A page displaying a list of content gets longer as the amount of content grows. Pagination is the technique of spreading that content across multiple pages.

The goal of pagination is to create multiple pages (from a single template) that show a limited number of items.

Each page will query GraphQL for those specific items.

The information needed to query for those specific items (i.e. values for `limit` and `skip`) will come from the `context` that is added when creating pages in `gatsby-node`.

Example

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

export default class BlogList extends React.Component {
  render() {
    const posts = this.props.data.allMarkdownRemark.edges
    return (
      <Layout>
        {posts.map(({ node }) => {
          const title = node.frontmatter.title || node.fields.slug
          return <div key={node.fields.slug}>{title}</div>
        })}
      </Layout>
    )
  }
}

export const blogListQuery = graphql`
// highlight-start
  query blogListQuery($skip: Int!, $limit: Int!) {
    allMarkdownRemark(
      sort: { fields: [frontmatter___date], order: DESC }
      limit: $limit
      skip: $skip
    ) {
// highlight-end
      edges {
        node {
          fields {
            slug
          }
          frontmatter {
            title
          }
        }
      }
    }
```

```
  }
`
```

```
const path = require("path")
const { createFilePath } = require("gatsby-source-filesystem")

exports.createPages = async ({ graphql, actions, reporter }) => {
  const { createPage } = actions

  const result = await graphql(
    `
      {
        allMarkdownRemark(
          sort: { fields: [frontmatter___date], order: DESC }
          limit: 1000
        ) {
          edges {
            node {
              fields {
                slug
              }
            }
          }
        }
      }
    `
  )

  if (result.errors) {
    reporter.panicOnBuild(`Error while running GraphQL query.`)
    return
  }

  // ...

  // Create blog-list pages
  // highlight-start
  const posts = result.data.allMarkdownRemark.edges
  const postsPerPage = 6
  const numPages = Math.ceil(posts.length / postsPerPage)

  Array.from({ length: numPages }).forEach((_, i) => {
    createPage({
      path: i === 0 ? `/blog` : `/blog/${i + 1}`,
      component: path.resolve("./src/templates/blog-list-template.js"),
      context: {
        limit: postsPerPage,
        skip: i * postsPerPage,
        numPages,
        currentPage: i + 1,
      },
```

```
    })
  })
  // highlight-end
}

exports.onCreateNode = ({ node, actions, getNode }) => {
  const { createNodeField } = actions
  if (node.internal.type === `MarkdownRemark`) {
    const value = createFilePath({ node, getNode })
    createNodeField({
      name: `slug`,
      node,
      value,
    })
  }
}
```

The code above will create a number of pages based on the total number of posts. Each page will list `postsPerPage` (6) posts, until there are less than `postsPerPage` (6) posts left. The path for the first page is `/blog`, following pages will have a path of the form: `/blog/2`, `/blog/3`, etc.

Other resources

- Follow this step-by-step tutorial to add links to the previous/next page and the traditional page-navigation at the bottom of the page
- See gatsby-paginated-blog (demo) for an extension of the official gatsby-starter-blog with pagination in place