DRUPAL OVERVIEW

Effective Web design is driven by the need to balance flexibility and simplicity. If a system is too simple, it can only be used for a single purpose - but if it is too flexible, it may be too difficult for new users to learn.

Drupal strives to reconcile these conflicting goals by providing its users with the tools they need to make their own content management solution, while still providing some pre-built components to help them get started. Thus, it can be described both as a content management system (CMS) and a content management framework (CMF) - one system which strives to have the strengths of both, without their deficiencies.

Most CMS's are like a toy boat or truck - specific assumptions have been made about their use, assumptions that would be hard for you to override. Frameworks, on the other hand, provide you with raw materials only - you need to know a programming language and have a clear design vision to put them together.

Drupal is like a Lego kit. Skilled developers have already made the building blocks - in the form of contributed modules - that you need to create a site that suits your needs, whether that is a news site, an online store, a social network, blog, wiki, or something else altogether.

DRUPAL IN ACTION

To make the contrast between Drupal and other CMS's more concrete, consider the example of a news site. You want to be able to post news articles on the site, and you want the homepage to have a section featuring the five most recent ones. Next, you decide that you want to add a blog section, and put a list of links to the five most recent blog entries on the homepage as well.

If you were using an ordinary CMS, first you would install a plugin that handled news articles and could put short blurbs on the homepage. Next, you'd install a plugin that would track the latest blog posts and put a list of those on the homepage. Each plugin would only be responsible for tracking and managing a particular kind of content, and would remain relatively isolated from the others.

But, what happens when you have that brilliant middle-of-the-night idea, and want to blend these two functions by showing a list of blog posts about the latest news items, ordered by most active contributor first? If you're using a "toy truck" CMS, you may be out of luck, or need to hire a developer to write you a custom plugin from scratch. But through the power of the Drupal way, the way of manageable abstraction, you can whip out a kit full of parts and knock this together pretty quickly. (Hint: just use Views.) Since Drupal's modules do things in a standard way, and interface with a common underlying system, building all sorts of clever, customized features is just a matter of snapping parts together. Of course, this flexibility comes at a certain cost. While a toy truck is instantly understandable and ready to use without much thought, a modular vehicle construction kit will by nature require you to read the instruction manual first. The building blocks are out there, but you'll need to learn how they fit together before you can take a paper prototype and turn it into a full-featured website.

Drupal core, and the thousands of contributed modules that build on it, require an initial investment to learn, but mastering the Drupal way is immensely rewarding; the passionate community is a testament to its power to liberate site builders from the simplicity/flexibility dilemma. Once you've tried Drupal, you'll likely leave your toy truck and boat in the closet gathering dust.

HOW DRUPAL DOES IT

Intrigued yet? Let's take a closer look at how Drupal works.

People often think of a website as a collection of static pages, with some functions (like a blog, or a news engine) thrown in to round it out. When they go to manage their site, they are thinking in terms of a tree-like hierarchy of pages that they will go in and edit.

Drupal, on the other hand, treats most content types as variations on the same concept: a node (more on this in a moment). Static pages, blog posts, and news items (some possible node types) are all stored in the same way, and the site's navigation structure is designed separately by editing menus, views (lists of content), and blocks (side content which often have links to different site sections).

It's a lot like the separation you find in standards-compliant page coding—XHTML provides the meaningful structure of the information, while CSS arranges it for presentation. In Drupal, nodes hold the structured information pertaining to a blog post (such as title, content, author, date) or a news item (title, content, go-live date, take-down date), while the menu system, as well as taxonomy (tagging of content) and views, create the information architecture. Finally, the theme system, along with display modules like Panels, controls how all this looks to site visitors.

Since these layers are kept separate, you can provide a completely different navigation and presentation of your content to different users based on their specific needs and roles. Pages can be grouped differently, prioritized in a different order, and various functions and content can be shown or hidden as needed.

NODES: THE SECRET TO DRUPAL’S FLEXIBILITY

We don't talk about "nodes" every day, but since they are at the heart of Drupal's design, they deserve further investigation. At its most basic, a node is a set of related information. When you create a new blog post, you are not only defining its body text, but also its title, content, author link, creation date, taxonomy (tags), etc. Some of these elements will be shown by the theme layer when the node is displayed. Others are meta-data that control when the node will show up at all - such as taxonomy or publishing status.

Since each item of content is stored as a node, and contains the same basic information, each can be handled in a standard way by both Drupal core and contributed modules. This allows site builders to choose exactly where they want content to show up, and exactly how they want it to look in each case. Most of a Drupal site builder's time is spent defining what kinds of information you want to store in your nodes, and configuring the structures (menus, taxonomy trees, views, panels) in which to display them.

As suggested before, you aren't limited to a single way of presenting your site's content. You can define as many navigation schemes, custom themes ("skins" for the site), blocks (small bits of content, such as the five most recent blog articles described earlier), and feature sets as there are distinct audiences for your site.

Comments are second-class citizens in Drupal compared to nodes, but they also illustrate the Drupal way. Comments aren't just part of the blog system, since there isn't a separate "blog system." Comments can be enabled on any node type you choose - blog posts, news items, book pages (which provide basic wiki features) and any other you may create.

COLLABORATIVE AT THE CORE

Creating an informational website that broadcasts from “one to many” is something that most CMSs do right out of the box. However, where Drupal really shines is when you want to empower site users to create content, and connect with each other - moving from "one to many" to "many to many.”

With some CMS's, you can set up a blog, and you can install plugins to handle having a community of users, but what happens when you want to give individual blogs to each of your users, sorting their contents so that they can be displayed individually with their own skins, while also generating cross-blog topical digests, top five lists, and links out to elaborate, customized user profiles? What if you want to also integrate that with forums, a wiki-like environment, and give each user their own gallery of taggable photos?

Drupal is designed from the ground up so site builders can delegate content creation, and even site administration, to users. All you have to do is define who gets to do what on your site (through user permissions), and then you can start collaborating.

GET STARTED QUICKLY

Customize extensively

Drupal's flexibility is incredible, but installing it is surprisingly easy. With a simple FTP upload and a few short web-based configuration questions, you can connect with your database and have your first Drupal site up and running within an hour.
Pick one of the included themes, and just start adding content. Do you want to have visitors log in? Switch authentication on or off. Want to switch on some of the included tools? Turn on forums; enable commenting on node types; turn on the book module for wiki-like collaboration; create forums and polls;

use taxonomy to give site content structured, hierarchical categorization or free-form tagging.
Do you want your own skin applied to the site? Drupal's theme system uses tiny snippets of PHP that you can insert into the appropriate spots in your design to replace your placeholder Lorem Ipsum text with dynamic content. Drupal's generated markup is clean, standards-compliant XHTML. No old-school tables. No cruft. No kidding.

THE DRUPAL FLOW

If you want to go deeper with Drupal, you should understand how information flows between the system's layers. There are five main layers to consider:

1. At the base of the system is the collection of nodes—the data pool. Before anything can be displayed on the site, it must be input as data.
2. The next layer up is where modules live. Modules are functional plugins that are either part of the Drupal core (they ship with Drupal) or they are contributed items that have been created by members of the Drupal community. Modules build on Drupal's core functionality, allowing you to customize the data items (fields) on your node types; set up e-commerce; programmatically sorting and display of content (custom output controlled by filters you define); and more. There are thousands of different options within the fast-growing repository of contributed Drupal modules. They represent the innovation and collaborative effort of everyone from individuals to large corporations.
3. At the next layer, we find blocks and menus. Blocks often provide the output from a module or can be created to display whatever you want, and then can be placed in various spots in your template (theme) layout. Blocks can be configured to output in various ways, as well as only showing on certain defined pages, or only for certain defined users. **Menus** are navigators in Drupal, which defines the content coming on each defined menu path (relative url). Menus are core element of drupal which gives all the pages created in Drupal
4. Next are user permissions. This is where settings are configured to determine what different kinds of users are allowed to do and see. Permissions are defined for various roles, and in turn, users are assigned to these roles in order to grant them the defined permissions.
5. On the top layer is the site theme (the "skin"). This is made up predominantly of XHTML and CSS, with some PHP variables intermixed, so Drupal-generated content can go in the appropriate spots. Also included with each theme is a set of functions that can be used to override standard functions in the modules in order to provide complete control over how the modules generate their markup at output time. Templates can also be assigned on-the-fly based on user permissions.

This directional flow from bottom to top controls how Drupal works. Is some new functionality you want not showing up? Perhaps you uploaded the module into the system but have not activated it yet, and this is making everything downstream non-functional (as in "A" in the diagram above).
Maybe the module is installed and activated, but you still don't see what you want on your site. Did you forget to place the block, as in "B"? Or are your user permission settings conflicting with what you want and your users are not set to see the output as in "C"?

Additionally—as mentioned earlier—getting the kind of granular control you want over the details of the XHTML module outputs requires understanding this flow. Are you using a module that does exactly what you want, only you wish the markup was just a little bit different? Maybe you'd like it to use different tags, or you'd like to assign a CSS class to something? You accomplish this by copying the output function from the module and pushing it up to the functions document in your theme. Modify the code there, and when the system goes to output, it will see your customized function and use that instead.