# Upgrade for Minor or Patch Releases

To keep up with the latest bug fixes, security patches, and minor releases from both Gatsby and its dependencies, you should upgrade often to the latest version of each one.

## Semantic versioning

Like many other packages, Gatsby uses semantic versioning to tag new versions and indicate what kind of changes are introduced in every new release.

This guide is meant to teach you how to upgrade Gatsby for minor or patch releases. For major changes you can refer to the Release and Migrations reference guide overview for the corresponding guide to upgrade.

## Why you should upgrade Gatsby and its dependencies

Every new version of every package comes with improvements on multiple categories from performance, accessibility, security, bug fixes, and more, so it is important to upgrade both Gatsby and its dependencies to get the latest improvements in every one of these categories.

Upgrading your dependencies often on minor or patch releases also helps you to make major upgrades easier and to identify soon-to-be-deprecated functionality or APIs.

## How to identify possible upgrades

To start, you can run the `outdated` command to identify new releases for all your dependencies.

```
npm outdated
```

This will output a table indicating which packages have new versions available and what is the latest version for each one.

```
Package                         Current   Wanted   Latest  Location
gatsby                          2.15.13  2.15.13  2.15.20
```

## Configure your dependencies for upgrades

Depending on whether you want to update Gatsby and its dependencies for minor or patch releases you need to modify your `package.json` accordingly.

If you only want to update **for patch releases**, you can add a tilde (`~`) before the version of your package:

```
"dependencies"{
  "gatsby": "~2.15.13",
}
```

**For both patch and minor updates**, add a caret (`^`) before the version of your package:

```
"dependencies"{
  "gatsby": "^2.15.13",
}
```

For major updates follow up with the corresponding guide from the Release and Migrations reference guide overview.

If you are updating Gatsby, you'll likely also need to update Gatsby related plugins, you can identify them by their names starting with `gatsby-`. This only applies to plugins managed in the gatsbyjs/gatsby repo; for community plugins check beforehand if there is a new version available for upgrading.

## Updating all your dependencies at once

After adding the corresponding annotations into your `package.json` file, you can run the update command:

```
npm update
```

This will upgrade all your packages to the latest wanted version, such as the latest patch, minor, or major update depending on your annotations in `package.json`.

## Upgrade individual dependencies

You can also update one package at the time with the `install` command in npm, alongside the version that you want to install:

```
npm install <package-name>@<version>
```

You can specify the version you want to install or upgrade to, in the following formats:

- A specific version after the `@`
- An annotated version with `*`,`^`,`~` to indicate that you want the latest major, minor or patch release respectively.
- Use an `x` instead of a number to indicate that you want the latest major (`x`), minor (`<major>.x`) or patch release (`<major>.<minor>.x`). For example,

to install the latest patch release for a given major and minor version: `npm install package-name@2.1.x`

For major upgrades, remember to follow up with the corresponding guide from the Release and Migrations reference guide overview.

## Upgrade Interactively

You can manually select which dependencies you want to update through the npm-check module. To do that, start by installing the module:

```
npm install npm-check --save-dev
```

Then add the corresponding script to your package.json file:

```
{
  "scripts": {
    "upgrade-interactive": "npm-check --update"
  }
}
```

And finally, run the recently added command:

```
npm run upgrade-interactive
```

## Troubleshooting

Aside from some specific cases, such as Gatsby's dropping of support for Node 6, upgrading for minor or patch releases should not require you to make changes to your code. It is recommended to run your suite of tests (in case you have one) after upgrading Gatsby or its dependencies.

In case you get stuck in dependencies conflicts, you can use the npm-force-resolutions package on npm.

## Related content

Check out these related guides for major upgrades of Gatsby:

- Migrating from v2 to v3
- Migrating from v1 to v2
- Migrating from v0 to v1