With webpack, using static assets like images and fonts works similarly to CSS.

You can `import` **a file right in a JavaScript module**. This tells webpack to include that file in the bundle. Unlike CSS imports, importing a file gives you a string value. This value is the final path you can reference in your code, e.g. as the `src` attribute of an image or the `href` of a link to a PDF.

To reduce the number of requests to the server, importing images that are less than 10,000 bytes returns a data URI instead of a path. This applies to the following file extensions: bmp, gif, jpg, jpeg, and png. SVG files are excluded due to #1153. You can control the 10,000 byte threshold by setting the `IMAGE_INLINE_SIZE_LIMIT` environment variable as documented in our advanced configuration.

Here is an example:

```
import React from 'react';
import logo from './logo.png'; // Tell webpack this JS file uses this image

console.log(logo); // /logo.84287d09.png

function Header() {
  // Import result is the URL of your image
  return <img src={logo} alt="Logo" />;
}

export default Header;
```

This ensures that when the project is built, webpack will correctly move the images into the build folder, and provide us with correct paths.

This works in CSS too:

```
.Logo {
  background-image: url(./logo.png);
}
```

webpack finds all relative module references in CSS (they start with `./`) and replaces them with the final paths from the compiled bundle. If you make a typo or accidentally delete an important file, you will see a compilation error, like when you import a non-existent JavaScript module. The final filenames in the compiled bundle are generated by webpack from content hashes. If the file content changes in the future, webpack will give it a different name in production so you don't need to worry about long-term caching of assets.

Please be advised that this is also a custom feature of webpack.

It is not required for React but many people enjoy it (and React Native uses a similar mechanism for images).

An alternative way of handling static assets is described in the next section.

Adding SVGs

> *Note: this feature is available with `react-scripts@2.0.0` and higher, and `react@16.3.0` and higher.*

One way to add SVG files was described in the section above. You can also import SVGs directly as React components. You can use either of the two approaches. In your code it would look like this:

```
import { ReactComponent as Logo } from './logo.svg';

function App() {
  return (
    <div>
      {/* Logo is an actual React component */}
      <Logo />
    </div>
  );
}
```

This is handy if you don't want to load SVG as a separate file. Don't forget the curly braces in the import! The `ReactComponent` import name is significant and tells Create React App that you want a React component that renders an SVG, rather than its filename.

> ***Tip:*** *The imported SVG React Component accepts a* `title` *prop along with other props that a* `svg` *element accepts. Use this prop to add an accessible title to your svg component.*