> *This documentation isn't up to date with the latest version of Gatsby.*
>
> *Outdated areas are:*
>
> - *`data.json` should be replaced with `page-data.json`*
> - *remove mentions of `pages.json`*
> - *describe `match-paths.json`*
>
> *You can help by making a PR to update this documentation.*

This is one of the last bootstrap stages before we hand off to webpack to perform code optimization and code splitting. webpack builds a web bundle. It has no knowledge of Gatsby's core code. Instead, it operates only on files in the `.cache` directory. It also doesn't have access to all the Redux information that was built up during bootstrap. So instead, we create dynamic JavaScript and JSON files that are dependent on by the webpack application in the `.cache` directory (see Building the JavaScript App).

You can think of this step as taking all the data that was generated during bootstrap and saving it to disk for consumption by webpack.

```
digraph {

  subgraph cluster_redux {
    label = "redux";

    pages [ label = "pages", shape = cylinder ];
    components [ label = "components", shape = cylinder ];
    jsonDataPaths [ label = "jsonDataPaths", shape = cylinder ];
  }

  writePages [ label = "pages-writer.js:writePages()" ];

  subgraph cluster_cache {
    label = "site/.cache/";
    pagesJson [ label = "pages.json", shape = cylinder ];
    syncRequires [ label = "sync-requires.js", shape = cylinder ];
    asyncRequires [ label = "async-requires.js", shape = cylinder ];
    dataJson [ label = "data.json", shape = cylinder ];
  }

  pages -> writePages;
  components -> writePages;
  jsonDataPaths -> writePages;

  writePages -> pagesJson;
  writePages -> syncRequires;
  writePages -> asyncRequires;
  writePages -> dataJson;
}
```

Most of the code backing this section is in pages-writer.js

The dynamic files that are created are (all under the `.cache` directory).

- pages.json
- sync-requires.js
- async-requires.js
- data.json

pages.json

This is a collection of page objects, created from Redux `pages` namespace. For each page it includes the

- componentChunkName
- jsonName
- path
- matchPath

The pages are sorted such that those with `matchPath` s come before those without. This is to assist find-page.js in selecting pages via regex before trying explicit paths. See matchPaths for more info.

e.g

```
;[
  {
    componentChunkName: "component---src-blog-2-js",
    jsonName: "blog-c06",
    path: "/blog",
  },
  // more pages
]
```

`pages.json` is generated for `gatsby develop` purposes only. In `npm run build`, we use data.json (below) which includes the pages info plus more.

sync-requires.js

This is a dynamically generated JavaScript file that exports `components`. It is an object created by iterating over the `components` Redux namespace. The keys are the componentChunkName (e.g. `component---src-blog-2-js`), and the values are expressions that require the component. E.g. `/home/site/src/blog/2.js`. The file will look something like this:

```
exports.components = {
  "component---src--blog-2-js": require("/home/site/src/blog/2.js"),
  // more components
}
```

It is used during static-entry.js so that it can map componentChunkNames to their component implementations. Whereas the production-app.js must use `async-requires.js` (below) since it performs code splitting.

async-requires.js

`async-requires.js` is very similar to `sync-requires.js`, in that it is a dynamically generated JavaScript file. The difference is that it is written to be used for code splitting via webpack. So, instead of using `require` with the

component's path, it uses `import` and adds a `webpackChunkName` hint so that we can eventually link the componentChunkName to its resulting file (more info in Code Splitting docs). `components` is a function, so that it can be lazily initialized.

`async-requires.js` also exports a `data` function that imports `data.json` (see below)

An example of async-requires is:

```
exports.components = {
  "component---src-blog-2-js": () =>
    import(
      "/home/site/src/blog/2.js" /* webpackChunkName: "component---src-blog-2-js" */
    ),
  // more components
}

exports.data = () => import("/home/site/.cache/data.json")
```

Remember, `sync-requires.js` is used during Page HTML Generation. And `async-requires.js` is used by Building the JavaScript App.

data.json

This is a generated JSON file. It contains the entire `pages.json` contents (as above), and the entire Redux `jsonDataPaths` which was created at the end of the Query Execution stage. So, it looks like:

```
{
  pages: [
    {
        "componentChunkName": "component---src-blog-2-js",
        "jsonName": "blog-2-c06",
        "path": "/blog/2"
    },
    // more pages
 ],

 // jsonName -> dataPath
 dataPaths: {
   "blog-2-c06":"952/path---blog-2-c06-meTS6Okzenz0aDEeI6epU4DPJuE",
   // more pages
 }
```

`data.json` is used in two places. First, it's lazily imported by `async-requires.js` (above), which in turn is used by `production-app` to load JSON results for a page.

It is also used by Page HTML Generation in two ways:

1. `static-entry.js` produces a `page-renderer.js` webpack bundle that generates the HTML for a path. It requires `data.json` and uses the `pages` to look up the page for the page.
2. To get the `jsonName` from the page object, and uses it to construct a resource path for the actual JSON result by looking it up in `data.json.dataPaths[jsonName]` .

Now that Gatsby has written out page data, it can start on the webpack section.