Debugging replaceRenderer API

Prerequisites

If you're not familiar with Gatsby's lifecycle, see the overview Gatsby Lifecycle APIs.

What is the `replaceRenderer` API?

The `replaceRenderer` API is one of Gatsby's Server Side Rendering (SSR) extension APIs. This API is called when you run `gatsby build` and is used to customize how Gatsby renders your static content. It can be implemented by any Gatsby plugin or your `gatsby-ssr.js` file - adding support for Redux, CSS-in-JS libraries or any code that needs to change Gatsby's default HTML output.

Why does it cause build errors?

If multiple plugins implement `replaceRenderer` in your project, only the last plugin implementing the API can be called - which will break your site builds.

Note that `replaceRenderer` is only used during the build lifecycle. It won't cause problems as you work on your site with the develop script.

If multiple plugins implement `replaceRenderer`, the build script will warn you:

```
warning replaceRenderer API found in these plugins:
warning plugin-name-a, default-site-plugin
warning This might be an error, see: https://www.gatsbyjs.com/docs/debugging-replace-rendere

warning Duplicate replaceRenderer found, skipping gatsby-ssr.js for plugin: plugin-name-a
```

Note that `default-site-plugin` refers to your local `gatsby-ssr.js` file, if this file exists it will always be used in favor of `gatsby-ssr.js` from other plugins.

Fixing `replaceRenderer` build errors

If you see errors during your build, you can fix them with the following steps.

1. Identify the plugins using `replaceRenderer`

Your error message should list which plugins implement `replaceRenderer`:

```
warning replaceRenderer API found in these plugins:
warning plugin-name-a, default-site-plugin
```

In this example, your `gatsby-ssr.js` file and `plugin-name-a` are both using `replaceRenderer`.

2. Copy the plugins' `replaceRenderer` functionality to your site's `gatsby-ssr.js` file

You'll need to override your plugins' `replaceRenderer` code in your `gatsby-ssr.js` file. This step will be different for each project, keep reading to see an example.

Example

Initial setup

In this example project you're using Redux and Gatsby's Styled Components plugin.

```
module.exports = {
  plugins: [`gatsby-plugin-styled-components`],
}
```

`gatsby-ssr.js` (based on the using Redux example)

```
import React from "react"
import { Provider } from "react-redux"
import { renderToString } from "react-dom/server"

import createStore from "./src/state/createStore"

exports.replaceRenderer = ({ bodyComponent, replaceBodyHTMLString }) => {
  const store = createStore()

  const ConnectedBody = () => <Provider store={store}>{bodyComponent}</Provider>
  replaceBodyHTMLString(renderToString(<ConnectedBody />))
}
```

Note that the Styled Components plugin uses `replaceRenderer`, and the code in `gatsby-ssr.js` also uses `replaceRenderer`.

Fixing the `replaceRenderer` error

Your `gatsby-config.js` file will remain unchanged. However, your `gatsby-ssr.js` file will update to include the `replaceRenderer` functionality from the Styled Components plugin

```
import React from "react"
import { Provider } from "react-redux"
import { renderToString } from "react-dom/server"
import { ServerStyleSheet, StyleSheetManager } from "styled-components"
import createStore from "./src/state/createStore"

exports.replaceRenderer = ({
  bodyComponent,
  replaceBodyHTMLString,
  setHeadComponents,
}) => {
  const sheet = new ServerStyleSheet()
  const store = createStore()

  const app = () => (
    <Provider store={store}>
      <StyleSheetManager sheet={sheet.instance}>
        {bodyComponent}
      </StyleSheetManager>
    </Provider>
  )
  replaceBodyHTMLString(renderToString(<app />))
  setHeadComponents([sheet.getStyleElement()])
}
```

Now `gatsby-ssr.js` implements the Styled Components and Redux functionality using one `replaceRenderer` instance. Run `npm run build` and the site will build correctly.

All the code from this example is available on GitHub.