> *This documentation isn't up to date with the latest version of Gatsby.*
>
> *Outdated areas are:*
>
> - *mention the page shell*
>
> *You can help by making a PR to update this documentation.*

This document aims to outline how resources are fetched and cached by various parts of Gatsby.

Offline Plugin (gatsby-plugin-offline)

Service worker (`sw.js`)

In the offline plugin, we specify which types of files should be automatically cached at runtime (i.e. resource files) so that whenever these are fetched from the network, they are saved in the cache.

We also keep an array stored in the IndexedDB which keeps a list of page paths (e.g. `/about/`) whose resources have all been cached. For these paths, we serve the offline shell rather than the full HTML, because we know that their resources are readily available (even if offline). For paths which aren't in this array, we load the HTML from the server as usual, which allows for native offline error pages to be displayed if the network is offline and allows static HTML to be rendered if the JS resources fail for some reason (e.g. due to an adblocker false-positive).

It's important that this array is accurate - if a page entered the array when its resources hadn't actually been cached, then the page could appear blank when loaded if the site updated in the meantime. A problem with the current approach is that devices may automatically remove cached files if they're running out of space, while the IDB array would still report these pages as being cached - in the future we aim to handle this behavior.

Browser APIs (`gatsby-browser.js`)

When all resources for a page have been successfully prefetched, we do *one* of the following:

- Add the page's path to a temporary array of prefetched paths, if the service worker has not yet installed
- Send a message to the service worker to let it know to allow the page's path, if it is installed

Upon initial install, we do the following:

- Loop through all scripts and stylesheets in the head and fetch these again, so that Workbox can now cache them
- Fetch all resources for pages whose paths are in the temporary prefetched paths array, so that Workbox can now cache them

Note that in both of the above cases, all these files should have already been downloaded once by the browser, so with proper HTTP caching setup we don't have to download any of the files again. However, one exception to this is `<style>` elements with a `data-href` attribute (indicating that the embedded stylesheet is the same as the stylesheet at the location specified) - we currently fetch the specified file rather than caching the contents of the element.

Another current problem is that we may start fetching the resources for a page before the service worker has finished installing, but finish fetching them all after it has installed - this could cause a page's path to be allowed even if some of its resources haven't been cached (since Gatsby assumes the service worker was installed at the start of fetching resources, if it was installed at the end).

Gatsby Core

Resource loader (`loader.js`)

There are two functions which perform a similar but distinct role in this file: `enqueue` and `loadPage`. The former of these, `enqueue`, is designed to speed up navigation by prefetching resources for a page, before we need to display the page, and hence it doesn't return anything. On the other hand, `loadPage` is used when we need the resources right now, usually in order to display the page, and therefore it fetches with higher priority than `enqueue` as well as returning them. Another difference between the two is that `loadPage` returns the resources for the 404 page if the specified page doesn't exist.

In the future, we could refactor these into a single function which takes parameters for whether or not to return the 404 page if the specified page is missing, and for whether to fetch with high or low priority.

EnsureResources (`ensure-resources.js`)

The `EnsureResources` component exists to (as its name suggest) ensure that we have the resources for a specified page, and load them if we currently don't. There are some valid reasons why we might not have resources at the time of navigation:

1. 404s resulting from site-internal links, when the user has not created a custom 404 page with `404.js`
2. When visiting a page (which hasn't been visited before), the site may have been updated since our last navigation and so the resources to the unvisited page may have changed location
3. Ad blockers may block access to certain paths in which case we're unable to load the JS resources for a page

Here is how the `EnsureResources` component handles each of these scenarios:

1. On non-initial renders, reload upon missing resources so that the browser can load the server's 404 page

2. On initial renders, flag the page as failed and throw an error to prevent rendering a blank page (static HTML will suffice), then reload it once the service worker updates

   On non-initial renders, reload upon missing resources so that the browser can load the latest page

3. On initial renders, throw an error to prevent rendering a blank page (static HTML will suffice)

The following are some invalid reasons why we might not have resources, i.e. things which we should never have to worry about:

1. 404s from external links, without a custom 404 page - the page will load from the server in the first place, so Gatsby won't even kick in at this point
2. 404s, with a custom 404 page - `loadPage` will automatically return the resources for the custom 404 page
3. Visiting a previously-visited page via an external link, when the site's resources have since updated - previously-visited pages are cached, so they'll work even if the site has updated since. Unvisited pages will always load from the server and get the latest resources.

Service worker update handling

The service worker updates automatically when the browser detects that the contents of the `sw.js` file have changed from the currently installed version. Upon an update, we clear all allowed paths to prevent old pages from loading after the update.

Blank pages can theoretically occur if we serve the app shell when resources are unavailable - however, this *should* never occur since we only serve the app shell with allowed paths (i.e. ones whose resources have been cached entirely). There may be some edge cases when this can occur, e.g. when the webpack runtime from the old site attempts to load a chunk which is unavailable on the updated site - we are currently investigating ways to prevent this, and make using service workers with Gatsby even more robust.

We should also never get incorrect 404 pages following a site update, since we never allow 404 pages to serve using the offline shell, meaning that a page which was previously a 404 should always load from the server. If it's no longer a 404, then it will be cached as usual.