Using Gatsby Image to Prevent Image Bloat

import ImageModel from "@components/layer-model/image-model"

This doc is for the deprecated `gatsby-image` package. See Using the Gatsby Image plugin for the new image plugin.

Using images in Gatsby components and pages requires four steps to take advantage of performance benefits.

`gatsby-image` is a React component designed to work seamlessly with Gatsby's GraphQL queries (`gatsby-image` plugin README). It combines Gatsby's native image processing capabilities with advanced image loading techniques to easily and completely optimize image loading for your sites. `gatsby-image` uses gatsby-plugin-sharp to power its image transformations.

> *Warning: gatsby-image is **not** a drop-in replacement for `<img />`. It's optimized for fixed width/height images and images that stretch the full width of a container. Some ways you can use `<img />` won't work with gatsby-image.*

Here's a demo site that uses the gatsby-image plugin

`gatsby-image` includes the tricks you'd expect from a modern image component. It:

- uses the new IntersectionObserver API to cheaply lazy load images
- holds an image's position so your page doesn't jump around as images load
- makes it possible to add a placeholder—either a gray background or a blurry version of the image.

For more complete API information, check out the Gatsby Image API docs.

Problem

Large, unoptimized images dramatically slow down your site.

But creating optimized images for websites has long been a thorny problem. Ideally, you would:

- Resize large images to the size needed by your design
- Generate multiple smaller images so smartphones and tablets don't download desktop-sized images

- Strip all unnecessary metadata and optimize JPEG and PNG compression
- Efficiently lazy load images to speed initial page load and save bandwidth
- Use the "blur-up" technique or a "traced placeholder" SVG to show a preview of the image while it loads
- Hold the image position so your page doesn't jump while the images load

Doing this consistently across a site feels like a task that can never be completed. You manually optimize your images and then... several images are swapped in at the last minute or a design-tweak shaves 100px of width off your images.

Most solutions involve a lot of manual labor and bookkeeping to ensure every image is optimized.

This isn't ideal. Optimized images should be easy and the default.

Solution

With Gatsby, you can make the experience of working with images way, way better.

`gatsby-image` is designed to work seamlessly with Gatsby's native image processing capabilities powered by GraphQL and Sharp. To produce perfect images with minimal effort, you can:

1. Install `gatsby-image` and other, necessary dependencies like `gatsby-plugin-sharp` and `gatsby-transformer-sharp`

```
npm install gatsby-image gatsby-transformer-sharp gatsby-plugin-sharp
```

2. Add the newly installed plugins and transformer plugins to your `gatsby-config.js`

```
module.exports = {
  plugins: [`gatsby-plugin-sharp`, `gatsby-transformer-sharp`],
}
```

3. Configure `gatsby-source-filesystem` to load images from a folder. In order to use GraphQL to query the image files, the files need to be in a location that is known to Gatsby. This requires an update to `gatsby-config.js` to configure the plugin. Feel free to replace the `path` option to reference wherever your images are located in your project.

```
module.exports = {
  plugins: [
    `gatsby-transformer-sharp`,
    `gatsby-plugin-sharp`,
    // highlight-start
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        path: `${__dirname}/src/data/`,
```

```
      },
    },
    // highlight-end
  ],
}
```

4. Write a GraphQL query using one of the included GraphQL "fragments" which specify the fields needed by `gatsby-image` to create a responsive, optimized image. This example queries for an image at a path relative to the location specified in the `gatsby-source-filesystem` options using the `GatsbyImageSharpFluid` fragment.

```
import React from "react"
import { graphql } from "gatsby" // highlight-line
import Layout from "../components/layout"

export default function MyDogs({ data }) {
  return (
    <Layout>
      <h1>I love my corgi!</h1>
    </Layout>
  )
}

// highlight-start
export const query = graphql`
  query MyQuery {
    file(relativePath: { eq: "images/corgi.jpg" }) {
      childImageSharp {
        # Specify the image processing specifications right in the query.
        fluid {
          ...GatsbyImageSharpFluid
        }
      }
    }
  }
`
// highlight-end
```

5. Import `Img` to display the fragment in JSX. There are additional features available with the `Img` tag as well, such as the `alt` attribute for accessibility.

```
import React from "react"
import { graphql } from "gatsby"
import Layout from "../components/layout"
import Img from "gatsby-image" // highlight-line

export default function MyDogs({ data }) {
```

```
  return (
    <Layout>
      <h1>I love my corgi!</h1>
      // highlight-start
      <Img
        fluid={data.file.childImageSharp.fluid}
        alt="A corgi smiling happily"
      />
      // highlight-end
    </Layout>
  )
}

export const query = graphql`
  query MyQuery {
    file(relativePath: { eq: "images/corgi.jpg" }) {
      childImageSharp {
        # Specify the image processing specifications right in the query.
        fluid {
          ...GatsbyImageSharpFluid
        }
      }
    }
  }
`
```

This GraphQL query creates multiple sizes of the image and when the page is rendered the image that is appropriate for the current screen resolution (e.g. desktop, mobile, and everything in between) is used. The `gatsby-image` component automatically enables a blur-up effect as well as lazy loading images that are not currently on screen.

So this is all very nice and it's far better to be able to use this from npm vs. implementing it yourself or cobbling together several standalone libraries.

Additional resources

- Gatsby Image API docs
- gatsby-image plugin README file
- Source code for an example site using gatsby-image
- Blog articles about gatsby-image
- Starters that use gatsby-image
- Other image plugins
- "Ridiculously easy image optimization with gatsby-image" by Kyle Gill