Example app with React Intl

This example app shows how to integrate React Intl with Next.js.

How to use

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

```
npx create-next-app --example with-react-intl with-react-intl-app
# or
yarn create next-app --example with-react-intl with-react-intl-app
# or
pnpm create next-app -- --example with-react-intl with-react-intl-app
```

Deploy it to the cloud with Vercel (Documentation).

Features of this example app

- React Intl integration with custom App component
- `<IntlProvider>` creation with `locale`, `messages` props
- Default message extraction via `@formatjs/cli` integration
- Pre-compile messages into AST with `babel-plugin-formatjs` for performance
- Translation management

Translation Management

This app stores translations and default strings in the `lang/` dir. The default messages (`en.json` in this example app) is also generated by the following script.

```
$ npm run i18n:extract
```

This file can then be sent to a translation service to perform localization for the other locales the app should support.

The translated messages files that exist at `lang/*.json` are only used during production, and are automatically provided to the `<IntlProvider>`. During development the `defaultMessage`s defined in the source code are used. To prepare the example app for localization and production run the build script and start the server in production mode:

```
$ npm run build
$ npm start
```

You can then switch your browser's language preferences to German or French and refresh the page to see the UI update accordingly.