How NOT to deploy a React site to Github Pages

#react #github

Ali Sherief May 1, 2020 · *Updated on May 7, 2020* · 4 min read

I'm writing this because it seems like a fairly big problem which I think several people have encountered during deployment. They push the compiled-by-npm site to their github.io repository but then only the homepage proceeds to work, and other paths return a 404 status code. So let's see how this can be fixed.

Fixing your Routes

If your routes look something like this:

```
// Last route is your 404 page
<BrowserRouter>
  <Switch>
    <Route path="/" render={props => <Index {...props} />} />
    <Route
      path="/page1"
      render={props => <Page1 {...props} />}
    />
    <Route
      render={props => <NotFoundPage {...props} />}
    />
  </Switch>
</BrowserRouter>
```

There are two problems with this router. First, the path "/" will overshadow the subpaths, which means it will render instead of any child paths it has, because the "/" route does not have the `exact` prop. But if you put the `exact` prop on the "/" route, then subpaths of "/page1", including paths that don't exist, will still render "/page1" instead of the correct path.

The solution is to always pass the `exact` prop to all the routes you make, except for your 404 route, the reason for that will be explained below. This will cause only that one corresponding JSX page to render for that path, and no others.

The second problem is because this router is a BrowserRouter, and the npm build

4 1 1

paths except for root "/" are missing.

To fix this, you need to change your BrowserRouter to HashRouter. This puts, as the name implies, a hash "#/" in front of the root path, and any subpath can be put after the hash to load its page in the DOM, thereby all rendering is done in Javascript instead of navigating across HTML files. You then have to change all of your `href` properties to have "/#" before the pathname, like "/#/" and "/#/path1", but you shouldn't need to change anything for Link components.

So now your router should look something like this:

```
<HashRouter>
  <Switch>
    <Route exact path="/" render={props => <Index {...props} />} />
    <Route exact
      path="/page1"
      render={props => <Page1 {...props} />}
    />
    <Route
      render={props => <NotFoundPage {...props} />}
    />
  </Switch>
</HashRouter>
```

Notice how the NotFoundPage route is not `exact`. It shouldn't be, because it is the catch-all route for invalid paths.

This will load all the pages correctly but breaks all of your HTML `id` anchors, because HashRouter is already using the anchor for its routing. Another way to scroll to the HTML anchors is needed. It will also cause anchorless links pointing within your site you click on to *not* scroll to the top, instead it keeps the scrolling position or else it leaves the scrollabar at the very bottom in case the previous page was longer than the next page.

UPDATE: If you use HashRouter, then only the front page will be indexed by Google. The hashed pages are not indexed by Google at all (see this Google forum question for details). There is no workaround for this unfortunately, except for using BrowserRouter without Github Pages.

Fixing your anchors

Fortunately there is some Javascript you can put after your component is rendered to fix this scrolling mess

4 1 1 ...

componentDidMount() function. The JS I'm about to write should go after it, preferably at the end.

First of all to fix the issue where anchorless links don't scroll at the top, you just need to put `window.scrollTo(0,0);` at the end of your `componentDidMount()` function in all of your pages. Since scrollTo has the parameter list (x-coordinate, y-coordinate) this has the side effect of scrolling to the leftmost of the page as well, but assuming your React app doesn't a have horizontal scroll bar, the 0 in the x-coordinate should be OK.

Then you need to put this in your `componentDidMount()`, so that it scrolls to anchors when a page component is rendered, and in your `componentDidUpdate()`, in case you scroll to anchors after the page is already rendered. This solution was found here.

```
// put this after window.scrollTo if applicable
let hash = this.props.location.hash.replace('#', '');
if (hash) {
  let node = ReactDOM.findDOMNode(this.refs[hash]);
  if (node) {
    node.scrollIntoView();
  }
}
```

And replace the `id`s in all of your divs with `ref`, so that `<h3 id="someanchor">scrolls here</h3>` becomes `<h3 ref="someanchor">scrolls here</h3>`.

So what this snippet does is that it takes the part of the anchor after the hash, so it takes "someanchor" from "#someanchor", and finds the DOM node that has a ref prop of that name. ref names need not be unique across components, only within the same component. If a matching DOM node is found, then it scrolls it into view.

And we're done

With these changes your React site should work perfectly on Github Pages. If this didn't work for you, let me know in the comments and I'll see what I can do.

Discussion

Subscribe

Add to the discussion

Code of Conduct • Report abuse

4 1 1

Ali Sherief

Backend developer by trade, frontend developer in the works. He/Him.

Follow

LOCATION
Khartoum, Sudan

EDUCATION
Self-made

JOINED
Nov 30, 2019

More from Ali Sherief

Using CSS Flexbox in Reactstrap

#react #webdev

4 1 1 •••