Decoupled vs. Progressively Decoupled

Last updated March 24, 2020

Theming Module Development 8.9.x/9.0.x

React and Drupal can be used together in two different ways: fully decoupled, also known as headless; or progressively decoupled.

In this tutorial we'll talk about the differences between these two approaches, including:

- Defining what each method refers to
- Considerations regarding hosting, performance, and access

Then we'll link to lots of additional reading materials so you can gain a deeper understanding of the subject.

By the end of this tutorial you should be able to define what *decoupled* and *progressively decoupled* mean, and how they differ from one another.

Goal

Understand the different ways in which React and Drupal can work together.

Prerequisites

- React Basics

Decoupled or progressively decoupled?

To determine which approach is being used, answer this question: is the JavaScript that makes up your React application part of an existing Drupal page?

If the answer is yes, than you're working on a progressively decoupled application wherein Drupal's theme layer and React are both responsible for what is on the page. In this scenario, React provides complex interactive elements, or pulls in and displays information from third party services, while Drupal and Twig control the majority of what is presented on the page.

If the answer is no, than you're using Drupal in a decoupled manner, wherein Drupal acts solely as the data store for your application and plays no part in determining the user experience. All the markup is provided by the React application.

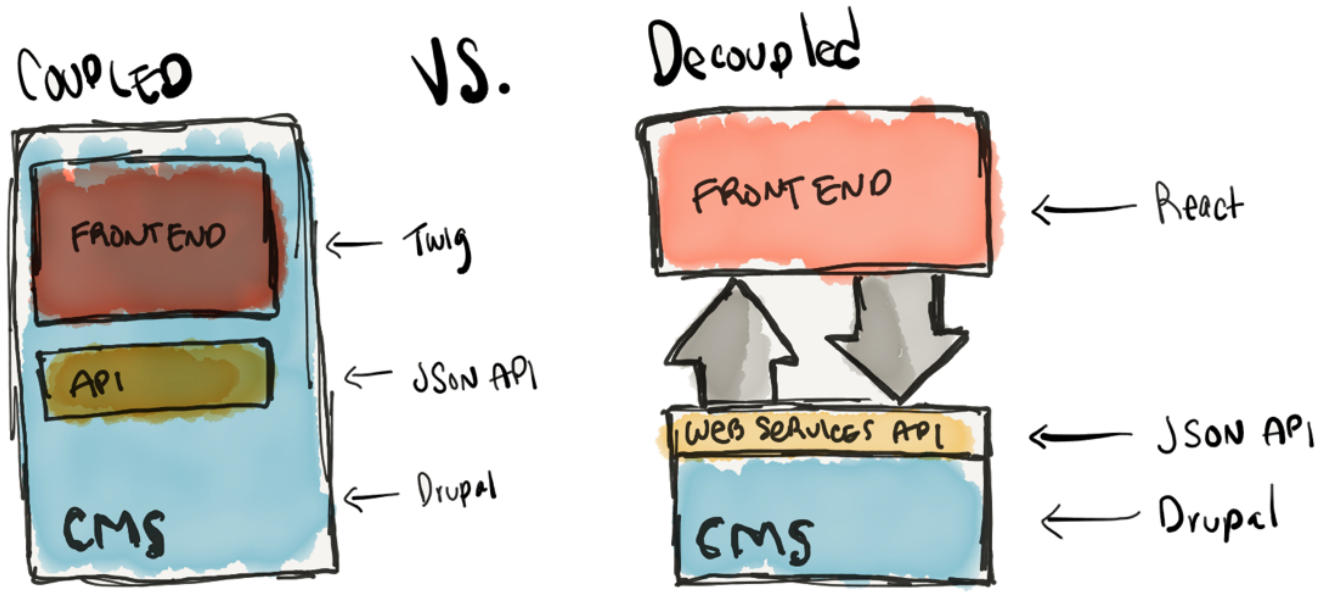
Both approaches are useful, and both have their pros and cons. We don't intend to tell you which one to use, or even to say one is better than the other. Rather, we'll cover how to do things with both approaches and let you determine which is going to be best for your use case.

In this series of tutorials we'll start by adding React code to an existing Drupal theme (progressively decoupled), and later move to creating a stand-alone React application (decoupled). Along the way we'll learn a whole lot about how both Drupal and React work.

Read more in our Decoupling Explained tutorial.

Decoupled

The terms *decoupled*, *headless*, and *stand-alone* all refer to a scenario in which a React application provides everything the user sees on the page. That application interacts with Drupal via a web services API to load content or authenticate users. Think of this as two independent applications -- one React, one Drupal -- that communicate with one another via an API.



Bringing data from Drupal into React requires setting up an API that exposes Drupal content at API endpoints, and then telling React to grab that data from the endpoint and parse that data as JSON.

Some of the pros and cons of this approach:

- Works well for development teams that are already well versed in React application development
- A component-based approach to designing and composing the UI
- Gives front-end developers complete control over the layout, look, and feel of the site. For example, a site administrator can't place a block into a region and change the UX of the site
- Front-end developers can use the tools they want to use in the way they are intended to be used rather then trying to coerce them into working within the context of a Drupal theme
- Your React application can potentially be hosted as a simple static site using GitHub pages, S3, or various other options

Progressively decoupled

The phrase *progressively decoupled* generally refers to any situation where Drupal's theme layer renders the initial state and a JavaScript framework like React performs further rendering on the client side. In this case Drupal, with a standard Drupal theme, outputs the primary user experience, and React enhances it. If you've built Drupal themes before, you've probably used

jQuery to provide user experience enhancements. This is the same thing, created with React instead of jQuery.

- You don't have to worry about doing server side rendering
- Drupal has powerful built-in tools for layout, views, displays, etc. If you need to provide editors or content creators the ability to control these things it's generally much easier to do with a traditional Drupal theme
- React code (which is just JavaScript) can ride along with the authentication that's already happening between your browser and Drupal

Recap

React can be integrated with Drupal in two different ways. You can write React code that integrates with, and enhances, an existing Drupal theme or module. Or, you can write a stand-alone React application that interfaces with Drupal via a web services API. Both approaches have their use cases, and it's up to you to decide which is appropriate for your project. The decision to decouple or not is nuanced, and requires a firm understanding of the end goals of your project, the team you're working with, and the advantages and disadvantages of both approaches.

Further your understanding

- Try to integrate React into a Drupal theme. Start with Connect React to a Drupal Theme.
- Create a stand-alone React application that interacts with Drupal via JSON:API.
- If you want us to cover this in more detail, please let us know.

Additional resources

- Decoupling Explained (Drupalize.Me)
- Decoupled Drupal: When, Why, and How (youtube.com)
- Should You Decouple? (lullabot.com)
- Decoupled Drupal - What You Need to Consider (chromatichq.com)
- Why Progressive Decoupling Drupal's Front-End Is a Bad Idea (chapterthree.com)
- Progressively Decoupled Drupal Approaches (acquia.com)
- Decoupled CMS: Why "Going Headless" Is Becoming So Popular (pantheon.io)

Was this helpful? Yes No

◀ Previous tutorial

Next tutorial ▶

Get Started Using React and Drupal Together

STAY INFORMED

Sign up for our mailing list to get Drupal tips and tricks in your inbox!

Subscribe

STAY CONNECTED

Powered by: