Gatsby Cheatsheet

A comprehensive cheat sheet for Gatsby, covering essential concepts, APIs, and commands to build performant React websites.

Gatsby Basics

Project Setup

Creating a new Gatsby site:

```
gatsby new my-gatsby-site
```

Starting the development server:

```
gatsby develop
```

Building for production:

```
gatsby build
```

Serving the production build:

```
gatsby serve
```

Cleaning the cache:

```
gatsby clean
```

Gatsby CLI help:

```
gatsby --help
```

File Structure

- `gatsby-config.js`: Main configuration file for the Gatsby site. Contains site metadata, plugins, etc.
- `gatsby-node.js`: Used for implementing Gatsby's Node APIs, such as creating pages programmatically from data.
- `gatsby-browser.js`: Used for implementing Gatsby's Browser APIs, allowing customization of the browser environment.
- `gatsby-ssr.js`: Used for implementing Gatsby's Server-Side Rendering (SSR) APIs.
- `src/pages`: Directory for React components that automatically become pages.
- `src/components`: Directory for reusable React components.
- `src/images`: Directory for static images.

Key Concepts

GraphQL	Gatsby uses GraphQL to query data. The `useStaticQuery` hook and the `graphql` tag are essential for fetching data in components.
Data Layer	Gatsby's data layer is built on GraphQL and allows you to source data from various sources like Markdown files, APIs, and databases.
Plugins	Plugins extend Gatsby's functionality, such as adding support for different file types (e.g., Markdown) or connecting to external APIs.

GraphQL Queries

Querying Data

Using `useStaticQuery` (for components):

```
import { useStaticQuery, graphql } from
'gatsby';

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

Using page queries (for pages):

```
import React from 'react';
import { graphql } from 'gatsby';

const MyPage = ({ data }) => {
  return <h1>{data.site.siteMetadata.title}
</h1>;
};

export const query = graphql`
  query {
    site {
      siteMetadata {
        title
      }
    }
  }
`;

export default MyPage;
```

Common GraphQL Queries

Site Metadata

```
query {
  site {
    siteMetadata {
      title
      description
      author
    }
  }
}
```

All Markdown Remark

```
query {
  allMarkdownRemark {
    edges {
      node {
        frontmatter {
          title
          date
        }
        excerpt
      }
    }
  }
}
```

File Query

```
query {
  allFile {
    edges {
      node {
        name
        relativePath
        size
      }
    }
  }
}
```

Image Sharp

```
query {
  allImageSharp {
    edges {
      node {
        fluid {
          src
          base64
          aspectRatio
        }
      }
    }
  }
}
```

Filtering and Sorting

Filtering Use the `filter` argument to filter results.

```
query {
  allMarkdownRemark(filter:
{frontmatter: {tags: {in:
["gatsby"]}}}) {
    edges {
      node {
        frontmatter {
          title
        }
      }
    }
  }
}
```

Sorting Use the `sort` argument to sort results.

```
query {
  allMarkdownRemark(sort: {fields:
frontmatter___date, order: DESC}) {
    edges {
      node {
        frontmatter {
          title
          date
        }
      }
    }
  }
}
```

Gatsby Plugins

Essential Plugins

- `gatsby-plugin-image`: For optimized image handling. Essential for performant websites.
- `gatsby-source-filesystem`: For sourcing data from the file system (e.g., Markdown files, images).
- `gatsby-transformer-remark`: For transforming Markdown files into HTML.
- `gatsby-plugin-sharp`: For image processing using Sharp.
- `gatsby-transformer-sharp`: Enables querying optimized images with GraphQL.
- `gatsby-plugin-react-helmet`: For managing document head metadata (e.g., title, meta tags).
- `gatsby-plugin-offline`: For enabling offline support via service workers.

Configuring Plugins

Plugins are configured in `gatsby-config.js`.

```
module.exports = {
  plugins: [
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `images`,
        path: `${__dirname}/src/images`,
      },
    },
    `gatsby-transformer-sharp`,
    `gatsby-plugin-sharp`,
  ],
};
```

Using `gatsby-plugin-image`

StaticImage Component

```
import { StaticImage } from "gatsby-plugin-image"

<StaticImage
src="../images/gatsby-
icon.png" alt="Gatsby icon"
/>
```

GatsbyImage Component

```
import { GatsbyImage,
getImage } from "gatsby-
plugin-image"

const MyComponent = ({ data
}) => {
  const image =
getImage(data.markdownRemark
.frontmatter.image)
  return (
    <GatsbyImage image=
{image} alt=
{data.markdownRemark.frontma
tter.title} />
  )
}

export const query =
graphql`
  query {
    markdownRemark(slug: {
eq: $slug }) {
      frontmatter {
        title
        image {
          childImageSharp {
            gatsbyImageData
          }
        }
      }
    }
  }
`
```

Gatsby Advanced

Gatsby Node API

`onCreateNode`: Called when a new node is created. Useful for transforming node data.

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

`createPages`: Called to create pages dynamically. Essential for blogs and content-heavy sites.

```
const path = require(`path`)

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

  const blogPostTemplate =
path.resolve(`src/templates/blog-post.js`)

  const result = await graphql(`
    query {
      allMarkdownRemark {
        edges {
          node {
            fields {
              slug
            }
          }
        }
      }
    }
  `)


result.data.allMarkdownRemark.edges.forEach(ed
ge => {
    createPage({
      path: edge.node.fields.slug,
      component: blogPostTemplate,
      context: {
        slug: edge.node.fields.slug,
      },
    })
  })
}
```

`onCreateWebpackConfig`: Customize Gatsby's webpack config.

```
exports.onCreateWebpackConfig = ({ actions })
=> {
  actions.setWebpackConfig({
    resolve: {
      alias: {
        "@components": path.resolve(__dirname,
"src/components"),
      },
    },
  })
}
```

Deploying Gatsby Sites

Gatsby sites can be deployed to various platforms like Netlify, Vercel, GitHub Pages, and AWS Amplify. Netlify and Vercel offer excellent built-in support for Gatsby.

Deploying to Netlify:

1. Build the site: `gatsby build`
2. Install Netlify CLI: `npm install -g netlify-cli`
3. Deploy: `netlify deploy --prod`

Deploying to Vercel:

1. Build the site: `gatsby build`
2. Install Vercel CLI: `npm install -g vercel`
3. Deploy: `vercel`

SEO Considerations

Meta Descriptions	Use `gatsby-plugin-react-helmet` to add meta descriptions to your pages. Ensure each page has a unique and relevant description.
Structured Data	Implement structured data markup (Schema.org) to help search engines understand your content.
Image Optimization	Use `gatsby-plugin-image` and `gatsby-transformer-sharp` to optimize images for the web. Properly sized and compressed images improve page load times.