夹中 **src** 目录下的 `FetchData.jsx` 文件。

2. 通过在 `FetchData` 箭头函数中传递特定的 URL 来实现自定义 Hook，从中获取数据。同时包含 `console.log` 以检查变量 `data` 是否已成功获取数据。在 `FetchData.jsx` 函数组件的 `return` 语句之前包含给定代码。

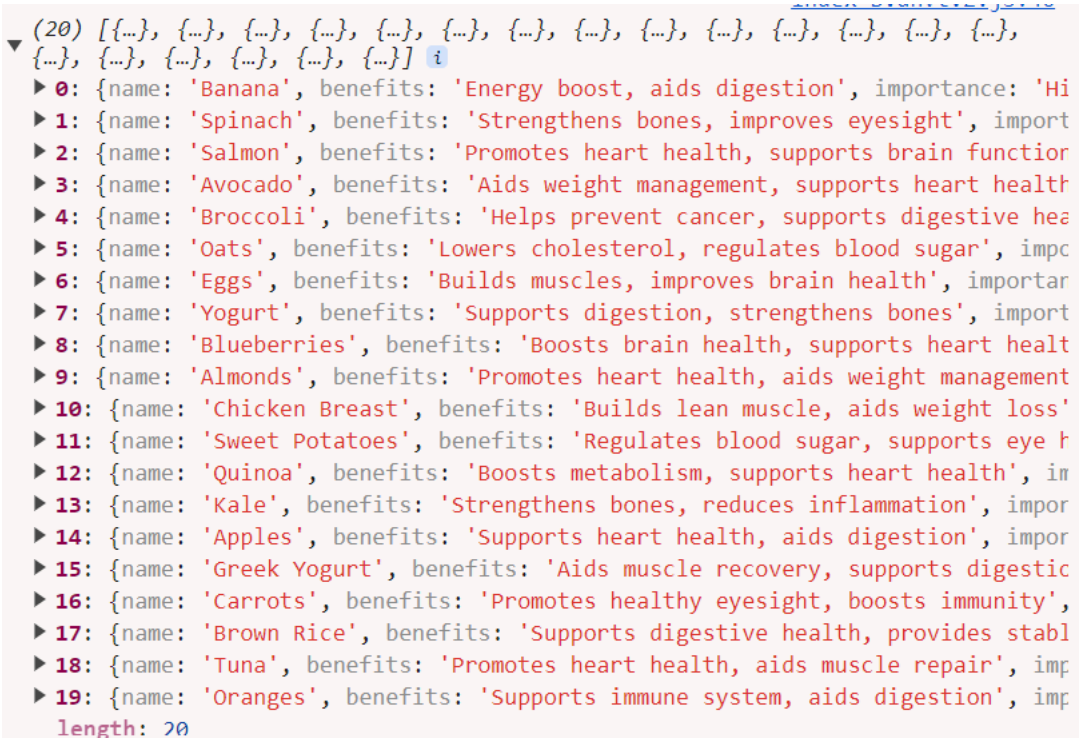
```
 const [data]=UseFetch('https://api.npoint.io/9045c260b1565daa9e15');
 console.log(data);
```

3. 导入 **UseFetch** 以确保该组件正常工作。为此，在 `FetchData.jsx` 组件的顶部包含给定代码。

```
import UseFetch from './UseFetch';
```

```

4. To check the output rerun the React application again. In the browser right-click and select **inspect** and click the **Console** tab. The output should display as shown in the given screenshot.
```

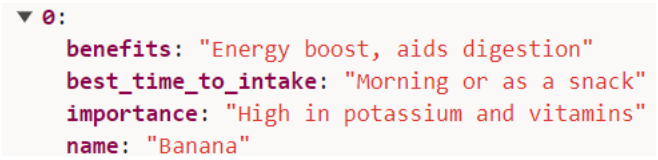


Retrieve Data in Front End

1. To retrieve the data, you need to perform an iteration of the data array to fetch the entire data. You need to apply the `map` array method within the `<ul>` tag after `<h1>...</h1>` tag of `FetchData.jsx` component as follows:

```
{data && data.map((e)=>(
     <>
     </>
   ))}
```

2. To implement this, create the `<li>` tag with the class name `list_data` within fragments `<></>` of `<ul>` tag. Afterwards, inspect the `console.log` output to determine which data you wish to display on the frontend. For example, in the given screenshot, upon expanding the first index object of the array, you'll notice that there are four distinct types of information that can be accessed.



3. These distinct types, then can fetch data using `e` variable that you have passed as parameter in the `map` method.

```
<li className='list_data'>
          <h3>{e.name}</h3>
          <p><strong>Importance: </strong>{e.importance}</p>
          <p><strong>Benefits: </strong>{e.benefits6}</p>
          <p><strong>Time to eat: </strong>{e.best_time_to_intake}</p>
          </li>
```

4. Include css file inside `FetchData.jsx` component using import statement. Include give statement before function component along with other import statements.

```
import './FetchData.css'
```

5. The entire code will look like below:

```
import React from 'react'
import UseFetch from './UseFetch'
import './FetchData.css'
 const FetchData = () => {
    const [data]=UseFetch('https://api.npoint.io/9045c260b1565daa9e15');
    console.log(data);
  return (
   <>
   <ul className='list_data_main'>
    {data && data.map(e=>(
        <>
        <li className='list_data'>
            <h3>{e.name}</h3>
            <p><strong>Importance: </strong>{e.importance}</p>
            <p><strong>Benefits: </strong>{e.benefits6}</p>
            <p><strong>Time to eat: </strong>{e.best_time_to_intake}</p>
            </li>
        </>
    ))}
   </ul>
```

```
    </>
  )
}
export default  FetchData
```

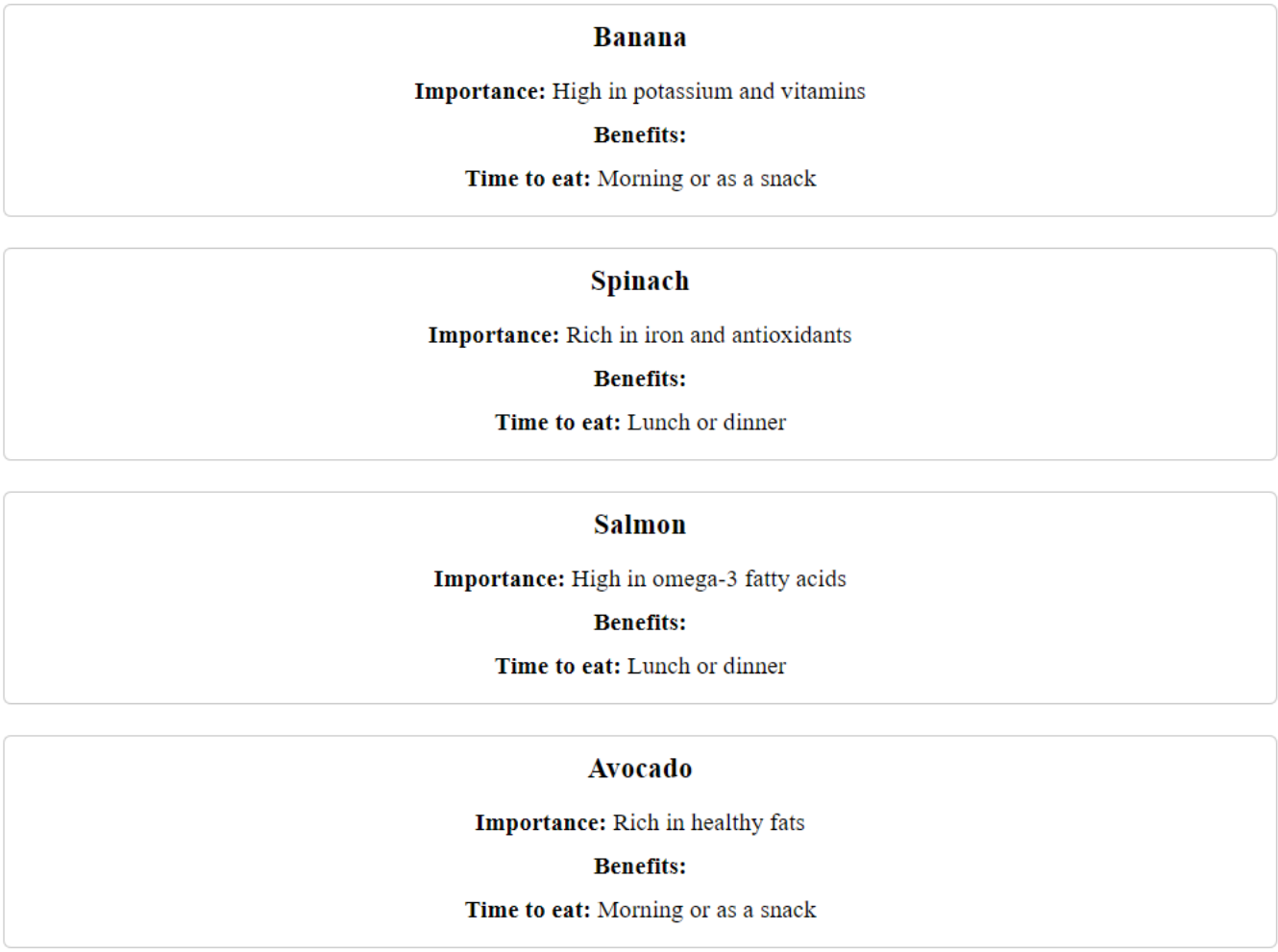
Note: The `e` parameter contains the value of each array index for every iteration.

Check the Output

1. To stop the execution of the React application in the terminal perform `ctrl+c` to quit.
2. Then write the given command in the terminal and hit Enter.

   ```
   npm run preview
   ```

3. To view your React application, refresh the already opened webpage for the React application on your browser. If it is not open then, click the Skills Network icon on the left panel. This action will open the "SKILLS NETWORK TOOLBOX." Next, select "Launch Application". Enter port number **4173** in "Application Port" and click .
4. The output will be displayed as shown in the given screenshot.



Note: To see the latest changes you need to execute `npm run preview` again in the terminal.

5. ▼ Click here to see the full sample solution for "`FetchData.jsx".

```
import React from 'react';
import UseFetch from './UseFetch';
import './FetchData.css'
const FetchData = () => {
    const [data]=UseFetch('https://api.npoint.io/9045c260b1565daa9e15');
    console.log(data);
  return (
    <>
     <ul className='list_data_main'>
        <h1 className='usefetch_heading'>Use Fetch Custom Hook</h1>
        {data && data.map((e)=>(
      <>
       <li className='list_data'>
            <h3>{e.name}</h3>
            <p><strong>Importance: </strong>{e.importance}</p>
            <p><strong>Benefits: </strong>{e.benefits6}</p>
            <p><strong>Time to eat: </strong>{e.best_time_to_intake}</p>
            </li>
      </>
    ))}
```

```
        </ul>
       </>
     )
}
export default FetchData
```

6. ▼ Click here to see the full sample solution for "UseFetch.jsx".

```
import {useState, useEffect } from "react";
const UseFetch = (url) => {
    const[data,setData]=useState();
    useEffect(()=>{
        fetch(url).then((res)=>res.json())
        .then((data)=>setData(data))
    },[])
return [data]
}
export default UseFetch
```

7. ▼ Click here to see the full sample solution for "App.jsx" parent component.

```
import React from 'react'
import FetchData from './Components/FetchData'
function App() {
  return (
 <>
 <FetchData/>
 </>
  )
}
export default App
```

Practice Exercise

1. In this practice exercise you will see how custom hook `UseFetch.jsx` component can be used in other component to fetch data.
2. You will need to fetch data for API related to **Yoga**. Click on this link Yoga-Benefits to see the api data.
3. First create one component by right clicking on **Components** folder after selecting it and then click on **New File**. Give the component named as `FetchYogaData.jsx`.
4. Then create basic layout for in `FetchYogaData.jsx` component.

   Hint: Use function component struture

▼ Click here for the sample solution

```
import React from 'react'
const FetchYogaData = () => {
  return (
   <>
   </>
  )
}
export default FetchYogaData
```

5. Now import custom hook component `UseFetch.jsx` into `FetchYogaData.jsx` component and pass api link as url to it.

   APi url is:

```
import UseFetch from './UseFetch';
```

   Hint: Use import statement to use custome hook component in current component and declare variable varibale to pass url.

▼ Click here for the sample solution

```
import UseFetch from './UseFetch';
const [data]=UseFetch('https://api.npoint.io/4459a9a10e43812e1152');
```

- Include import statement at the top of the component and data variable before the return statement of the function component.

6. Write `console.log` statement to check if data is coming in console tab in browswer or not.

   Hint: write `console.log(data)` before return statement

7. Include `FetchYogaData.jsx` component in the `App.jsx` parent component and then Check the output by re-running the application in terminal again.

   Hint: use import statement in.

8. Create `<ul>` tag with className as list_data_main. Within this, create logic to map through the fetched data and display each item inside `<li>` tags.

Conclusion

Congratulations! You have created your React application to fetch the data using your own personlaized custom hook!

- In this lab, you created a React component named **FetchData** responsible for rendering fetched data onto the user interface. It uses the **UseFetch** custom hook to retrieve data from an API endpoint specified by a URL.
- The custom React hook named **UseFetch** encapsulates the logic for fetching data from a specified URL. This hook uses **useState** and **useEffect** to manage the data fetching process, ensuring that data is retrieved asynchronously and stored in the component's state.
- You have learned to implement the map method through the fetched data array that renders each item onto the UI using JSX. It displays various attributes of each data item, such as name, importance, benefits, and the best time to intake, within styled list items.
- By separating the data fetching logic into a custom hook (UseFetch), you have adopted a modular and reusable approach. This approach allows components throughout the application to easily fetch data from different API endpoints by simply providing the desired URL as an argument to the UseFetch hook, promoting code efficiency and maintainability.

Author(s)

Richa Arora

© IBM Corporation. All rights reserved.