React Error Handling

When working with a front end connected to an API / some APIs, sometimes things go wrong.

With a bit of luck and a well designed back end, these will come with an error code (`4xx` or `5xx`). First consider what would happen if an entity of yours - such as an article - failed to return. How would that impact your render? Will you be trying to access properties that don't exist / are undefined? Worse, access properties of ***those*** properties, which will lead to a Type Error?

Clearly we need to handle these catches on the front end, and provide sensible feedback to the user. `console.log` won't cut it - we can't expect a user to open the console and see what has happened.

Client Side vs Server Side Errors

Sometimes errors that can occur could be dealt with ***before*** a request is sent to a backend - ***client side***. Sometimes the only time you can know something is wrong is when the server responds with an error - ***server side***.

We ***definitely*** don't want to make any requests to the backend if we know the request will error in the first place. For example, you know if there's missing information on a post your backend should respond with `status 400`, so you could prevent that request being sent in the first place until all inputs are filled in.

Some errors to look out for:

Client Side	Server Side
Post information like `body` is missing	The server isn't working as it normally should
The user has gone to a url that has no page or information associated with it	An incorrect id is entered in the URL, so an article is requested that doesn't exist.
-	The request in the URL is badly formatted in some way e.g. bad id
-	A user isn't authorised to access a resource.

Default Routes

To deal with the user going to a url that has no information or pages associated with it, `react-router` `Route` components provide a `path` attribute that when set to `"*"` will render a component on any user url that isn't explicitly defined in other specified front-end routes.

```
<Route path="*" element={ErrorPage} />
```

Error Components

```
const Page = (props) => {
  const [error, setError] = useState(null);

  useEffect(() => {
    getData()
      .then((stuff) => {
        // do stuff
      })
      .catch((err) => {
        setError({ err });
      });
  }, []);

  if (error) {
    return <ErrorComponent message={error.something.keyForTheErrorMessage} />;
  }
  return <Stuff />;
};
```

In this situation, our external api request either works or doesn't. If it doesn't, we record this in `state`, and on the `re-render`, we spot that there is an error and render an error component instead of the regular stuff.

Reusable ErrorPages

Why waste all the effort we expended in making tailored errors on our APIs if we're just going to make a generic error component? Even worse, we don't want to have to make a separate `404ErrorPage` and `400ErrorPage` and conditionally render those...

Our API errors will come with messages, codes - information we can pass to our `ErrorComponent` component on `props` to render a helpful message to the user.