MDX Remote Example

This example shows how a simple blog might be built using the next-mdx-remote library, which allows mdx content to be loaded via `getStaticProps` or `getServerSideProps`. The mdx content is loaded from a local folder, but it could be loaded from a database or anywhere else.

The example also showcases next-remote-watch, a library that allows next.js to watch files outside the `pages` folder that are not explicitly imported, which enables the mdx content here to trigger a live reload on change.

Since `next-remote-watch` uses undocumented Next.js APIs, it doesn't replace the default `dev` script for this example. To use it, run `npm run dev:watch` or `yarn dev:watch`.

Deploy your own

Deploy the example using Vercel:

How to use

Execute `create-next-app` with npm or Yarn to bootstrap the example:

```
npx create-next-app --example with-mdx-remote with-mdx-remote-app
# or
yarn create next-app --example with-mdx-remote with-mdx-remote-app
# or
pnpm create next-app -- --example with-mdx-remote with-mdx-remote-app
```

Deploy it to the cloud with Vercel (Documentation).

Notes

Conditional custom components

When using `next-mdx-remote`, you can pass custom components to the MDX renderer. However, some pages/MDX files might use components that are used infrequently, or only on a single page. To avoid loading those components on every MDX page, you can use `next/dynamic` to conditionally load them.

For example, here's how you can change `getStaticProps` to pass a list of component names, checking the names in the page render function to see which components need to be dynamically loaded.

```
import dynamic from 'next/dynamic'
import Test from '../components/test'

const SomeHeavyComponent = dynamic(() => import('SomeHeavyComponent'))

const defaultComponents = { Test }
```

```
export function SomePage({ mdxSource, componentNames }) {
  const components = {
    ...defaultComponents,
    SomeHeavyComponent: componentNames.includes('SomeHeavyComponent')
      ? SomeHeavyComponent
      : null,
  }

  return <MDXRemote {...mdxSource} components={components} />
}

export async function getStaticProps() {
  const source = `---
  title: Conditional custom components
  ---

  Some **mdx** text, with a default component <Test name={title}/> and a Heavy
component <SomeHeavyComponent />
  `

  const { content, data } = matter(source)

  const componentNames = [
    /<SomeHeavyComponent/.test(content) ? 'SomeHeavyComponent' : null,
  ].filter(Boolean)

  const mdxSource = await serialize(content)

  return {
    props: {
      mdxSource,
      componentNames,
    },
  }
}
```