gatsby-plugin-manifest

The web app manifest (part of the PWA specification) enabled by this plugin allows users to add your site to their home screen on most mobile browsers — see here. The manifest provides configuration and icons to the phone.

This plugin provides several features beyond manifest configuration to make your life easier, they are:

- Auto icon generation - generates multiple icon sizes from a single source
- Favicon support
- Legacy icon support (iOS)[^1]
- Cache busting
- Localization - Provides unique manifests for path-based localization (Gatsby Example)

Each of these features has extensive configuration available so you are always in control.

This guide focuses on configuring the plugin. For more information on the web app manifest, check out the additional resources below.

Install

```
npm install gatsby-plugin-manifest
```

How to use

Add plugin and manifest settings - Required

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
        theme_color: `#a2466c`,
        display: `standalone`,
      },
    },
  ],
}
```

If you're using this plugin together with `gatsby-plugin-offline` (recommended), this plugin should be listed *before* the offline plugin so that it can cache the created `manifest.webmanifest`.

For more information on configuring your web app see here.

Configure icons and their generations - Required

There are three modes in which icon generation can function: automatic, hybrid, and manual(disabled). These modes can affect other configurations defaults.

- Automatic - Generate a pre-configured set of icons from a single source icon.
    - Favicon - yes
    - Legacy icon support - yes
    - Cache busting - yes
    - Localization - optional
- Hybrid - Generate a manually configured set of icons from a single source icon.
    - Favicon - yes
    - Legacy icon support - yes
    - Cache busting - yes
    - Localization - optional
- Manual - Don't generate or pre-configure any icons.
    - Favicon - never
    - Legacy icon support - yes
    - Cache busting - never
    - Localization - optional

IMPORTANT: For best results, if you're providing an icon for generation it should be...

- ...at least as big as the largest icon being generated (512x512 by default).
- ...square (if it's not, transparent bars will automatically be added to make it square).
- ...of one of the following formats: JPEG, PNG, WebP, TIFF, GIF or SVG.

Automatic mode configuration

Add the following line to the plugin options

```
icon: `src/images/icon.png`, // This path is relative to the root of the site.
```

Automatic mode is the easiest option for most people.

Hybrid mode configuration

Add the following line to the plugin options

```
icon: `src/images/icon.png`, // This path is relative to the root of the site.
icons: [
  {
    src: `/favicons/android-chrome-192x192.png`,
    sizes: `192x192`,
    type: `image/png`,
  },
  {
    src: `/favicons/android-chrome-512x512.png`,
    sizes: `512x512`,
    type: `image/png`,
  },
], // Add or remove icon sizes as desired
```

If you want to include more or fewer sizes, then the hybrid option is for you. Like automatic mode, you include a high-resolution icon from which to generate smaller icons. But unlike automatic mode, you provide the `icons` array config and icons are generated based on the sizes defined in your config.

The hybrid option allows the most flexibility while still not requiring you to create all icon sizes manually.

Manual mode configuration

Add the following line to the plugin options

```
icons: [
  {
    src: `/favicons/android-chrome-192x192.png`,
    sizes: `192x192`,
    type: `image/png`,
  },
  {
    src: `/favicons/android-chrome-512x512.png`,
    sizes: `512x512`,
    type: `image/png`,
  },
], // Add or remove icon sizes as desired
```

In the manual mode, you are responsible for defining the entire web app manifest and providing the defined icons in the static folder. Only icons you provide will be available. There is no automatic resizing done for you.

Feature configuration - Optional

Localization configuration

Localization allows you to create unique manifests for each localized version of your site. You can add as many languages as you want. Localization requires unique paths for each language (e.g. if your default about page is at `/about`, the German (`de`) version would be `/de/about`).

The default site language should be configured in your root plugin options. Any additional languages should be defined in the `localize` array. The root settings will be used as defaults if not overridden in a locale. Any configuration option available in the root is also available in the `localize` array.

`lang` and `start_url` are the only *required* options in the array objects. `name` , `short_name` , and `description` are recommended to be translated if being used in the default language. All other config options are optional. This is helpful if you want to provide unique icons for each locale.

The `lang` option is part of the web app manifest specification and thus is required to be a valid language tag.

Using localization requires name-based cache busting when using a unique icon in automatic mode for a specific locale. This is automatically enabled if you provide an `icon` in a specific locale without uniquely defining `icons` . If you're using icon creation in hybrid or manual mode for your locales, remember to provide unique icon paths.

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
```

```
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `The Cool Application`,
        short_name: `Cool App`,
        description: `The application does cool things and makes your life better.`,
        lang: `en`,
        display: `standalone`,
        icon: `src/images/icon.png`,
        start_url: `/`,
        background_color: `#663399`,
        theme_color: `#fff`,
        localize: [
          {
            start_url: `/de/`,
            lang: `de`,
            name: `Die coole Anwendung`,
            short_name: `Coole Anwendung`,
            description: `Die Anwendung macht coole Dinge und macht Ihr Leben
besser.`,
          },
        ],
      },
    },
  ],
}
```

Iterative icon options

The `icon_options` object may be used to iteratively add configuration items to the `icons` array. Any options included in this object will be merged with each object of the `icons` array (custom or default). Key value pairs already in the `icons` array will take precedence over duplicate items in the `icon_options` array.

`icon_options` may be used as follows:

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
        theme_color: `#a2466c`,
        display: `standalone`,
        icon: `src/images/icon.png`,
        icon_options: {
          // For all the options available,
          // please see the section "Additional Resources" below.
          purpose: `any maskable`,
```

```
        },
      },
    },
  ],
}
```

Disable legacy icons

iOS 11.3 added support for the web app manifest specification. Previous iOS versions won't recognize the icons defined in the webmanifest and the creation of `apple-touch-icon` links in `<head>` is needed. This plugin creates them by default. If you don't want those icons to be generated you can set the `legacy` option to `false` in the plugin configuration:

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
        theme_color: `#a2466c`,
        display: `standalone`,
        icon: `src/images/icon.png`,
        legacy: false, // this will not add apple-touch-icon links to <head>
      },
    },
  ],
}
```

Disable favicon

A favicon is generated by default in automatic and hybrid modes (a 32x32 PNG, included via a `<link rel="icon" />` tag in the document head). Additionally, if an SVG icon is provided as the source, it will be used in the document head without modification as a favicon. The PNG will still be created and included as a fallback. Including the SVG icon allows creating a responsive icon with CSS Media Queries such as dark mode and others.

You can set the `include_favicon` plugin option to `false` to opt-out of this behavior.

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
```

```
        theme_color: `#a2466c`,
        display: `standalone`,
        icon: `src/images/icon.png`, // This path is relative to the root of the
site.
        include_favicon: false, // This will exclude favicon link tag
      },
    },
  ],
}
```

Disable or configure "[cache busting](#)"

Cache Busting allows your updated icon to be quickly/easily visible to your site's visitors. HTTP caches could otherwise keep an old icon around for days and weeks. Cache busting can only be done in 'automatic' and 'hybrid' modes.

Cache busting works by calculating a unique "digest" of the provided icon and modifying links or file names of generated images with that unique digest. If you ever update your icon, the digest will change and caches will be busted.

Options:

- **`query`** - This is the default mode. File names are unmodified but a URL query is appended to all links. e.g. `icons/icon-48x48.png?digest=abc123`.
- **`name`** - Changes the cache busting mode to be done by file name. File names and links are modified with the icon digest. e.g. `icons/icon-48x48-abc123.png` (only needed if your CDN does not support URL query based cache busting). This mode is required and automatically enabled for a locale's icons if you are providing a unique icon for a specific locale in automatic mode using the localization features.
- **`none`** - Disables cache busting. File names and links remain unmodified.

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
        theme_color: `#a2466c`,
        display: `standalone`,
        icon: `src/images/icon.png`,
        cache_busting_mode: `none`, // `query`(default), `name`, or `none`
      },
    },
  ],
}
```

Using with gatsby-plugin-offline

If using this plugin with `gatsby-plugin-offline` 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

Remove `theme-color` meta tag

By default a `<meta content={theme_color} name="theme-color" />` tag is inserted into the html output. This can be problematic if you want to programmatically control that tag (e.g. when implementing light/dark themes in your project). You can set `theme_color_in_head` plugin option to `false` to opt-out of this behavior.

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
        theme_color: `#a2466c`,
        display: `standalone`,
        icon: `src/images/icon.png`,
        theme_color_in_head: false, // This will avoid adding theme-color meta tag.
      },
    },
  ],
}
```

Enable CORS using `crossorigin` attribute

Add a `crossorigin` attribute to the manifest `<link rel="manifest" crossorigin="use-credentials" href="/manifest.webmanifest" />` link tag.

You can set `crossOrigin` plugin option to `'use-credentials'` to enable sharing resources via cookies. Any invalid keyword or empty string will fallback to `'anonymous'`.

You can find more information about `crossorigin` on MDN.

```
// in gatsby-config.js
module.exports = {
  plugins: [
    {
      resolve: `gatsby-plugin-manifest`,
      options: {
        name: `GatsbyJS`,
        short_name: `GatsbyJS`,
        start_url: `/`,
        background_color: `#f7f0eb`,
        theme_color: `#a2466c`,
        display: `standalone`,
        icon: `src/images/icon.png`,
        crossOrigin: `use-credentials`, // `use-credentials` or `anonymous`
      },
    },
  ],
}
```

Appendices

Additional information that may be interesting or valuable.

Default icon config

When in automatic mode the following json array is injected into the manifest configuration you provide and the icons are generated from it.

```
[
  {
    "src": "icons/icon-48x48.png",
    "sizes": "48x48",
    "type": "image/png"
  },
  {
    "src": "icons/icon-72x72.png",
    "sizes": "72x72",
    "type": "image/png"
  },
  {
    "src": "icons/icon-96x96.png",
```

```
    "sizes": "96x96",
    "type": "image/png"
  },
  {
    "src": "icons/icon-144x144.png",
    "sizes": "144x144",
    "type": "image/png"
  },
  {
    "src": "icons/icon-192x192.png",
    "sizes": "192x192",
    "type": "image/png"
  },
  {
    "src": "icons/icon-256x256.png",
    "sizes": "256x256",
    "type": "image/png"
  },
  {
    "src": "icons/icon-384x384.png",
    "sizes": "384x384",
    "type": "image/png"
  },
  {
    "src": "icons/icon-512x512.png",
    "sizes": "512x512",
    "type": "image/png"
  }
]
```

Legacy icon support coverage

Currently this feature only covers older versions of iOS Safari.

Internet Explorer is the only other major browser that doesn't support the web app manifest, and its market share is so small no one has contributed support.

Additional resources

This article from the Chrome DevRel team is a good intro to the web app manifest — https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/

For more information, see the W3C specification, Mozilla documentation or Web.Dev guide. More information about `icon_options` can be found in this Web.Dev guide.

Plugin options validation

This plugin validates plugin options set in the `gatsby-config.js`. It validates the options used by the plugin and the entire WebAppManifest spec. To see the exact implementation of the validator see src/pluginOptionsSchema.js.

The WebAppManifest spec is not stable at the time of writing. This version of the validator adheres the most recent version of the specification available.

Troubleshooting

Incompatible library version: sharp.node requires version X or later, but Z provides version Y

This means that there are multiple incompatible versions of the `sharp` package installed in `node_modules`. The complete error typically looks like this:

```
Something went wrong installing the "sharp" module

dlopen(/Users/misiek/dev/gatsby-starter-
blog/node_modules/sharp/build/Release/sharp.node, 1): Library not loaded:
@rpath/libglib-2.0.dylib
  Referenced from: /Users/misiek/dev/gatsby-starter-
blog/node_modules/sharp/build/Release/sharp.node
  Reason: Incompatible library version: sharp.node requires version 6001.0.0 or
later, but libglib-2.0.dylib provides version 5801.0.0
```

To fix this, you'll need to update all Gatsby plugins in the current project that depend on the `sharp` package. Here's a list of official plugins that you might need to update in case your projects use them:

- `gatsby-plugin-sharp`
- `gatsby-plugin-manifest`
- `gatsby-remark-images-contentful`
- `gatsby-source-contentful`
- `gatsby-transformer-sharp`
- `gatsby-transformer-sqip`

To update these packages, run:

```
npm install gatsby-plugin-sharp gatsby-plugin-manifest gatsby-remark-images-
contentful gatsby-source-contentful gatsby-transformer-sharp gatsby-transformer-sqip
```

If updating these doesn't fix the issue, your project probably uses other plugins from the community that depend on a different version of `sharp`. Try running `npm list sharp` or `yarn why sharp` to see all packages in the current project that use `sharp` and try updating them as well.

APACHE: Error while trying to use the following icon from the Manifest

If you are on an Apache webserver and utilizing the automatic solution, you may receive an error when attempting to access an icon listed in the manifest as residing in the `/icons` folder. This is because Apache restricts access to all `/icons` folders by default. You can fix this by switching to the hybrid solution and changing the folder name from `icons` to something like `favicons`.

Example:

```
// In the gatsby-plugin-manifest section of your gatsby-config.js
  icon: `src/images/icon.png`, // This path is relative to the root of the site.
  icons: [
    {
        "src": "favicons/icon-144x144.png",
```

```
        "sizes": "144x144",
        "type": "image/png"
    },  // Add or remove icon sizes as desired
  ]
```

Alternatively, if you have access to modify Apache, you can resolve this issue by removing the restriction on `/icons` folders.

On Debian based systems

Backup the `/etc/apache2/mods-available/alias.conf` file:

```
cp /etc/apache2/mods-available/alias.conf /etc/apache2/mods-
available/alias.conf.back
```

Comment out the Alias `/icons/ "/usr/share/apache2/icons/"` row in `/etc/apache2/mods-available/alias.conf` file:

```
cat /etc/apache2/mods-available/alias.conf | grep "Alias /icons/"
```

Reload Apache service:

```
service apache2 reload
```

On Red Hat or CentOS systems

Create a backup of `/etc/httpd/conf.d/autoindex.conf` :

```
cp /etc/httpd/conf.d/autoindex.conf /etc/httpd/conf.d/autoindex.conf.back
```

Comment out `"Alias /icons/"` in `/etc/httpd/conf.d/autoindex.conf` :

```
cat /etc/httpd/conf.d/autoindex.conf | grep "Alias /icons/"
```

Reload Apache or restart the server:

```
service httpd reload
```