Use create-react-app to Start a Decoupled React Application

Last updated March 31, 2020

Theming Module Development 8.9.x/9.0.x

Using create-react-app we can scaffold a stand-alone React application with boilerplate configuration and organization already in place. It's a great way to get started using React, as well as the ecosystem of associated tools like Webpack, Jest, and Babel. After creating the scaffolding, we'll port the code we wrote in previous tutorials to the new structure.

In this tutorial we'll:

- Use `create-react-app` to scaffold a new React project
- Refactor existing code into the organizational structure used by `create-react-app`
- Confirm that our code runs

By the end of this tutorial, you should know what `create-react-app` is and how to get started using it.

Goal

Use `create-react-app` to set up a boilerplate React application with hot loading and build tools already configured.

Prerequisites

- NodeJS version 8.9.1 or higher installed. (We recommend using `nvm` to install NodeJS.)
- Drupal 8.5.x+ installed, with some *article* nodes created, JSON:API module installed and enabled, and configured to allow CORS. Note: If you are using Drupal 8.7 or higher installed, JSON:API is already included.

What is `create-react-app`?

The `create-react-app` application makes it easier to start a React project that follows current best practices with regard to use of tools like Webpack and Babel, and organization of code within the project. Prior to `create-react-app` developers needed to manually create a lot of boilerplate configuration to get started. Now you can let `create-react-app` do it for you. The majority of what it sets up isn't required, but is considered best practice. In the past it often made learning React more confusing because it's hard to distinguish which parts of a project were React, and which parts were just other tools or organizational best practices.

What do you get? A boilerplate React project with configuration in place for Webpack and Babel, a built-in development server that supports hot reloading, a testing framework, a suggested pattern for organizing your code, and a lot more.

If you're familiar with Drupal console it's like using the `generate` commands that it provides to scaffold a module or theme.

You can also use your React framework of choice to scaffold your application -- be it create-react-app, Next.js, or something else that hasn't been invented yet.

Overview

Our goal is to focus more on teaching you how to integrate a fully decoupled React application with Drupal, and less on the React code itself. To that end we'll use code that we wrote in previous tutorials and update it to work as part of a decoupled application. Everything you learn here will apply just as well to whatever custom React code you might be working on.

We're going to:

- Use `create-react-app` to start a new project.
- Migrate our components from previous tutorials that are part of a Drupal theme into the new application.
- Run the hot-loading development server.
- Create a production-ready build.
- Optionally, connect the build code to Drupal via an asset library.
- Optionally, modify the `create-react-app`'s Webpack configuration and scripts so that we can easily view the content in Drupal.

1 Scaffold a project with `create-react-app`

- In a Terminal, navigate to the directory on your machine where you would like your React application to live, e.g. `cd ~/Sites/decoupled-drupal/react`
- Ensure you have NodeJS version 6 or higher installed via `node --version`. TIP: use `nvm` to manage your node versions, e.g. `nvm install --lts; npm use --lts`. `npm` and `npx` are included when you install NodeJS.
- Run `npx create-react-app app` to create a new `create-react-app` application in a folder called `app`. This will scaffold a basic React application that includes a bunch of standard defaults for file organization and tooling related to React app development. Note the use of `npx` and not `npm`. You can use a different name for the directory than *app/*.
- `cd app`
- `yarn start` (If you don't have `yarn` installed you can install it, or use `npm run start` instead.) When this is finished, a new browser window should open, running your React application. Try making a change to *app/src/App.js*. It should update the interface in the browser automatically via hot reloading.
- Optionally, run `yarn build` to create code that is ready to deploy to production.

If you have trouble, the official documentation for `create-react-app` is quite good.

2 Get the example code

If you're just starting the project now you can grab the completed example code from previous tutorials here.

If you've been following along and already written an example application as part of a Drupal theme or module you can copy the *js/src/components* directory, and *js/src/utils* directory from your Drupal theme into the *src/* directory of your new application.

Alternatively, create an empty *src/components* directory; this is where your custom React components will live.

3 Update *src/App.js*

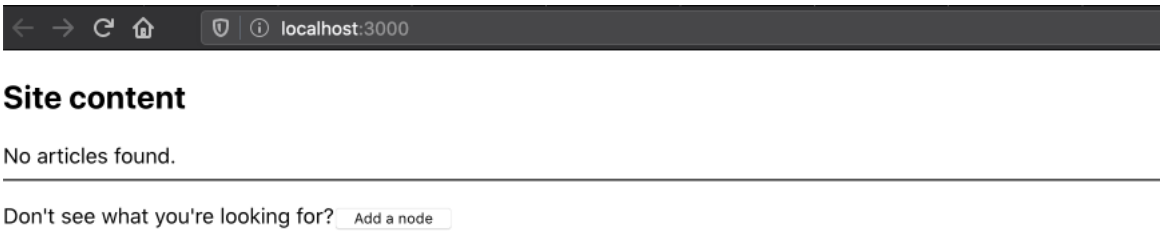
Edit the *src/App.js* file created by create-react-app, and use the `NodeReadWrite` component, or your own custom component.

Example *src/App.js*:

```
import React from 'react';
import './App.css';
import NodeReadWrite from './components/NodeReadWrite';

function App() {
  return (
    <NodeReadWrite />
  );
}

export default App;
```



At this point your application should load, but may not be able to `GET` data from, or `POST` data to, Drupal. This is a result of the code being fully decoupled and no longer integrated into Drupal directly.

When making a `GET` request to Drupal you might see an error like the following, indicating that Drupal needs to be configured to support CORS (cross-origin) requests:

```
Cross-Origin Request Blocked: The Same Origin Policy disallows reading the remote resource at https://drupal.ddev.site/jsonapi/node/article
```

Additionally, if you try to add a new article via a `POST` request, that will fail with an authorization error. We will need to set up authentication and modify our `fetch` requests to get this working. Our current example code assumes it has access to Drupal's session cookie via the browser. But since we're now running a fully decoupled app, from a different domain, that won't work.

Integrate `create-react-app` code with a Drupal theme or module.

A previous version of this tutorial contained instructions on how to use a create-react-app codebase embedded in a Drupal theme. We've since decided to stop recommending this approach since it's fragile and error prone. We've updated Connect React to a Drupal Theme or Module with information about how to configure build tools like Webpack in the context of a Drupal theme or module.

If you're still interested in using create-react-app in this way, check out the example code in this GitHub project. The use of `hook_library_info_alter()` seems to be the most robust way to go about including create-react-app generated bundles as Drupal asset libraries.

Recap

In this tutorial we used `create-react-app` to scaffold a new React application. Then we refactored our existing code from previous tutorials into the recommended organizational structure. Finally, we started the application to make sure that it runs -- noting that it doesn't fully work yet.

Further your understanding

- Take some time to explore the files that the `create-react-app` command generated. What's in the */public* folder? Can you figure out what each of the files does?
- Check out the Create React App documentation

Additional resources

- Create React App (github.com)
- Yarn installation instructions (yarnpakg.com)
- Creating a React App (codecademy.com)
- Learning React With Create-React-App (Part 1) (medium.com)
- Webpack (webpack.js.org)
- JavaScript `import` (developer.mozilla.org)
- JavaScript `export` (developer.mozilla.org)

Was this helpful? Yes No

◀ Previous tutorial

Next tutorial ▶

Get Started Using React and Drupal Together

1 Introduction to React and Drupal Free

2 React Basics

3 Decoupled vs. Progressively Decoupled

4 Connect React to a Drupal Theme or Module Free

5 Create a React Component

6 Add Webpack Hot Module Replacement (HMR) to a Drupal Theme

7 Retrieve Data from an API with React

8 Use React to List Content from Drupal

9 Create, Update, and Delete Drupal Content with JavaScript

10 Build an Interface to Edit Nodes with React

11 Create a Fully Decoupled React Application Free

12 Use create-react-app to Start a Decoupled React Application

13 Make API Requests with OAuth

14 Use Fetch and OAuth to Make Authenticated Requests

Contact us

Powered by: