Gatsby themes introduce a concept called "shadowing". This feature allows users to replace a file in the `src` directory that is included in the webpack bundle with their own implementation. This works for React components, pages in `src/pages`, JSON files, TypeScript files, as well as any other imported file (such as `.css`) in your site.

A practical use case is when you've installed `gatsby-theme-blog` and want to customize the author `Bio` component to add your own biographical content. Shadowing lets you replace the theme's original file, `gatsby-theme-blog/src/components/bio.js`, with your own file to make any changes you need.

Shadowing example

If you've installed `gatsby-theme-blog` you'll notice that it renders a `Bio` component which is used in the `BlogPost` template. If you'd like to change the `Bio` component you can do so with the shadowing API.

Theme file structure

You can inspect `gatsby-theme-blog`'s file structure to determine the file path for the file you want to shadow.

```
├── gatsby-browser.js
├── gatsby-config.js
├── gatsby-node.js
└── src
    ├── components
    │   ├── bio-content.js
    │   ├── bio.js // highlight-line
    │   ├── header.js
    │   ├── home-footer.js
    │   ├── layout.js
    │   ├── post-date.js
    │   ├── post-footer.js
    │   ├── post-hero.js
    │   ├── post-link.js
    │   ├── post-list.js
    │   ├── post-title.js
    │   ├── post.js
    │   ├── posts.js
    │   └── seo.js
    ├── gatsby-plugin-theme-ui
    │   └── components.js
    └── gatsby-theme-blog-core
        └── components
            ├── post.js
            └── posts.js
```

Customizing the `Bio` component

In this case, the file to shadow is `gatsby-theme-blog/src/components/bio.js`.

The shadowing API uses a deterministic file structure to determine which component will be rendered. In order to override the `Bio` component in `gatsby-theme-blog`, create a file named `user-site/src/gatsby-theme-blog/components/bio.js`.

Any file that lives in the `src/gatsby-theme-blog` directory of the user's site will be used instead of a file with the same name located in the theme's src directory: `gatsby-theme-blog/src`. This replaces the entire file: to re-use parts of the original file from the theme such as functionality or styling, check out the sections of this doc on extending and importing shadowed files.

This means that `user-site/src/gatsby-theme-blog/components/bio.js` will be rendered in place of `gatsby-theme-blog/src/components/bio.js`:

```
import React from "react"
export default function Bio() {
  return <h1>My new bio component!</h1>
}
```

A successful shadow of the Bio component will result in the following directory tree:

```
user-site
└── src
    └── gatsby-theme-blog
        └── components
            └── bio.js // highlight-line
```

Shadowing other files

Some themes, including `gatsby-theme-blog`, install additional plugins. `gatsby-theme-blog` uses `gatsby-plugin-theme-ui` with the `gatsby-theme-ui-preset` preset. Shadowing is one way to customize the styling of a theme.

For example, to shadow `index.js` from `gatsby-plugin-theme-ui`, create a file named `user-site/src/gatsby-plugin-theme-ui/index.js`. The styles in this file will be automatically merged with those in `gatsby-theme-ui-preset`. For conflicting styles, your local shadowed settings take precedence.

```
export default {
  fontSizes: [12, 14, 16, 24, 32, 48, 64, 96, 128],
  space: [0, 4, 8, 16, 32, 64, 128, 256],
  colors: {
    primary: `tomato`,
  },
}
```

> *Note that any styles in shadowed files will automatically get deep-merged with your `preset` theme. Shadowed styles take precedence.*

Which will result in the following directory tree:

```
user-site
└── src
    └── gatsby-plugin-theme-ui
        └──index.js // highlight-line
```

Any source file is shadowable

The shadowing API isn't restricted to React components; you can override any JavaScript, Markdown, MDX, or CSS file in the `src` directory. This gives you fine-grained control of all functionality, content, and styling that a theme provides.

If you wanted to shadow a CSS file in `gatsby-theme-awesome-css` that's found at `src/styles/bio.css` you can do so by creating `user-site/src/gatsby-theme-awesome-css/styles/bio.css`

```
.bio {
  border: 10px solid tomato;
}
```

The theme's `bio.css` file would then be replaced with your new CSS file.

File extensions can be overridden

As long as the theme author imports components/files without the file extension, users are able to shadow these with other types of files. For example the theme author created a TypeScript file at `src/components/bio.tsx` and uses it in another file:

```
import Bio from "./bio"

/* Rest of the code */
```

You'll be able to shadow the Bio file by creating e.g. a JavaScript file at `src/gatsby-theme-blog/components/bio.js` as the file extension wasn't used in the import.

Extending shadowed files

In addition to overriding files, you can *extend* shadowable files.

This means that you can import the component you're shadowing and then render it. Consider a scenario where you have a custom `Card` component that you want to wrap the author's bio in.

Without extending the component, you would have to manually copy over the entire component implementation from the theme to wrap it with your custom shadowed component. It might look something like:

```
import React from "react"
import { Avatar, MediaObject, Icon } from "gatsby-theme-blog"
import Card from "../components/card"

export default function Bio({ name, bio, avatar, twitterUrl, githubUrl }) {
  return (
    <Card>
      <MediaObject>
        <Avatar {...avatar} />
        <div>
          <h3>{name}</h3>
          <p>{bio}</p>
```

```
          <a href={twitterUrl}>
            <Icon name="twitter" />
          </a>
          <a href={githubUrl}>
            <Icon name="github" />
          </a>
        </div>
      </MediaObject>
    </Card>
  )
}
```

This workflow isn't too bad, especially since the component is relatively straightforward. However, it could be optimized in scenarios where you want to wrap a component or pass a different prop without having to worry about the component's internals.

Importing the shadowed component

In the above example it might be preferable to be able to import the `Bio` component and wrap it with your `Card`. When importing, you can do the following instead:

```
import React from "react"
import { Author } from "gatsby-theme-blog/src/components/bio"
import Card from "../components/card"

export default function Bio(props) {
  return (
    <Card>
      <Author {...props} />
    </Card>
  )
}
```

This is a quick and efficient way to customize rendering without needing to worry about the implementation details of the component you're looking to customize. Importing the shadowed component means you can use composition, leveraging a great feature from React.

Applying new props

In some cases components offer prop APIs to change their behavior. To extend a component you can import it and then add a new prop.

For example, if `NewsletterCTA` accepts a `variant` prop which changes the look and colors of the call to action, you can use it when you extend the component. Below, `NewsletterCTA` is re-exported and `variant="link"` is added in the shadowed file to override its default value.

```
import { NewsletterCTA } from "gatsby-theme-blog/src/components/newsletter"

export default function CallToAction(props) {
  return <NewsletterCTA {...props} variant="link" />
}
```

Using the CSS prop

In addition to passing a different prop to a component you're extending, you might want to apply CSS using the Emotion CSS prop. This will allow you to change the styling of a particular component without changing any of its functionality.

```
import { NewsletterCTA } from "gatsby-theme-blog/src/components/newsletter"

export default function CallToAction(props) {
  return (
    <NewsletterCTA
      css={{
        backgroundColor: "rebeccapurple",
        color: "white",
        boxShadow: "none",
      }}
      {...props}
    />
  )
}
```

Note: For this approach to work NewsletterCTA has to accept a `className` property to apply styles after the CSS prop is transformed by the Emotion babel plugin.

```
export default {
  colors: {
    primary: "tomato",
  },
}
```

This provides a nice interface to extend an object if you want to change a couple values from the defaults.

How much shadowing is too much shadowing?

If you find yourself shadowing a large number of components in a particular theme, it might make sense to fork and modify the theme instead. The official Gatsby themes support this pattern using a set of `-core` themes. For example, `gatsby-theme-blog` relies on `gatsby-theme-blog-core` so you can fork `gatsby-theme-blog` (or skip it completely) to render your own components without having to worry about dealing with any of the page creation or data sourcing logic.