Caching Static Sites

An important part of creating a very fast website is setting up proper HTTP caching. HTTP caching allows browsers to cache resources from a website so that when the user returns to a site, very few parts of the website have to be downloaded.

Different types of resources are cached differently. Let's examine how the different types of files built to `public/` should be cached.

HTML

HTML files should never be cached by the browser. When you rebuild your Gatsby site, you often update the contents of HTML files. Because of this, browsers should be instructed to check on every request if they need to download a newer version of the HTML file.

The `cache-control` header should be `cache-control: public, max-age=0, must-revalidate`[1]

Page data

Similar to HTML files, the JSON files in the `public/page-data/` directory should never be cached by the browser. In fact, it's possible for these files to be updated even without doing a redeploy of your site. Because of this, browsers should be instructed to check on every request if the data in your application has changed.

The `cache-control` header should be `cache-control: public, max-age=0, must-revalidate`[1]

App data

The new `app-data.json` file which contains the build hash for the current deployment of the site should also share the same `cache-control` header as `page-data.json`. This is to ensure that the version of the site loaded in the browser is always synchronized with the currently deployed version.

The `cache-control` header should be `cache-control: public, max-age=0, must-revalidate`[1]

Static files

All files in `static/` should be cached forever. For files in this directory, Gatsby creates paths that are directly tied to the content of the file. Meaning that if the file content changes, then the file path changes also. These paths look weird e.g. `reactnext-gatsby-performance.001-a3e9d70183ff294e097c4319d0f8cff6-0b1ba.png` but since the same file will always be returned when that path is requested, Gatsby can cache it forever.

The `cache-control` header should be `cache-control: public, max-age=31536000, immutable`

JavaScript and CSS

JavaScript and CSS files *generated by webpack* should also be cached forever. Like static files, Gatsby creates JS & CSS file names (as a hash!) based on the file content. If the file content is changed, the file hash will change, therefore these files *generated by webpack* are safe to cache.

The `cache-control` header should be `cache-control: public, max-age=31536000, immutable`

The only exception to this is the file `/sw.js`, which needs to be revalidated upon each load to check if a new version of the site is available. This file is generated by `gatsby-plugin-offline` and other service worker plugins, in order to serve content offline. Its `cache-control` header should be `cache-control: public, max-age=0, must-revalidate`[1]

Setting up caching on different hosts

How you set up your caching depends on how you host your site. We encourage people to create Gatsby plugins per host to automate the creation of caching headers.

The following plugins have been created:

- gatsby-plugin-netlify
- gatsby-plugin-s3

When deploying with Vercel, follow the instructions in the Vercel documentation.

---

1 You can use 'no-cache' instead of 'max-age=0, must-revalidate'. Despite what the name might imply, 'no-cache' permits a cache to serve cached content as long as it validates the cache freshness first. 2 In either case, clients have to make a round trip to the origin server on each request. However, if you are correctly utilizing ETags or Last-Modified validation you will avoid downloading assets when the cached copy is still valid (e.g. the file hasn't changed on the origin server since it was cached).