As you encounter errors while developing with Gatsby, it is likely you'll run into something that other users have already stumbled upon. Some errors may require fixes to Gatsby's core processes, these are best filed as issues. Many errors you encounter will mean adjusting how you've configured a plugin or API in your site.

This guide is meant as a reference for common errors that have tripped up other Gatsby users.

Problems with the cache

Running a site in `gatsby develop` will set up a server locally that enables features like hot-module replacement. Gatsby keeps a cache of data and rendered assets in the `.cache` folder at the root of a Gatsby site so that it doesn't have to repeat work processing optimized resources. If you see errors about not being able to find a resource in the cache it may be enough to clear your cache and restart your server. You can clear Gatsby's cache by running:

```
gatsby clean
```

This will delete the `.cache` folder, as well as the `public` folder for you. Running `gatsby develop` will recreate the cache and process all resources again.

You may also want to refer to the dedicated guide on Debugging Cache Issues.

> *For additional discussion on caching and persistence issues, refer to the umbrella issue.*

Errors with common plugin configurations

Plugins extend Gatsby's functionality, because they introduce new behavior it's possible that an installed plugin introduces errors.

Installing plugins for styling results in Generating SSR bundle failed

If you encounter a webpack error that says `Generating SSR bundle failed` after installing a plugin and trying to run `gatsby develop` or `gatsby build`, it's possible you haven't yet installed all the packages you need.

For some plugins like emotion, styled-components, or Sass, it won't be enough to only install the plugin, you also need to install libraries they rely on. The official installation instructions should guide you to install all needed libraries when you install the plugin, but some tutorials or blog posts you find at other sources may not.

Here are some examples of plugins that require you to install more than just the plugin:

- gatsby-plugin-emotion: `@emotion/react`, and `@emotion/styled`
- gatsby-plugin-styled-components: `styled-components`, and `babel-plugin-styled-components`
- gatsby-plugin-sass: `node-sass`, or `sass`
- gatsby-plugin-material-ui: `@material-ui/styles`
- gatsby-plugin-image: `gatsby-source-filesystem`, `gatsby-transformer-sharp`, and `gatsby-plugin-sharp`

Rather than packaging up the other dependent libraries alongside these plugins, they can stay smaller in size when they are published and are able to rely on alternative implementations. One example is `gatsby-plugin-sass` that can use either the Node.js or Dart implementations of Sass.

To resolve these errors, identify the packages that haven't been installed, the error message might look like this:

```
...
success run queries - 0.095s - 8/8 84.63/s

 ERROR #98123  WEBPACK

Generating SSR bundle failed

Can't resolve '@emotion/react' in '/Users/you/tmp/gatsby-site/.cache' // highlight-
line

File: .cache/develop-static-entry.js
```

This error is a result of Gatsby having failed to find `@emotion/react` because `gatsby-plugin-emotion` has been installed and added to the `gatsby-config`, without installing the emotion library. Install it like this:

```
npm install @emotion/react
```

Or replace `@emotion/react` with the name of the library that is missing. Installing the plugin and any necessary libraries as well as adding the plugin to your `gatsby-config` should resolve this error.

Issues with `fs` resolution

You may see this error because you're attempting to use `fs` inside a React component. Additionally, it often shows up when working with `@mdx-js/runtime`.

This error may be a top level `Cannot resolve module 'fs'` or part of a webpack error like `Can't resolve 'fs'`.

`fs` stands for filesystem and it's a Node.js library that's used to access files on your computer. However, when your packaged Gatsby code runs, your computer is but a distant memory.

Some packages, like Babel, bring `fs` along for the ride anyway. In order to prevent it from causing errors, you can add the following to your `gatsby-node.js` file.

```
exports.onCreateWebpackConfig = ({ actions }) => {
  actions.setWebpackConfig({
    resolve: {
      fallback: {
        fs: false // highlight-line
      }
    }
  })
}
```

Errors in styling

The following errors are related to styles in your site, using CSS, preprocessors, or CSS-in-JS solutions.

Inconsistent CSS styles between develop and build using styled-components or emotion

NOTE: We're in process of adding SSR support to the develop server. To use it now, enable the `DEV_SSR` flag in your gatsby-config.js — https://github.com/gatsbyjs/gatsby/discussions/28138

A common problem that trips up users that install and begin to use styled-components or emotion is not including the related plugin in the config. Because `gatsby develop` doesn't run server-side rendering, the build may look different if the plugin is not included to tell Gatsby to server-side render the styles for the CSS-in-JS solution being used.

Adding `gatsby-plugin-styled-components` (in the case of styled-components) or `gatsby-plugin-emotion` (in the case of emotion) to `gatsby-config.js` will inform Gatsby to process the styles server-side so they display correctly in the final build.

Errors with GraphQL

Gatsby's GraphQL data layer provides access to build time data, there are sometimes errors you may encounter while implementing plugins that are sourcing data or adding nodes to the schema yourself.

Unknown field 'A' on type 'B'

If the data you are requesting in a GraphQL query differs from what has been sourced in the GraphQL schema you might encounter an error like `Unknown field 'A' on type 'B'`. As the error suggests, a field you are asking for is not defined under the type that is listed. If your site is still building okay, you can open up `http://localhost:8000/___graphql` and examine your schema, which includes the definition of what fields are included on the type provided by the error. This can help you identify what fields aren't being created and locate where those fields should be created, whether by a plugin or in your code.

If the error is describing an `Unknown field 'X' on type 'Query'`, the content type you are trying to source is likely not processing correctly. The `Query` type represents the top-level root queries that are included in the GraphQL schema. Source plugins will often create root nodes that you can query like `mdx` (created by `gatsby-plugin-mdx`) or for a collection of root nodes like `allFile` (created by `gatsby-source-filesystem`).

Some ideas for debugging these errors include verifying the following:

- if you are using a transformer plugin (like `gatsby-transformer-yaml`), the data you need is pulled in using a source plugin (like `gatsby-source-filesystem`)

```
{
  plugins: [
    `gatsby-transformer-yaml`,
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        path: `./src/data/`, // location of yaml files
      },
    },
  ]
}
```

- the structure of your content being sourced matches your GraphQL schema as well as the way you are querying the data

Comparing your GraphQL query to your site's schema in `http://localhost:8000/___graphql` and whatever plugin or code you are using to source data is a great way to find these errors as they should all express the data in the same shape.

- neither any source plugins you are using nor your own implementation of the [`sourceNodes` API] are misconfigured

Errors using gatsby-plugin-image and sharp

Gatsby's image processing is broken up into different packages which need to work together to source images and transform them into different optimized versions. You might run into these errors getting them to play together nicely.

Field "image" must not have a selection since type "String" has no subfields

This error message `Field "image" must not have a selection since type "String" has no subfields.` comes up when a GraphQL query is trying to query a field for subfields, but none exist. This generally happens when plugins that are used together are added in the `gatsby-config` in the wrong order, or haven't been added at all.

The query is trying to access fields that don't exist because they weren't set up at build time. In the following code, a query is looking to find the subfield `childImageSharp` of the `image` field, like the error states. The problematic GraphQL schema looks like this:

```
allMdx {
  nodes {
    id
    title
    image
  }
}
```

And the expected GraphQL schema looks this:

```
allMdx {
  nodes {
    id
    title
    image {
      childImageSharp {
        gatsbyImageData
      }
    }
  }
}
```

In the first code example, the `image` field was not transformed (*modified*) by a plugin to add subfields, so it would only return a string. `gatsby-plugin-sharp` and `gatsby-transformer-sharp` can be included before other plugins that would manipulate or create image nodes (like `gatsby-source-filesystem` or `gatsby-source-contentful`) to ensure that they are present before Gatsby tries to modify them and add the needed fields like `childImageSharp` .

You can read more about how images are added to the GraphQL schema in the guide on processing external images.

Another possibility that could cause this issue is from empty strings used for image paths somewhere in your site. If this is the case, when Gatsby constructs a GraphQL schema it may infer the wrong type because the empty string doesn't look like a file path.

Problems installing `sharp` with `gatsby-plugin-sharp` - gyp ERR! build error

If you see an error message in the console when installing dependencies that look related to sharp like `gyp ERR! build error` and `npm ERR! Failed at the sharp@x.x.x install script`, they can often be resolved by deleting the `node_modules` folder in the root of your project and installing dependencies again:

```
# be careful as this command will delete all files recursively in
# the folder you provide, in this case, node_modules
rm -rf node_modules

# this command will install libraries from your package.json file
# and place them in the node_modules folder
npm install
```

The version of Node.js that's used to install sharp needs to match the version of Node.js that is run, so clearing `node_modules` and reinstalling often resolves the problem.

> *For more discussion around problems with sharp installation and image processing, refer to this issue.*

Incompatible library version: sharp.node requires version X or later, but Y provides version Z

The error `Incompatible library version: sharp.node requires version X or later, but Y provides version Z` means that there are multiple incompatible versions of the `sharp` package installed in `node_modules`. Your error may look something like this:

```
Something went wrong installing the "sharp" module
dlopen(/Users/you/gatsby-site/node_modules/sharp/build/Release/sharp.node, 1):
Library not loaded: @rpath/libglib-2.0.dylib
  Referenced from: /Users/you/gatsby-
site/node_modules/sharp/build/Release/sharp.node
  Reason: Incompatible library version: sharp.node requires version 6001.0.0 or
later, but libglib-2.0.dylib provides version 5801.0.0
```

To fix this, you'll need to update all Gatsby plugins in the current project that depend on the `sharp` package. Here's a list of official plugins that you might need to update in case your project uses them:

- `gatsby-plugin-sharp`
- `gatsby-plugin-manifest`
- `gatsby-remark-images-contentful`
- `gatsby-source-contentful`
- `gatsby-transformer-sharp`
- `gatsby-transformer-sqip`

To update these packages, run:

```
npm install gatsby-plugin-sharp gatsby-plugin-manifest gatsby-remark-images-
contentful gatsby-source-contentful gatsby-transformer-sharp gatsby-transformer-sqip
```

If updating these doesn't fix the issue, your project probably uses other plugins from the community that depend on a different version of `sharp`. Try running `npm list sharp` or `yarn why sharp` to see all packages in the current project that use `sharp` and try updating them as well.

Errors building and deploying

The process of building your site varies slightly from the development process. Some errors can arise when you build your site if you include references to the browser, though almost all problems should be caught by error messages in `develop` mode.

For more information on common problems while building your site, refer to the Debugging HTML Builds guide.

Error: ReferenceError: window is not defined when running `gatsby build`

You may encounter an error like `Error: ReferenceError: window is not defined` that you didn't see in development if you reference browser globals like `window` or `document` in your code. Because the build is not running in a browser, it will not have access to a browser, which is why objects like `window` will not be defined.

Exact steps for fixing this issue can be found on in the Debugging HTML Builds guide in the section on checking if `window` is defined.

Build problems from Field 'browser' doesn't contain a valid alias configuration

If you are seeing an error like:

```
Module not found: Error: Can't resolve `../..SomeFile.svg`
  Field 'browser' doesn't contain a valid alias configuration
  ...
```

The build is failing to find the file at `../..SomeFile.svg`. This can be frustrating if your site works when you when run it locally with `gatsby develop`, and even works when you run `gatsby build` and `gatsby serve` locally. A likely problem is that the operating system you are running locally is different than the one where your site is deployed. Oftentimes your deployment target is running some distribution and flavor of Linux.

The most common culprit to prompt this issue is with filepaths having mixed capitalization. In the example above, check to make sure that the file is actually named `SomeFile.svg` and not something different like `Somefile.svg` or `somefile.svg`. Some operating systems will pick up on this discrepancy for you and find the image without any problems. Your deployment environment may not.

Checking the capitalization of files output in your build logs and redeploying is the best next step.

Error: ENOSPC: System limit for number of file watchers reached

You may have encountered a system limit on the number of files you can monitor.

To fix it, increase your system's file watchers limit (which is the number of processes that check for changes to files in your site while it's running) with the following command:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo
sysctl -p
```

You may be able to find more information for your circumstances in the GitHub issue corresponding to this error.