Debugging Incremental Builds

With the release of Gatsby v3 Incremental Builds is available to everyone. This improvement is (re)generating only subset of HTML files that need to be generated. To be able to use it you will need to keep the `.cache` and `public` directories from previous builds.

How does it work?

To be able to understand why you might see more generated pages than expected it's helpful to know how Gatsby tracks "inputs" when generating HTML files. In particular Gatsby tracks:

- which page template the page is using
- results of page query
- results of static queries used by a page template
- frontend source code (shared and also browser `gatsby-browser.js` / SSR `gatsby-ssr.js` specifically)

When those inputs change since the last build, the HTML files are marked to be regenerated. If they don't change, Gatsby can reuse HTML files generated in previous build.

Expected behaviors

There are certain changes that you can make to your project that trigger more changes than anticipated or even a full rebuild. For the following cases this is expected behavior:

- Changing a React component / page. This results in a changed webpack compilation where some assets generated by webpack will have changed hashes in filenames and thus Gatsby needs to regenerate pages to update `<link>` and `<script>` references.
- Changing content queried by a static query. This results in one or more `.html` files being regenerated. In case of static query used by all page templates it would regenerate all pages (for example static query used by shared layout component)

Debugging

To be able to see the changed pages inside console, run `gatsby build` with some additional flags:

```
gatsby build --verbose --log-pages
```

At the end of the build you will see the changed pages:

```
info Built pages:
Updated page: /foo/bar/
```

If you also want to see what and how files actually did change between builds you can look at the `diff` between two builds. Inside your project directory run these commands:

```
gatsby clean
gatsby build --no-uglify
cp -r public public-first-build
```

If you are debugging why some changes result in more `.html` files being regenerated than expected, this is the time to make your changes to the files. If you are debugging why builds always regenerate all `.html` files, skip making any changes.

Continue with these commands:

```
gatsby build --no-uglify --verbose --log-pages
diff -u -r public-first-build public > build-diff.diff
```

Open the `build-diff.diff` file in the root of your site to see the difference between both builds.

> **Note:** To exclude all `.html` files in the diff you can run this command at the end instead: `diff -u -r --exclude="*.html" public-first-build public > build-diff.diff`

Understanding the `diff`

Generating the `diff` between two builds is only the first step, understanding and interpreting the `diff` is the second step. You can use tools like diffchecker.com to see the differences more easily (or your code editor might have a feature/plugin for that).

Possible avenues to look into:

- **Check if the JS bundle changed**. If you see `chunk-map.json`, `webpack.stats.json`, and contents of `app-data.json` changed you'll see lines like `Only in public: component---src-path-to-file-[hash].js`. This tells you that the component itself changed between builds. Compare both `.js` files to see what exactly changed.

- **Check if anything in `/page-data` changed**. This means that a static query and/or page query changed and thus the page(s) rebuilt. A static query change is recognizable by a change of `page-data/sq/d/[hash].json`, a page query by a change of `page-data/[page-title]/page-data.json`. You can look at the files to figure out what exact query is the culprit.

Tips

Avoid direct filesystem calls

As we mentioned, Gatsby tracks "inputs" used to generate HTML files. However, the `gatsby-ssr` file allows some arbitrary code execution like using the `fs` module. For example:

```
const fs = require(`fs`)
const someUntrackedInput = fs.readFileSync(`some-path.txt`)

// Rest of gatsby-ssr.js file
```

While Gatsby could also track files that are read, the custom code that does those reads might have some special logic that Gatsby is not aware of. In the above example the filename could be generated and changed between builds, or the file read itself could change – so Gatsby sees this as "arbitrary" file reading. If Gatsby discovers that `fs` modules are used, it will disable "Incremental Builds"-mode to stay on the safe side (there will be warnings mentioning "unsafe builtin method").

If your `gatsby-ssr` (either site itself or plugin) make use of `fs` reads, head over to migrating from v2 to v3 guide and check how to migrate.

Avoid relying on current date and/or time

Using `Date` inside your `gatsby-node.js` or `gatsby-config.js` files to get and use current date (`new Date()`, `Date.now()`) could be the reason for a rebuild of all pages as the date changes between builds.

This is also true for querying the `buildTime`:

```
{
  site {
    buildTime
  }
}
```

Querying that information in e.g. a static query inside a layout component that is used across all pages will result in a rebuild of all pages.