BÁO CÁO TÌM HIỂU GATSBYJS

I. Giới thiệu chung

Gatsbyjs là 1 framework cho phép build nhanh chóng 1 static site React với data từ bất kỳ nguồn nào (markdown, JSON, database, …).

II. Set up môi trường dev

- Cài đặt Nodejs
- Cài đặt Git
- Cài đặt Gatsby CLI bằng lệnh:

    npm install --global gatsby-cli

III. Bắt đầu với một project mới

Vào terminal, gõ lần lượt các lệnh:

gatsby new hello-world

cd hello-world

gatsby develop

- Trang web sẽ được start ở địa chỉ http://localhost:8000
- Để tạo 1 production build:

    gatsby build

IV. Cấu trúc của một gatsby project

Cấu trúc của một project như sau:

```
|-- /.cache
|-- /node_modules
|-- /public
|-- /src
        |-- /components
        |-- /pages
|-- gatsby-config.js
|-- gatsby-node.js
|-- gatsby-ssr.js
|-- gatsby-browser.js
```

- /.cache: là 1 internal cache được tự động tạo bởi gatsby
- /node_modules: thư mục chứa các node module
- /public: thư mục chứa output của quá trình build
- /src: thư mục chứa toàn bộ code liên quan đến front-end của site. Bên trong có thể bao gồm các thư mục như components, pages, ultis, ...
- gatsby-config.js: file main config cho site
- gatsby-node.js: nơi implement các node api
- gatsby-browser.js: nơi implement các browser api
- gatsby-ssr: nơi implenment các server side rendering api

V. Các thao tác chính

1. Create pages và link pages

a. Create pages

Có 2 cách tạo page: định nghĩa React component trong /src/pages hoặc sử dụng API createpages của gatsby.

- Sử dụng React component:

```
import React from 'react'

const HomePage = () => (
    <div>This is home page</div>
)

export default HomePage
```

- Sử dụng createpages API (chi tiết ở phần ví dụ transform data).

b. Link pages

Link giữa các page sử dụng <Link /> của gastby

file *src/pages/index.js*

```
import React from 'react'
import { Link } from 'gatsby'


const HomePage = () => (
    <div>
        <div>This is home page</div>
        <Link to="/page-2">Go to page 2</Link>
    </div>
)

export default HomePage
```

file *src/pages/page-2.js*

```
import React from 'react'
import { Link } from 'gatsby'

const SecondPage  = () => (
    <div>
        <div>This is page 2</div>
        <Link to="/">Go to home page</Link>
    </div>
)

export default SecondPage
```

2. **Create components:** Định nghĩa các React component trong thư mục /src/components

```
import React from 'react'
import PropTypes from 'prop-types'

const Header = ({siteTitle}) => (
    <div>
        <h1>{siteTitle}</h1>
    </div>
)

Header.propTypes = {
    siteTitle: PropTypes.string,
}

Header.defaultProps = {
    siteTitle: '',
}

export default Header
```

3. **Create styles**

Gatsby hỗ trợ hầu hết các styles thông qua các plugin.

a. SASS

- Sử dụng plugin *gatsby-plugin-sass:*

npm install --save node-sass gatsby-plugin-sass

- Include plugin vào file *gatsby-config.js*

```
// in gatsby-config.js
plugins: [`gatsby-plugin-sass`]
```

- Viết stylesheet SASS/SCSS và import tương tự trong React.

b. CSS Module

Sử dụng tương tự như React.

file *src/components/container.js*

```
import React from "react"
import containerStyles from "./container.module.css"

export default ({ children }) => (
  <div className={containerStyles.container}>{children}</div>
)
```

file *src/components/container.module.css*

```
.container {
  margin: 3rem auto;
  max-width: 600px;
}
```

4. **Querying, sourcing và transforming data**

a. Querying data

- Gatsby query data dựa trên ngôn ngữ truy vấn GraphQL. Thử truy vấn title site tại địa chỉ http://localhost:8000/___graphql

```
{
  site {
    siteMetadata {
      title
    }
  }
}
```

Kết quả thu được là:

```
{
  "data": {
    "site": {
      "siteMetadata": {
        "title": "Gatsby Default Starter"
      }
    }
  }
}
```

- Sử dụng graphql trong page:

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

const HomePage = ({data}) => {
    return (
        <div>
            {data.site.siteMetadata.title}
        </div>
    )
}

export const query = graphql`
    query HomePageQuery {
        site {
            siteMetadata {
                title
            }
        }
    }
`
export default HomePage
```

- Sau khi thực hiện xong truy vấn query, dữ liệu của truy vấn sẽ được tự động chứa trong data prop.
- Sử dụng <StaticQuery />

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

const HomePage = () => (
    <StaticQuery
        query={graphql`
            query HomePageQuery {
                site {
                    siteMetadata {
                        title
                    }
                }
            }
        `}
        render={data => (
            <div>
                {data.site.siteMetadata.title}
            </div>
        )}
    />
)

export default HomePage
```

b. Sourcing data

- Source plugin cho phép fetch data từ nhiều nguồn khác nhau
- Vd: Sử dụng plugin *gatsby-source-filesystem* để fetch data trong file local

npm install --save gatsby-source-filesystem

Sau đó thêm vào file *gatsby-config.js* trong plugins:

```
    {
      resolve: `gatsby-source-filesystem`,
      options: {
        name: `src`,
        path: `${__dirname}/src/`,
      },
    },
```

Thử truy vấn bằng graphql trên http://localhost:8000/___graphql

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

Kết quả thu được là:


```
{
  "data": {
    "allFile": {
      "edges": [
        {
          "node": {
            "name": "header",
            "relativePath": "components/header.js"
          }
        },
        {
          "node": {
            "name": "image",
            "relativePath": "components/image.js"
          }
        },
        {
          "node": {
            "name": "layout",
            "relativePath": "components/layout.css"
          }
        },
        {
          "node": {
            "name": "layout",
            "relativePath": "components/layout.js"
          }
        },
        {
          "node": {
            "name": "seo",
            "relativePath": "components/seo.js"
          }
        },
```


- Ngoài ra còn có các plugin khác như *gatsby-source-wordpress*, *gatsby-source-mongodb*,…

c. Transforming data

Transform data sử dụng các plugin để xử lý dữ liệu được fetch từ sourcing data trở nên dễ sử dụng hơn (như chuyển JSON thành 1 js object hay markdown thành HTML).

Ví dụ: Transform data markdown được fetch từ local file và tự động generate site cho 1 blog.

link project github: https://github.com/gatsbyjs/gatsby-starter-blog

- Muốn fetch được data từ file markdown, cần phải cài plugin *gatsby-transformer-remark*

  npm install --save gatsby-transformer-remark

- Muốn access dữ liệu local thông qua graphql, cần phải cài plugin g*atsby-source-filesystem*

  npm install --save gatsby-source-filesystem

- Sau đó thêm 2 plugin trên vào file *gatsby-config.js* mục plugin

```
{
  resolve: `gatsby-source-filesystem`,
  options: {
    path: `${__dirname}/content/blog`,
    name: 'blog',
  },
},
```

```
{
  resolve: `gatsby-transformer-remark`,
  options: {
    plugins: [
      {
        resolve: `gatsby-remark-images`,
        options: {
          maxWidth: 590,
        },
      },
      {
        resolve: `gatsby-remark-responsive-iframe`,
        options: {
          wrapperStyle: `margin-bottom: 1.0725rem`,
        },
      },
      'gatsby-remark-prismjs',
      'gatsby-remark-copy-linked-files',
      'gatsby-remark-smartypants',
    ],
  },
},
```

- Trong option path của plugin *gatsby-source-filesystem* là */content/blog* vì những file markdown sẽ được chứa trong thư mục này.
- Trong file */src/pages/index.js* khai báo 1 query:

```
export const pageQuery = graphql`
  query {
    site {
      siteMetadata {
        title
      }
    }
    allMarkdownRemark(sort: { fields: [frontmatter___date], order: DESC }) {
      edges {
        node {
          excerpt
          fields {
            slug
          }
          frontmatter {
            date(formatString: "MMMM DD, YYYY")
            title
          }
        }
      }
    }
  }
`
```

- Thử query tại localhost:8000/___graphql thu được kết quả:


```
{
  "data": {
    "site": {
      "siteMetadata": {
        "title": "Gatsby Starter Blog"
      }
    },
    "allMarkdownRemark": {
      "edges": [
        {
          "node": {
            "excerpt": "Far far away, behind the word mountains, far from the
countries Vokalia and\nConsonantia, there live the blind texts. Separated
they live in…",
            "fields": {
              "slug": "/hi-folks/"
            },
            "frontmatter": {
              "date": "May 28, 2015",
              "title": "New Beginnings"
            }
          }
        },
        {
          "node": {
            "excerpt": "Wow! I love blogging so much already. Did you know
that “despite its name, salted duck eggs can also be made from\nchicken eggs,
though the…",
            "fields": {
              "slug": "/my-second-post/"
            },
            "frontmatter": {
              "date": "May 06, 2015",
              "title": "My Second Post!"
            }
          }
        },
        {
```

- Dựng component BlogIndex trong file *src/pages/index.js*. Trong page index sẽ chứa toàn bộ danh sách các bài viết và link tới mỗi bài viết.

```
class BlogIndex extends React.Component {
  render() {
    const { data } = this.props;
    const siteTitle = data.site.siteMetadata.title
    const posts = data.allMarkdownRemark.edges

    return (
      <Layout location={this.props.location} title={siteTitle}>
        <SEO title="All posts" keywords={['blog', 'gatsby', 'javascript', 'react']} />
        <Bio />
        {posts.map(({ node }) => {
          const title = node.frontmatter.title || node.fields.slug
          return (
            <div key={node.fields.slug}>
              <h3
                style={{
                  marginBottom: rhythm(1 / 4),
                }}
              >
                <Link style={{ boxShadow: 'none' }} to={node.fields.slug}>
                  {title}
                </Link>
              </h3>
              <small>{node.frontmatter.date}</small>
              <p dangerouslySetInnerHTML={{ __html: node.excerpt }} />
            </div>
          )
        })}
      </Layout>
    )
  }
}

export default BlogIndex
```

- Để link tới mỗi bài viết cần phải tạo slug cho mỗi bài viết. Trong file */gatsby-node.js* thêm vào onCreateNode.
- onCreateNode là 1 Gatsby API được gọi khi mỗi node được thêm mới hoặc chỉnh sửa (ở đây node là file .md)
- Sau khi tạo node, bước tiếp theo sử dụng createPages để tạo page tương ứng với mỗi file .md, path chính là slug của mỗi node, template của mỗi page được tạo trong file */src/templates/blog-post.js*.

File */gatsby-node.js*

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

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

  return new Promise((resolve, reject) => {
    const blogPost = path.resolve('./src/templates/blog-post.js')
    resolve(
      graphql(
        `
          {
            allMarkdownRemark(sort: { fields: [frontmatter___date], order: DESC }, limit:
              edges {
                node {
                  fields {
                    slug
                  }
                  frontmatter {
                    title
                  }
                }
              }
            }
          }
        `
      ).then(result => {
        if (result.errors) {
          console.log(result.errors)
          reject(result.errors)
        }

        // Create blog posts pages.
        const posts = result.data.allMarkdownRemark.edges;
```

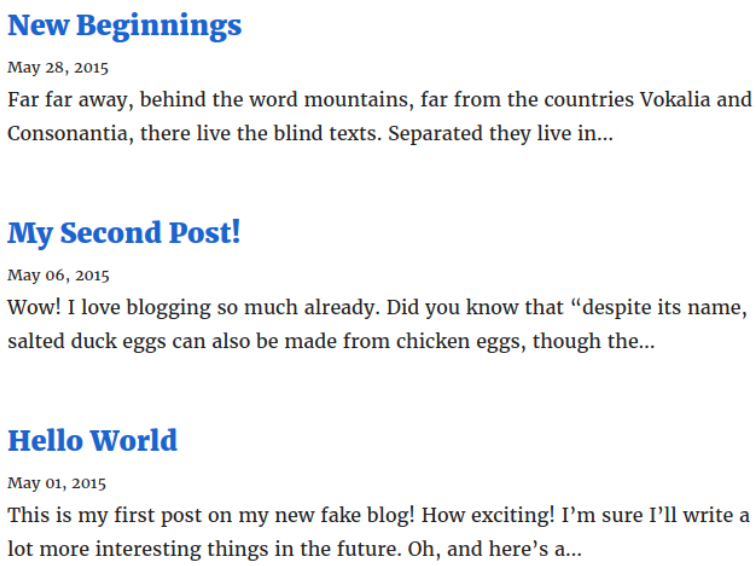
```

        posts.forEach((post, index) => {
          const previous = index === posts.length - 1 ? null : posts[index + 1].node;
          const next = index === 0 ? null : posts[index - 1].node;

          createPage({
            path: post.node.fields.slug,
            component: blogPost,
            context: {
              slug: post.node.fields.slug,
              previous,
              next,
            },
          })
        })
      })
    )
  })
}

exports.onCreateNode = ({ node, actions, getNode }) => {
  const { createNodeField } = actions

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

File */src/templates/blog-post.js*

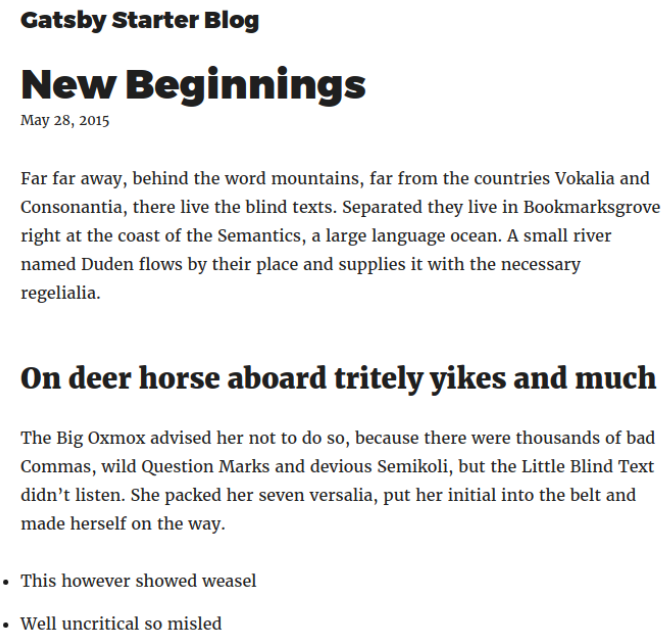
```
class BlogPostTemplate extends React.Component {
  render() {
    const post = this.props.data.markdownRemark
    const siteTitle = this.props.data.site.siteMetadata.title
    const { previous, next } = this.props.pageContext

    return (
      <Layout location={this.props.location} title={siteTitle}>
        <SEO title={post.frontmatter.title} description={post.excerpt} />
        <h1>{post.frontmatter.title}</h1>
        <p
          style={{
            ...scale(-1 / 5),
            display: 'block',
            marginBottom: rhythm(1),
            marginTop: rhythm(-1),
          }}
        >
          {post.frontmatter.date}
        </p>
        <div dangerouslySetInnerHTML={{ __html: post.html }} />
        <hr
          style={{
            marginBottom: rhythm(1),
          }}
        />
        <Bio />

        <ul
          style={{
            display: 'flex',
            flexWrap: 'wrap',
            justifyContent: 'space-between',
            listStyle: 'none',
            padding: 0,
          }}
        >
          <li>
            {
              previous &&
              <Link to={previous.fields.slug} rel="prev">
                ← {previous.frontmatter.title}
              </Link>
            }
          </li>
          <li>
            {
              next &&
              <Link to={next.fields.slug} rel="next">
                {next.frontmatter.title} →
              </Link>
            }
          </li>
        </ul>
      </Layout>
    )
  }
}

export default BlogPostTemplate
```

```
export const pageQuery = graphql`
  query BlogPostBySlug($slug: String!) {
    site {
      siteMetadata {
        title
        author
      }
    }
    markdownRemark(fields: { slug: { eq: $slug } }) {
      id
      excerpt(pruneLength: 160)
      html
      frontmatter {
        title
        date(formatString: "MMMM DD, YYYY")
      }
    }
  }
`
```

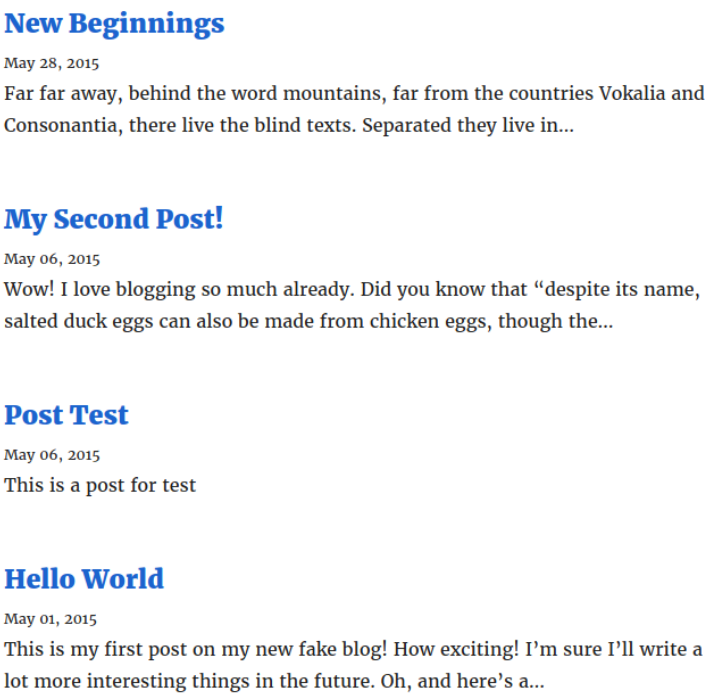
- Sau khi thực hiện xong các bước trên, gatsby sẽ tự động generate site dựa trên các file data markdown và template page đã code sẵn. Giao diện của trang chủ sau khi run site:



- Thử click vào 1 post:



- Thử thêm 1 file *test-post.md* mới và reload lại trang:



Thử click vào post test:

Kết luận: Gatsby pros and cons

- Pros:
  + Trang tạo bằng gatsby chạy khá nhanh, chỉ cần load trang web 1 lần là có thể chạy được tới tất cả các path. Khi thêm dữ liệu không cần sửa lại code.
  + Định tuyến dễ dàng.
  + Hỗ trợ nhiều plugin.
  + Khả năng tái sử dụng code tốt, khi muốn thay đổi trang chỉ cần thay đổi template hoặc thay đổi query.
  + Không yêu cầu server nên có thể deploy ở nhiều nền tảng miễn phí.
- Cons:
  + Gatsby chuyên dùng để tạo static site => không có các xử lý tương tác với site.
  + Yêu cầu có kiến thức về HTML, CSS, React, GraphQL.

Tham khảo

Tài liệu chính thức của GatsbyJS

https://www.gatsbyjs.org/docs/

Tạo trang blog GatsbyJS

https://completejavascript.com/tao-trang-blog-voi-gatsby-js/

Source code template blog từ GatsbyJS

https://github.com/gatsbyjs/gatsby-starter-blog