Gatsby lets you access your data across all sources using a single GraphQL interface.

Querying data with a Page Query

You can use the `graphql` tag to query data in the pages of your Gatsby site. This gives you access to anything included in Gatsby's data layer, such as site metadata, source plugins, images, and more.

Directions

1. Import `graphql` from `gatsby`.
2. Export a constant named `query` and set its value to be a `graphql` template with the query between two backticks.
3. Pass in `data` as a prop to the component.
4. The `data` variable holds the queried data and can be referenced in JSX to output HTML.

```
import React from "react"
// highlight-next-line
import { graphql } from "gatsby"

import Layout from "../components/layout"

// highlight-start
export const query = graphql`
  query HomePageQuery {
    site {
      siteMetadata {
        title
      }
    }
  }
`
// highlight-end

// highlight-next-line
const IndexPage = ({ data }) => (
  <Layout>
    // highlight-next-line
    <h1>{data.site.siteMetadata.title}</h1>
  </Layout>
)

export default IndexPage
```

Additional resources

- GraphQL and Gatsby: understanding the expected shape of your data
- More on querying data in pages with GraphQL
- MDN on Tagged Template Literals like the ones used in GraphQL

Querying data with the StaticQuery Component

`StaticQuery` is a component for retrieving data from Gatsby's data layer in non-page components, such as a header, navigation, or any other child component.

Directions

1. The `StaticQuery` Component requires two render props: `query` and `render`.

```
import React from "react"
import { StaticQuery, graphql } from "gatsby"

const NonPageComponent = () => (
  <StaticQuery
    query={graphql` // highlight-line
      query NonPageQuery {
        site {
          siteMetadata {
            title
          }
        }
      }
    `}
    render={(
      data // highlight-line
    ) => (
      <h1>
        Querying title from NonPageComponent with StaticQuery:
        {data.site.siteMetadata.title}
      </h1>
    )}
  />
)

export default NonPageComponent
```

2. You can now use this component as you would any other component by importing it into a larger page of JSX components and HTML markup.

Querying data with the useStaticQuery hook

Since Gatsby v2.1.0, you can use the `useStaticQuery` hook to query data with a JavaScript function instead of a component. The syntax removes the need for a `<StaticQuery>` component to wrap everything, which some people find simpler to write.

The `useStaticQuery` hook takes a GraphQL query and returns the requested data. It can be stored in a variable and used later in your JSX templates.

Prerequisites

- You'll need React and ReactDOM 16.8.0 or later (keeping Gatsby updated handles this)
- Recommended reading: the Rules of React Hooks

Directions

1. Import `useStaticQuery` and `graphql` from `gatsby` in order to use the hook query the data.

2. In the start of a stateless functional component, assign a variable to the value of `useStaticQuery` with your `graphql` query passed as an argument.

3. In the JSX code returned from your component, you can reference that variable to handle the returned data.

```
import React from "react"
import { useStaticQuery, graphql } from "gatsby" //highlight-line

const NonPageComponent = () => {
  // highlight-start
  const data = useStaticQuery(graphql`
    query NonPageQuery {
      site {
        siteMetadata {
          title
        }
      }
    }
  `)
  // highlight-end
  return (
    <h1>
      Querying title from NonPageComponent: {data.site.siteMetadata.title}{" "}
      //highlight-line
    </h1>
  )
}

export default NonPageComponent
```

Additional resources

- More on Static Query for querying data in components
- The difference between a static query and a page query
- More on the useStaticQuery hook
- Visualize your data with GraphiQL

Limiting with GraphQL

When querying for data with GraphQL, you can limit the number of results returned with a specific number. This is helpful if you only need a few pieces of data or need to paginate data.

To limit data, you'll need a Gatsby site with some nodes in the GraphQL data layer. All sites have some nodes like `allSitePage` and `sitePage` created automatically: more can be added by installing source plugins like `gatsby-source-filesystem` in `gatsby-config.js`.

Prerequisites

- A Gatsby site

Directions

1. Run `gatsby develop` to start the development server.
2. Open a tab in your browser at: `http://localhost:8000/___graphql`.
3. Add a query in the editor with the following fields on `allSitePage` to start off:

```
{
  allSitePage {
    edges {
      node {
        id
        path
      }
    }
  }
}
```

4. Add a `limit` argument to the `allSitePage` field and give it an integer value `3`.

```
{
  allSitePage(limit: 3) { // highlight-line
    edges {
      node {
        id
        path
      }
    }
  }
}
```

5. Click the play button in the GraphiQL page and the data in the `edges` field will be limited to the number specified.

Additional resources

- Learn about nodes in Gatsby's GraphQL data API
- Gatsby GraphQL reference for limiting
- Live example: