How to create a Hacker News clone using React

Are you looking to upgrade your React programming skills? Then try to build your own version of Hacker News with the help of the guide below.

Hacker News is a popular website among entrepreneurs and programmers. This site focuses on computer science and business.

Hacker News' simple layout may be suitable for some individuals. However, if you want a more engaging and personalized version, you can use the helpful API to create a customized Hacker News experience. Also, building Hacker News copy can help you strengthen your React skills.

Setting up the project and development server

Packages needed in programming include:

1. React Router to handle routing in Single Page Application (SPA).
2. HTMLReactParser to parse the HTML returned by the Application Programming Interface (API).
3. MomentJS to handle the date returned by the API.

Open this terminal and run:

```
yarn create vite
```

You can also use Node Package Manager (NPM) if you prefer it over yarn. The above command will use the Vite tool to build a basic project. Give the project a name and when prompted for the framework, select **React** and set the variable to **JavaScript** .

Now put **the cd** into the project directory and install the packages mentioned earlier by running the following command in the terminal:

```
yarn add html-react-parser yarn add react-router-dom yarn add moment yarn dev
```

After installing all the packages and starting the server programming, open the project in any code editor and create 3 folders in the **src** named respectively: **components** , **hooks** , and **pages** .

In the components folder , add two files: **Comments.jsx** and **Navbar.jsx** . **In the hooks** folder , add a file **useFetch.jsx** . Then, in the **pages** folder , add two files **ListPage.jsx** and **PostPage.jsx** .

Delete the **App.css** file and replace the content of the **main.jsx** file with:

```
yarn add html-react-parser yarn add react-router-dom yarn add moment yarn dev
```

In the **App.jsx** file , remove all boilerplate code and edit the file so that you have only functional components:

```
function App() { return ( >  ) } export default App
```

Import the required modules:

```
import { Routes, Route } from 'react-router-dom' import ListPage from './pages/Li
```

In the React fragment, add Routes components with 3 **Route** child components with paths: / , **/:type** , and **/item/:id** respectively.

```
 }> }> }>
```

Create custom hook useFetch

This project uses 2 APIs. The first API is responsible for fetching the list of articles in the provided directory, and the second is the Algolia API, which is responsible for fetching a specific article and its comments.

Open the file **useFetch.jsx** , define the hook as the default export choice and import **the useState** and **useEffect** hooks .

```
import { useState, useEffect } from "react"; export default function useFetch(typ
```

Define 3 state variables named: **data** , **error** and **loading** with their respective setup functions.

const [data, setData] = useState();
const [error, setError] = useState(false);
const [loading, setLoading] = useState(true);

Then add the **useEffect** hook with the dependencies: **id** and **type** .

```
useEffect(() => { }, [id, type])
```

Next in the callback, add the **fetchData()** function to fetch data from the appropriate API. If the parameter passed is **type** , use the first API. If not, use the second API.

```
async function fetchData() { let response, url, parameter; if (type) { url = "ht
```

Finally, return the **loading** , **error** , **data** state variables as an object.

```
return { loading, error, data };
```

Show list of posts by queryed category

Whenever the user navigates to / or **/:type** , React renders the **ListPage** component . To implement this feature, first import the required modules:

```
import { useNavigate, useParams } from "react-router-dom"; import useFetch from "
```

Then define the function element, and then assign the dynamic parameter, **type** to the variable **type** . If there are no dynamic parameters available, set **the type** variable to news. Then call the **useFetch** hook .

```
export default function ListPage() { let { type } = useParams(); const navigate =
```

Next, return the appropriate JSX code depending on whether one of the loading , **error** or **data** state variables is **true** .

```
if (error) { return Something went wrong! } if (loading) { return Loading } if (
```

Create a PostPage Component

First, import the appropriate modules and components, then define the default function element, attach the dynamic parameter **id** to the variable **id** , call the **useFetch** hook . Make sure you unstructure the response.

```
import { Link, useParams } from "react-router-dom"; import parse from 'html-react
```

And like the **ListPage component, JSX** now matches based on the state of the following variables: **loading** , **error** and **data** .

```
if (error) { return Something went wrong! } if (loading) { return Loading } if (
```

Show comment section with nested replies

Import the parse and **moment** modules . Defines the default function element **Comments** , takes the **commentsData** array as an attribute, traverses the arrays, and shows the Node component for each element.

```
import parse from 'html-react-parser'; import moment from "moment"; export defau
```

Next, define the Node function element just below **the Comments** . **The Node** component now renders comments, metadata, and replies to each comment (if any) by recursively rendering itself.

```
function Node({ commentData }) { return { commentData.text && > {commentData.autl
```

In the above code block, **parse** is responsible for parsing the HTML stored in **commentData.text** , while moment is responsible for parsing the comment time and returning the associated time using the **fromNow()** method .

Create a Navbar . component

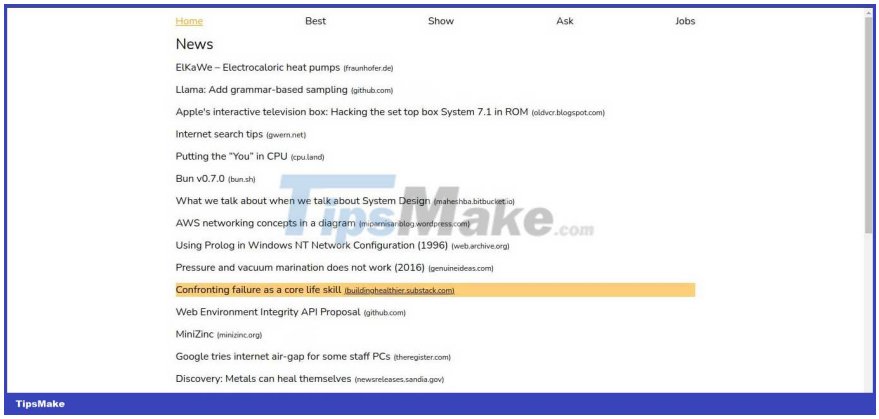
Open the **Navbar.jsx file and import the NavLink** module from **react-router-dom** . Finally, define the function element and return a parent **nav with 5 NavLink** elements pointing to the appropriate categories (or styles).

```
import { NavLink } from "react-router-dom" export default function Navbar() { re
```

Home Best Show Ask Jobs

```
}
```

Congratulations! You just built a front-end client for Hacker News.



Hope the article is useful to you!

You finished reading the article "**How to create a Hacker News clone using React**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.
