Gatsby handles three varieties of GraphQL queries: Page queries (sometimes for simplicity referred to as "normal queries"), static queries using the `<StaticQuery />` component, and static queries using the `useStaticQuery` hook.

Differences between varieties of GraphQL queries

Static queries differ from Gatsby page queries in a number of ways. For pages, Gatsby is capable of handling queries with variables because of its awareness of page context. However, **page queries can only be made in top-level page components**.

In contrast, static queries do not take variables. This is because static queries are used inside specific components, and can appear lower in the component tree. Data fetched with a static query won't be dynamic (i.e. **they can't use variables**, hence the name "static" query), but they can be called at any level in the component tree.

For more information on the practical differences in usage between static and normal queries, refer to the guide on static query. This guide discusses the differences in how they are handled internally by Gatsby

Keeping track of site queries during build in Redux stores

Gatsby stores the queries from your site in Redux stores called `components` and `staticQueryComponents`. This process and a flowchart illustrating it are explained in the query extraction guide.

In Redux, `staticQueryComponents` is a `Map` from component `jsonName` to `StaticQueryObject`. An example entry in that data structure looks like this:

```
{
  `blog-2018-07-17-announcing-gatsby-preview-995` : {
    name: `/path/to/component/file`,
    componentPath: `/path/to/component/file`,
    id: `blog-2018-07-17-announcing-gatsby-preview-995`,
    jsonName: `blog-2018-07-17-announcing-gatsby-preview-995`,
    query: `raw GraphQL Query text including fragments`,
    hash: `hash of graphql text`
  }
}
```

In the example above, `blog-2018-07-17-announcing-gatsby-preview-995` is the key, with the object as its value in the `Map`.

The `staticQueryComponents` Redux namespace watches for updates to queries while you are developing, and adds new data to your cache when queries change.

Replacing queries with JSON imports

With the final build, there isn't a GraphQL server running to query for data. Gatsby has already extracted and run the queries, storing them in files based on hashes in `/public/static/d/<hash>.json`. It can now remove code for GraphQL queries, because the data is already available.

Distinguishing between static and normal queries

Babel traverses all of your source code looking for queries during query extraction. In order for Gatsby to handle static and normal queries differently, it looks for 3 specific cases in `babel-plugin-remove-graphql-queries`:

1. JSX nodes with the name `StaticQuery`
2. Calls to functions with the name `useStaticQuery`
3. Tagged template expressions using the `gql` tag

Adding imports for page data

Code that is specific for querying the GraphQL server set up during build time is no longer relevant, and can be swapped out in exchange for the JSON data that has been extracted for each query.

The imports related to GraphQL and query declarations are changed to imports for the JSON that correspond to the query result that Gatsby found when it ran the query. Consider the following component with a static query written using the `useStaticQuery` hook:

```
import { useStaticQuery, graphql } from "gatsby"

export () => {
  const data = useStaticQuery(graphql`
      siteMetadata {
        site {
            title
        }
      }
  `)
  return (
    <h1>
        {data.siteMetadata.site.title}
    </h1>
  )
}
```

This component will have the query string removed and replaced with an import for the JSON file created and named with its specific hash. The Redux stores tracking the queries link the correct data to the page they correspond to.

The above component is rewritten like this:

```
- import { useStaticQuery, graphql } from "gatsby"
+ import dataJson from `/d/<hash>.json`

export () => {
- const data = useStaticQuery(graphql`
-     siteMetadata {
-       site {
-           title
-       }
-     }
- `)
+ const data = dataJson
```

```
  return (
    <h1>
        {data.siteMetadata.site.title}
    </h1>
  )
}
```

A page query would be updated in a similar fashion. Although the exact specifics of what has to change are different, the idea is the same:

```
- import { graphql } from "gatsby"
+ import dataJson from `/d/<hash>.json`

- export const query = graphql`
-   query HomePageQuery {
-     site {
-       siteMetadata {
-         description
-       }
-     }
-   }
- `

- export ({ data }) => {
+ export ({ data = dataJson }) => {
    return (
      <h1>
          {data.siteMetadata.site.title}
      </h1>
    )
  }
```

Gatsby will remove unnecessary imports like `useStaticQuery` and `graphql` from your pages along with the strings containing queries as well.