# Lab 18: HTTP GET

## Objectives

*  Create an API object that loads data from an REST API
*  Update a component to use the API object
*  Add Pagination

## Steps

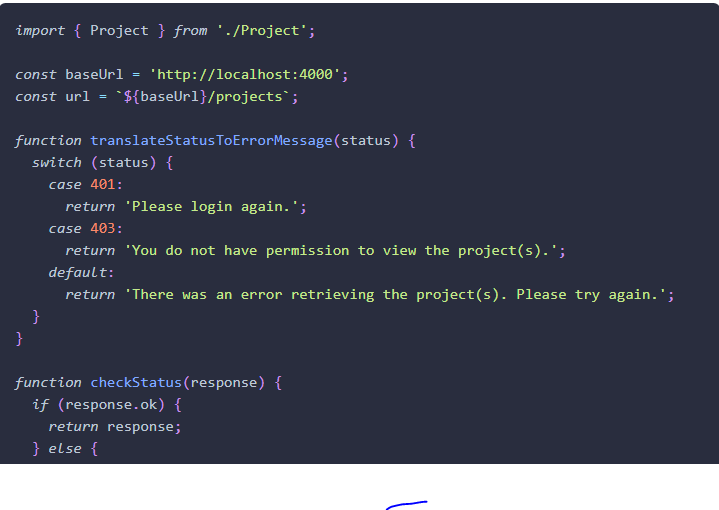
### Create an API object that loads data from an REST API

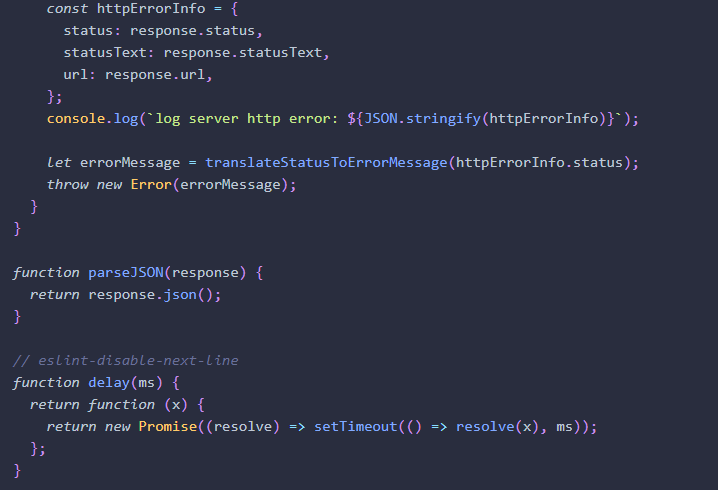
1. Create the file src\projects\projectAPI.js.
2. Create a projectAPI object and export it from the file.
3. Implement a get method that requires page and limit parameters and sets the default to page = 1 and limit=20. The projects should be sorted by name.

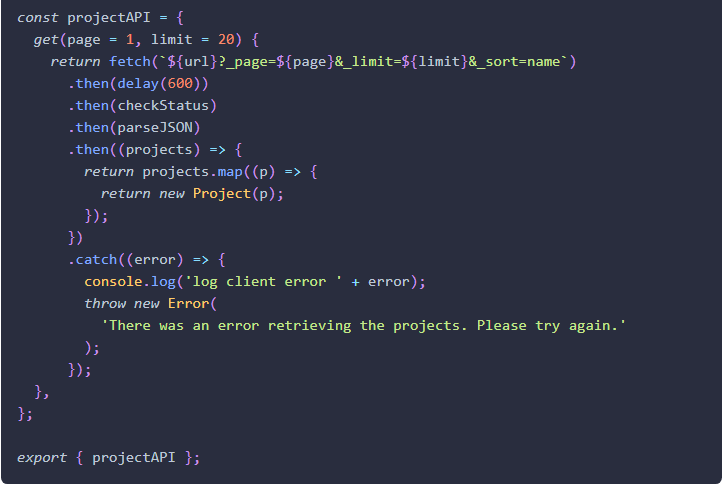
[json-server](https://github.com/typicode/json-server/blob/master/README.md" \t "https://handsonreact.com/docs/labs/js/_blank)supports sorting and paging using the following syntax.



#### src\projects\projectAPI.js



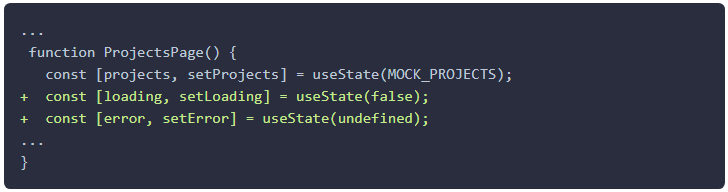




### Update a component to use the API object

1. **Open** the file src\projects\ProjectsPage.js.
2. **Use** the useState function to create two additonal state variables loading and error.

#### src\projects\ProjectsPage.js



**DO NOT DELETE** the file src\projects\MockProjects.js. We will use it in our unit testing.

1. **Change** the projects **state** to be an empty array [] **(be sure to remove the mock data)**.

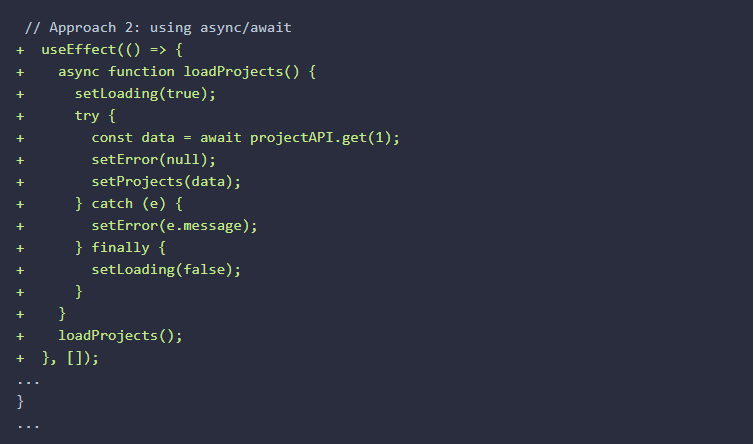
#### src\projects\ProjectsPage.js



1. **Implement** the loading of the data from the API after the intial component render in a useEffect hook. Follow these specifications.
   1. **Set** state of loading to true
   2. Call the API: projectAPI.get(1).
   3. If **successful**, **set** the returned data into the components projects state variable and set the loading state variable to false.
   4. If an **error occurs**, **set** the returned error's message error.message to the components error state and loading to false.

#### src\projects\ProjectsPage.js





1. **Display** the **loading** indicator **below** the <ProjectList />. Only display the indicator when loading=true.

If you want to try it yourself first before looking at the solution code use the HTML snippet below to format the loading indicator.

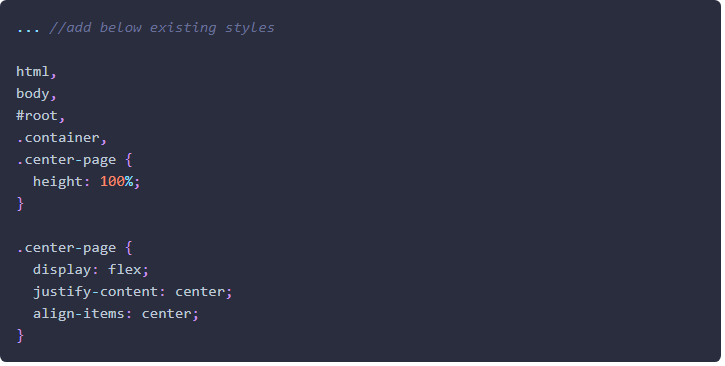


#### src\projects\ProjectsPage.js



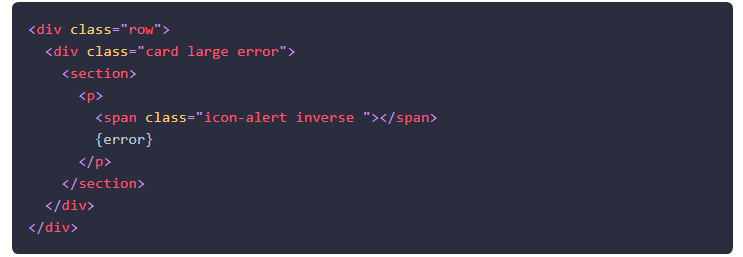
1. Add these CSS styles to center the loading indicator on the page.

#### src\index.css

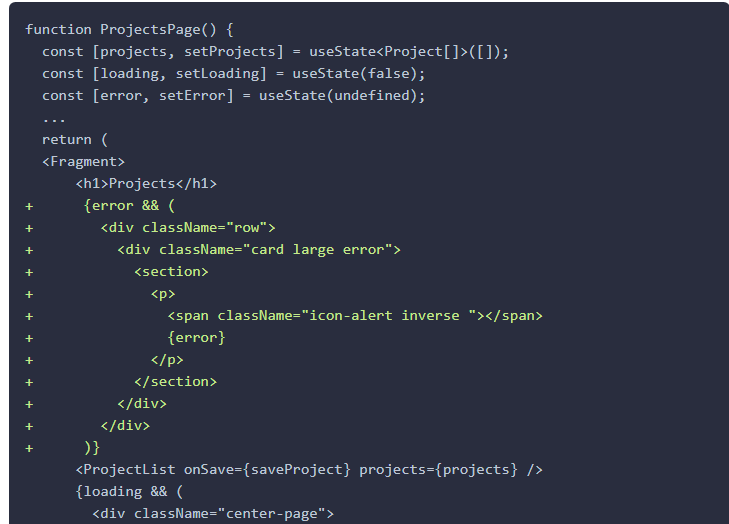


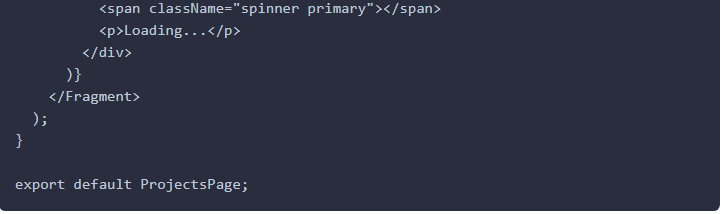
1. **Display** the **error** message **above** the <ProjectList /> using the HTML snippet below. Only display the indicator when error is defined.

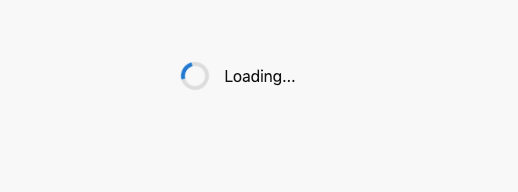
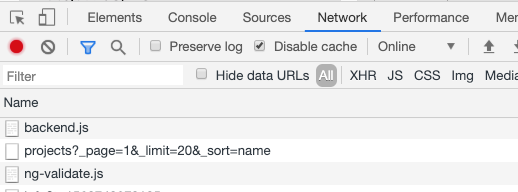
If you want to try it yourself first before looking at the solution code use the HTML snippet below to format the error.



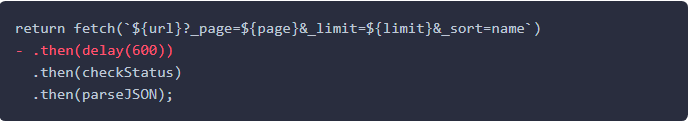
#### src\projects\ProjectsPage.js





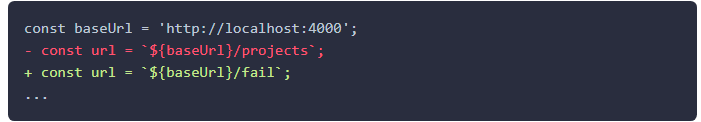
1. **Verify** the application is working by **following these steps** in your Chrome **browser**.
2. **Open** the application on http://localhost:3000.
3. **Open** Chrome DevTools.
4. **Refresh** the page.
5. For a brief second, a **loading indicator** should **appear**. 
6. Then, a list of **projects** should **appear**.
7. **Click** on the Chrome DevTools Network tab.
8. **Verify** the **request** to /projects?\_page=1&\_limit=20&\_sort=name is happening. 
9. We are using a delay function in projectAPI.get() to delay the returning of data so it is easier to see the loading indicator. You can remove the delay at this point.

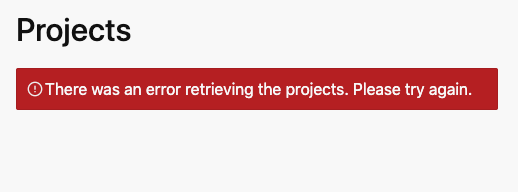
#### src\projects\projectAPI.js



1. **Change** the **URL** so the API endpoint cannot be reached.

#### src\projects\projectAPI.js



1. In your browser, you should see the following **error message** **displayed**. 
2. **Fix** the **URL** to the backend API before continuing to the next lab.

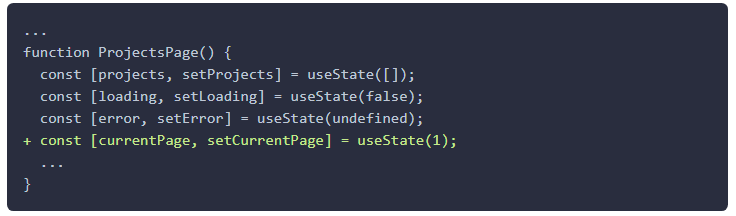
#### src\projects\projectAPI.js



### Add Pagination

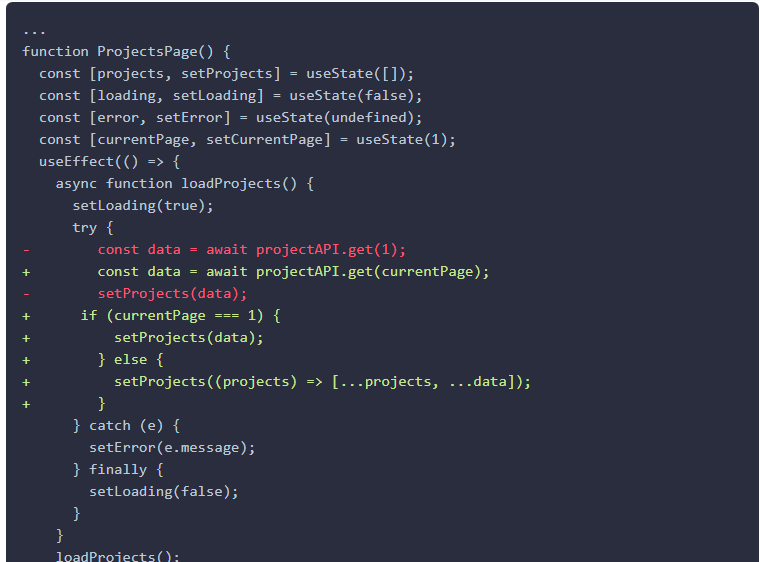
1. **Use** the useState function to create an additonal state variable currentPage.

#### src\projects\ProjectsPage.js



1. **Update** the useEffect method to make currentPage a dependency and use it when fetching the data.

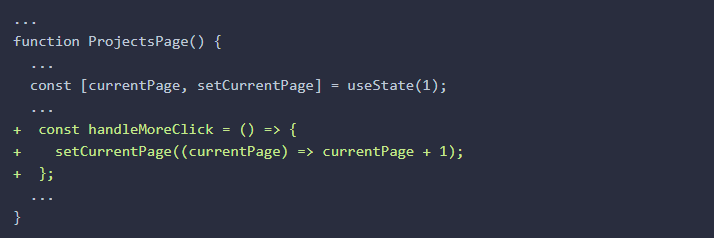
#### src\projects\ProjectsPage.js





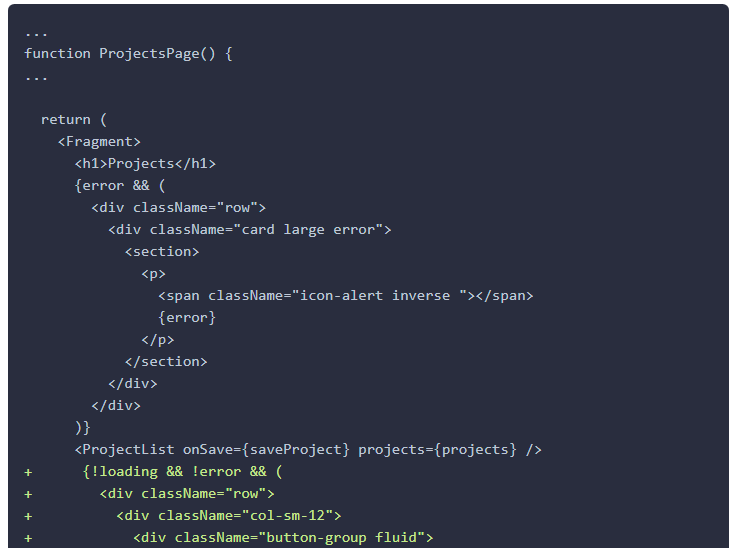
1. **Implement** a handleMoreClick event handler and implement it by incrementing the page and then calling loadProjects.

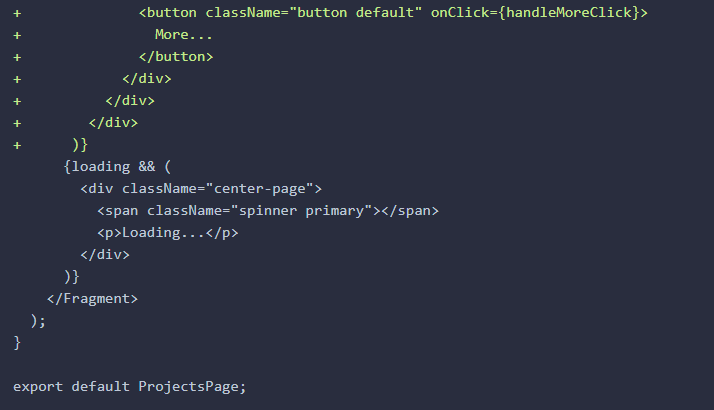
#### src\projects\ProjectsPage.js



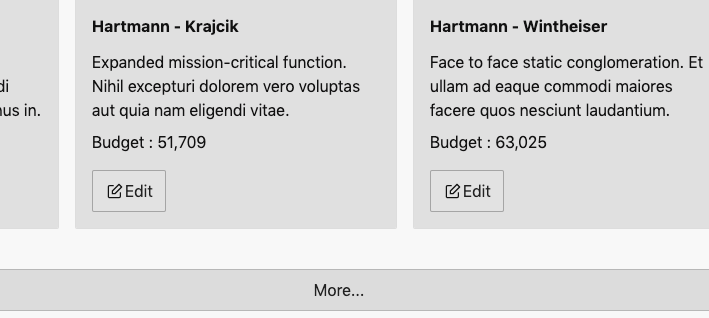
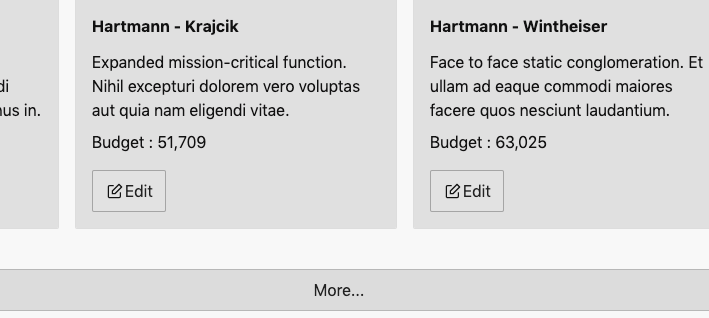
1. Add a More... button below the <ProjectList /> . Display the More... button only when not loading and there is not an error and handle the More... button's click event and call handleMoreClick.

#### src\projects\ProjectsPage.js





1. **Verify** the application is working by **following these steps** in your **browser**.
   1. **Refresh** the page.
   2. A list of **projects** should **appear**.
   3. **Click** on the More... **button**.
   4. **Verify** that 20 additional projects are appended to the end of the list.
   5. **Click** on the More... **button** again.
   6. **Verify** that another 20 projects are appended to the end of the list.

### ✔ You have completed the Lab 18