gatsby-remark-images

Processes images in markdown so they can be used in the production build.

In the processing, it makes images responsive by:

- Adding an elastic container to hold the size of the image while it loads to avoid layout jumps.
- Generating multiple versions of images at different widths and sets the `srcset` and `sizes` of the `img` element so regardless of the width of the device, the correct image is downloaded.
- Using the "blur up" technique popularized by Medium and Facebook where a small 20px wide version of the image is shown as a placeholder until the actual image is downloaded.

Install

```
npm install gatsby-remark-images gatsby-plugin-sharp
```

How to use

```
// In your gatsby-config.js
plugins: [
  `gatsby-plugin-sharp`,
  {
    resolve: `gatsby-transformer-remark`,
    options: {
      plugins: [
        {
          resolve: `gatsby-remark-images`,
          options: {
            // It's important to specify the maxWidth (in pixels) of
            // the content container as this plugin uses this as the
            // base for generating different widths of each image.
            maxWidth: 590,
          },
        },
      ],
    },
  },
]
```

Usage in Markdown

You can reference an image using the relative path, where that path is relative to the location of the Markdown file you're typing in.

```
![Alt text here](./image.jpg)
```

By default, the text `Alt text here` will be used as the alt attribute of the generated `img` tag. If an empty alt attribute like `alt=""` is wished, a reserved keyword `GATSBY_EMPTY_ALT` can be used.

```
![GATSBY_EMPTY_ALT](./image.png)
```

Options

Name	Default	Description
`maxWidth`	`650`	The `maxWidth` in pixels of the div where the markdown will be displayed. This value is used when deciding what the width of the various responsive thumbnails should be.
`linkImagesToOriginal`	`true`	Add a link to each image to the original image. Sometimes people want to see a full-sized version of an image e.g. to see extra detail on a part of the image and this is a convenient and common pattern for enabling this. Set this option to `false` to disable this behavior.
`showCaptions`	`false`	Add a caption to each image with the contents of the title attribute, when this is not empty. If the title attribute is empty but the alt attribute is not, it will be used instead. Set this option to true to enable this behavior. You can also pass an array instead to specify which value should be used for the caption — for example, passing `['alt', 'title']` would use the alt attribute first, and then the title. When this is set to `true` it is the same as passing `['title', 'alt']`. If you just want to use the title (and omit captions for images that have alt attributes but no title), pass `['title']`.
`markdownCaptions`	`false`	Parse the caption as markdown instead of raw text. Ignored if `showCaptions` is `false`.
`wrapperStyle`		Add custom styles to the div wrapping the responsive images. Use the syntax for the style attribute e.g. `margin-bottom:10px; background: red;` or a function returning a style string which receives the information about the image you can use to dynamically set styles based on the `aspectRatio` for example.
`backgroundColor`	`white`	Set the background color of the image to match the background image of your design. **Note:** - set this option to `transparent` for a transparent image background. - set this option to `none` to completely remove the image background.
`quality`	`50`	The quality level of the generated files.
`withWebp`	`false`	Additionally generate WebP versions alongside your chosen file format. They are added as a srcset with the appropriate mimetype and will be loaded in browsers that support the format. Pass `true` for default support, or an object of options to specifically override those for the WebP files. For example, pass `{ quality: 80 }` to have the WebP images be at quality level 80.
`withAvif`	`false`	Additionally generate AVIF versions alongside your chosen file format. They are added as a srcset with the appropriate mimetype and will be

		loaded in browsers that support the format. Pass `true` for default support, or an object of options to specifically override those for the AVIF files. For example, pass `{ quality: 80 }` to have the AVIF images be at quality level 80.
`tracedSVG`	`false`	Use traced SVGs for placeholder images instead of the "blur up" effect. Pass `true` for traced SVGs with the default settings (seen here), or an object of options to override the defaults. For example, pass `{ color: "#F00", turnPolicy: "TURNPOLICY_MAJORITY" }` to change the color of the trace to red and the turn policy to `TURNPOLICY_MAJORITY`. See `node-potrace` parameter documentation for a full listing and explanation of the available options.
`loading`	`lazy`	Set the browser's native lazy loading attribute. One of `lazy`, `eager` or `auto`.
`decoding`	`async`	Set the browser's native decoding attribute. One of `async`, `sync` or `auto`.
`disableBgImageOnAlpha`	`false`	Images containing transparent pixels around the edges results in images with blurry edges. As a result, these images do not work well with the "blur up" technique used in this plugin. As a workaround to disable background images with blurry edges on images containing transparent pixels, enable this setting.
`disableBgImage`	`false`	Remove background image and its' inline style. Useful to prevent `Stylesheet too long` error on AMP.
`srcSetBreakpoints`		By default gatsby generates 0.25x, 0.5x, 1x, 1.5x, 2x, and 3x sizes of thumbnails. If you want more control over which sizes are output you can use the `srcSetBreakpoints` parameter. For example, if you want images that are 200, 340, 520, and 890 wide you can add `srcSetBreakpoints: [ 200, 340, 520, 890 ]` as a parameter. You will also get `maxWidth` as a breakpoint (which is 650 by default), so you will actually get `[ 200, 340, 520, 650, 890 ]` as breakpoints.

dynamic wrapperStyle example

```
{
  resolve: `gatsby-remark-images`,
  options: {
    maxWidth: 800,
    wrapperStyle: fluidResult => `flex:${_.round(fluidResult.aspectRatio, 2)};`,
  },
}
```

Supported Formats

This plugin will support the following formats:

- JPEG
- PNG

Since Sharp is used for image processing, this plugin will not support GIFs or SVGs. If you would like to render these file types with the image markdown syntax, use the `gatsby-remark-copy-linked-files` plugin. Do note with this it will load in the images, but won't use the features of Sharp such as the elastic container or the blur-up enhancements.