Next.js & Netlify-cms

Blog site build

David Richards - February 2021

jamstack?

- Javascript, API (application program interface), Markup (html, css)
- Uses “someone else’s” compute resources
  - No need to run backend server (AWS, Azure, GCP, DigitalOcean, Heroku)
  - Configure for automatic scaling (up or down demand)

Reactjs

- Reactjs is often referred to as a component library
  - Since it has a set of rules it probably should be considered a framework
  - React apps are built from code components (button, navbar, page)
  - React is very un-opinionated for constructing projects allowing use of many different methods & libraries (e.g. routing: react-router, reach-router)
  - Flexibility can result in complexity of choices
  - Functional components replacing class components
  - Permits different styling methods
    - CSS-modules, styled-components, global styling, css-in-js

Nextjs

- Framework around React
  - Static Site Generation
    - Creates complete code bundle delivered to browser
      - No interaction except navigation
  - Server Side Rendered
    - Builds ahead of browser request - complete page
    - Client Side rendering delivers initial HTML then loads javascript (hydrates)
    - Hybrid
      - Static (product page), Server (Cart)

Nextjs Benefits

- SEO - Rendered pages actually contain all html
- Code splitting
  - Only loads JS & CSS for viewed page
- HMR - hot module replacement - only reloads changed files
- Built-in routing - just add page
- Built-in CSS support
- API routes for API endpoints with Serverless Functions

SEO Demo

- Open VSCode
- Launch terminal (ctrl + `)
- If using alias go to projects folder/directory otherwise navigate to where you want to create projects (cd folder && mkdir)
- Create working directory (‘mkdir cftw’)
- >npx create-react-app react_example
  - >cd react_example
  - >npm install
  - Will open web page at localhost:3000
  - Inspect page with Dev Tools

SEO Demo

- >npx create-next-app next_example
  - >cd next_example
  - >code .
  - Edit package.json
    - “dev”: “next dev -p 3001”

  - In new window open terminal (ctrl + `)
  - >npm run dev
  - Will open web page at localhost:3001
  -

Mods

- npm install - -save-dev frontmatter-markdown-loader
  - Alt: npm i -D frontmatter-markdown-loader
- npm install - -save-dev @babel/core @babel/preset-react
- mkdir content
- touch content/home.md
- package.json
  - under scripts
    - “export”: “next build && next export”

content/horses.md

```
---
title: Great Horses
date: 2021-02-22T23:31:20:551Z
horses:
  - name: Excepzional
    description: 'Excepzional is an experienced competitor, Polish-Crabbet,
grey Arabian stallion'
    character: Intelligent, opionated.
  - name: Chanz
    description: 'Chanz is a chestnut colored Arabian out of the Rushcreek
line'
    character: Good sense of humor, curious, easily spooked, challenge to stay
in saddle
  - name: Aris
    description: 'Aris is a grey Arabian and half brother to Chanz'
    character: Earnest, cautious.
---

This is the profile page for some of my horses.

Also, this features Next.js as a React.js framework
```

next.config.js

```
module.exports = {
  webpack: (cfg) => {
    cfg.module.rules.push({
      test: /\.md$/,
      loader: 'frontmatter-markdown-loader',
      options: { mode: ['react-component'] },
    })
    return cfg
  },
}
```

index.js

```
import Head from 'next/head'
import { attributes, react as HorseContent } from '../content/horses.md'
export default function Home() {
  let { title, horses } = attributes
  return (
    <>
      <Head>
        <script src='https://identity.netlify.com/v1/netlify-identity-widget.js'></script>
        <title>Horses</title> //! modify
      </Head>
      <article>
        <h1>{title}</h1>
        <HorseContent />
        <ul>
          {horses.map((horse, k) => (
            <li key={k}>
              <h2>{horse.name}</h2>
              <p>{horse.description}</p>
              <p>{horse.character}</p>
            </li>
          ))}
        </ul>
      </article>
    </>
  )
}
```

Add netlify-cms

- create directory in public
  - mkdir -p public/admin
- create two files
  - index.html
  - config.yml

public/admin/index.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Content Manager</title>
    <script src="https://identity.netlify.com/v1/netlify-identity-widget.js"></script>
  </head>
  <body>
    <!-- Include the script that builds the page and powers Netlify CMS -->
    <script src="https://unpkg.com/netlify-cms@^2.0.0/dist/netlify-cms.js"></script>
  </body>
</html>
```

public/admin/config.yml

```yaml
backend:
  name: test-repo
  branch: master
media_folder: public/img
public_folder: img
collections:
  - name: 'pages'
    label: 'Pages'
    files:
      - label: 'Home'
        name: 'home'
        file: 'content/horses.md'
        fields:
          - { label: 'Title', name: 'title', widget: 'string' }
          - { label: 'Publish Date', name: 'date', widget: 'datetime' }
          - { label: 'Body', name: 'body', widget: 'markdown' }
          - label: 'Horses'
            name: 'horses'
            widget: list
            fields:
              - { label: 'Name', name: 'name', widget: 'string' }
              - { label: 'Description', name: 'description', widget: 'text' }
              - { label: 'Character', name: 'character', widget: 'text' }
```

Github

- Create account
- Create Github repo
- Link local project to remote repo

Netlify

- Select: New site from git
- Select Github repo
- Build options:
  - build command: npm run export
  - publish directory: out

Netlify

- Settings > identity > enable identity services
- Registration preferences > Open / Invite
- External providers > select method (github, google)
- Services > Git Gateway > enable
- Authenticates with your github account

Component Styling

- Global styling using stylesheet
- CSS modules (specific to a component)
  - Avoids classname collisions
- Styled components (styling embedded in component) isolates style
- JSX - Javascript as styling (embedded in component) isolates style

Global CSS

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
    <link rel="stylesheet" href="./styles/buttonGlobal.css">
  </head>
  <body>
    <button class="button" type="button">Submit</button>
  </body>
</html>
```

```css
.button {
  display: inline-block;
  background-color: #7b38d8;
  border-radius: 10px;
  border: 4px double #cccccc;
  color: #eeeeee;
  text-align: center;
  font-size: 28px;
  padding: 20px;
  width: 200px;
  margin: 5px;
}
```

Nav.module.css

```jsx
import navStyles from '../styles/Nav.module.css'
const Nav = () => {
  return (
    <nav className={navStyles.nav}>
      <ul>
        <li>
          <Link href='/'>Home</Link>
        </li>
        <li>
          <Link href='/about'>About</Link>
        </li>
      </ul>
    </nav>
  )}
```

```css
.nav {
  height: 50px;
  padding: 10px;
  background: #000;
  color: #fff;
  display: flex;
  align-items: center;
  justify-content: flex-start;
}

.nav ul {
  display: flex;
  justify-content: center;
  align-items: center;
  list-style: none;
}

.nav ul li a {
  margin: 5px 15px;
}
```

Styled Component


```
const Button = styled.a`
display: inline-block;
padding: 0.5rem 0;
margin: 0.5rem 1rem;
width: 11rem;
background: transparent;
color: white;
`
```

Tagged Template Literal

```
render(
<div>
  <Button
    href="https://github.com/styled-components/styled-components"
    primary
  >
    GitHub
  </Button>

  <Button as={Link} href="/docs">
    Documentation
  </Button>
</div>
)
```

JSX Styling

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

```
function getGreeting(user) {
  if (user) {
    return <h1>Hello, {formatName(user)}!</h1>;
  }
  return <h1>Hello, Stranger.</h1>;
}
```