The Gatsby community thrives on the power of plugins. You often find that there is a plugin for almost everything ranging from source plugins to transformer plugins. A useful way of understanding the purpose of a plugin at a glance is by its name. Check out the guide to naming conventions for plugins to learn more.

> *Wondering how to create a plugin? Look no further. Start contributing.*

What is a plugin?

Gatsby plugins are Node.js packages that implement Gatsby APIs. To learn more take a look at the plugin documentation.

In creating plugins, the bulk of the work is during the development phase. However, there's still a need to take a look at the dependencies and security features from time to time. Keeping plugins up to date can become really tasking and it is important to be careful when updating dependencies, seeing as this could potentially break users' apps.

However, the consequences of not updating and maintaining a plugin can introduce security issues to your users. The following are some recommendations for maintaining plugins:

Handling patches and improvements

Most plugins generally follow Semantic Versioning to determine how many releases to do and the type of releases.

The first public release of a plugin is 0.1.0 or 1.0.0. From there, bugs are fixed in patch releases (e.g. 0.1.1) and features are added or changed in minor releases (e.g. 0.2.0).

Version 1.0.0 should be released when the plugin's API is considered stable. Version 2.0.0 (and subsequent major releases) would introduce breaking API changes.

> *Consider a version format of X.Y.Z (Major.Minor.Patch). Bug fixes not affecting the API increment the patch version, backwards compatible API additions/changes increment the minor version, and backwards incompatible API changes increment the major version.*

Read more about this concept in this paper on semantic versioning.

Update README and document use cases

Every major version update should also be reflected in the README of the plugin as well as the use case examples.

It would be great for users to be able to reference several versions of the plugin with the updated examples to see if they want to keep the current version or upgrade and also to understand what the new version offers. Although this is good:

- Try to not clog your release repository with older versions of the plugin as you update, as they're not often needed. Instead, keep the last few in place.
- Try not to push every iterative change from Git live

Subscribe to updates from the Gatsby team

Please subscribe to the For Community Plugin Authors and Maintainers discussion on GitHub. This is where the Gatsby Core team will be communicating information regarding upcoming / in-process changes to Gatsby that may require updates to the plugins you maintain.

Tools for dependency version maintenance

There are a couple of useful tools that can help with keeping dependencies up to date.

- Version Lens enables you to update your dependencies from your `package.json` by allowing you see the available version number the package can be updated to, right above the current version each package in dependencies or devDependencies uses.
- The npm check updates command line tool helps to check for outdated dependencies. You can also use this tool to update those dependencies by following these steps:

1. Install the tool

```
npm install -g npm-check-updates
```

2. Run the command to update dependencies

```
ncu -u
```

Community supporting content and feedback

Having example repos and support forums on Discord, Twitter or Reddit are great ways to offer community support for your plugin. This would offer a pool of resources for users to get more information on implementations and contribute.