

Acquia
EXPERIENCE DIGITAL FREEDOM

THE ULTIMATE GUIDE TO DRUPAL 8

8.8 — December 2019



CONTENTS

- 3 **Getting Started with Drupal**
- 8 **Content Authoring Experience**
- 11 **Media Management**
- 14 **Workflow**
- 19 **Layout Builder**
- 23 **Multilingual**
- 29 **Mobile Experience**
- 34 **Building And Managing Your Site**
- 40 **Front-end Developer Experience**
- 45 **Back-end Developer Experience**
- 51 **The Future of Drupal 8**

Welcome to *The Ultimate Guide to Drupal*, hot off the presses for **Drupal 8.8**

Drupal 8 has a lot in store for you, whatever you do with Drupal. This e-book will enumerate the major changes, features, and updates in Drupal 8 for service providers and end users, site builders, designers, theme- and front-end developers and for module and back-end developers.

GETTING STARTED WITH DRUPAL

Out-of-the-Box Demonstration

As you're installing Drupal 8, one of the first things you'll notice is that it offers an option during installation to select a demo option for Umami Food magazine.

Select an installation profile

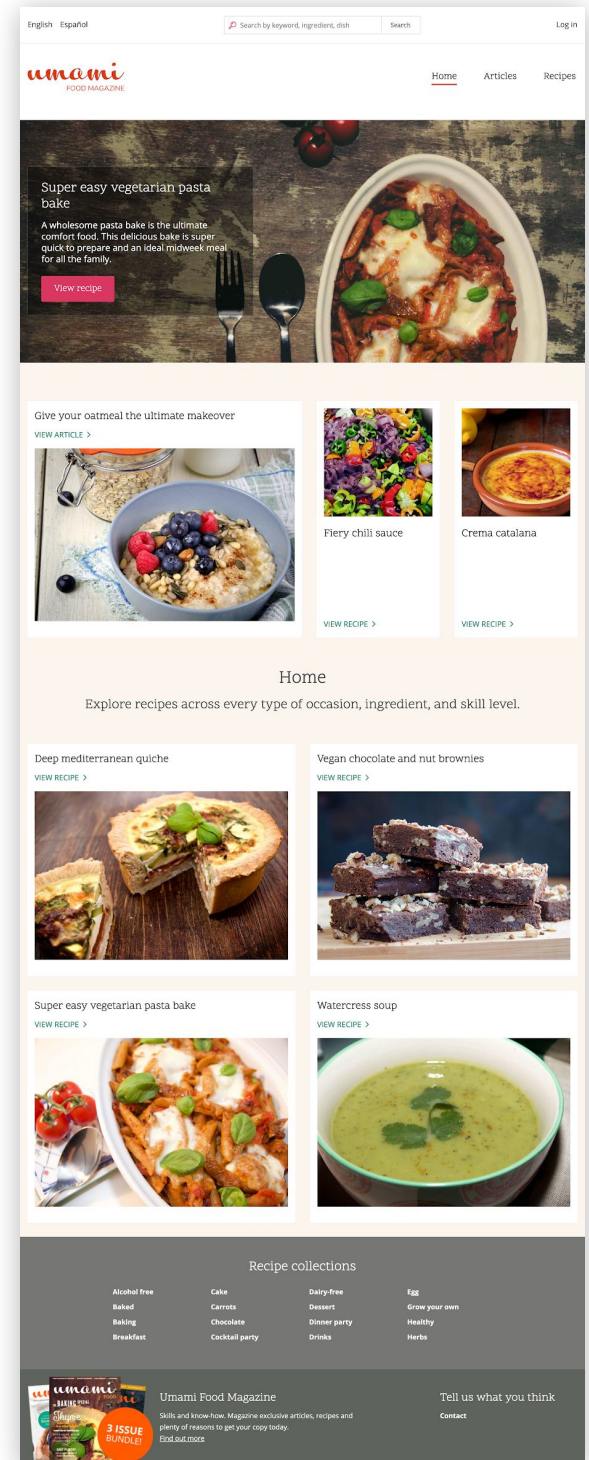
- Standard
Install with commonly used features pre-configured.
- Minimal
Build a custom site without pre-configured functionality. Suitable for advanced users.
- Demo: Umami Food Magazine (Experimental)

⚠ This profile is intended for demonstration purposes only.

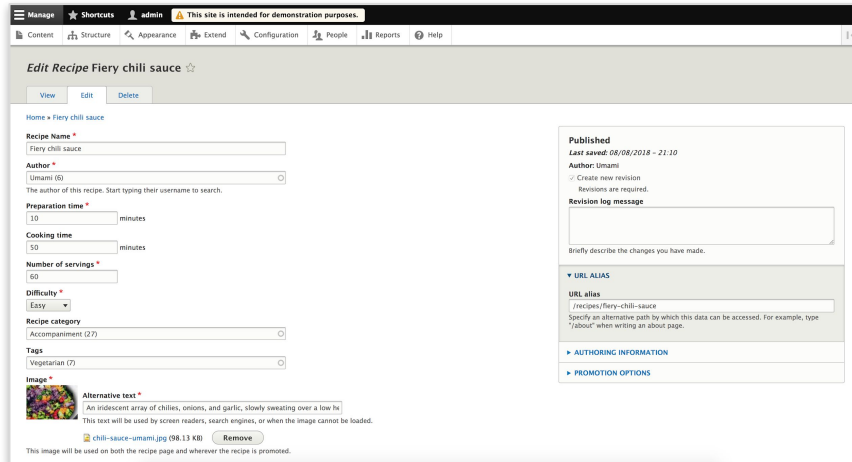
Install an example site that shows off some of Drupal's capabilities.

Save and continue

While "Demo: Umami Food Magazine" is not intended as a starting point for new Drupal sites (the Standard profile, as displayed in the image, is usually best) this option can be very useful the first time you install Drupal to get an idea of what it can do. When the "Demo: Umami Food Magazine" installation completes, a wide range of Drupal's features and functionalities will be on display, including structured content, layouts, content moderation and multilingual capabilities. Umami also shows off new features that are unique to Drupal 8, including Layout Builder and Media Library.



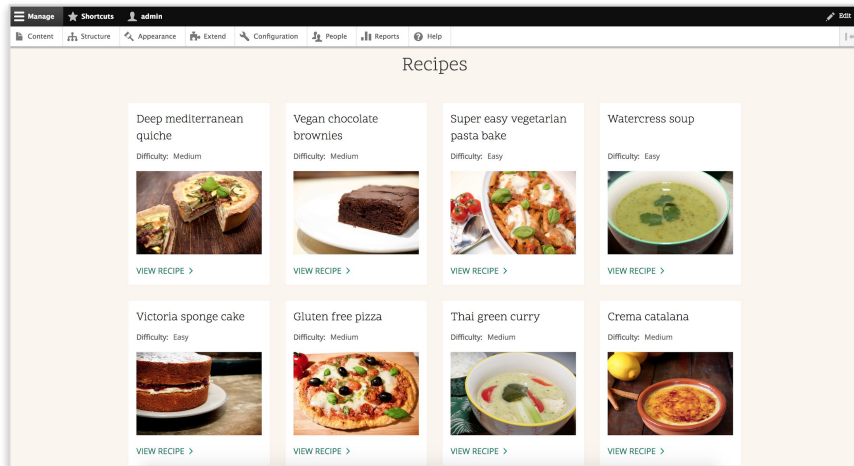
A Brief Introduction to Drupal Terminology and Basic Capabilities



As a magazine website, Umami is set up with some default **content types** — Article, Basic Page and Recipe — which contain relevant **fields** for capturing structured data, such as Cooking Time (Numeric), Difficulty (List), Image (File), and Tags (Reference).

Data within these fields are then exposed as **views**, which are essentially "smart" content listings, for example, a sidebar **block** of featured articles, or a gallery page of the most recent recipes on the site.

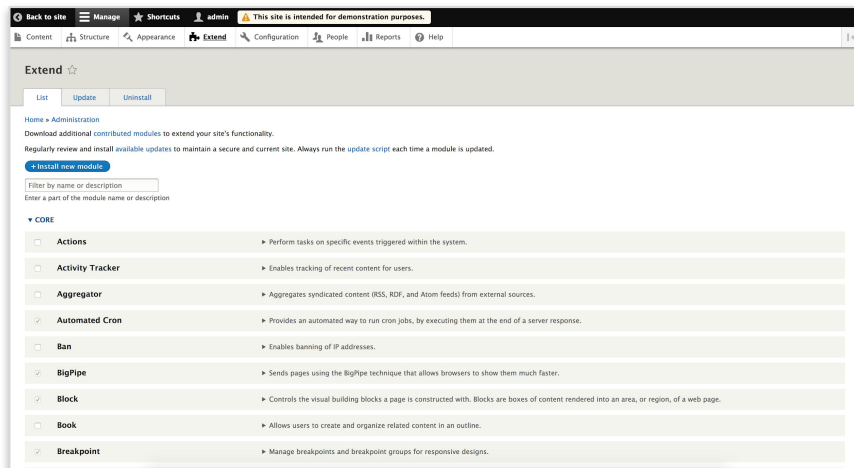
A Brief Introduction to Drupal Terminology and Basic Capabilities



Connecting all of these capabilities is a nicely designed **theme** (custom design) for the site to make it visually look like an actual food magazine. Multiple **user roles**, each with distinctly assigned **permissions** along with sample users, are available to produce the magazine. Authors can create draft articles and recipes, while editors can review, publish and archive them.

A number of modules that are included in the Umami profile are enabled to provide various capabilities, such as a WYSIWYG editor, and you can further extend Drupal by turning on others or downloading additional modules from Drupal.org.

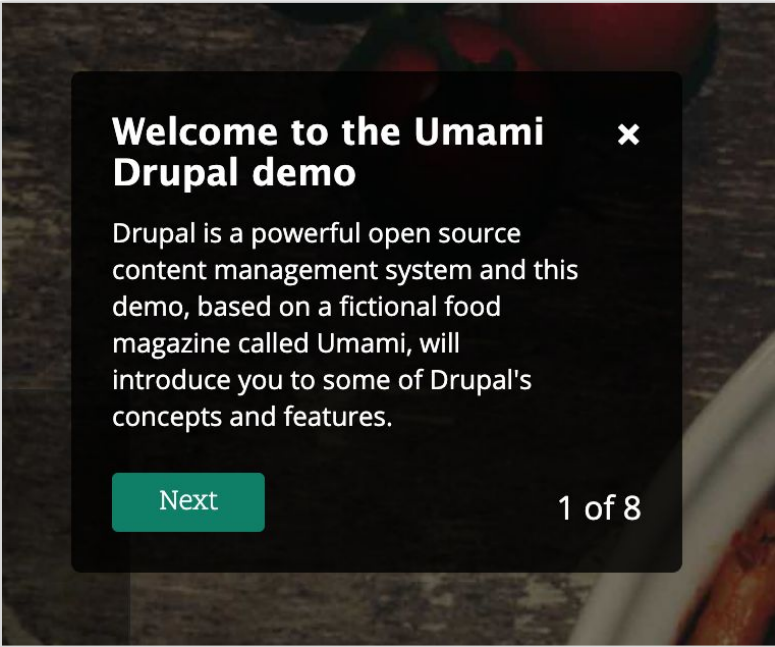
If you ever need a quick review of this overview and / or terminology, Umami provides a tour that is available on the homepage located in the upper right side of the Toolbar that provides in-context cues to various parts of Drupal that are on display.



Drupal 8 User Guide

DRUPAL 8 USER GUIDE

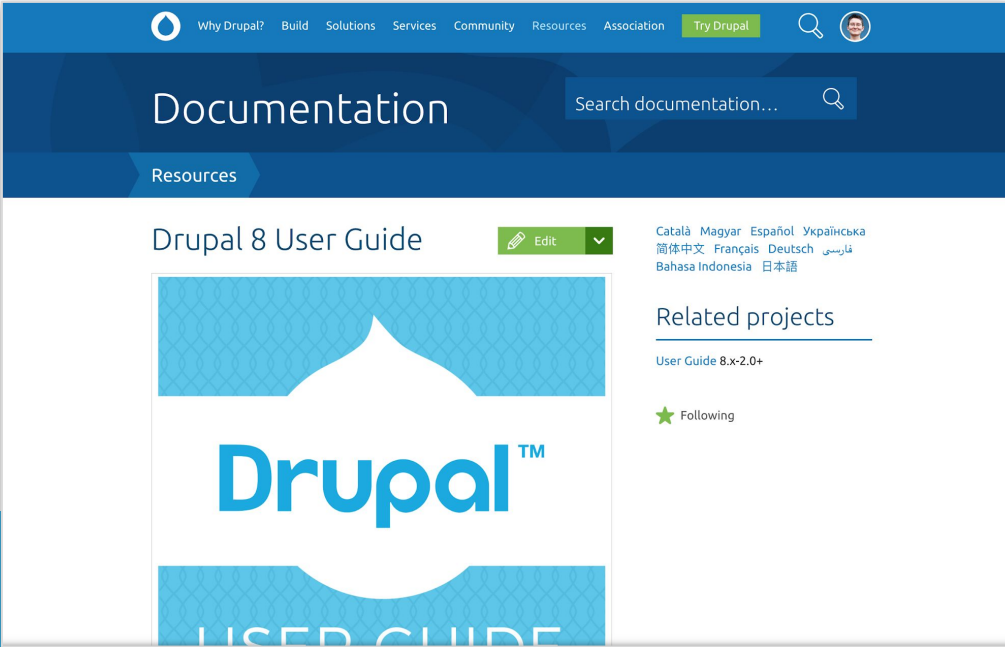
For additional information about getting started with Drupal, check out the [Drupal 8 User Guide](#), which is maintained by Drupal's documentation team. It provides an excellent overview of Drupal 8 from a site-builder perspective, introducing Drupal concepts with a project-based, step-by-step approach. It's available in several languages and downloadable as a PDF for offline viewing. Chapters have video commentary, as well.



Welcome to the Umami Drupal demo ✕

Drupal is a powerful open source content management system and this demo, based on a fictional food magazine called Umami, will introduce you to some of Drupal's concepts and features.

Next 1 of 8



Why Drupal? Build Solutions Services Community Resources Association Try Drupal

Documentation

Search documentation...

Resources

Drupal 8 User Guide

Edit

Català Magyar Español Українська
简体中文 Français Deutsch فارسی
Bahasa Indonesia 日本語

Related projects

User Guide 8.x-2.0+

★ Following

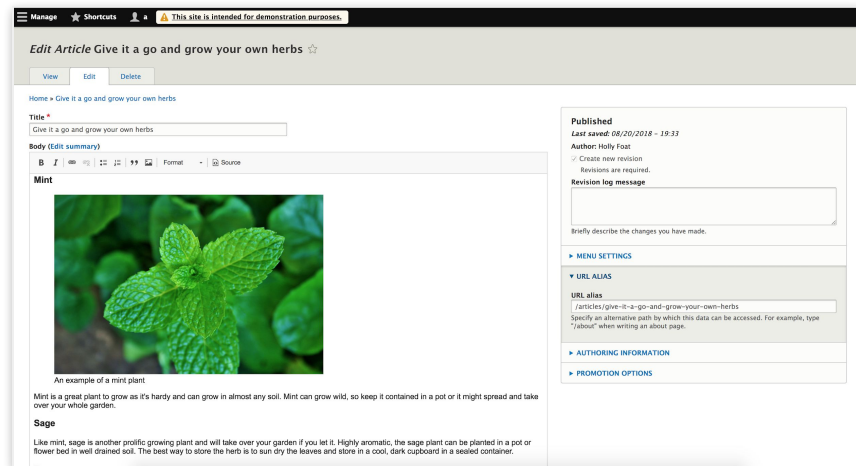
To learn more about Acquia's Drupal 8 expertise, visit dev.acquia.com

CONTENT AUTHORIZING EXPERIENCE

A major area of focus in developing Drupal 8 related to the out-of-the-box experience for content authors and editors — the folks who actually use Drupal websites every day. Here are some of the changes you'll see.

WYSIWYG Editor

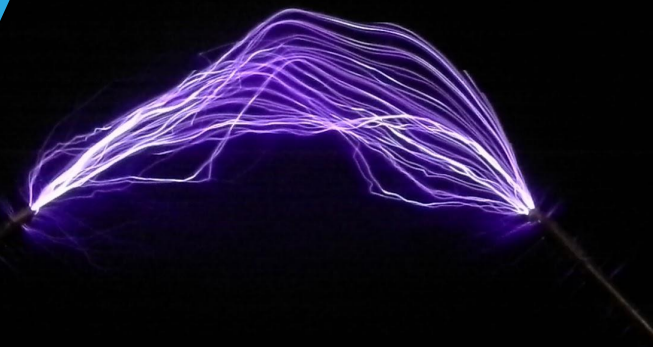
Drupal 8 ships with the [CKEditor](#) WYSIWYG editor in the default installation.



In addition to supporting what you'd expect in a WYSIWYG editor — buttons for bold, italic, images and links — it supports extras, such as easily editable image captions, thanks to CKEditor's new [widgets](#) feature, developed specifically for Drupal's use. It is fully integrated into Drupal 8, from Drupal-styled dialogs to utilizing Drupal's user roles and permissions and image management, and it [ensures that we keep the benefits of Drupal's structured content concepts in our WYSIWYG implementation.](#)

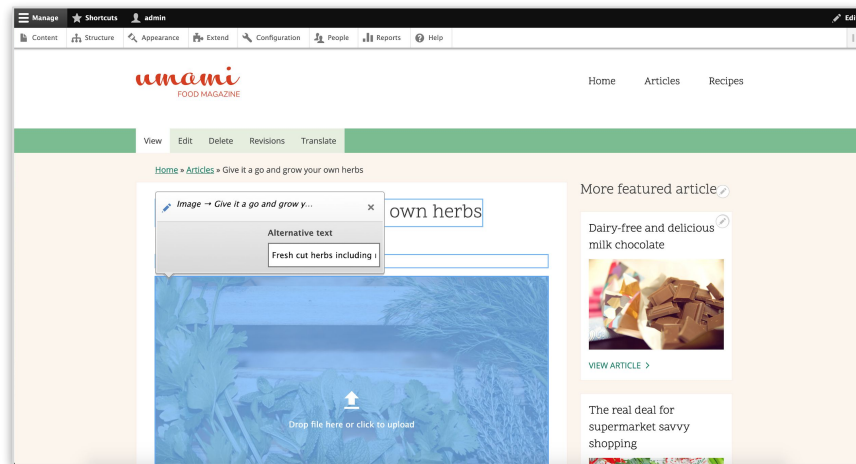
Drupal 8 also sports a drag-and-drop admin interface for customizing the WYSIWYG toolbar. Adding and removing buttons in the customized toolbar automatically syncs the allowed HTML tags for a given text format. Buttons are contained in "button groups" with labels that are invisible to the eye but can be read by screen readers, providing an accessible editing experience for visually impaired users. Though core only supports CKEditor, Drupal 8's Text Editor module wraps around the WYSIWYG integration, so other text editors, libraries and contributed modules can be tightly integrated, as well.

Though core only supports CKEditor, Drupal 8's Editor module wraps around the WYSIWYG integration, so other text editors, libraries and contributed modules can be used and tightly integrated.

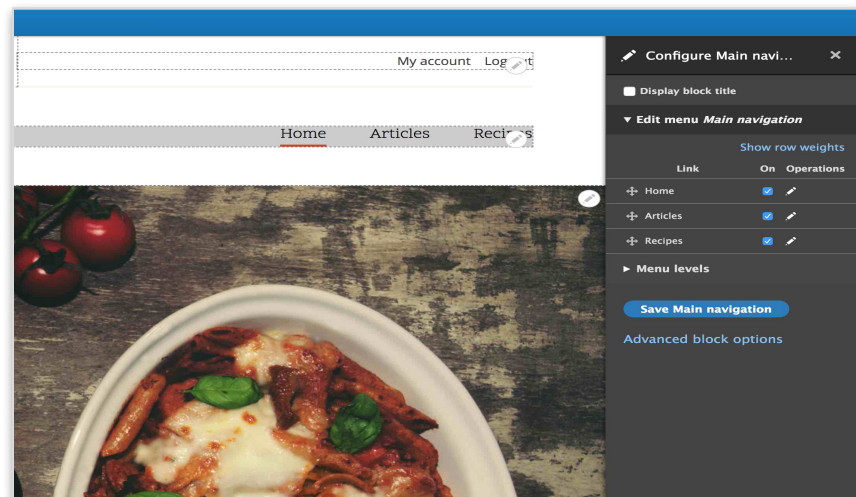


In-Place Editing

Drupal 8's Quick Edit in-place editing feature allows editors to click into any field within a piece of content that appears on the front-end of the site and edit without having to visit the back-end editing form. User profiles and custom blocks are just some of the areas you can use the in-place editing feature.



Additionally, the Settings Tray module allows for quick configuration changes, such as the title of a sidebar block or the number of records shown. Other modules also leverage the Settings Tray module to expose their configuration, including Drupal 8 core's Layout Builder and Workspaces modules.



Claro Administrative Theme (Experimental)

Drupal's current default administrative theme, Seven, was designed in 2009 and could use a fresh coat of paint after a decade. Enter the new experimental theme, Claro, which not only provides a modern design following standard patterns, but also greatly enhances the accessibility of Drupal's backend.

The screenshot shows the Drupal Administration interface for the 'Appearance' section. At the top, there are navigation links for 'Home', 'Administration', and 'Appearance'. Below this, there are tabs for 'List', 'Update', and 'Settings'. A warning message is displayed: 'Warning message: There was a problem checking available updates for Drupal.' Below the warning, there is a section for setting the default theme, with a '+ Install new theme' button. The 'Installed themes' section shows two themes: 'Umami 8.8.0-dev (default theme)' and 'Claro 8.8.0-dev (administration theme, experimental theme)'. The 'Claro' theme is currently selected as the default, indicated by a checkmark next to 'Set as default'.

Home > Administration

Appearance

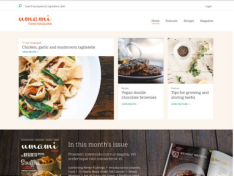
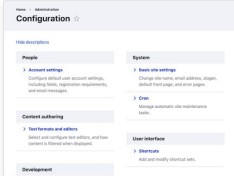
List Update Settings

Warning message
There was a problem checking [available updates](#) for Drupal.

Set and configure the default theme for your website. Alternative [themes](#) are available.
You can place blocks for each theme on the [block layout](#) page.

+ Install new theme

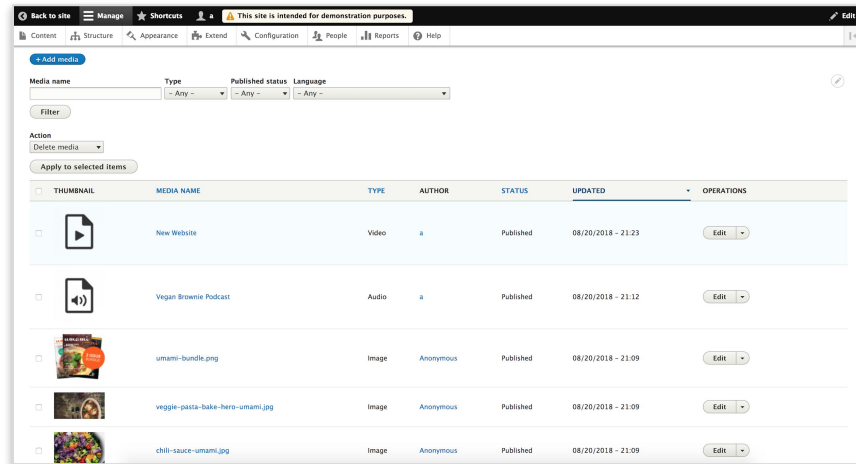
Installed themes

 <p>Umami 8.8.0-dev (default theme) The theme used for the Umami food magazine demonstration site.</p> <p>Settings</p>	 <p>Claro 8.8.0-dev (administration theme, experimental theme) A clean, accessible, and powerful Drupal administration theme.</p> <p>Settings <input checked="" type="checkbox"/> Set as default</p>
--	--

MEDIA MANAGEMENT

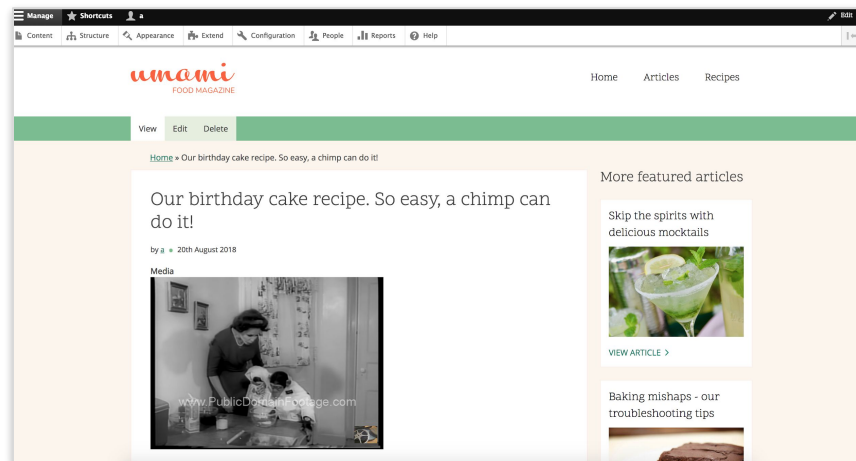
Drupal 8 offers richer media management capabilities than previous versions.

Additional Media Type Support



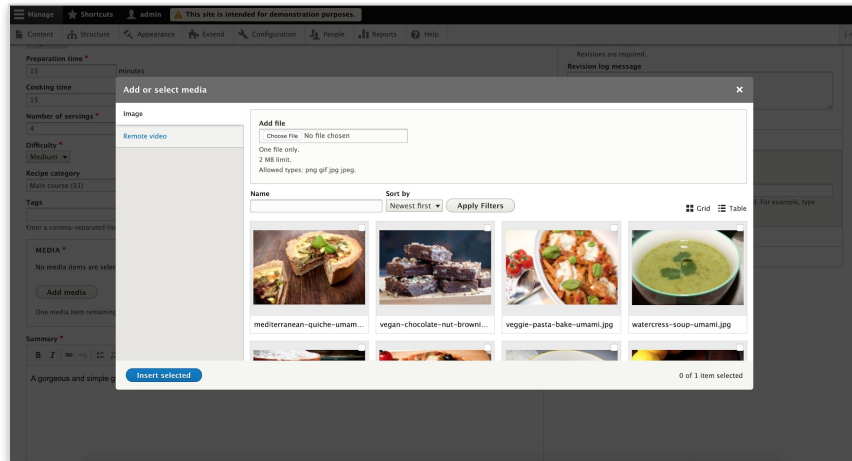
While Drupal has long supported images and generic files, Drupal 8 expands on this functionality with a generic Media field, with support for local audio, video, images and generic files.

Embedding Remote Content



Drupal 8 also ships with [oEmbed](#) support, which allows the embedding of external media, such as [YouTube](#) and [Vimeo](#) videos.

Media Library



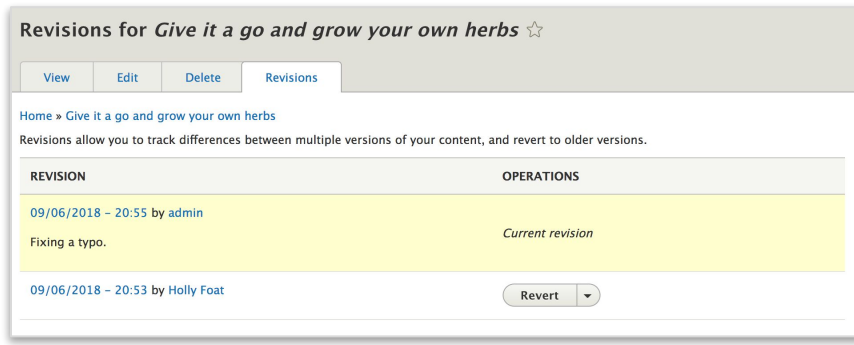
The Media Library module lets you select from existing media to place into your post, as well as upload new items directly into the library. You can choose between media types to browse and upload (for example Images vs. Remote video). There are also accessibility enhancements such as the ability to toggle between grid and table view. As of Drupal 8.8, you can now open the Media Library from your WYSIWYG editor, too!

The media library is powered by views, which allows site builders to customize its display and what sorting and filtering options are available.

WORKFLOW

A number of Drupal 8's most exciting new features for content authors relate to workflow and moderation.

Content Revisions



Revisions allow tracking of content edits and also provides a log report to inform other content authors about why a change was made. The system keeps track of all revisions and allows users to revert to prior versions if the need arises.

Content types and custom block types are configured by default to require revisions. In the backend of your Drupal installation, there is revisions support for menu links and taxonomy terms as well.

Workflows

The lifecycle of content production and management of media such as blog posts begin as outlines that become drafts, during which there may be multiple reviewers to edit and provide feedback to prepare the posts for publication.

After publishing, workflow processes may include translation copy and archiving the content. Drupal is generally utilized for more complex scenarios. For example, a shop's inventory management of products that are backordered arrive in inventory and are then sold, or users that begin as newcomers and can increasingly gain higher status on the system.

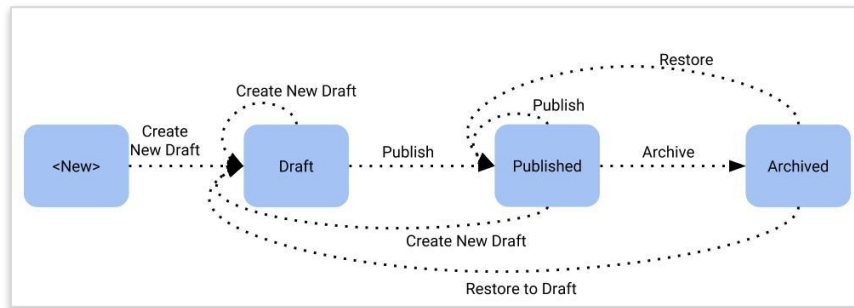
In the Workflows module it allows you to define multiple workflows, show their states and the transitions allowed between them. In the above examples, blog posts can have a publishing workflow, inventory items can have a warehouse workflow, and user profiles can have a user stature workflow, each of which has its own states.

However, these examples are theoretical as the Workflows module merely provides a framework to manage them and the behaviors should be provided by other modules. This is where the Content Moderation module comes in.

Content Moderation

By default, content in Drupal has two states: published and unpublished. Content in the published state is visible on site. Content in the unpublished state is visible only to the author and site admins. However, certain sites may have more complex publishing needs.

The Content Moderation module builds on top of the Workflows module and Drupal 8's ability to create content revisions that are not live yet but newer than the last live version. It ships with a default editorial workflow with draft, published, and archived states, which are useful for state tracking in simple publishing scenarios.



This workflow can either be modified and extended for more complex use cases, or another publishing workflow can be created if you plan to apply a more simple workflow for your blog posts and a more complex workflow for your press releases. Your workflow for press releases could be designed as the following model: new => draft => media / asset review => legal team sign-off => published.

You can also create multiple editorial roles, and restrict users' ability to move content to and from these workflow states, e.g., only admins can use the restore and restore-to-draft transitions.

The content moderation module builds on top of the workflows module and Drupal 8's ability to create content revisions that are not live yet but newer than the last live version.



Content Revisions

Label *
Editorial Machine name: editorial

▼ STATES Show row weights

STATE	OPERATIONS
↕ Draft	<input type="button" value="Edit"/>
↕ Published	<input type="button" value="Edit"/>
↕ Archived	<input type="button" value="Edit"/>

[Add a new state](#)

▼ TRANSITIONS Show row weights

LABEL	FROM	TO	OPERATIONS
↕ Create New Draft	Draft, Published	Draft	<input type="button" value="Edit"/>
↕ Publish	Draft, Published	Published	<input type="button" value="Edit"/>
↕ Archive	Published	Archived	<input type="button" value="Edit"/>
↕ Restore to Draft	Archived	Draft	<input type="button" value="Edit"/>
↕ Restore	Archived	Published	<input type="button" value="Edit"/>

[Add a new transition](#)

User interface elements are also provided on content items that are configured for the workflow, allowing content editors to run transitions. The Content Moderation module ties the states back to publishing statuses, so drafts and archived content will remain unpublished while content in other states is published. You can also define, in additional added states, their publishing status and whether content in that state becomes the default revision.

Within the Drupal admin interface, an overview for managing unpublished content that's awaiting moderation is also available. (as seen in the attached image)

Moderated content ☆

Content Files Media

Overview Moderated content

Home » Administration » Content

Title Content type Moderation state Language

TITLE	CONTENT TYPE	AUTHOR	MODERATION STATE	UPDATED	OPERATIONS
Thai green curry	Recipe	Umami	Archived	09/06/2018 - 21:19	<input type="button" value="Edit"/>
Give it a go and grow your own herbs	Article	Holly Foat	Draft	09/06/2018 - 21:17	<input type="button" value="Edit"/>

Content Staging (Experimental)

Although content moderation works great for staging single pieces of content, sites often have a need for staging multiple pieces of content at once, e.g., an article with callout blocks and video assets, and previewing the changes that will be made to content listings or landing pages.

Fortunately, the robust Workspaces module provides full content staging capabilities for these scenarios. You can define multiple workspaces such as "staging" and "live," which are copies of your site, and create content or modifications to existing content that is visible only within that workspace. Then, once the changes have been verified, you can deploy the content to another workspace at the click of a button.



This functionality is particularly useful for events where there are multiple possible outcomes (such as elections or sporting events), that require vastly different hero images, featured articles and advertisements, depending on the conclusion. Workspaces allow you to build out the entire content for either scenario in advance and then publish the lot at once, once the winner is announced. Additionally, a new feature as of Drupal 8.8 is the concept of "hierarchical" workspaces, which allow you to synchronize proposed changes across all scenarios at once.

Workspaces allow you to build out the entire content for either scenario ahead of time and then publish the lot at once.

LAYOUT BUILDER

Drupal core comes with a flexible block system to allow placing discrete pieces of content, such as a listing of who is online or an advertisement, within various page regions.



Layout Options for Content

The Layout Builder core module provides layout capabilities to content. Layout Builder is unique in offering a single, powerful visual design tool for the following three use cases:

1. Layouts for templated content.

The creation of "layout" templates that will be used to layout all instances of a specific content type, e.g., blog posts, product pages.

2. Customizations to templated layouts.

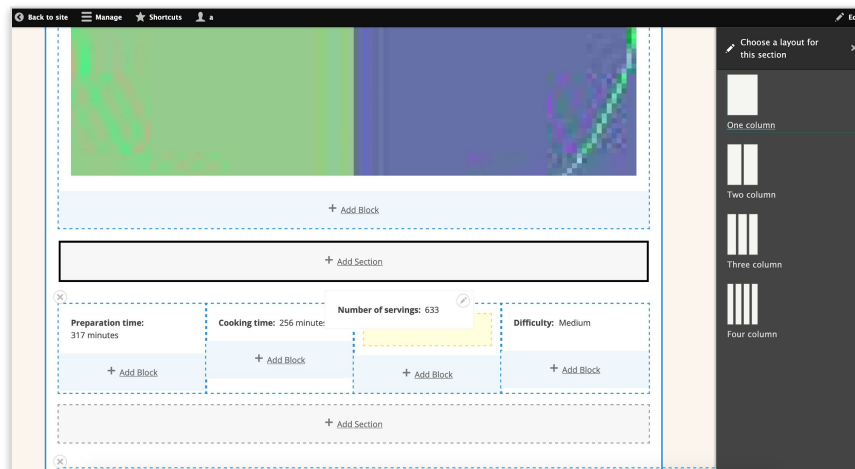
The ability to override these layout templates on a case-by-case basis, e.g., the ability to override the layout of a standardized product page.

3. Custom pages.

The creation of custom, one-off landing pages not tied to a content type or structured content, e.g., a single "About us" page.

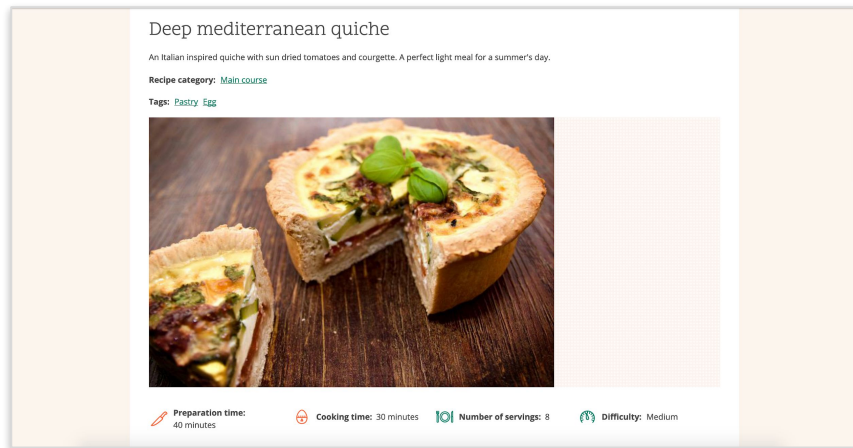
Using Layout Builder to Customize Page Display

By default, Drupal will output all fields of a content type stacked one on top of each other. Layout Builder allows the creation of one or more "Sections" (pictured in the right sidebar), each of which consists of columns that can have one or more "blocks" added to them. All fields from a content type (in this example, recipes) can be placed individually. Global blocks like Views listings or menus can be placed within the content display layout.



All details of the layout are edited in-line using a preview-first approach, though you can also toggle a "show content preview" checkbox to gain a more compact view for easier moving of exceptionally large areas. Additionally, there is ample support for assistive technologies: blocks can be placed, reordered, and moved into different sections all by using keyboard navigation. Throughout this process, ARIA-labels are present to orient screen reader users.

Here, we are changing the site-wide layout for all recipes, and placing various fields into a four-column section. When editing the layout for a content type (as opposed to an individual piece of content), mock field values are automatically generated -- this maintains Layout Builder's preview-first approach even when field content is unavailable.



The resulting four-column section can be viewed on an actual recipe on the site.

A detailed set of Layout Builder permissions is provided to meet different use cases. You can limit users' ability to customize layouts to specific content types, or only to content to which they have edit access. Another permission determines whether or not they can create custom in-line blocks.

All details of the layout are edited inline using a preview-first approach.

MULTILINGUAL



Drupal 8 comes with no less than four modules for language support, each with a different role.

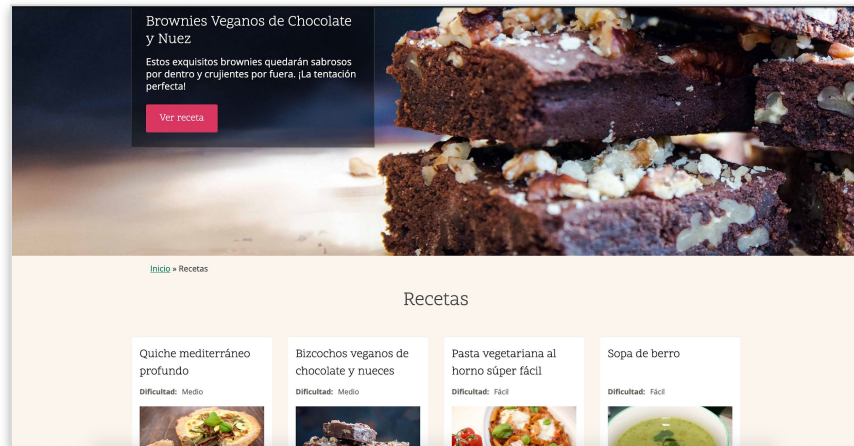
An Overview of Drupal 8's Multilingual Capabilities

Drupal 8 comes with no less than four modules for language support, each with a different role.

1. **Language** This module enables content and configuration to be produced in multiple languages in the system. If you are building an asset repository with an English interface, you will only need this module to maintain language metadata on your assets.
2. **Interface Translation** This module provides translations of the built-in interfaces of Drupal on the back end and front end.
3. **Content Translation** This feature provides translation for content items, such as nodes, taxonomy terms, menu items and user profiles.
4. **Configuration Translation** This provides an interface to translate website configuration to multiple languages.

For a monolingual, non-English website, the language and interface translation modules should be enabled. When you install Drupal, after selecting a foreign language, these two are enabled automatically, and Drupal is installed entirely in that foreign language. For a multilingual website, the Content Translation and Configuration Translation modules are essential.

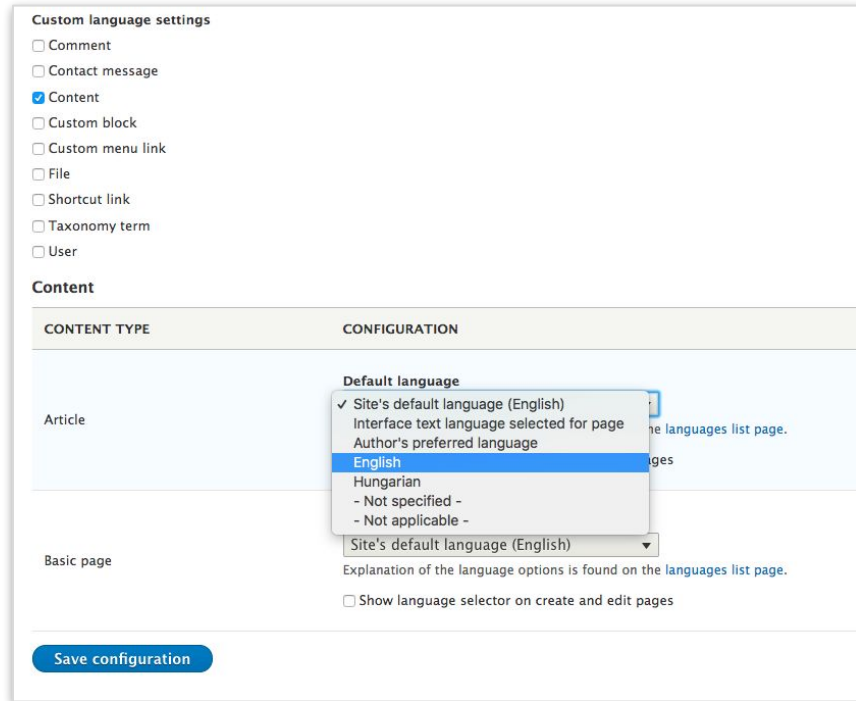
Multilingual Out of the Box



Drupal 8's Umami profile provides a demonstration of the above concepts with both an English and Spanish translation. Users can follow the site's navigation to seamlessly switch between the two languages and specify their preferred language.

Language Management

Drupal 8 is capable of managing any number of languages. After enabling the language module, it becomes possible to assign language information to content to enable tracking of the language of each content item. By default, Drupal assigns the site default language to newly created content, however you can customize this further with various dynamic presets, such as the author's preferred language.



The language module is also responsible for selecting the language used for the page, which may be based on the visitor's browser preferences or a part of the URL for example.

Finally, blocks are also enhanced with language visibility settings, so you can show or hide specific blocks for a group of languages. This feature can be used to provide different navigation options, for example when the foreign language version of the website is less capable than the original language.

Interface Translation

Drupal has built-in interface features for both the back and front end of the site. For many languages, these are already translated by the community at <https://localize.drupal.org/> — the interface translation module downloads translations automatically when a new language is added, as well as during the Drupal installation process.

However, not all translations are complete, and not all translations match the needs of all sites, so there is a built-in user interface to customize these translations further. As the module keeps the translations updated from community sources, you can choose to protect your local customizations from being overwritten by such updates.

The screenshot shows the 'FILTER TRANSLATABLE STRINGS' interface. At the top, there is a search box labeled 'String contains' with the text '@count' entered. Below it, a note says 'Leave blank to show all strings. The search is case sensitive.' There are three dropdown menus: 'Translation language' set to 'German', 'Search in' set to 'Only translated strings', and 'Translation type' set to 'All'. There are 'Filter' and 'Reset' buttons. Below the filters is a table with two columns: 'SOURCE STRING' and 'TRANSLATION FOR GERMAN'. The table contains two rows of data, each with 'Singular form' and 'Plural form' entries.

SOURCE STRING	TRANSLATION FOR GERMAN
Singular form 1 minute	Singular form 1 Minute
Plural form @count minutes	Plural form @count Minuten
Singular form 1 hour	Singular form 1 Stunde
Plural form @count hours	Plural form @count Stunden

It is also possible to enable interface translation for English to make small modifications to the English user interface, such as when "sign in" is needed in place of "log in."

All of these customizations may be exported and reused on other sites. Translation updates may be disabled on live sites and made part of a deployment process.

Content Translation

The Content Translation module builds on the language assignment functionality of the language module and allows content items, such as nodes, user profiles, taxonomy terms, custom blocks and menu items, to be translated into multiple languages. This functionality is configurable on the field level and, in some cases, within fields. Such as for images, the file itself may be the same across translations while the title and alternate text would be translated. This is practical for a user profile picture. For a figure with text on it, the file would also need to be replaced in translations.

The screenshot shows the configuration page for the 'User' entity. It is divided into three main sections: 'TRANSLATABLE', 'USER BUNDLE', and 'CONFIGURATION'. The 'TRANSLATABLE' section has a checked checkbox. The 'USER BUNDLE' section lists 'User'. The 'CONFIGURATION' section includes a 'Default language' dropdown menu set to 'Site's default language (English)', a link to the 'languages list page', and two checkboxes: 'Show language selector on create and edit pages' (checked) and 'Hide non translatable fields on translation forms' (unchecked). Below this is a table of fields with checkboxes indicating their translatable status.

TRANSLATABLE	USER BUNDLE	CONFIGURATION
<input checked="" type="checkbox"/>	User	<p>Default language</p> <p>Site's default language (English) ▼</p> <p>Explanation of the language options is found on the languages list page.</p> <p><input checked="" type="checkbox"/> Show language selector on create and edit pages</p> <p><input type="checkbox"/> Hide non translatable fields on translation forms</p>
<input checked="" type="checkbox"/>	Changed	
<input type="checkbox"/>	Full name	
<input checked="" type="checkbox"/>	Hobby	
<input checked="" type="checkbox"/>	Job title	
<input checked="" type="checkbox"/>	Picture	
<input type="checkbox"/>	File	
<input checked="" type="checkbox"/>	Alt	
<input checked="" type="checkbox"/>	Title	

Translations of a content item are managed using the translate tab content page. Once the translations are complete, you can adjust the listings and pages using the Views module, which generates dynamic content listings, to adhere to your language requirements. Built-in pages like the default front page are created with Views, so you can customize their language behavior. Views has filters to show only content in specific languages, and the rendering of results is also customizable in terms of language used.

The combination of Content Translation, Views and Block Visibility modules can be used to build highly complex multilingual experiences.

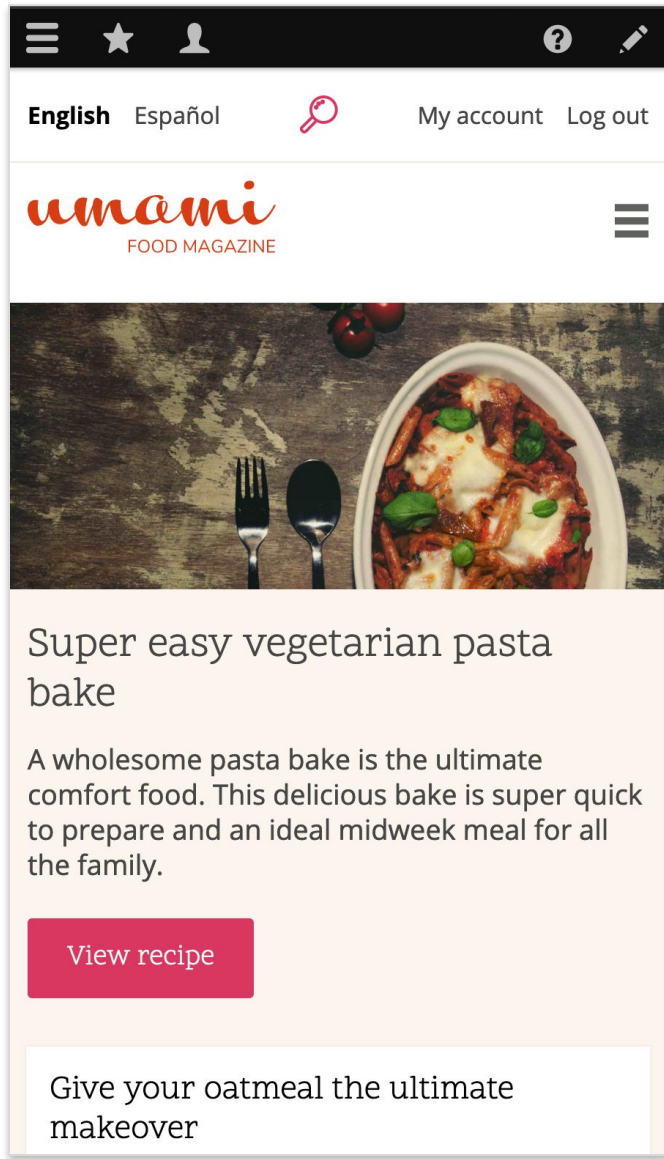
Configuration Translation

Finally, a major highlight of Drupal 8 is the configuration system that manages deployable configuration of content types, fields, menus, forms, views and block placement. These also need translation. The Configuration Translation module provides a user interface to manage the translation of these pieces and it stores the language files within the regular configuration storage system so translations participate in the deployment process as well.

MOBILE EXPERIENCE

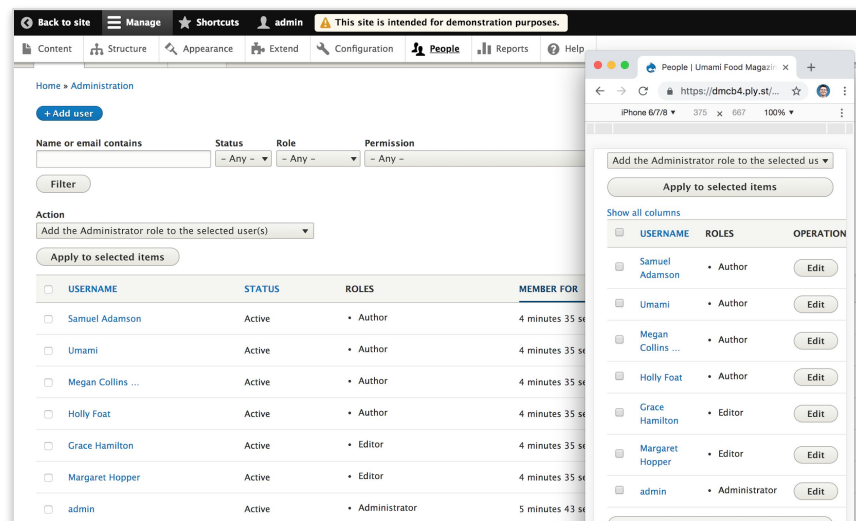
A key goal for Drupal 8 was to make both the front and back end entirely “mobile friendly” to provide the best experience for site visitors and administrators alike. Drupal 8 supports site visitors’ needs as they surf the web on their tablets and phones as well as enables authors and editors to work productively on their sites from mobile devices.

Mobile-First



Drupal 8 has been designed with mobile in mind, from the installer to the modules page. And new features, such as in-place editing, and oEmbed, have been designed to work on the smallest of screens.

Responsive-ize ALL Things: Themes, Images and Tables



To support the innumerable array of internet-enabled devices released, Drupal 8 incorporates responsive design into all of its functionality.

To begin, all core themes are responsive and automatically reflow elements, such as menus and blocks, to fit well on mobile devices, e.g., if the viewport is too narrow, horizontal elements will switch to a vertical orientation instead. Images that show up large on a desktop shrink down to fit on a tablet or smartphone thanks to built-in support for responsive images.

Drupal 8 also provides support for responsive tables with table columns that can be declared with high, medium or low importance. On wide screens, all the columns show, but as the screen size narrows, the less important columns start dropping off, so everything will fit nicely. This API is also built into the Views module, so you can configure your own responsive admin screens.

The responsive tables API is built into the views module, so you can configure your own responsive admin screens.

Mobile-Friendly Toolbar

Drupal 8 sports a responsive administrative toolbar that automatically expands and orients itself horizontally on wide screens and collapses down to icons and orients itself vertically on smaller screens. Like all new front-end features in Drupal 8, this toolbar was designed with accessibility in mind. The toolbar allows screen-reader users to easily navigate to various parts of a site.

Fast By Default

One of the primary factors that can make or break the mobile experience is the raw performance of a website, so an important focus of Drupal 8 is minimizing its front-end footprint. Acquia's own Wim Leers said that the best way to make the internet faster is to make the leading [CMSes fast by default](#). This means that CMSes need to have their high-performance settings enabled out-of-the-box rather than require users to be savvy enough to find them in all their various locations, and in Drupal 8, that's exactly what we've done. Drupal 8 ships with features such as CSS and JavaScript aggregation and Page Cache and Dynamic Page Cache are already turned on due to improved functionality in Drupal 8 for an accelerated default installation.

Additionally, page loads are increased by replacing jQuery with efficient, targeted and native JavaScript in many cases, and out-of-the-box Drupal 8 does not load JavaScript files at all for anonymous visitors.

Drupal 8 also ships with the BigPipe page delivery system enabled as part of a standard install, which improves the user experience for your site visitors by reducing the perceived page loading time. Drupal 8 includes cacheability metadata and intuitively knows which parts of every page are static and which are dynamic. BigPipe sends the unchanging parts of a page to the browser immediately while rendering and delivering the dynamic parts later as soon as they are ready. Essentially, your site visitors see what they came to see almost immediately — the main content and images, for example — while the uncacheable, personalized page elements, such as a shopping cart block, are delivered once ready.

Native Mobile Data Entry

ROGERS 1:15 PM 60%

dmcb4.ply.st

▼ AUTHORIZING INFORMATION

Authored by
Umami (6)
The username of the content author.

Authored on
Sep 24, 2018 7:36 PM
Format: 2018-09-24 20:14:47. Leave blank to use the time of form submission.

Clear Done

June	21	2015
July	22	2016
August	23	2017
September	24	2018
October	25	2019
November	26	2020
December	27	2021

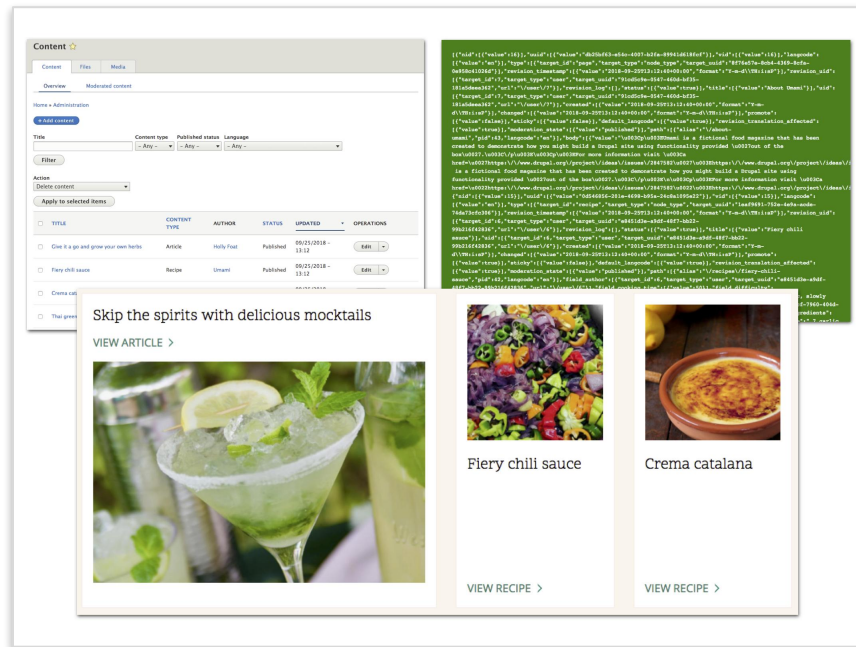
Data entry in Drupal 8 is seamless thanks to HTML5 form elements that provide targeted user interfaces on mobile devices for fields such as date/time, phone numbers and email addresses.

BUILDING AND MANAGING YOUR SITE

Although the [authoring experience improvements](#) and [mobile improvements](#) in Drupal 8 tend to focus on end users and content authors of Drupal websites, Drupal 8 also includes a massive push to improve the site building tools.



Views in Core



The Views module, a query-builder UI used for creating dynamic content listings in a variety of formats, is part of Drupal 8 core and is more tightly integrated into Drupal than ever before. Numerous previously hardcoded admin pages are now Views listings, which gives site builders the power to customize most of the main administrative listings or even build brand new ones. For example, adding a “full name” search to the people listing, or thumbnails next to items in the Content listing, is just a few clicks away. It is possible to create responsive table listings and to turn any listing into a REST export that can be consumed by a mobile application or other external service when the core RESTful Web Services module is also enabled.

Organize Pages with Blocks

Most page elements are displayed through blocks, including breadcrumbs, site name and slogan. This makes it easy to adjust page organization in the user interface, enables in-place editing and Settings Tray compatibility and makes for easier styling of these elements. It is also possible to reuse blocks. You can place a block in multiple places, for example, a “Navigation” block can be placed in both the header and footer. And finally, you can create custom block types just as you can for custom content types to allow for granular control over different styling and different fields. For example, this allows you to create an “Ad” block type with an “Ad code” field that contains JavaScript snippets from a remote ad service, which will allow you to add and place as many different blocks of that type on your site as needed.

Blocks can also be placed in sections within the Layout Builder, as discussed in the Layout Builder Chapter on page 19.

Structure Content with Entity Fields

Content, user profiles and comments are examples of Entities. You can add fields to all entities, including references to other entities. This makes it easier than ever to build data models for the structured content you want to manage using Drupal.

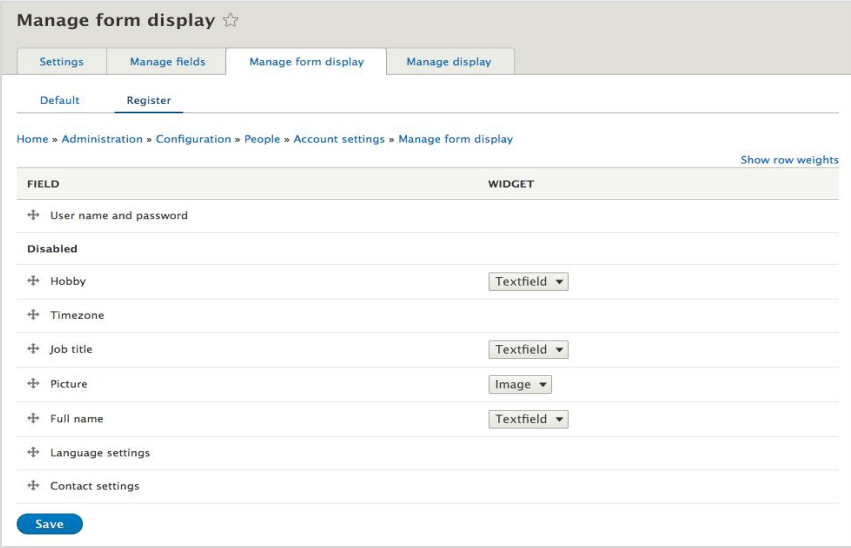
To build data models, Drupal 8 includes a plethora of fundamental, semantic field types like Taxonomy, Image, Phone, Email, Link and File, as well as more powerful fields such as Entity Reference and Date Range. Also, the setting for whether comments are open or closed has been moved to a field, making any entity type commentable.

Drupal 8 allows you to add fields to all entities, including references to other entities. This makes it easier than ever to build data models for the structured content you want to manage using Drupal.

View Modes and Form Modes

Once you set up your entity structures with fields, “view modes” allows for creating multiple display options for content in different contexts, for example, showing a thumbnail image on your content’s teaser view and a full-size image on the default view.

Drupal 8 also has the notion of “form modes” for data-entry forms. For example, you can make your user registration form very simple and only show more complex fields on the user edit form.



In summary, Drupal 8 provides a lot of flexibility with more general purpose components, so when building sites, you can work with the same concepts everywhere. This is demonstrated by the Umami demo profile in core that is outlined in the [Getting Started with Drupal](#) post on Acquia.com.

Migrating to Drupal 8

Drupal core has built-in support for migrating data to the system from third-party systems, e.g., WordPress, as well as specific solutions built to facilitate migration from Drupal 6 and 7 to Drupal 8. For Drupal-to-Drupal migrations, there is a Migrate Drupal UI module that allows the user to migrate using a web interface, similar to the update.php interface. Unlike update.php, however, the migration path allows you to keep your Drupal 6/7 site running while you build your new Drupal 8 site, so you can avoid downtimes entirely.

The migration paths from Drupal 6 and 7 are stable for monolingual websites, with multilingual support available experimentally through the Drupal Migrate Multilingual module.

Drupal 7's end-of-life date has been set for November 2021, though commercial support will be available until at least 2024. Given all of the stellar advantages of Drupal 8, it is a perfect time to evaluate the readiness of your site to plan for the migration to Drupal 8 when it makes sense for you or your organization to do so.

Migrating to Drupal 8 involves three parts:

1. **Evaluating the readiness of your contributed modules and themes.** Drupal 7 sites can use the [Upgrade Status module](#), which provides a simple red/yellow/green report on the Drupal 8 readiness of each module. In some cases, it will show alternative modules that provide suitable replacements in Drupal 8.
2. **Converting your custom code and themes to Drupal 8.** [Drupal Module Upgrader](#) is a set of [Drush](#) commands to analyze your custom code and provide a list of items that need to be converted. In some cases, it can automatically convert up to 60% to 70% or the necessary code changes. For the remainder, API change records can be helpful to point in the right direction.
3. **Migrating data and content.** For this, the built-in Migrate API and Migrate Drupal modules can be used, along with either Drush or the Drupal Migrate UI. This will execute all of the core + contributed + custom code migrations and transfer over the earlier site's content to the new site. See the [Migrate Tools module](#) for a set of helpful command-line extensions.

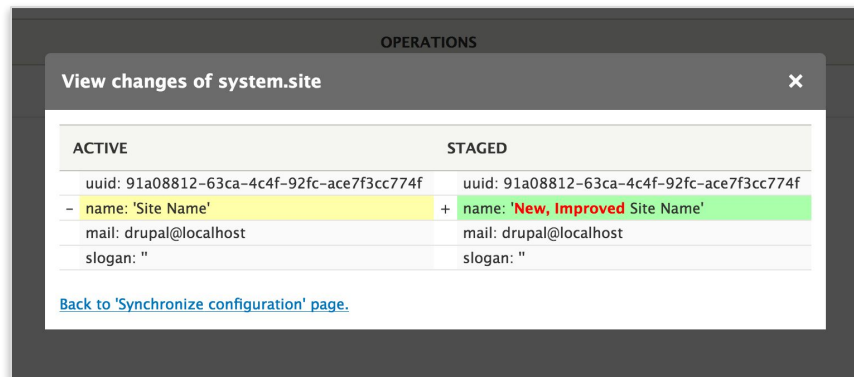
For more details, see the helpful Upgrading to Drupal 8 section of the Drupal.org documentation.

Configuration Management System

One of the major improvements in Drupal 8, for both developers and site builders, is the new configuration management system. This allows sites that are following the best practice of having a "development," "staging" and "production" environment to seamlessly move configuration from one environment to another.

All configuration changes, both standard admin settings forms, such as site name, as well as any "configuration entity," such as Views or Content Types, are run through a standard configuration API. Each environment has a "sync" directory to hold configuration changes from other environments that will be imported for review. For performance, active configuration is stored in a configuration table in the database, though the storage location is swappable, e.g., to Redis or Cassandra, and when importing/exporting configuration, configuration is stored in YAML files.

Drupal 8 also ships with a basic UI to complete both single and full configuration imports and exports and view differences between them prior to importing changes.



Configuration can also be moved around via the command line with tools such as [Drush](#), which is useful when keeping your configuration in version control systems such as Git.

There are also a variety of contributed projects that build on top of the configuration management system, such as [Configuration Split](#) module, which provides the ability to keep different sets of configuration on one environment versus another. Plans for future versions of Drupal 8 are expected to include more of this functionality as part of core.

Regarding content deployment, the goal is for the new [Workspaces](#) module to manage that use case. See the Workflows chapter for more information.

FRONT-END DEVELOPER EXPERIENCE

Drupal 8 contains a lot of improvements for front-end developers, including HTML5, additional helper libraries, accessibility enhancements, new themes and UI elements and accelerated performance, to name a few.

HTML5



All of Drupal's output uses semantic HTML5 markup by default, as part of an overarching effort to simplify Drupal's default markup. This means you'll find tags such as `<nav>`, `<header>`, `<main>` and `<section>` in Drupal's default templates, and you'll find HTML5/CSS3 replacements for several aspects that previously needed custom workarounds: resizing of text areas and first/last/odd/even classes are now covered by CSS3 pseudo-selectors; and collapsible fieldsets are largely replaced by the `<details>` element.

New Front-End Libraries and Helpers

Drupal 8 brings with it an update to the latest version of [jQuery](#) and [jQuery UI](#), as well as an expanded array of front-end libraries. Together, these additional libraries allow for creating mobile-friendly, rich front-end applications in Drupal, and they're used by several of the authoring experience and mobile feature improvements to Drupal 8. These include:

- [Modernizr](#) (detects if a browser supports touch or HTML5/CSS3 features)
- [Underscore.js](#) (a lightweight JS-helper library)
- [Backbone.js](#) (a model-view-controller JavaScript framework).

Additionally, as part of the [Admin UI & JavaScript Modernization](#) initiative, work is underway to bring a [React](#)-based revamped administration experience for Drupal.

Improved Accessibility

Edit Recipe Fiery chili sauce ☆

[View](#) [Edit](#) [Delete](#)

[Home](#) » [Fiery chili sauce](#)

✖ 1 error has been found: [Cooking time](#)

Recipe Name *

Author *

The author of this recipe. Start typing their username to search.

Preparation time *
 minutes

Cooking time
 minutes
✖ **Cooking time must be lower than or equal to 20.**

Number of servings *

Drupal 8 has expanded on prior versions' stellar accessibility record with even more improvements. Drupal 8 extensively uses WAI-ARIA attributes to provide semantic meaning to elements in rich, front-end applications, such as the in-place editor and responsive toolbar. On the back end, Drupal 8 provides a variety of new Accessibility tools for JavaScript (JS) that allow module developers to create accessible applications easily. The Inline Form Errors module can also be used to place error messages adjacent to form inputs for improved usability and accessibility.

New Theme System: Twig

Drupal 8 introduces [Twig](#), a widely adopted theme system in the PHP world. Twig's syntax allows designers and themers with HTML/CSS knowledge to modify markup without needing to be a PHP expert. Twig effectively forces a separation of presentation and business logic, and all variables going into template files are automatically escaped, far reducing the risk of dangers like XSS vulnerabilities and making theming in Drupal 8 more secure. Drupal 8.8 and above optionally allow you to use Twig version 1 or 2.

Another attribute of Twig is when turning on debug mode, helpful code comments will be displayed throughout Drupal's generated markup to inform you where to find the template for the markup you're trying to change and which particular "theme suggestion" is being used to generate the markup.

Here is an example from page.html.twig, showing basic Twig syntax, as well as some HTML5 elements:

```
<main role="main">
  <a id="main-content" tabindex="-1"></a>{#
link is in html.html.twig #}

  <div class="layout-content">
    {{ page.content }}
  </div>{# /.layout-content #}

  {% if page.sidebar_first %}
    <aside class="layout-sidebar-first"
role="complementary">
      {{ page.sidebar_first }}
    </aside>
  {% endif %}

  {% if page.sidebar_second %}
    <aside class="layout-sidebar-second"
role="complementary">
      {{ page.sidebar_second }}
    </aside>
  {% endif %}

</main>
```

Theme Responsively

As mentioned in the [mobile improvements](#) section of this e-book, Drupal 8 ships with numerous new responsive features, including responsive themes, toolbar, images, and tables.

To support these features, themes can now declare [breakpoints](#) (the height, width, and resolution at which a design changes to meet shifting browsers and devices) that can be used by responsive features.

Drupal 8 also ships with the new [responsive image module](#), which contains support for the HTML5's [<picture> and <source>](#) elements, as well as the sizes, srcset and type attributes. This will make for a significant front-end performance improvement, particularly on mobile devices, because it allows delivering smaller images (typically the heaviest part of any page load) for smaller screens, saving data.

R.I.P. IE < 11



Another big improvement for front-end developers and designers is that Drupal 8 core has officially dropped support for earlier versions of Internet Explorer, enabling the use of newer front-end libraries and other code that assumes modern HTML5/CSS3 browser support.

As a parting gift, [html5shiv](#) (an HTML5 polyfill for less capable browsers) is included in D8 core so at least earlier browsers aren't completely ignored, for those who absolutely must have compatible versions of core front-end features on Drupal 8 websites. For the rest of us, we're looking forward to snappier front-end code that doesn't have to worry about limitations in unsupported browsers.

For more information on browser compatibility, see <https://www.drupal.org/docs/8/system-requirements/browser-requirements>.

BACK-END DEVELOPER EXPERIENCE

Drupal 8 offers numerous back-end developer improvements. These include object-oriented code, improved caching, better integration with third-party services, and extensive built-in web services features.

Composer Support

New as of Drupal 8.8, Drupal now ships with native support for [Composer](#), a popular PHP dependency management tool. Previously, Composer support was provided by a variety of third-party plugins, but this support is now part of core. Composer support makes installation and updates easier from the command line.

With one command, similar to:

```
composer create-project drupal/recommended-project:~8.8.0
```

Drupal can be downloaded for you and ready to go. See the [Composer instructions on Drupal.org](#) for more information.

Note that Composer is not required, but is the recommended way to install and manage Drupal going forward, since more and more developers are adding Composer dependencies to their modules..

Quickstart Installation

```
> cd docroot && php core/scripts/drupal install demo_umami --no-interaction
0/18 [ ..... ]
Installing Drupal

1/18 [ ..... ]
Choose profile

2/18 [ ..... ]
Choose profile

3/18 [ ..... ]
Verify requirements

4/18 [ ..... ]
Set up database

5/18 [ ..... ]
Set up database

6/18 [ ..... ]
Set up database

7/18 [ ..... ]
Set up database

8/18 [ ..... ]
Set up database
```

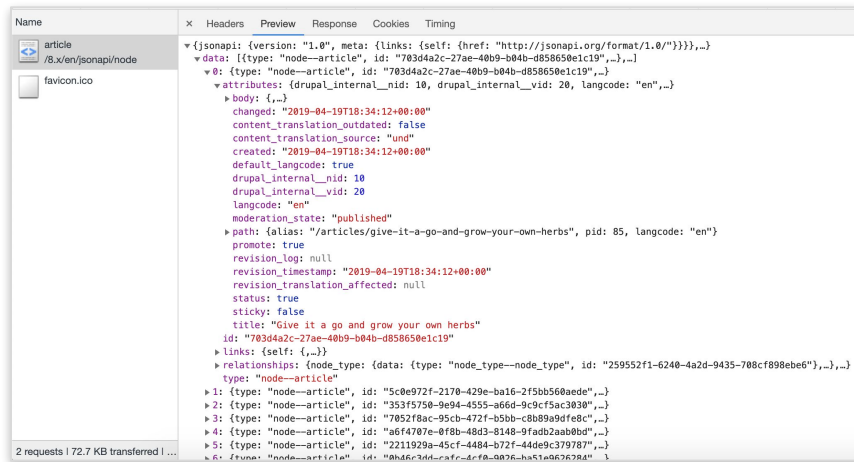
If you are comfortable working from the command line, you can also install Drupal 8 from a single command, including a local web server and database:

```
php core/scripts/drupal install <profile_name>
--no-interaction
```

Web Services

A major focus of Drupal 8 is to have an API-first development strategy to allow developers to produce and consume web services for the creation of Drupal-powered decoupled and mobile applications and to facilitate cross-site communication and better integration with third-party resources, such as in microservices-style applications. To that end, Drupal 8 now ships with the JSON:API module, which exposes entities as a JSON:API-specification-compliant web API.

When this module is enabled, you can navigate to the /jsonapi path and see a list of all available resources and drill down from there to find the specific content you are seeking. For example, /jsonapi/node/article returns a list of articles in the



By default, all resources, i.e., JSON:API terminology for what Drupal calls “entities,” are exposed as read-only, which is perfect for a set up where Drupal acts as a central content store and there’s a decoupled front end in JavaScript or a mobile application consuming the data. However, there’s also a simple toggle in the admin settings to enable full CRUD (create, read, update, and delete) operations, which allows performing the content creation and editing process in decoupled applications.

Also see the [JSON:API Extras](#) contributed module that provides more granular configuration options, for example, the ability to turn on/off certain resource types and fields, and the ability to overwrite names and paths of resources.

Drupal 8 also ships with the [Guzzle PHP HTTP](#) library, which provides an easy PHP API to retrieve and post data to Drupal or to talk to third-party web services, such as Twitter or Github.

Another web services feature in Drupal 8 offered by the RESTful Web Services module is the ability to add a “REST export” display to any view. This means you can easily create JSON or XML feeds of custom dynamic content from your Drupal site, just by clicking them together.

Improved Caching

Caching in Drupal 8 has been improved across the board. The following functions and features are notable for their updated design:

- **Speed-enhancing features.** All caching features such as CSS/JS aggregation, Page Cache and Dynamic Page Cache are turned on out of the box, as well as other speed-enhancing features such as BigPipe.
- **Entity cache.** The module is now in core, which allows queries to be offloaded from the database onto alternative storage, such as Memcache or Redis.
- **Cache tags.** Cache tags allow for much more granular cache clearing when content or settings are updated on the site. This feature works on reverse proxies like Varnish and CDNs such as Fastly, Cloudflare or KeyCDN.

Proudly Found Elsewhere

The philosophy largely embraced by Drupal 8 is “proudly invented elsewhere,” defined as actively seeking out adjacent projects that specialize in solving a particular problem and adopting this code in our code base.

One of the great strengths of open source software is to not have to reinvent the wheel and to be able to build better solutions “on the shoulders of giants.” This means finding the best tool for the job rather than creating something custom and specific. By using third-party libraries, we benefit from having a broader community of contributors and users, and when we make improvements to these open source tools, everyone who uses them benefits, as well.

You’ll see this philosophy present in many aspects of Drupal 8. Among the external libraries we’ve pulled in are [PHPUnit](#) for unit testing, [Guzzle](#) for performing HTTP (web service) requests, a variety of [Symfony](#) components (please see the [Create your own PHP framework](#) tutorial on the Symfony website), and [Composer](#) for external dependencies and class autoloading.

This philosophy also extends to the code base itself. We made [big architecture changes](#) in Drupal 8 to embrace the way the rest of the world is writing code, i.e., decoupled and object-oriented (OO) and to create modern language features of PHP, such as namespaces and traits.

Getting OOP-y with It

Let's look at a few code samples to illustrate Drupal 8's architecture in action.

Registering a Module

Drupal's functionality is extended through the use of modules. Step 1 to writing a module is describing metadata to register it in the system. This is done in a [YAML](#) file.

example.yaml.info

```
name: Example
type: module
description: 'An example module'
core: 8.x
dependencies:
  - user
```

Defining a URL and content

A common function of modules is to expose URLs that produce some sort of output. This is done in two parts: a routing.yml file, which utilizes the [Symfony Routing system](#) to define Route (URL) metadata, and a "Controller" class, as in the standard [model-view-controller](#) pattern:

example.routing.yml

```
example.hello:
  path: '/hello'
  defaults:
    _controller: '\Drupal\example\Controller\HelloController::hello'
  requirements:
    _permission: 'access content'
```

src/Controller/HelloController.php

```
<?php

namespace Drupal\example\Controller;

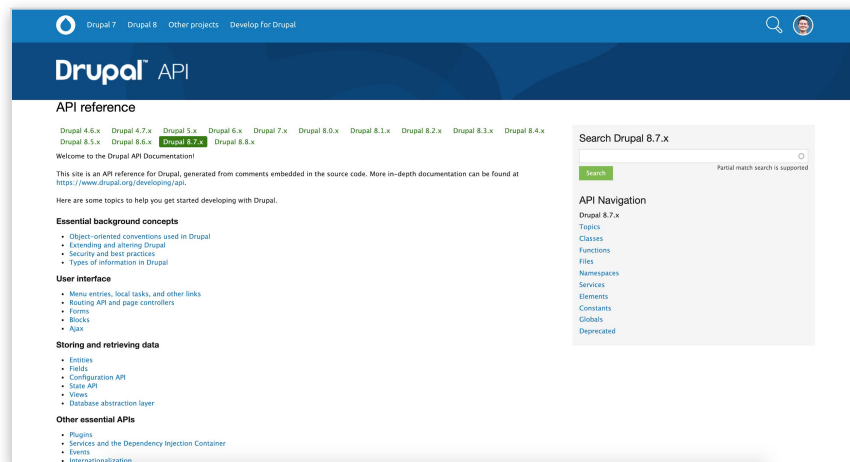
use Drupal\Core\Controller\ControllerBase;

/**
 * Greet the user.
 */
class HelloController extends ControllerBase {

    public function hello() {
        return ['#markup' => $this->t('Hello world')];
    }

}
```

In this example, we are defining a route at `http://example.com/hello` to confirm if the currently logged-in user has "access content" permission. If the access check passes, control is passed to the HelloController class, which handles returning "Hello world" page content.



Numerous modern PHP best practices are at use here:

The controller class uses a specific directory and naming convention, per the [PSR-4 class autoloading standard](#), to allow its code to load when needed.

The class declares a PHP [namespace](#) to prevent naming conflicts with other modules and third-party libraries.

The class pulls in the logic from the ControllerBase class in via the use statement and extends it. This gives the HelloController access to all ControllerBase's convenient methods and capabilities, such as `$this->t()`, to handle translatable content.

You can find a great introduction to Drupal 8's APIs on [api.drupal.org](#), which provides a list of topics to orient developers to Drupal 8.

You can also see <https://drupal.org/list-changes> for API changes between releases. Each API change record includes before and after code examples to help you migrate your code, as well as explanations to which issue(s) introduced the change and why.

THE FUTURE OF DRUPAL 8

The Future of Drupal 8

Drupal 8.0.0 was released on November 19, 2015. While many feature updates noted are considered important, the most compelling changes relate to how releases are executed and how innovation happens within those releases. We made four major changes to support innovation within Drupal 8:

1. **Semantic versioning:** The Drupal community has moved to support innovation within major releases with a new semantic versioning release system. “Minor” versions such as 8.5.0 and 8.6.0 are released, which can include backwards-compatible new features and functional improvements. This is alongside the expected bug fix, “patch” releases between minor versions, e.g., Drupal 8.5.1 and 8.5.2.
2. **Scheduled releases:** Minor releases follow planned timelines and are released approximately every six months. To ensure quality, each of these minor releases gets its own beta releases and release candidates with strict guidelines on allowed changes.
3. **Deprecation:** As new features and API additions are introduced, old functionality and APIs may be marked deprecated and targeted for removal in Drupal 9. All releases of Drupal 8 should retain backwards compatibility, but Drupal 9 will remove outdated approaches when there are better solutions.
4. **Experimental modules:** To further accelerate the pace of innovation, Drupal 8 includes experimental core modules. While these modules don’t necessarily conform to the rigorous requirements for full Drupal core modules, they allow core contributors to iterate quickly and get real-world feedback on important new functionality that may be fully supported in future versions of Drupal 8. All experimental modules shipped in core releases are required to at least be beta quality, supporting backwards compatibility. Experimental core modules also must undergo a stabilization/polish process to become full modules within a year of their introduction, two minor releases, or to be removed from core again. Various stable modules of Drupal 8 began as core experimental modules, such as BigPipe, Migrate, Inline Form Errors and Layout Builder.

These practices allow core developers to validate pre-release functionality with frequent releases as well as indicate APIs and features slated for removal in favor of more modern solutions.

The Limits of Drupal 8

This release and innovation model worked so well for Drupal 8 core that some users have said Drupal 8 should be the final major version. However, Drupal 8 is not fully in the core team's control in that the system depends on various third-party projects such as Symfony, Twig, PHPUnit and Guzzle. This is helpful in that best practices and solutions are shared with a wider community. However, support, in particular, security support, of Drupal 8 is dependent on the support of those third-party systems, as well.

The Road to Drupal 9

Drupal core's biggest dependency is Symfony 3, which reaches its end of life in November 2021, meaning Drupal 8 sites need to be moved to Drupal 9 before that to retain security support. Consequently, the planned release date for Drupal 9 is June 3, 2020. In a recent blog post, Dries Buytaert, co-founder of Acquia, shared insight to the design and functionality Drupal 9 will bring, as well as the release schedule. "We are building Drupal 9 in Drupal 8, which means the technology in Drupal 9 will have been battle-tested in Drupal 8. Furthermore, we are adding new functionality as backwards-compatible code and experimental features. Once the code becomes stable, we will deprecate any old functionality."

With only a year to migrate from Drupal 8 to Drupal 9, APIs and features removed are continually identified. You can prepare for Drupal 9 by not using deprecated features or APIs in your own custom code and contribute code to upstream modules to help them follow best practices as well. The Drupal 8 version of [Upgrade Status module](#) provides a helpful report on deprecated code usages. You can alternatively use the command-line version at <https://github.com/mglaman/drupal-check>.

The planned release date for Drupal 9 is in June 2020. Drupal 9 will mainly be the last release of Drupal 8, minus the backwards-compatibility layers.

If you are following this process, your Drupal 8.8 site will be very easy to migrate to Drupal 9 as your modules and custom code will already work with Drupal 9. For additional reading, check out cofounder Dries Buytaert's blog post [Drupal 7.8 and 9](#). For a global overview of how the Drupal contributed project ecosystem as a whole is doing on Drupal 9 compatibility, check out the [Drupal 9 Deprecation Status](#).

Back to site Manage Shortcuts admin

Upgrade status

Home > Administration > Reports

Analyze your site's Drupal 9 readiness. Run the report to find out if there are detectable compatibility errors with the modules and themes installed on your site. [Read more about preparing your site for Drupal 9.](#)

Report last ran on Tue, 04/09/2019 - 20:01

Restart full scan Export full report

▼ CUSTOM MODULES AND THEMES

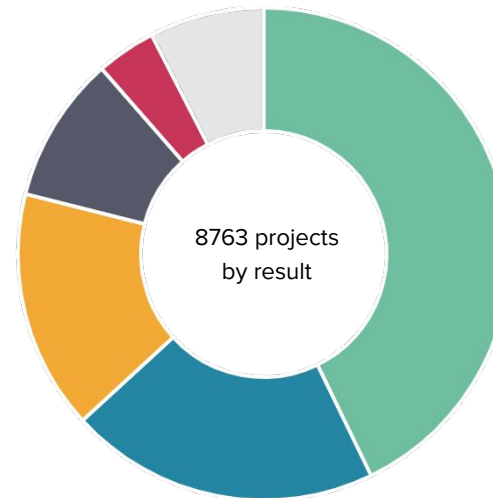
Custom code is specific to your site, and must be upgraded manually. [Read more about how developers can upgrade their code to Drupal 9.](#)

PROJECT	STATUS	OPERATIONS
Example.com Form Tools	2 errors	View errors
Example.com site API	No known errors	Re-scan

▼ CONTRIBUTED MODULES AND THEMES

Contributed code is available from drupal.org. Problems here may be partially resolved by updating to the latest version. [Read more about how to update contributed projects.](#)

PROJECT	STATUS	AVAILABLE UPDATE	OPERATIONS
Chaos Tools 8.x-3.0	5 errors	8.x-3.2	View errors
Pathauto 8.x-1.2	16 errors	8.x-1.4	View errors
Token 8.x-1.5	47 errors	Up to date	View errors



- No problems found
- All problems fixable now
- Some problems fixable now
- Needs manual review
- Only problems fixable later
- No results

IT'S A WRAP

This concludes our review of the updated features and functionality of **Drupal 8.8**. As you continue to deepen your knowledge of the Drupal open source platform, we hope this guide will remain a valuable resource as you create the next generation of innovative digital experiences.

To learn more about Acquia's Drupal 8 expertise, visit dev.acquia.com

ABOUT ACQUIA

Acquia is the open source digital experience company. We provide the world's most ambitious brands with technology that allows them to embrace innovation and create customer moments that matter. At Acquia, we believe in the power of community — giving our customers the freedom to build tomorrow on their terms.



[acquia.com](https://www.acquia.com)

ACQUIA