gatsby-plugin-offline

Adds drop-in support for making a Gatsby site work offline and more resistant to bad network connections. It uses Workbox Build to create a service worker for the site and loads the service worker into the client.

If you're using this plugin with `gatsby-plugin-manifest` (recommended) this plugin should be listed *after* that plugin so the manifest file can be included in the service worker.

Install

`npm install gatsby-plugin-offline`

How to use

```
// In your gatsby-config.js
plugins: [`gatsby-plugin-offline`]
```

Available options

In `gatsby-plugin-offline` 3.x, the following options are available:

- `precachePages` lets you specify pages whose resources should be precached by the service worker, using an array of globs. For example:

  ```
  plugins: [
    {
      resolve: `gatsby-plugin-offline`,
      options: {
        precachePages: [`/about-us/`, `/projects/*`],
      },
    },
  ]
  ```

  Note: while essential resources of specified pages will be precached, such as JavaScript and CSS, non-essential resources such as fonts and images will not be included. Instead, these will be cached at runtime when a user visits a given page that includes these resources.

- `appendScript` lets you specify a file to be appended at the end of the generated service worker (`sw.js`). For example:

  ```
  plugins: [
    {
      resolve: `gatsby-plugin-offline`,
      options: {
        appendScript: require.resolve(`src/custom-sw-code.js`),
      },
    },
  ]
  ```

```
// show a notification after 15 seconds (the notification
// permission must be granted first)
setTimeout(() => {
  self.registration.showNotification("Hello, world!")
}, 15000)

// register a custom navigation route
const customRoute = new workbox.routing.NavigationRoute(({ event }) => {
  // ...
})
workbox.routing.registerRoute(customRoute)
```

- `debug` specifies whether Workbox should show debugging output in the browser console at runtime. When undefined, defaults to showing debug messages on `localhost` only.
- `workboxConfig` allows you to override the default Workbox options - see Overriding Workbox configuration. For example:

```
plugins: [
  {
    resolve: `gatsby-plugin-offline`,
    options: {
      workboxConfig: {
        importWorkboxFrom: `cdn`,
      },
    },
  },
]
```

Upgrading from 2.x

To upgrade from 2.x to 3.x, move any existing options into the `workboxConfig` option. If you haven't specified any options, you have nothing to do.

For example, here is a 2.x config:

```
plugins: [
  {
    resolve: `gatsby-plugin-offline`,
    options: {
      importWorkboxFrom: `cdn`,
    },
  },
]
```

Here is the equivalent 3.x config:

```
plugins: [
  {
    resolve: `gatsby-plugin-offline`,
    options: {
      workboxConfig: {
        importWorkboxFrom: `cdn`,
      },
    },
  },
]
```

In version 3, Workbox is also upgraded to version 4 so you may need to update your `workboxConfig` if any of those changes apply to you. Please see the docs on Google Developers for more information.

Overriding Workbox configuration

When adding this plugin to your `gatsby-config.js`, you can use the option `workboxConfig` to override the default Workbox config. To see the full list of options, see this article on Google Developers.

The default `workboxConfig` is as follows. Note that some of these options are configured automatically, e.g. `globPatterns`. If you're not sure about what all of these options mean, it's best to leave them as-is - otherwise, you may end up causing errors on your site, causing old files to be remain cached, or even breaking offline support.

```
const options = {
  importWorkboxFrom: `local`,
  globDirectory: rootDir,
  globPatterns,
  modifyURLPrefix: {
    // If `pathPrefix` is configured by user, we should replace
    // the default prefix with `pathPrefix`.
    "/": `${pathPrefix}/`,
  },
  cacheId: `gatsby-plugin-offline`,
  // Don't cache-bust JS or CSS files, and anything in the static directory,
  // since these files have unique URLs and their contents will never change
  dontCacheBustURLsMatching: /(\.js$|\.css$|static\/)/,
  runtimeCaching: [
    {
      // Use cacheFirst since these don't need to be revalidated (same RegExp
      // and same reason as above)
      urlPattern: /(\.js$|\.css$|static\/)/,
      handler: `CacheFirst`,
    },
    {
      // page-data.json files, static query results and app-data.json
      // are not content hashed
      urlPattern: /^https?:.*\/page-data\/.*\.json/,
      handler: `StaleWhileRevalidate`,
    },
    {
```

```
      // Add runtime caching of various other page resources
      urlPattern:
        /^https?:.*\.(png|jpg|jpeg|webp|svg|gif|tiff|js|woff|woff2|json|css)$/,
      handler: `StaleWhileRevalidate`,
    },
    {
      // Google Fonts CSS (doesn't end in .css so we need to specify it)
      urlPattern: /^https?:\/\/fonts\.googleapis\.com\/css/,
      handler: `StaleWhileRevalidate`,
    },
  ],
  skipWaiting: true,
  clientsClaim: true,
}
```

Remove

If you want to remove `gatsby-plugin-offline` from your site at a later point, substitute it with `gatsby-plugin-remove-serviceworker` to safely remove the service worker. First, install the new package:

```
npm install gatsby-plugin-remove-serviceworker
npm uninstall gatsby-plugin-offline
```

Then, update your `gatsby-config.js`:

```
 plugins: [
-  `gatsby-plugin-offline`,
+  `gatsby-plugin-remove-serviceworker`,
 ]
```

This will ensure that the worker is properly unregistered, instead of leaving an outdated version registered in users' browsers.

Notes

Empty View Source and SEO

Gatsby offers great SEO capabilities and that is no different with `gatsby-plugin-offline`. However, you shouldn't think that Gatsby doesn't serve HTML tags anymore when looking at your source code in the browser (with `Right click` => `View source`). `View source` doesn't represent the actual HTML data since `gatsby-plugin-offline` registers and loads a service worker that will cache and handle this differently. Your site is loaded from the service worker, not from its actual source (check your `Network` tab in the DevTools for that).

To see the HTML data that crawlers will receive, run this in your terminal:

on Windows (using powershell):

```
Invoke-WebRequest https://www.yourdomain.tld | Select -ExpandProperty Content
```

on Mac OS/Linux:

```
curl https://www.yourdomain.tld
```

Alternatively you can have a look at the `/public/index.html` file in your project folder.

App shell and server logs

Server logs (like from Netlify analytics) may show a large number of pageviews to a route like `/offline-plugin-app-shell-fallback/index.html`, this is a result of `gatsby-plugin-offline` adding an app shell to the page. The app shell is a minimal amount of user interface that can be cached offline for reliable performance loading on repeat visits. The shell can be loaded from the cache, and the content of the site loaded into the shell by the service worker.

Using with gatsby-plugin-manifest

If using this plugin with `gatsby-plugin-manifest` you may find that your icons are not cached. In order to solve this, update your `gatsby-config.js` as follows:

```
// gatsby-config.js
{
   resolve: 'gatsby-plugin-manifest',
   options: {
      icon: 'icon.svg',
      cache_busting_mode: 'none'
   }
},
{
   resolve: 'gatsby-plugin-offline',
   options: {
      workboxConfig: {
         globPatterns: ['**/icon-path*']
      }
   }
}
```

Updating `cache_busting_mode` is necessary. Otherwise, workbox will break while attempting to find the cached URLs. Adding the `globPatterns` makes sure that the offline plugin will cache everything. Note that you have to prefix your icon with `icon-path` or whatever you may call it