Profiling Site Performance with React Profiler

React profiling captures timing information that can help identify performance issues within your gatsby site.

Requirements

React's profiling API was introduced in **React 16.5**. Therefore you must be running React 16.5 or higher.

Using the profiler

Profiling will be enabled automatically in Development. If you have React DevTools it will show a "Profiler" tab. Caution should be given to the validity of profiling within Development as this does not reflect the performance of the Production build and therefore the experience of your users.

To enable profiling for a Production build an additional CLI option `--profile` must be provided when running the build command. See gatsby build command for further information.

Performance impact

Although Profiler is a light-weight component, it should be used only when necessary; each use adds some CPU and memory overhead to an application.

A warning will be issued in a Gatsby Production build if the `--profile` option is provided.

Using the Profiler API

If you just want to see some high level results, your React DevTools will have a profiling tab which provides some helpful performance information. However if you want to be able to capture more information programmatically, then React exports a `profiler` component which allows your site to tap into the performance metrics.

The React profiler component can be used anywhere in a React tree to measure the performance of that part of the tree.

Profiling a slow component

```
import * as React from "react"
import { Profiler } from "react"

export const MyComponent = props => (
  // See https://reactjs.org/docs/profiler.html#onrender-callback for onRender parameters
  <Profiler id={props.someUniqueId} onRender={capturePageMetrics}>
    <SlowComponent />
  </Profiler>
)
```

Profiling Gatsby pages

Capturing page performance can be achieved by using the wrapPageElement API to profile each page.

```
//gatsby-browser.js

import * as React from "react"
import { Profiler } from "react"

export const wrapPageElement = ({ element, props }) => (
  // See https://reactjs.org/docs/profiler.html#onrender-callback for onRender parameters
  <Profiler id={props.someUniqueId} onRender={capturePageMetrics}>
    {element}
  </Profiler>
)
```

References:

- React Profiler
- React Profiler Introduction Blog