Gatsby has lots of amazing features that make it easier to add new functionality to your websites. But, like any technology, that means there is some jargon that can be confusing to newcomers. This guide breaks down the Gatsby-isms to make it easier to learn what everything means.

Starters

Starters are optional precompiled Gatsby sites that are maintained by Gatsby's core team and growing open source community; they can be used for blogging, e-commerce, design, and documentation to allow users to create blazing fast sites. Gatsby offers over a hundred opinionated starters from a variety of categories such as Blog, SEO, Portfolio, WordPress, and Markdown. You can browse the selection in Gatsby's Starter Library.

A starter is a fully functional Gatsby site that can run on its own but is designed to be used by you as a jumping-off point for creating your own site. After selecting a starter, you can create your own site locally based on that project using the `gatsby new` CLI command, which handles cloning the project from Git for you. A cloned starter does not maintain a connection to the original source code, so they are helpful as starting points to customize heavily and create an entirely new Gatsby site.

You can find more information on how to get started, how to modify existing starters, and how to create your very own in the starter documentation.

Plugins

Gatsby plugins are Node.js packages that implement Gatsby APIs to add extra functionality to a Gatsby site. Plugins are commonly installed through a registry like npm and configured in a `gatsby-config.js` file. There are many types of plugins, including data sourcing, SEO, responsive images, offline support, support for Sass, TypeScript, sitemaps, and RSS, Google Analytics, and more. You can also make your own plugins and either distribute them for fellow Gatsby developers to use or install them locally.

Plugins are different from starters in that plugins package up functionality for Gatsby's data layer and other structural site code into installable parts. Additionally, plugins remain tied to the original source code and can be updated accordingly. Starters, in contrast, break their connection to the original source code, making them a one time snapshot, and include all source code for a Gatsby site including CSS styles and JSX markup.

To learn more about Gatsby's plugin system, check out the plugins section of the docs.

Themes

Gatsby themes are a type of plugin with their own `gatsby-config.js` file, effectively making them into Gatsby sites you can install from a package manager like npm. Themes provide a way to package up and distribute functionality for both Gatsby's data layer and UI components. Since themes are a type of plugin, your Gatsby site can make use of multiple themes at the same time. They have the added benefit of maintaining their tie to the original source code, allowing you to update and receive changes to them. This is how themes differ from starters, which do not maintain a connection to their source. Themes also come with a powerful API called Shadowing, which allows you to structure themes in a customizable hierarchy.

Themes for various use cases are added to the Gatsby ecosystem daily, such as documentation themes, UI library implementations, and more. It's also possible to create your own theme and either distribute it for people in the community to use or keep it for internal use.

To learn more about Gatsby themes including themes available in the plugin library, check out the themes section of the docs.

Routing

Gatsby sites are full React web applications, meaning they have access to client-side APIs including routing (a fancy word for matching a URL with a part of a web page using JavaScript). Gatsby creates paths to access content, handling dynamic routing for you and providing performance benefits through content prefetching. A lot of the work is done at build time for a site using mostly static content. To handle applications that include functionality that cannot be handled at build time or through rehydration, including authentication or retrieving dynamic content, you can use client-only routes using @reach/router which is built into Gatsby.

Gatsby provides a few different ways to create HTML pages that hook into its routing system. Here are some examples of how to create pages in Gatsby:

- In your site's gatsby-node.js file by implementing the API createPages
- Gatsby core automatically turns React components in src/pages into pages
- Plugins can also implement `createPages` for you, which will create additional pages

When Gatsby sites get built static HTML pages are compiled. When a user initially navigates to a page on the site, the HTML page is fetched from a server and then rehydrated into a full React web application when browser JavaScript takes over. This is where client-side routing comes into the picture with @reach/router, as that's how Gatsby links pages together.

Routes in Gatsby are generated for you under the hood by creating pages. Changing a route would depend on how that page was created: either changing a filename in `src/pages`, updating a slug in a post's frontmatter metadata for a dynamic blog route, or changing the path you chose when generating a page with the `createPages` API.

Here is where you can find more information on Routing in Gatsby, including details on how to customize and authenticate routes.

Redux

Redux is a state container and is used inside Gatsby to keep its internal state in one place. Redux provides a predictable flow on how and when state updates are done. Gatsby can easily hook into these state transitions to update multiple actions to keep everything consistent.

When you implement a Gatsby Node API, you are passed a collection of actions. When using the supplied actions, you are manipulating state that is created, and relied upon, by Gatsby itself such as created pages and webpack config.

Once the build is complete, the work of Redux is done and it is not part of the bundle delivered to the browser unless required independently by the application itself.

If using a data store such as Redux in your Gatsby application, check out Adding a Redux Store.

GraphQL queries

GraphQL is a query language (the QL part of its name) that Gatsby uses to generate a data layer available to your site's components. If you're familiar with SQL, it works in a very similar way at build time.

Using a special syntax, you describe the data you want in your component and then that data is given to you, such as site metadata from your `gatsby-config.js`, connected WordPress posts, Markdown files, images, and more. Gatsby uses GraphQL to enable components to declare the data they need and apply it to render on a page. Using GraphQL in Gatsby provides many benefits, such as the ability to return data from multiple sources in one query, and transform that data at the same time (such as using Gatsby Image).

Here is how you get started using GraphQL in Gatsby: Tutorial - Part 4

webpack

webpack is an open-source JavaScript module bundler that Gatsby uses under the hood to package up your site content and assets into a static bundle. You don't have to do anything to webpack directly for it to work on your Gatsby site, but you do have the option to customize it if necessary to provide a special configuration.

When Gatsby creates its default webpack config, a function is called allowing you to modify the config using a package called webpack-merge. Gatsby does multiple webpack builds with a somewhat different configuration; each build type is referred to as a "stage".

You can learn more about webpack including how to add a custom webpack Config in Gatsby by visiting the docs.

Gatsby Cloud

Gatsby Cloud is a new product offering from the same team that brought you the Gatsby open-source framework. Gatsby Cloud makes it easier than ever to build and maintain a professional, large-scale website by delivering a growing suite of tools designed to optimize and streamline website building.

Features include real-time Preview, simplifying content collaboration, and Automatic Provisioning that empowers new and non-technical users to get started easily with a CMS, sample content, and a connected Gatsby starter.

To learn more about Gatsby Cloud and its evolving feature set, visit https://www.gatsbyjs.com/cloud.