Notes

When the user navigates to the web application in the browser, the Web Server will return the necessary resources to run the application. There are two approaches to serving code and resources in Single Page Applications.

1. When the browser requests the application, return and load all necessary HTML, CSS and JavaScript immediately. This is known as bundling.
2. When the browser requests the application, return only the minimum HTML, CSS and JavaScript needed to load the application. Additional resources are downloaded as required by the application, for example, when a user navigates to a specific section of the application. This is known as lazy loading or code splitting.

Both approaches are valid and are used depending on the size, complexity and bandwidth requirements of the application.

React Router library

However, with the default React library,

these anchor tags won't work as expected.

This is because React can't imitate multi-page websites.

However, I can make this possible with the help

of another library known as React Router.

As you may have guessed from the name React Router

gives you more control over the routing of components.

I'll install it using the npm command,

**npm i react-router-dom@6**

nav a {

  color: #edefee;

  text-decoration: none;

  font-size: 20px;

  margin-right: 20px;

}

This is a CSS style for anchor links (**<a>** elements).

Here's what each property does:

* **color: #edefee;** sets the text color of the link to a light grayish-blue (#edefee).
* **text-decoration: none;** removes the default underline from the link.
* **font-size: 20px;** sets the font size of the link text to 20 pixels.
* **margin-right: 20px;** adds a 20-pixel margin to the right of the link.

an SPA comes with its own special implementation of anchor tags and links, which only give an illusion of loading different pages to the end user, when in fact they simply load different components into a single element of the real DOM into which the virtual DOM tree gets mounted and updated

Bundling is a process that takes all the imported files in an app and joins them into a single file, referred to as a **bundle** .

webpack is a module bundler.

This means that depending on other files, all of these files can create a **dependency graph**. The order in which all these files are loading is essential. That dependency graph can get so complex that it becomes almost impossible for a human to structure a complex project and bundle all those dependencies properly.This is the reason you need tools like webpack.

So, webpack builds a dependency graph and bundles modules into one or more files that a browser can consume.

While it is doing that, it also does the following:

* It converts modern JS code - which can only be understood by modern browsers - into older versions of JavaScript so that older browsers can understand your code. This process is known as transpiling. For example, you can transpile ES7 code to ES5 code using webpack.
* It optimizes your code to load as quickly as possible when a user visits your web pages.
* It can process your SCSS code into the regular CSS, which browsers can understand.
* It can build source maps of the bundle's building blocks
* It can produce various kinds of files based on rules and templates. This includes HTML files, among others.

Another significant characteristic of webpack is that it helps developers create modern web apps.

It helps you achieve this using two modes: **production** mode or **development** mode.

There are several ways to tackle this issue of a large bundle. One such approach is code-splitting, a practice where a module bundler like webpack splits the single bundle file into multiple bundles, which are then loaded on an as-needed basis. With the help of code-splitting, you can **lazy load** only the parts that the visitor to the app needs to have at any given time. This approach significantly reduces the download times and allows React-powered apps to get much better speeds.

There are other ways to tackle these problems.

An example of a viable alternative is SSR (Server-side rendering).With SSR, React components are rendered to HTML on the server, and the visitor downloads the finished HTML code. An alternative to SSR is client-side rendering, which downloads the index.html file and then lets React inject its own code into a dedicated HTML element (the **root** element in **create-react-app**).

Sometimes, you can combine client-side rendering and server-side rendering. This approach results in what’s referred to as **isomorphic apps**.