Drupal 8

CONTENTS

WRITTEN BY CINDY MCCOURT
CONSULTING

Drupal is an open-source content management system used to create multilingual websites and online applications. Drupal 8 is better than ever.

Its configurable features have increased significantly with each version, enabling non-coders to build sophisticated websites without touching the code. At the same time, Drupal's transition to an object-oriented coding strategy is just one significant improvement on the backend that has made this powerful system more customizable than ever before.

What does this dual advantage mean for you as a developer new to Drupal? It means you need to wear two hats --- builder and coder --- if you want to deliver websites and online applications that remain ready for change and growth with minimal custom intervention.

Given Drupal's legacy and new features, it's impossible to cover everything in this guide. However, the following will get you started with important concepts and resources so that you can wear each hat.

- How Drupal Works
- Installing Drupal
- Building Drupal Sites
- Customizing Drupal Sites
- Maintenance
- Migrating to Drupal 8

How Drupal Works

Before diving into the details of how to install Drupal 8 and develop a site, you need to start thinking in Drupal. That means an introduction to how Drupal works.

Thinking in Drupal can ignite ideas regarding your site that you had already planned. You might find you want to revisit aspects of your plan in order to take advantage of Drupal features you hadn't considered.

Your first step to thinking in Drupal is understand the Drupal web page and how it comes together. This section introduces the following.

- **URL paths** - more than just the page address.
- **Drupal pages** - pages which are not the full web page.
- **Drupal page structure** - centers on the content region that displays the page.
- **From data to web page** - involves several Drupal systems.
- **From web page to site architecture** - more than just the menu and site sections.

PANTHEON
Build your first website on Drupal 8.
"Getting Started with Drupal" training series, presented by Pantheon.
Free Training

PANTHEON

Power your Drupal websites with the leading WebOps Platform.

With WebOps, Drupal developers can:

Build. Quickly get started on Drupal with simple installation.

Launch. Use Dev and Test environments to develop and test your site until you're ready to go live.

Iterate. Focus on iterating on your website and providing more value to your users.

Try Pantheon's WebOps Platform today.

- **Inner workings** - includes the backend and front-end before an HTML page is rendered.

Let's start with the concept of pages and how they are accessed.

URL PATHS

It may seem rudimentary to talk about URL paths, but you can't have a web page in Drupal without one. Keeping it rudimentary, consider the following example:

http://www.example.com/?q=node/3

This is the default path for requesting a content page, also known as a node. Nodes are just one way pages are created in Drupal. This URL query for content is not a friendly way to ask for a page, therefore, clean URLs are created by the server to allow the following to work as well.

http://www.example.com/node/3

Unfortunately, the clean URL isn't as friendly as it could be. Drupal comes with the Path module installed. This feature enables you to create a human-friendly URL alias. For example, if `/node/3` represented an article with the title, *My First Article*, you can set the URL path to be:

http://www.example.com/my-first-article

Aliasing a URL doesn't stop there. Automated URL aliases help ensure consistency. With the contributed module called Pathauto, coupled with the Token and Chaos Tools modules, you can better meet your SEO goals.

Let's take a look at the types of pages that go along with the URL.

DRUPAL PAGES

As mentioned above, a node page is one way to create a page in Drupal. It's actually the most common method used to create pages. Before we dive into the details of this popular strategy, there are other ways to create pages using Drupal.

- **Views pages** - Views is a default functionality in Drupal 8 that enables you to query the database and create a page to show the results. Drupal's homepage is made possible by a Views page with the path `/node`.
- **Core module pages** - Pages created by one or more core modules with pages such as `/search`, `/forum`, `/book`, and `/admin`.
- **Contributed module pages** - Some modules, such as [Panels] (https://www.drupal.org/project/panels), help you build pages with unique layouts and elements, as well as a page path.
- **Custom module pages** - You can write code to create your own pages, however, this should be your last resort. When you think in Drupal, the opportunities to develop using existing and tested code is huge.

Which strategy you use depends on your needs. Figure 1.1 illustrates two example solutions to a page whose requirement is a list of links to other pages on the site.

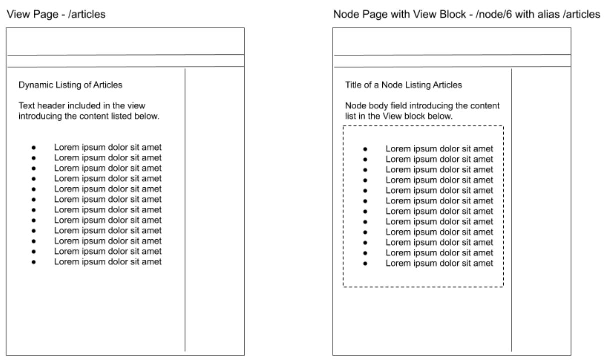

View Page - /articles
Dynamic Listing of Articles
Text header included in the view introducing the content listed below.
Lorem ipsum dolor sit amet
Node Page with View Block - /node/6 with alias /articles
Title of a Node Listing Articles
Node body field introducing the content list in the View block below.
Lorem ipsum dolor sit amet

Figure 1.1: Two wire frames of simple content pages. One made with Views page and the other with a combination of a node with a Views block.

In the first example, you use Views to query the database, get a list of page titles as links, and then display the results as a page. In the second scenario, you create a node page and use the same Views query but display the results as a block before setting it to render below the node. We will talk about blocks and fields shortly, but first we'll review page structure and the region that does the heavy lifting on any Drupal page.

DRUPAL PAGE STRUCTURE

Now that you are thinking in URLs and pages, let's look under the hood at what makes a page possible. Drupal web pages are made up of a series of nested Twig templates. Twig is a PHP engine used to convert Drupal's templates into HTML. A more detailed explanation of the theme and its templates is offered later is this Refcard.

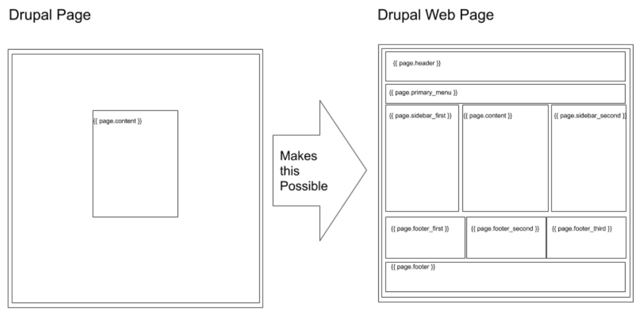


Figure 1.2: The role of the page template content region

Technically, if you want the content associated with the Drupal page to render, the content region is all you need to include in your page.html.twig template. The Drupal page (be it a node, Views page, search, etc.) is like a coat rack ready to carry whatever you hang on it, such as a header, menu, sidebar content, and various footers (see Figure 1.2).

So far, we've considered pages as a whole. Let's look at a node page and what it takes to create the full Drupal web page.

FROM DATA TO WEB PAGE

Like any content management system, you enter data and the system returns a page. The way pages are created varies from one framework to another, therefore, knowing how Drupal does it is imperative to developing a site full of pages that can flex and grow over time without having to refactor your code.

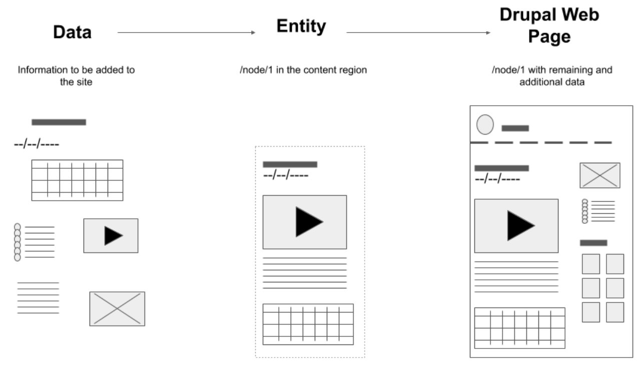

Data
Information to be added to the site
Entity
/node/1 in the content region
Drupal Web Page
/node/1 with remaining and additional data
--/--/----

Figure 1.3: Illustration of the data to include in a web page, that some or all data items would be loaded to a node that displays in the content region, which is then wrapped with other regions and their content items.

A Drupal web page is created via several Drupal systems working together.

- **Entity types** are used to collect data via **fields** and manage the display of the data **field** (e.g., formatting and styling).
- **Blocks** are used to tell Drupal what to display (e.g., entity fields, custom content blocks, system blocks, menu blocks, dynamic list blocks, etc.).
- **Regions** that provide a page section for the blocks (e.g., header, content, sidebar, etc.).
- **Theme**, which controls which regions are available and how they are rendered in the page.html.twig template.

Let's consider each briefly.

ENTITY TYPES

In Drupal 8, almost all functionality is either a content entity or configuration entity. Content entities focus on collecting and storing your content data, whereas, configuration entities focus on collecting and storing configuration settings.

Not all content entity types render a page. In order to create a page, the entity must have a URL property. The URL is present in the following entity types.

- **Content type (a.k.a. Node type)** - used to create a content/node page. There are two default content types enabled in Drupal: Article and Basic page. Two other core content types are part of the Book and Forum modules that you can enable.
- **Media type** - used to add various forms of media to the media library. The core Media module is not enabled by default as of writing this Refcard.
- **Taxonomy term** - used within other entity types to categorize content. It also produces a page created by Views, Drupal's powerful query and display module.
- **User** - accounts and their pages can display the same types of fields used on other entity types to create a user profile page where you can add a Views block that lists the user's blog posts, for instance.
- **Contact form** - used by site visitors to send a message to one or more email addresses via the site. It has the option to add fields as well, however, the contributed module, Webform, might yield better results for more robust contact forms.
- **Custom entity types you create** - used to add data to the site that doesn't need all the propertes that an existing entity type offers. One place to start to learn the process is on *Create a Custom Content Entity* at https://www.drupal.org/docs/8/api/entity-api/create-a-custom-content-entity.

FIELDS

Some entity types wear two hats: the entity page and the entity field. As a field, it is dependent upon another entity type to display, such as on pages and blocks.

- **Taxonomy term reference field** - uses terms, collected in Vocabularies, to categorize and/or describe an entity via the tagging process.
- **Media fields** - used to add an image, video, or file to an entity.
- **Comment types** - used to allow site visitors to comment on an entity.

As entities, even fields can have fields. For example, comment types are made up of field sets, such as the subject line and the comment field. You could add additional fields, just like the entities that are also pages and blocks. As an example, imagine Taxonomy terms with images, such as a picture of an orange for the term orange.

BLOCKS

Blocks are very similar to content types, as you can have various types and multiple fields. They are also like fields in that they are dependent upon entity pages to become visible.

Other types of blocks include:

- **System blocks** - helpful built-in blocks that you can turn on and off and configure (e.g., Messages, Main page content, and the site branding block that manages the display of the logo, site name, and site slogan).
- **Menu blocks** - there are five menu blocks (Administration, Footer, Main navigation, Tools, User account menu), but you can create your own custom menu for static lists of links. All menus are blocks, no matter where they are rendered in the template or styled via CSS.

- **Module blocks** - created for you by a module such as the Search and Book modules do. When you install a contributed module, use the Place block button on the Block layout admin page to see if it offers a block.
- **Views blocks**. Queries on the database can be rendered as a block as well as a page and other types of displays.

There is more to learn about entities and their properties, especially if you are going to create your own. Start by reviewing the entity resources on Drupal.org.

- Entity API
- 2.3. Concept: Content Entities and Fields

REGIONS AND THE THEME

The Drupal Page Structure topic above touched briefly on the page.html.twig template. Let's take a closer look at how the various nested theme templates work together to display blocks and fields.

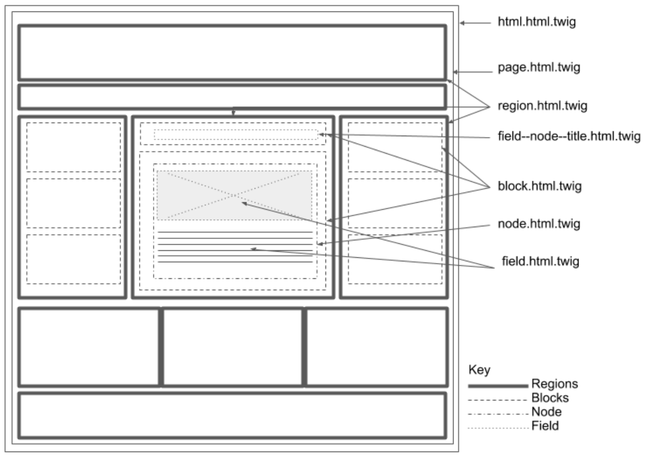
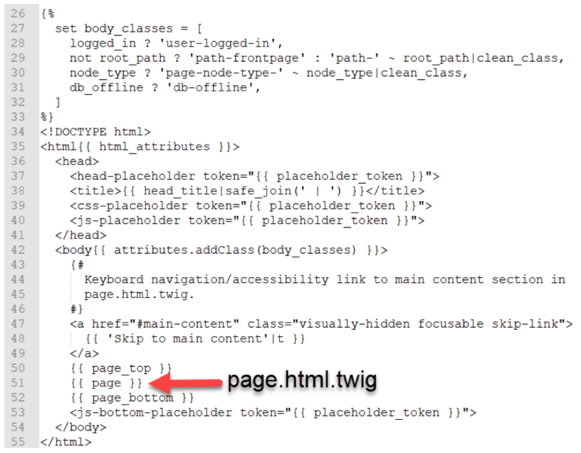


Figure 1.4: Conceptual illustration of nesting templates

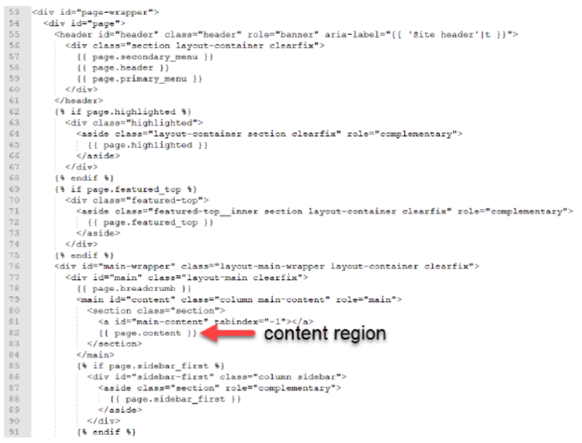
The hierarchy of the theme templates works as follows:

- **html.html.twig** - sets up the web page and renders the content for the page.html.twig template.



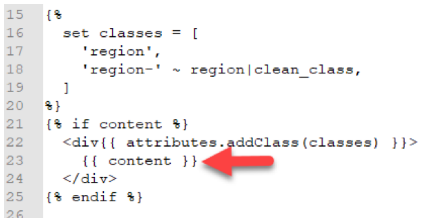
```
{%
  set body_classes = [
    logged_in ? 'user-logged-in',
    not root_path ? 'path-frontpage' : 'path-' ~ root_path|clean_class,
    node_type ? 'page-node-type-' ~ node_type|clean_class,
    db_offline ? 'db-offline',
  ]
%}
<!DOCTYPE html>
<html{{ html_attributes }}>
  <head>
    <head-placeholder token="{{ placeholder_token }}">
    <title>{{ head_title|safe_join(' | ') }}</title>
    <css-placeholder token="{{ placeholder_token }}">
    <js-placeholder token="{{ placeholder_token }}">
  </head>
  <body{{ attributes.addClass(body_classes) }}>
    {#
      Keyboard navigation/accessibility link to main content section in
      page.html.twig.
    #}
    <a href="#main-content" class="visually-hidden focusable skip-link">
      {{ 'Skip to main content'|t }}
    </a>
    {{ page_top }}
    {{ page }}
    {{ page_bottom }}
    <js-bottom-placeholder token="{{ placeholder_token }}">
  </body>
</html>
```

- **page.html.twig** - renders the regions that organize blocks, as well as any other region you need.



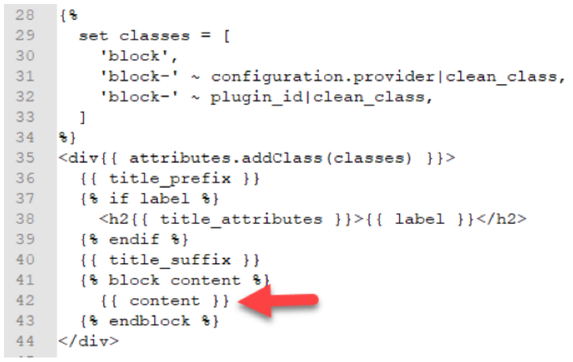
```
<div id="page-wrapper">
  <div id="page">
    <header id="header" class="header" role="banner" aria-label="{{ 'Site header'|t }}">
      <div class="section layout-container clearfix">
        {{ page.secondary_menu }}
        {{ page.header }}
        {{ page.primary_menu }}
      </div>
    </header>
    {% if page.highlighted %}
      <div class="highlighted">
        <aside class="layout-container section clearfix" role="complementary">
          {{ page.highlighted }}
        </aside>
      </div>
    {% endif %}
    {% if page.featured_top %}
      <div class="featured-top">
        <aside class="featured-top__inner section layout-container clearfix" role="complementary">
          {{ page.featured_top }}
        </aside>
      </div>
    {% endif %}
    <div id="main-wrapper" class="layout-main-wrapper layout-container clearfix">
      <div id="main" class="layout-main clearfix">
        {{ page.breadcrumb }}
        <main id="content" class="column main-content" role="main">
          <section class="section">
            <a id="main-content" tabindex="-1"></a>
            {{ page.content }}
          </section>
        </main>
        {% if page.sidebar_first %}
          <div id="sidebar-first" class="column sidebar">
            <aside class="section" role="complementary">
              {{ page.sidebar_first }}
            </aside>
          </div>
        {% endif %}
```

- **region.html.twig** - a section of HTML that renders blocks whose data is contained within double curly brackes, i.e. `{{ content }}`.



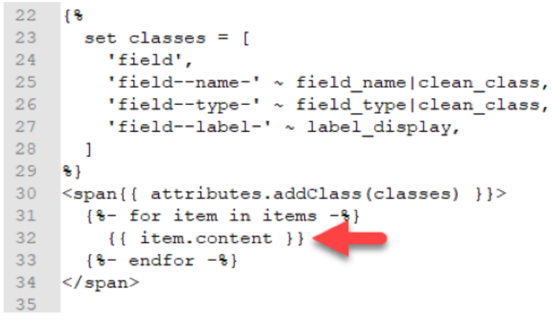
```
{%
  set classes = [
    'region',
    'region-' ~ region|clean_class,
  ]
%}
{% if content %}
  <div{{ attributes.addClass(classes) }}>
    {{ content }}
  </div>
{% endif %}
```

- **block.html.twig** - renders various types of content, depending on the block, again, stored in double curly brackes, i.e `{{ content }}`.



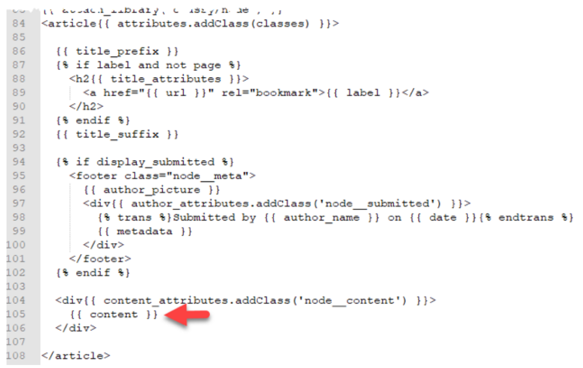
```
{%
  set classes = [
    'block',
    'block-' ~ configuration.provider|clean_class,
    'block-' ~ plugin_id|clean_class,
  ]
%}
<div{{ attributes.addClass(classes) }}>
  {{ title_prefix }}
  {% if label %}
    <h2{{ title_attributes }}>{{ label }}</h2>
  {% endif %}
  {{ title_suffix }}
  {% block content %}
    {{ content }}
  {% endblock %}
</div>
```

- What is included in a block will vary. For example:
- **field--node--title.html.twig** is a variation on field.html.twig and will render the node title.



```
{%
  set classes = [
    'field',
    'field--name-' ~ field_name|clean_class,
    'field--type-' ~ field_type|clean_class,
    'field--label-' ~ label_display,
  ]
%}
<span{{ attributes.addClass(classes) }}>
  {%- for item in items -%}
    {{ item.content }}
  {%- endfor -%}
</span>
```

- **node.html.twig** will render the other populated fields for the content type via the **Main page content** system block. This system block will then render the fields.



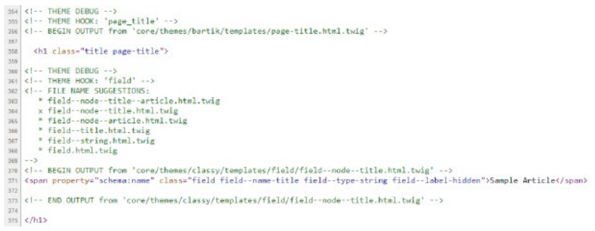
```
<article{{ attributes.addClass(classes) }}>

  {{ title_prefix }}
  {% if label and not page %}
    <h2{{ title_attributes }}>
      <a href="{{ url }}" rel="bookmark">{{ label }}</a>
    </h2>
  {% endif %}
  {{ title_suffix }}

  {% if display_submitted %}
    <footer class="node__meta">
      {{ author_picture }}
      <div{{ author_attributes.addClass('node__submitted') }}>
        {% trans %}Submitted by {{ author_name }} on {{ date }}{% endtrans %}
        {{ metadata }}
      </div>
    </footer>
  {% endif %}

  <div{{ content_attributes.addClass('node__content') }}>
    {{ content }}
  </div>

</article>
```

- **field.html.twig** - renders the data as declared by the Manage display settings for the entity.

TEMPLATES IN USE

To see these nesting relationships and alternative naming conventions for your custom theming efforts, enable the debugging feature that comes standard with Drupal. After you install your site:

1. Go to your code base.
2. Locate `\sites\default\default.services.yml`.
3. Rename the file services.yml.
4. Change **debug: false** to **debug: true** and save it.
5. Clear the site cache via configuration>performance>clear cache.
6. Create an Article node and view its page source.
7. Observe:



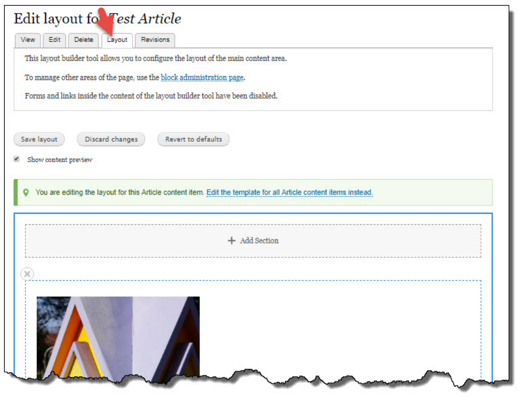
```
<!-- THEME DEBUG -->
<!-- THEME HOOK: 'page_title' -->
<!-- BEGIN OUTPUT from 'core/themes/bartik/templates/page-title.html.twig' -->

  <h1 class="title page-title">

<!-- THEME DEBUG -->
<!-- THEME HOOK: 'field' -->
<!-- FILE NAME SUGGESTIONS:
   * field--node--title--article.html.twig
   x field--node--title.html.twig
   * field--node--article.html.twig
   * field--title.html.twig
   * field--string.html.twig
   * field.html.twig
-->
<!-- BEGIN OUTPUT from 'core/themes/classy/templates/field/field--node--title.html.twig' -->
<span property="schema:name" class="field field--name-title field--type-string field--label-hidden">Sample Article</span>

<!-- END OUTPUT from 'core/themes/classy/templates/field/field--node--title.html.twig' -->

</h1>
```

   - `X` marks the template that is in use. In the example shown in the screenshot above, start with the generic `field`. Narrow the focus to `node`. Then, go even tighter and select the title `field` where `title` is the machine name.
   - `*` indicates other template names that can be used.
   - `field.html.twig` is the most generic file name for the field template.

To learn more about how the options in the debugging feature are generated, visit Working With Twig Templates.

PAGE-BY-PAGE CUSTOMIZATION

It's likely that most of your pages, if not all, will be rendered via theme templates. However, sometimes there is cause to enable a content editor to customize the layout of a node page. The core Layout Builder module allows you to do just that via the Manage Display option for each content type. With Layout Builder installed and enabled for a content type, each node page created using that content type will have the option to customize the content displayed.



Edit layout for *Test Article*

This layout builder tool allows you to configure the layout of the main content area.

To manage other areas of the page, use the block administration page.

Forms and links inside the content of the layout builder tool have been disabled.

☑ Show content preview

You are editing the layout for this Article content item. Edit the template for all Article content items instead.

Read up on *Layout Builder* here.

HEADLESS DRUPAL

The solution discussed so far assumes a *coupled* Drupal. In other words, a traditional Drupal sight that uses Drupal's theming layer to present the site's pages. However, Drupal is not locked into such a solution.

Decoupling the backend of a CMS from its front-end empowers content owners to establish and manage one content resource, thus enabling them to reuse content in multiple presentation layers, as suggested in Figure 1.5.

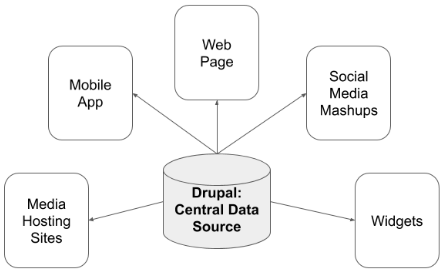


Figure 1.5: Conceptual illustration of Headless Drupal

By integrating RESTful (Representational State Transfer) Web Services into Drupal 8's core, decoupling is easier than it has ever been. The RESTful Web Services module exposes entities and other resources such as RESTful web APIs.

Decoupling is not an all-or-nothing process. In his How to Decouple Drupal in 2019 blog post, Dries Buytaert, the inventor of Drupal,explains the advances in headless Drupal, as well as the degrees with which decoupling is now possible.

- Coupled
- Progressively decoupled
- Fully decoupled static site
- Fully decoupled app

There are other Web Services included in Drupal 8 that make customization easier.

- **HAL** - Serializing entities using Hypertext Application Language.
- **HTTP Basic Authentication** - Provides the HTTP Basic authentication provider.
- **JSON API** - Exposes entities as a JSON API specification compliant web API.
- **Serialization** - Provides a service for (de)serializing data to/from formats such as JSON and XML.

FROM WEB PAGE TO SITE ARCHITECTURE

A key success factor in any website is a multifaceted and friendly information architecture. There are various strategies available in Drupal to organize and relate content pages.

- **Menus** for a fixed and structured organization of Drupal web pages.
- **Taxonomy** to organize entities into one or more categories.
- **References** to manually create relationships between chosen entities (e.g., node-to-node and/or node-to-user).
- **Dynamic lists created with Views** to display links to entities that share something in common such as a date, an image, a topic, and/or a content type.

INNER WORKINGS

There are several types of interaction processes that need to be considered when standing up a Drupal site. For example:

- Input Tasks
  - Site builder configures the site.
  - Site author creates pages.
  - Site visitor/user requests a page and/or posts data.
- Output Tasks
  - Site builder receives confirmation.
  - Site author views the results saved.
  - Site visitor receives a page.

Whether the request is to process input or display output, Drupal executes the same process to return the applicable interface. The following is a high-level, simplified illustration of the process.

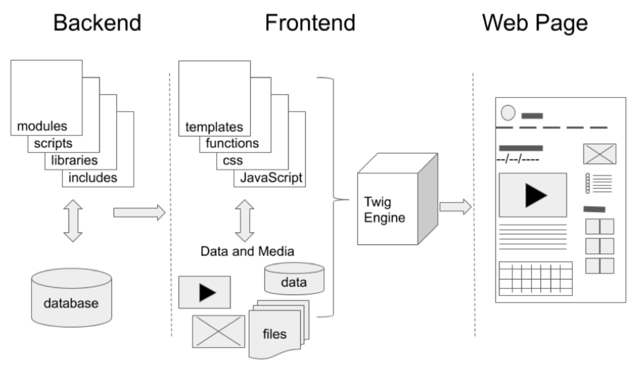


Figure 1.6: Conceptual illustration of Dupal's inner workings

The backend determines the requirements for the page based on the URL path being requested. The appropriate modules, scripts, libraries, and includes are called into action, as well as the retrieval of applicable data from the database.

The results from the backend are fed into the theming layer (if the site is not decoupled) where templates are blended with the applicable data and converted from PHP to HTML. The HTML and its associated CSS, pre-processing functions, and JavaScript and jQuery code are combined with the media files for the page and sent to the requester's browser.

DISTRIBUTIONS

According to the Drupal 8 docs, "Distributions are full copies of Drupal that include Drupal Core, along with additional software such as themes, modules, libraries, and installation profiles." Why do you need to care about distributions? Well, if the type of site you need already exists and you can tweak it into completion versus starting from scratch, you could save some time.

Drupal distributions are defined by their profiles. According to Drupal's `profile\readme.txt` file, "Installation profiles define additional steps that run after the base installation of Drupal is completed. They may also offer additional functionality and change the behavior of the site."

For a list of distributions, visit https://www.drupal.org/project/project_distribution. For example:

- **Lightning** - the distribution for enterprise authoring.
- **Open Social** - used to quickly stand up online communities.
- **Thunder** - built for professional publishing.

Whether you use a distribution or start with Drupal default installation, you will benefit by installing one or more distributions that have features of interest. There is more than one way to meet requirements when using Drupal and, often, a distribution reflects best practices used by seasoned Drupal developers. Learn from others by looking at what they have done.

Installing Drupal

Now that you have a big picture view of key Drupal concepts, let's look at what is takes to install a Drupal 8 site, starting with the server that will host your site.

SERVER REQUIREMENTS

There are three perspectives to consider when creating or choosing a Drupal compatible server.

- Drupal specific requirements.
- Server operating system.
- Web server application.

The following three server requirements sections are a summary from two Drupal.org resources: https://www.drupal.org/docs/user_guide/en/

install-requirements.html and the pages listed on https://www.drupal.org/docs/8/system-requirements.

DRUPAL SPECIFIC REQUIREMENTS

There are several requirements that must be present or configured for Drupal 8 to work.

- PHP 7.2 or higher.
- Minimum PHP memory of 64MB; but be ready to go to 256MB as you add functionality.
- Minimum of 100MB of disk space for the base files alone. If you plan on adding functionality or uploading images, for instance, you will need more.
- Drupal and PHP need read/write access to the `\sites\default\files` directory. Universal permissions, like 0777, are not secure.
- A database. The default Drupal database is "MySQL 5.5.3/MariaDB 5.5.20/Percona Server 5.5.8 or higher with InnoDB as the primary storage engine and requires the PDO database extension." PostgreSQL and SQLite are also options. See the sources linked above for more details.

SERVER OPERATING SYSTEMS

You are not restricted in your choice of operating systems, however, some are better suited than others. And, if you query the Drupal community, you will see various versions of Linux OS, Cent OS, Ubuntu OS, Unix OS, Microsoft IIS, and Mac OS mentioned, each with anecdotal evidence as to which is better.

Drupal was developed to run an open source stack, therefore Linux is often preferred.

WEB SERVER

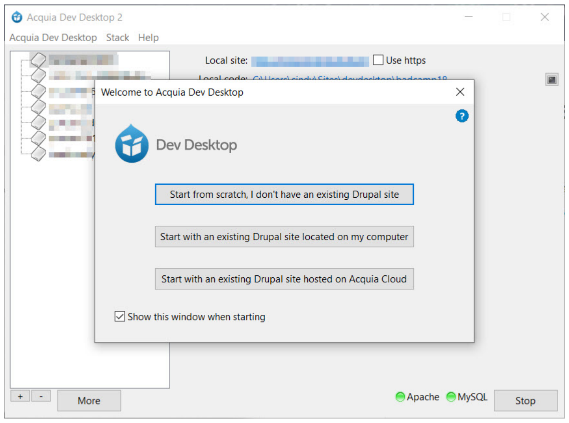
There are several web servers to choose from, as expected, given the range of operating systems that can be used. Keeping with Drupal's open source perspective, Apache is the default recommendation. The source links above provide alternatives and what you need to know in order to use them successfully.

HOSTING OPTIONS

You have two environments to consider: a local server for local development and a remote web server for actual hosting (which you will likely use for your dev, stage, and prod environments).

LOCAL SERVER

A great way to practice developing Drupal sites is on your computer. Turning it into a local web server is easier than you might imagine. If you are brand new to server configuration, start with Acquia's dev desktop. You can download it for Windows or Mac from here. However, feel free to use whatever local server configuration with which you are familiar.


Acquia Dev Desktop 2
Acquia Dev Desktop Stack Help
Local site:
Use https
Local code:
Welcome to Acquia Dev Desktop
Dev Desktop
Start from scratch, I don't have an existing Drupal site
Start with an existing Drupal site located on my computer
Start with an existing Drupal site hosted on Acquia Cloud
Show this window when starting
+ -
More
Apache
MySQL
Stop

There are numerous tutorials online on how to use the Acquia dev desktop. You will need to learn two processes: installing it on your computer and site setup. The tutorial, Drupal 8 Beginner, Lesson 5: Installing Acquia's Dev Desktop addresses each of these processes.

Note: There have been times when students in Drupal training classes have had to switch to the Acquia dev desktop because their local environment isn't configured properly.

DRUPAL INSTALLATION ON A REMOTE SERVER

There are several factors to consider when installing Drupal 8. For example, will you install Drupal using git? Then you will need Composer installed in the same environment where you host your site. (https://getcomposer.org/doc/00-intro.md).

The *Installing Drupal 8* instructions at https://www.drupal.org/docs/8/install provide a comprehensive set of instructions, depending on your requirements.

With that said, you have installation options.

- **Do it yourself**. Follow the instructions spelled out at https://www.drupal.org/docs/8/install. You will see that you have two options for getting Drupal onto your web server.
    - **Use Composer** - See *Download Drupal core using Composer* at https://www.drupal.org/docs/develop/using-composer/using-composer-to-manage-drupal-site-dependencies#download-core.
    - **Do it manually** - See *Section 3.3 of the Drupal 8 User Guide* at https://www.drupal.org/docs/user_guide/en/install-prepare.html
- **Click to Install.** Some hosting providers will offer you a one-click service that will install (and possibly manage) the Drupal code and database. This is a nice solution for non-developers. However, in order to manage updates and troubleshoot successfully, it's better to understand your hosting environment and where you can access your Drupal code and data, therefore, use one of the above Do it Yourself options.

- **Managed**. Pay someone to set you up and teach you how to work within their hosting environments. Such providers are going to be more expensive, but sometimes the investment is worth it.

Already have a site and want to use Composer? Drupal.org has you covered. Learn how to add Composer to an existing site at https://www.drupal.org/docs/8/install/add-composer-to-an-existing-site.

CHOOSING A HOSTING SERVICE

If you are savvy enough to build your own web server, start with https://www.drupal.org/docs/8/system-requirements for the information you need. If you need a hosting service, start by reviewing the *Marketplace* at https://www.drupal.org/hosting. Not all hosting providers offer environments that Drupal will like. Selection considerations include:

- **Limitations** - Shared hosting has its limitations regarding control over application versions, PHP memory allocation, and disk space, to name a few. Read what they offer carefully.
- **Flexibility** - There is a significant difference between shared hosting, VPS hosting, dedicated server hosting, and cloud hosting. Each has its pros and cons, including price, support, ability to meet your needs, and your required skill level to use them. Do your research before choosing services you don't understand or need.
- **Drupal Support** - Support by Drupal's knowledgeable staff. If the hosting service promotes Drupal, you will likely have more success regarding the appropriate server configuration and get applicable help when needed.
- **Multi-site Hosting** - Are you hosting multiple Drupal sites that need individual site owner access? Some hosting services offer such plans, but you should ask.
- **Site size** - Anticipate how much server space you will need and if the hosting options offer what you need.
- **Site performance** - Anticipate traffic on your site and your caching needs. Will the hosting service be able to handle large volumes of page requests?
- **Email** - Will you manage your email with the hosting service? Make sure to investigate your options.
- **Domain management** - Do you need to purchase a domain? If you already have one, ensure that you are not under an obligation to host your site with the domain provider. That great deal you got might come with strings attached: "Free domain with purchase of a hosting plan."
- **Price** - Don't be fooled by low prices. Sometimes you get what you pay for and that isn't enough.

Building Drupal Sites

The sheer volume of possible configuration tasks you can perform before touching the code is more than what can be covered in this quick reference. Not only is Drupal 8 more robust than any other version before it, but it's a moving target - in a good way.

DRUPAL 8 DEVELOPMENT STRATEGY

Every six months from its initial launch in November of 2015 new functionality has been added. Instead of jumping from Drupal 8 to Drupal 9, the development plan is moving in increments. From 8.0.0 to 8.1.0 and so on to 8.7.6 (as of August 2019). The second number indicates core functionality changes, whereas the third number indicates required fixes, such as security patches.

In June 2020, Drupal 8.9 and Drupal 9.0 will be released. Learn more about the process from *Plan for Drupal 9*.

LEARNING TO BUILD WITH DRUPAL 8

When Drupal 8 was released, Acquia teamed up with OSTraining to develop 63 brief but informative Drupal 8 beginner tutorials and placed them on YouTube for free. If you don't want to sign up and pay for a Drupal 8 training course, this is a great place to start to learn about the builder hat you need to wear, in addition to your coder hat. The key concepts are introduced along with the Drupal 8 interface.

NEW DRUPAL FUNCTIONALITY

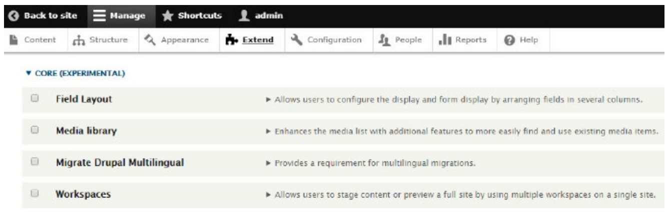
What's not covered in the beginner videos are the new features that have been added to Drupal 8's core since its launch. As of Drupal 8.7, the following experimental modules have been added to Drupal core.

- **BigPipe** - Sends pages using the BigPipe technique that allows browsers to show them much faster.
- **Content Moderation** - Provides moderation states for content.
- **Contextual moderation** - Provides contextual links to perform actions related to elements on a page.
- **Inline form errors** - Places error messages adjacent to form inputs, for improved usability and accessibility.
- **Layout Builder** - Allows users to add and arrange blocks and content fields directly on the content.
- **Layout Discovery** - Provides a way for modules or themes to register layouts.
- **Media** - Manages the creation, configuration, and display of media items.
- **Migrate** - Handles migrations.
- **Migrate Drupal** - Contains migrations from older Drupal versions.
- **Migrate Drupal UI** - Provides a user interface for migrating from older Drupal versions.
- **Settings Tray** - Allows users to directly edit the configuration of blocks on the current page.

- **Workflows** -- Provides a UI and API for managing workflows. This module can be used with the Content moderation module to add highly customizable workflows to content.

The list of Core (experimental) modules in Drupal 8.7 includes:

- **Field Layout** - Allows users to configure the display and form display by arranging fields in several columns.
- **Media Library** - Enhances the media list with additional features to more easily find and use existing media items.
- **Migrate Drupal Multilingual** - Provides a requirement for multi-lingual migrations.
- **Workspaces** - Allows users to stage content or preview a full site by using multiple workspaces on a single site.



▼ CORE (EXPERIMENTAL)

☐	Field Layout	▸ Allows users to configure the display and form display by arranging fields in several columns.
☐	Media library	▸ Enhances the media list with additional features to more easily find and use existing media items.
☐	Migrate Drupal Multilingual	▸ Provides a requirement for multilingual migrations.
☐	Workspaces	▸ Allows users to stage content or preview a full site by using multiple workspaces on a single site.

The experimental list of modules has changed with each update so keep watch as Drupal 8.7 moves towards Drupal 9.

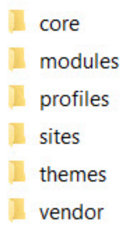
Customizing Drupal Sites

Customizing a Drupal site happens both on the backend and front-end. A key to a successful Drupal site is using Drupal's flexible object-oriented framework versus jumping straight into custom coded solutions.

PREREQUISITES

When you put your coder hat on, you need to understand how Drupal is coded before you put your existing PHP skills and experiences to work. Begin by learning. The following resources will start down this path.

- **Module Development**
  - Read Getting Started - Background & Prerequisites (Drupal 8)
  - Review Drupal's existing code base by visiting your web server.



core
modules
profiles
sites
themes
vendor

  - Explore *Drupal 8 APIs* at https://www.drupal.org/docs/8/api
- **Theme Development**
  - Review the Drupal 8 Theme folder structure to learn about the files and code used to create a Drupal theme.
  - Get an introduction to how YAML and Twig play a major role in Drupal 8 theming.
  - The Drupal 8 Theming Guide on GitHub
    - YAML file format
    - Twig in Drupal 8

CUSTOM MODULES

Finally, it's time to put on your coder hat. Suppose you need to integrate with another system or create a unique entity, Drupal has coding requirements you have to follow in order for your custom module to be recognized and supported by Drupal core. The community also has standards. You might think you can do anything you want, because it's your site --- and you would be right. However, in the long run, following Drupal's requirements and standards can help you keep your site secure.

Drupal.org, once again, has excellent resources to get you started with Creating Custom Modules.

An internet search will yield multiple resources to help you learn to develop Drupal modules using Drupal coding best practices. Not all are free. Below are randomly selected examples.

- Drupal 8 Module Development
- Learn Drupal 8 module development with examples

CUSTOM THEME

We've already explored aspects of the theme in Regions and the Theme, Inner Workings, and Figure 1.6. Let's continue exploring the Drupal theme.

The theme is, most often, where customization efforts are applied. However, that doesn't mean you start from scratch. There are many great, free themes and base themes on Drupal.org. That is where you should start.

Customization can include adding or subtracting regions, changing the CSS, overriding core or contributed module templates, adding pre-processing functions, and more. Therefore, the next concept to understand is where Drupal looks for the theme code it needs to render the page requested as not everything is included in one theme.

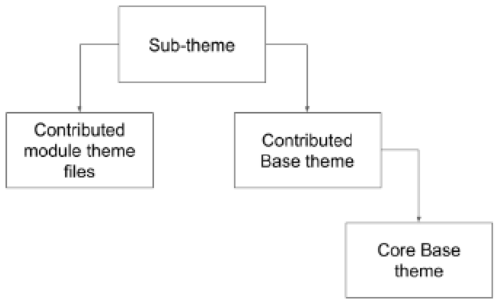


Figure 3.1: Drupal's Front-end uses a Cascading Process

The process of assembling the necessary templates, CSS, JavaScript files, media, and more starts with Drupal locating the default theme set under the Appearance settings.

- If the default theme has all the information (e.g., templates) needed to create the HTML page, the process ends, and the page is sent to the browser.
- Otherwise, it checks for a declared base theme in the info.yml file that defines the theme.
- If the declared base theme has what it needs, the process ends, and the page is sent to the browser.
- If the necessary information is still missing or no base theme is declared, it checks Drupal's default base theme, Stable, for what it needs.
- If Stable has what it needs, the process ends, and the page is sent to the browser.
- Otherwise, it checks the contributed and custom modules for what it needs and then sends the page.

Just like custom modules, there are rules you need to follow in order to create a theme that will work. Start with Theming Drupal 8 at https://www.drupal.org/docs/8/theming.

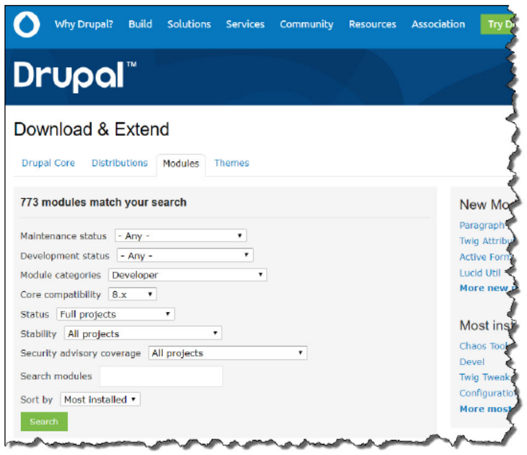
Note: If you want to buy a theme, there are some for sale online. Be careful. Some are conversions from other content management systems and might not work as you hoped. In other instances, the theme only works with the Drupal site that downloads it. Stick to vendors who contribute themes on Drupal.org, look at the way they code. If you like what you see, consider their products.

DRUPAL DEVELOPMENT TOOLS

Composer was mentioned earlier as the command line tool to use to install Drupal 8. However, there are many other tools used by the Drupal community. You can find information about them at the following Drupal.org resource pages.

- 3.2. Concept: Additional Tools
- Development Tools

Your tool options don't end with those listed on the above resources. Drupal modules that will help you develop your site can be installed. Go to Drupal.org > Build > Modules. Filter by selecting Developer from the Module categories and 8.x from Core compatibility.



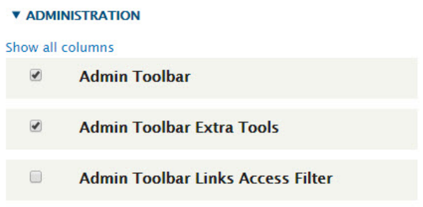
Download & Extend

773 modules match your search

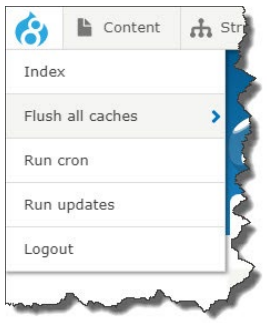
Maintenance status	- Any -
Development status	- Any -
Module categories	Developer
Core compatibility	8.x
Status	Full projects
Stability	All projects
Security advisory coverage	All projects
Search modules	
Sort by	Most installed

The Admin Toolbar is not included in the list, but it is one of the first modules you will install. The project has three modules wrapped into one install. Many "modules" are actually comprised of more than one.

You'll need to enable the Admin Toolbar and Admin Toolbar Extra Tools to gain access to quick links, such as Flush all caches.



✓	Admin Toolbar
✓	Admin Toolbar Extra Tools
	Admin Toolbar Links Access Filter



Lastly, check out Drupal 8 Debugging Techniquesfor options not included in the resources above.

Maintenance

Updating code might be the first thing that comes to mind with the word maintenance and you would be right. However, as a developer, you need to think about updating the site's content. If your plan doesn't already include such a strategy, something to consider would be a workflow. Drupal 8 core ships with the Workflows module and its companion, Content Moderation.

PLANNING UPDATES

Back to the Drupal code, there are a few basics to understand before you start updating the core and modules.

- **Drupal Core vs. Contributed vs. Custom** - Given the changes that occur with each minor update (e.g., 8.7 to 8.8), it's possible that such an update would break a contributed or custom module.
  - Review each contributed module for their updates or upgrades. If they are ready, do they change or upgrade functionality? Does the update make adjustments for changes in core?
  - Make a plan before you simply update. What contributed or custom module isn't ready for the core changes? When can each be performed without creating issues?
- **Drupal Bugs vs. API vs. Security** - What is the update doing? Is it a bug fix? A simple security update? A change to the API? Check out the instructions in Drupal docs' Chapter 13. Security and Maintenance

- **Pre- vs. Post-launch** - Running updates on core, contributed, and custom modules should be a regular part of the development plan. This is often overlooked until the end and can cause additional work.
- **Server Updates** - Don't forget updates to PHP, your operating system, your web server application, and the database you are using. As Drupal moves forward, the version of PHP and MySQL will, at times, move along with it.

UPDATE TOOLS

Drupal 8's core has a feature to update modules for you, but not core. Before you consider using this feature, read up on what the modules updates include, review the applicable text files included in the module to ensure a simple code replacement is all you need.

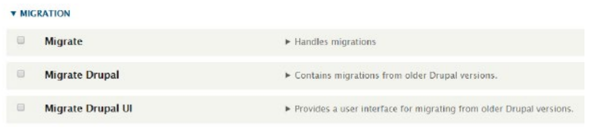
Regarding core updates, start by reviewing your options at https://www.drupal.org/docs/8/update/updating-drupal-8-overview-of-options#before-you-begin. Then, review the details of the options on Updating Drupal 8, go to https://www.drupal.org/docs/8/update.

Note: Installing Drupal has options. Using Composer to update Drupal core is preferred given the dependencies that have likely increased as development proceeded.

Migrating to Drupal 8/9

Whether you are moving from a previous version of Drupal or another source, you will need to plan and build the new Drupal 8 site first. There are three resources to review before starting the migration process.

- **Migrate API** - Review the resources on Migrate API for guidance on how to migrate from older Drupal sites and non-Drupal sites.
- **Migration Modules** - Drupal 8.7 ships with four migration modules, three in core and one in core (experimental).



▾ MIGRATION

☐	**Migrate**	▸ Handles migrations
☐	**Migrate Drupal**	▸ Contains migrations from older Drupal versions.
☐	**Migrate Drupal UI**	▸ Provides a user interface for migrating from older Drupal versions.

- **Upgrade Process** - The details of what you need to consider when migrating your legacy site data to your new Drupal 8 site is explained in detail in Upgrading to Drupal 8.

Does any of the information included in this Refcard mean you should wait until Drupal 9 is released before building your new site? No. Drupal 9 will continue as Drupal 8 has, with incremental development to Drupal 10. Hear what Dries Buytaert has to say about moving to Drupal 9.

Conclusion

Remember, think in Drupal. Be open to the fact that there is more than one way to develop/configure a Drupal site and you can learn from others before cutting custom code. Lastly, never assume that Drupal and/or its contributed modules can't meet one or more of your requirements. You would be amazed at what you can accomplish with Drupal 8.

Written by **Cindy McCourt,** Consulting

Cindy McCourt is a Drupal planner, builder, and trainer with the Promet Source Drupal Training Practice. She built her first site with Drupal 4.5 and didn't look back. In 2011, she shared her planning and building experience in the book, Drupal: The Guide to Planning and Building Websites. She also coauthored Drupal 8 Explained and Drupal 7 Explained with Steve Burge.

Devada
600 Park Offices Drive
Suite 150
Research Triangle Park, NC

888.678.0399   919.678.0300