URL Routing and Routing with React

Hunter Jorgensen

Anatomy of a URL

URLs are easy to ignore, but tell developers a lot about data hierarchy and can provide different ways to pass information to the backend. Additionally, they allow us to organize our site in a somewhat readable and meaningful way.

Anatomy of a URL

This is an example of a URL

http://www.example.com:80/resource/specificResource?key1=value1&key2=value2#SomewhereInTheDocument

- **http**: this is the protocol the browser must use when making this request; we'll learn more about this in the next module
- **www.example.com**: this is the domain name and represents the web server being requested
- **:80**: is the port. You see this locally when developing (port 3000, 5000, etc.) It is omitted when using ports standard to the protocol

Anatomy of a URL

http://www.example.com:80/resource/specificResource?key1=value1&key2=value2#SomewhereInTheDocument

- **resource** and **specificResource**: in our first assignment, these were addresses to physical resources. Now they are abstracted, where resource is part of the path and specificResource is known as a path parameter (we'll see this more detail in the next slide)
- **?key1=value1&key2=value2**: these are query parameters and should always be optional. We'll see these a bit more in the next module, but are used very sparingly on the frontend. They take the form of key/value pairs.
- **#SomewhereInTheDocument**: this is an anchor to a specific part of the document. This will bring users to the first HTML tag with the matching ID and is not sent to the server

Example: Amazon.com, Redfin.com

Let’s look into some websites and see what their URLs look like

Setting Up React Routing

Go into the directory where you have your React app

```
npm install --save react-router-dom
```

Adding Routes to Our index.js

The react-router-dom library provides us with some components and logic to simplify creating meaningful URLs. We'll use BrowserRouter's Route, Router and Switch components.

Router and Route are special components that parse the url and return a specific component.

Switch is another component that essentially works like a switch statement, but in matching URL patterns. Note the implied default at the end.

Demo

Creating Our Views

Now we want to create some simple views to experience what it would be like to navigate around the app.

Our first page is a simple about me page that just has some text.

The home page is using Link and NavLink (instead of the anchor <a href=...> tag!) to link. NavLink has some nice additional functionality, but you can use them interchangeably for this project.

Demo

Accessing the URL Path Params

Typically, path and query parameters can be parsed by accessing the props (see the console.log in the demo), but this can provide a hard-to-read string.

react-router-dom provides us with a useParams function to access any path parameters in your URL (but NOT any query params)

Demo

Accessing Query Parameters

react-router no longer provides support for query parameters, so you'll need to parse the string yourself or use a 3rd party library.

```
npm install query-string -s
```