Create a Fully Decoupled React Application

Last updated April 1, 2020

Theming Module Development 8.9.x/9.0.x

Now that we know how to build a working application in React and embed the application in Drupal, let's make a stand-alone version of our application which can be used outside of the context of a Drupal module or theme. In the next few tutorials we'll look at how to create a fully decoupled React application whose only interactions with Drupal happen via API requests.

In this tutorial we'll:

- Introduce differences we need to account for in a fully decoupled application
- Provide an example of what the final project will look like

By the end of this tutorial you should have a better understanding of what we're trying to create in the rest of this series.

Goal

Understand the modifications needed to build a stand-alone version of our React application.

Prerequisites

The remaining tutorials build on the code for the progressively decoupled React application created in the previous tutorials. If you're just starting from here, you'll want to make sure you grab that example code as a starting point.

Example code

You can find the complete example code for this application in the Git repository: https://github.com/DrupalizeMe/react-and-drupal-examples.

Overview

In the previous tutorial, we built a "progressively decoupled" React application. This means we embedded the React application inside Drupal. We connected to Drupal's JSON:API and used the `same-origin` option with `fetch()` to send the user's Drupal session cookie and authenticate API requests. This allowed us to side-step some things that we'll now have to account for in our code. Most of these things are primarily related to authentication. Our previous code relied on the fact that it was running inside the scope of an existing Drupal theme or module, and could make use of existing cookie and session handling for authentication. If you are logged into Drupal, you are also logged into our React app. For a fully decoupled application, we'll have to handle authentication ourselves.

To do this we'll want to start using OAuth to handle authentication and authorization. With OAuth, we're no longer dependent on the browser's somewhat opaque handling of cookies for authentication. Our code will also work in other contexts, for example, a React Native app where the code isn't executed inside a browser. The downside is we'll have to write a bunch of code to handle something the browser had been doing for us automatically.

You can use this same approach with any JavaScript front-end framework -- it's not specific to React. In fact, you can use this same approach with any decoupled application that needs to communicate via HTTP with Drupal. It doesn't even have to be JavaScript. This opens doors to all kinds of possibilities.

Benefits of a decoupled React application

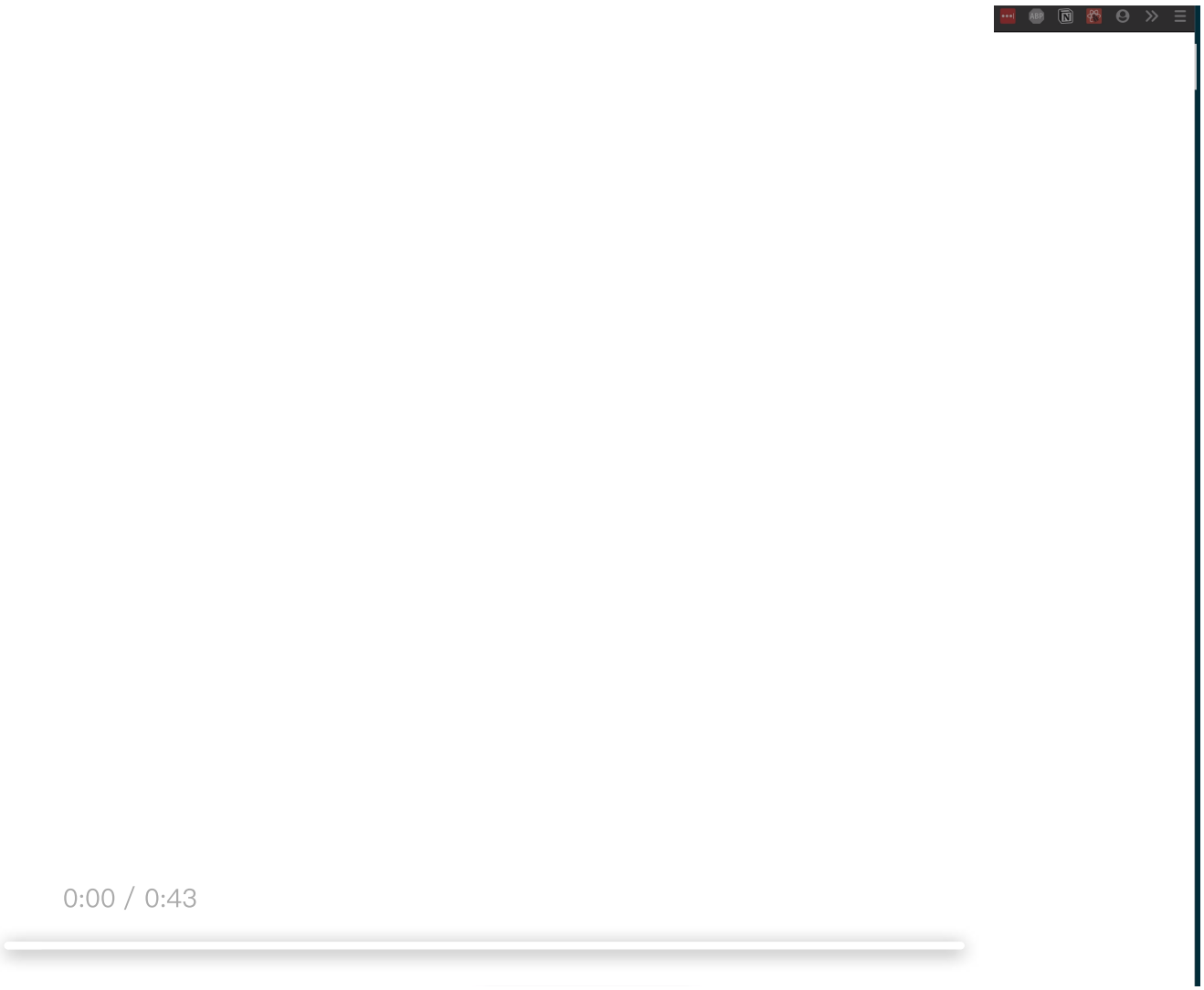
The primary benefit, as far as writing code goes, is that it's much easier to follow best practices established by the React community. This can go a long way towards helping find useful

documentation, or even getting help from others, when working on your code base.

When you introduce a build tool to your React code, you get developer experience benefits. You can code with hot-reloading, where your JavaScript will update automatically in the browser whenever you save the file. You can use linting tools with your code editor to write cleaner code. You can use a variety of CSS preprocessing tools. You can break up your React code into files for each component, and use that to organize your application in a sensible way.

You can also leverage third party packages. When you set up a build tool, you can import packages into React from `npm`. With a progressively decoupled application, you might be able to add some JavaScript libraries to your page in the HTML headers, but this can get cumbersome. With a decoupled application and a working build tool, you can load libraries into components when you need them, and your entire application will be self-contained.

What we're building



This tutorial will cover how to set up build tools to compile a React application and allow for development processes that more closely match what JavaScript developers are used to. We will use the create-react-app application to scaffold a new project, then port our existing code into that framework.

On the Drupal side, we'll install and configure the Simple OAuth module to allow for making authenticated requests to the JSON:API module. This will allow our decoupled React application to do things like POST, PATCH, and DELETE content in Drupal. We will need to refactor some of the HTTP request code to handle OAuth tokens.

Let's get started

Start with create-react-app -- Use `create-react-app` to scaffold a new fully decoupled React codebase. Optionally, learn how to integrate that into a Drupal theme or module.

Make API requests with OAuth -- Learn how to configure the Simple OAuth Drupal module and use a password grant flow to create authenticated API requests.

Use Fetch and OAuth to Make Authenticated Requests -- Add a login form to collect a username and password, exchange those for an OAuth access token, and update existing `fetch` requests to use the OAuth access token.

Recap

In this tutorial we got an overview of the decoupled React application that we'll build. We discussed some of the differences we'll need to address when creating a fully decoupled React application, compared to React code embedded in a Drupal module or theme.

Further your understanding

- Make a list of the changes you think you'll need to make to fully decouple your application.

Additional resources

You won't need to understand all the build tools for these tutorials, but if you are curious and want to understand some of the terms you might come across, here are some good

introductory articles:

- I finally made sense of front end build tools. You can, too. (medium.freecodecamp.org)
- JavaScript Build Tools and Automation Systems (hackernoon.com)
- The Ultimate Guide to JavaScript Build Tools (stackchief.com)

Was this helpful? Yes No

◀ Previous tutorial

Next tutorial ▶

Get Started Using React and Drupal Together

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