Open Source Solutions: Wordpress vs. Drupal

The NewCity Development Perspective

If you're considering your next website project and trying to decide what open source solution to go with, there's a lot to consider. We've created this guide based on our experience working with clients in both Wordpress and Drupal and our research into each platform. This document covers our thoughts and findings in the following areas:

Note: This document was compiled during the summer of 2018. Both Wordpress and Drupal go through regular updates, so some of this information may change.

Platform Basics

	Drupal	Wordpress
First Release	2000	2003
Creators	Dries Buytaert	Matt Mullenweg Mike Little
License	GPL 2, strict about it	GPL 2, relaxed about it
CMS Usage Statistics Rank (Top 100k Sites)	2nd (8%)	1st (41%)
Managed By	The Drupal Association	The Wordpress Foundation
Associated For-profit Company	Acquia	Automattic

Wordpress is an open-source PHP-based content management system created by Matt Mullenweg and Mike Little and released in 2003. According to CMS Usage Statistics, Wordpress holds the largest part of the CMS market share across all brackets, although this may be driven in part by its popularity among small businesses, Internet marketers, and bloggers. Wordpress is associated with, although not owned or managed by, Matt Mullenweg's company "Automattic."

Drupal is an open-sourced PHP-based content management system created by Dries Buytaert and released in 2000. According to CMS Usage Statistics, Drupal is the second most popular CMS across most market segments, but it runs a distant second to Wordpress. Drupal is associated with, although not owned or managed by, Dries Buytaert's company Acquia.

Platform Stability & Security

	Drupal	Wordpress
Versions Supported for Security Updates	Most recent two	Most recent
Can Migrate Between Major Versions	In theory	Yes
Stability	Recovering from platform rewrite	Upcoming overhaul, possible breaking changes
Next Major Release	9, unannounced	5, behind schedule
Security Team	~ 30 people, core & some contributed plugins	~ 50 people, focus on core
Automatic Updates	Planned	Yes
Regular Update Window	Yes (Wednesdays)	No

Although both platforms are mature, both are changing rapidly.

Wordpress upgrades have a reputation for being fairly smooth, but much of the functionality required for enterprise-scale configuration and deployment means minor point upgrades can be potential support headaches if plugin developers are not keeping up with the changes. Some point upgrades have been so significant that they broke many websites without much warning. Wordpress is currently preparing to release a significant overhaul of its content editor, which may involve

some breaking changes and significantly change both how Wordpress is used and how sites are implemented in the system.

Drupal has traditionally required re-implementations of sites with each full-version upgrade, which has left some large institutional users of Drupal stuck. Drupal 8 was rewritten using the Symfony framework and object-oriented PHP in an effort to make major version updates and migrations simpler. his theory will not really be tested until the next major version is released.

Both **Wordpress** and **Drupal** are open-source systems, which leaves them more vulnerable to attacks based on code analysis than closed-source, proprietary systems. On the other hand, both tools can draw from a small army of volunteers that seek out and close these security issues on their own.

Whether proprietary or open-source, all CMSs that dynamically build pages on demand using server-side code and database connections are vulnerable to hacking; server infrastructure needs to be maintained.

Wordpress has a reputation for being an insecure platform in large part because the platform is so often used by individual publishers and small businesspeople who may not have the resources or understanding to keep these systems up-to-date.

Wordpress has a 50-person security team who focus on core security updates, providing training, and documenting best practices for Wordpress plugin developers. Security releases happen when needed. Minor releases are security releases only. Site managers can opt in to automatic updates of Wordpress core, which helps mitigate security issues for many people, but the potential for breaking changes

encourages many enterprise-level Wordpress users to disable this feature.

Drupal has had several high-profile security incidents typically called "Drupalgeddon." Drupal responded to these threats by announcing ahead of time through multiple channels that vital security updates were being prepared and that site maintainers needed to apply the updates immediately; in some cases they provided security-only updates for older site versions.

Drupal has a 30-ish person team devoted to security (https://security.drupal.org/team-members). Stable modules in the Drupal ecosystem are verified and checked for security issues (https://www.drupal.org/node/475848) and advisories are announced during the maintenance windows, announcing modules that fail these checks. Drupal schedules a regular window (usually Wednesdays) to make scheduling maintenance and patching easier. One place where Wordpress does better than Drupal is the application of automatic updates, though the Drupal core development team is currently focusing on this — https://www.drupal.org/about/strategic-initiatives.

It is important to update both CMSs as soon as new security patches are released. In **Wordpress** this typically means having the most recent version. **Drupal** still provides security updates for version 7 as well as version 8 and has plans to provide continued security updates for 8 once 9 is released, so keeping up to date with the most recent dot-version of either of these major versions is sufficient. Security updates are discontinued for older versions of Drupal three months after the latest major version comes out.

Content Types / Data Storage

	Drupal	Wordpress
Field Manager	Yes, core	Yes, plugins (ACF)
Content-type Nesting	Yes	No
Dynamic Fields Based on Data	No	Yes
Structured (Typed) Storage	Yes	No
Content Query Builder	Yes, core (Views)	Yes, plugins (WPQuery UIs)
Content Tagging	Yes	Yes
Page Builders	Yes, core*	Yes, plugins; (core in 5.0)
Data Migration	Yes, core	Yes, several plugins
Custom URL Patterns	Yes, plugin	Yes, plugin

Wordpress and Drupal address content types using significantly different strategies. This is one of the biggest differences between the two systems and affects the available feature sets, ease of migration, plugin development, and site architecture

Wordpress content types typically have a title, a summary, and a body. All other fields are commonly referred to as “post-meta” and are stored as key-value pairs in a single table in the database. All non-binary field data are stored in text format.

While this is extremely inconvenient from a data querying, data portability, and database performance perspective, it does let users design more fluid Wordpress content type. It is possible to set up Wordpress so you can change content types mid-stream. You can even change the fields presented to content managers based on data entered into other fields.

The Wordpress approach of storing all of a page/post's content and metadata as fields attached to that page results in a generally self-contained unit that can be moved or duplicated with ease (albeit usually with the help of a plugin). This is often closer to many content authors' mental models of how "web pages" work.

Drupal's approach is pretty close to opposite of Wordpress'. Wordpress stores every custom field in the same table; Drupal creates a table for each custom field. This means field data is stored in a format suitable to the content and the database can be relied on for sorting and filtering this content correctly. Fields can also be reused between content types, making it easy to build database queries across content types.

While this means data is simpler to relate and query than it is in Wordpress, it means content types are more strictly structured and it is impossible to change the content type of a node in mid-edit. NewCity's approach to flexible content types in Drupal is to take advantage of Drupal's entity system. This lets us create content type fragments that can be added to individual pieces of content as the need arises. But since these pages are assembled from smaller groups of content, some of it potentially referenced from other pages, it can be difficult to duplicate a page to create similar pieces of content.

Special note about data migration

It is possible to automate data migration from one CMS to another, but how much labor it requires depends on the orderliness of data at the source and the ability of the source CMS to produce a structured data output format in some fashion.

Although there are some exceptions, generally speaking it is much easier to write a migration for Drupal than Wordpress because Drupal provides a better mechanism for maintaining content relationships.

Theme Creation

	Drupal	Wordpress
Theme Language	Twig	PHP (core) Twig (plugin)
Theme Strategy	Very hierarchical	Page-based
Executable PHP in Templates	No	Yes (core) No (Twig)
Template Fallbacks	Yes	Yes (page, archive only)
Theme Inheritance	Yes, required	Yes, optional

Both content management systems can use a templating language called Twig. Twig support is native (and the default) in Drupal 8. Wordpress supports Twig with a plugin. Twig is a powerful and expressive templating language that has significant benefits over PHP. It is also the templating language for our pattern library systems.

In vanilla **Wordpress,** a template is a PHP file that defines all rendering behavior for a particular page or post type, archive type, or even an individual post. In the typical NewCity installs, a “template” is two files: a PHP file that defines the name of the template and accesses and modifies data from the database, and a .twig file that accepts variables and defines the HTML that they will be rendered into. In more advanced situations, the same PHP file can direct data to different .twig files based on conditional logic (e.g., direct all “page” posts to the “page.twig” render template, except for the homepage, which should go to “homepage.twig”).

Each post type, including the “page” post type, has a default template as well as the ability to choose a different template during post creation. Changing a post’s template can affect not only the front-end rendering of the post, but also the set of content fields on the post’s edit screen (mostly controlled with the Advanced Custom Fields Pro plugin).

Templates can also be set up for automatic use based on almost any post attribute, such as taxonomy terms.

Drupal 8 uses Twig as the default templating language and extends the language with several filters and functions that work with the Drupal API. Drupal’s templating architecture is very granular with even the simplest page loading dozens of templates. Templates are provided at the page, page region, content structure (“entity”), and field level. Templates can also be provided for specific contexts — for example, Drupal enables you to provide different templates for the same field depending on display mode and entity type.

In addition to the theming functions, Drupal provides many opportunities in the administration area to change the presentation. NewCity tends to avoid these features in favor of making changes in templates because relying on Drupal’s admin presentation tools distributes the design implementation over potentially thousands of configuration screens. It also removes templates from source control and makes it more difficult to police and control design.

Third-party Ecosystem

	Drupal	Wordpress
Centralized Distribution	Yes	Yes
Premium Plugins	No	Yes
Pre-release Distribution	Yes	No
Community Philosophy	Collaborative	Competitive

Most **Wordpress** plugins are distributed through the main Wordpress site; they can be searched and downloaded within the content management system, but most enterprise hosting is configured in a way to prevent this on live sites. The Wordpress plugin developer community is highly competitive, so there may be several plugins available to accomplish a specific task. Plugins also have many opportunities to inject promotional messages, advertisements, notifications, or other alerts into the administrative screens.

Some plugin developers have created “pro” versions of their plugins which require either a purchase or an annual subscription; at least one such plugin, Advanced Custom Fields, is used in every NewCity Wordpress implementation.

Another excellent example is the “User Role Editor,” which lets you edit user capabilities on roles. It is advertising-supported in the Wordpress admin screens; the pro version removes ads, adds many more functions, and costs between $29 and $159 annually depending on the number of sites served.

Plugins distributed through Wordpress.org have been leveraged as an attack vector; in some cases, common plugins have been sold to new maintainers who take advantage of the plugin's wide distribution to inject advertising, manipulate Google search results, or spread malware. (See also: https://www.theregister.co.uk/2017/12/20/backdoor_wordpress_captcha/)

Wordpress plugins are distributed as release only, so there are no opportunities to install beta software through typical channels.

Drupal also distributes plugins through the main Drupal website, but the developer culture rewards collaboration over competition. If two plugins do basically the same thing, the maintainers of those plugins will be encouraged (some might say "hounded") to merge their efforts. As a result, most mission-critical plugins are the *only* choice for that task, and those plugins may have dozens (or even hundreds) of maintainers. In addition, the community does not tolerate premium plugins — if someone offers a premium version of the plugin, a community member will purchase it, then add the source code to the main Drupal.org website to ensure it is free for everyone to use. (See "Why Paid Drupal Modules Fail.")

Perhaps as a side-effect of this collectivist approach to module development, Drupal plugins are frequently "permanently pre-release," and often people are running patched plugins. Most of our Drupal builds have one or more pre-released and/or patched plugins running in production.

Workflow (Roles & Permissions)

	Drupal	Wordpress
Role-based Permissions	Yes	Yes
Custom Roles	Yes (core)	Yes (plugin)
Customize Role Capabilities	Yes (core)	Yes (plugin)
Custom Workflows	Yes (core)	Yes (plugin)

Roles & Permissions

Wordpress' "out-of-the-box" user roles were originally created to accommodate blogging sites with multiple authors. The permissions system is very hierarchical, and each role includes the capabilities of the role below it while adding additional capabilities. For example, a *contributor* can only write and manage posts that they created themselves, and can not publish any posts. An *author* can manage their own posts, and they *can* publish them. An *editor* can manage and publish *everybody's* posts.

With the help of certain plugins like "User Role Editor" it is possible to create new roles or edit existing ones. Other plugins allow users to be assigned multiple roles, which is not allowed by default. This allows developers to set the site up so that roles can be granted permission based on content type. For example, a certain role could grant edit access to news stories but not to faculty bios.

Out of the box, Wordpress workflows are limited to a rudimentary approval workflow; more advanced or custom workflows will require research into the appropriate plugin or plugins. NewCity does not have

a go-to solution for creating more advanced or custom workflows in Wordpress.

Drupal's workflow has fewer roles initially, but you can create new roles, configure roles, and assign members to multiple roles without any additional plugins. Every plugin has an opportunity to check role capabilities and create their own, so those frequently come with their own permissions.

For many years, the go-to solution for custom workflows in Drupal was a plugin called "Workbench." In 2018 many workflow functions were added as a core Drupal module of the same name. Workbench lets users make custom workflows by creating and organizing document states and transitions. We have not yet had an opportunity to use the new core module on a production site.

Neither CMS is particularly good at "section" based permissions — for example, giving a user role permission to edit anything along a particular logically-grouped set of pages at or below a specific URL structure. This is generally because site "sections" represent a more conceptual than programmatic group of content, which may span multiple content types or pieces of site functionality.

Single Site vs. Multisite

Wordpress Multisite is a specific configuration of Wordpress wherein multiple Wordpress sites are contained within the same database. Sub-sites can be folders within a single domain structure ("https://mainsite.edu/sub-site/"), subdomains ("https://sub-site.mainsite.edu"), or standalone sites ("https://sub-site.edu"). To administrators and editors of individual sub-sites, the interface is essentially the same as any stand-alone Wordpress site, while a multisite-specific "super admin" has access to tools that affect all sites across the multisite network. This allows for centralized control over what themes and plugins are available, disabled, or even required for all sub-sites or for certain subsets of them.

Access to data across sub-sites is limited, complicating tools like site search and consolidated directories. However, this is still a greater amount of cross-site data sharing than would be possible with Wordpress sites hosted in completely separate Wordpress installations.

Unfortunately, Wordpress multisite's single database approach results in a very large database that is difficult to navigate and a collection of sites that are hard to split apart should there be a need to do so. Multisite Wordpress also requires a specialized server configuration that is not supported by some major managed hosting companies and comes with a high extra cost at others.

For the most part, **Drupal** multisite works in a similar fashion to Wordpress; many sites are served by a single installation of the CMS.

Due to a number of complexities with multisite installations, we generally recommend managing multiple single-site installations with

automated deployment of updates managed through local scripting or custom upstreams.

Wordpress vs. Drupal Selection Recommendations

Both Wordpress and Drupal are solid, mature content management systems with large and dedicated followings; both have significant presence in the higher education market. There is also significant overlap between the two — for certain site designs and implementations, the difference between the two may just be a matter of team preference.

Though the overlap is significant, there are many good reasons to choose one over the other. NewCity recommends and implements both content management systems largely because we have found neither to be a perfect choice in all circumstances.

In general, we recommend Wordpress for projects that:

- Have few editors (or just one!)
- Require content and presentational flexibility
- Have a more publishing-focused mindset when building a website
- Require little content reuse or content filtering

Such clients can make very good use of Wordpress' content flexibility but do not need Drupal's more advanced features, like complex user role configuration and ability to extract, repurpose, and assemble content from content records throughout the system.

We tend to recommend Drupal for projects that:

- Have a large editorial team
- May require a custom workflow
- Want to be able to repurpose / reuse content in multiple places
- Tend to think of content and design as separate components
- Want to build tight associations between content (for example: faculty profiles to departments, programs, student organizations, and news stories)
- Want to be able to query the content store as though it were a database

User-friendliness vs. Complexity

Wordpress has a reputation for being very user-friendly with a “pretty” administration area. On the other hand, Drupal has a reputation for being extremely complicated with an “ugly” administration area.

There is some truth to this. But as site configurations become more complex, packing in more custom functionalities, Wordpress begins to lose a lot of its user-friendliness and attractiveness. The secret sauce behind Wordpress’s user experience is a simplistic storage framework that gives plugins free reign to modify core functionality. This is the Wild West of content management and user interfaces, but plugins can build you a beautiful (if architecturally hodgepodge) homestead.

Drupal, on the other hand, looks more unfriendly for two reasons: it exposes a lot of the complexity of the CMS up-front and its APIs encourage standardization. For simple workflows or design-based sites, this means users may be exposed to choices or features they do not

care about or need. Novice users may have a hard time adapting to the bespoke Drupal admin UI design language. It is very hard to “strip down” the content editing experience to a Wordpress level but, on the other hand, the UI is functionally consistently throughout the administration area.

Why not both?

As an institution, standardizing and supporting a single CMS makes a lot of sense. But it usually comes at the cost of increased governance and enforcement. Since institutions are diverse places, it can also make sense to support implementations of websites in either CMS. This is made simpler (not easy!) with the help of an independent technical pattern library that site integrators can use to apply the institution’s official design elements, but it may also require a larger support team versed in both content management systems.

From our perspective, this is the ideal choice — letting stakeholders who need their own sites decide which tool supports them and their staff the best. However, we recognize that this is not always feasible. In those cases, it may be necessary to choose one or the other based on the recommendations outlined above.