



Drupal™

Why I'm looking forward to
Drupal 8

Who Am I?

Jim Taylor

drupal.org/u/bigjim

@jalama

Currently:

Manager, Web Development @ Highlights for Children

Yes We're Hiring!

Started with Drupal in 2007 (Drupal 5.2)

Past Roles:

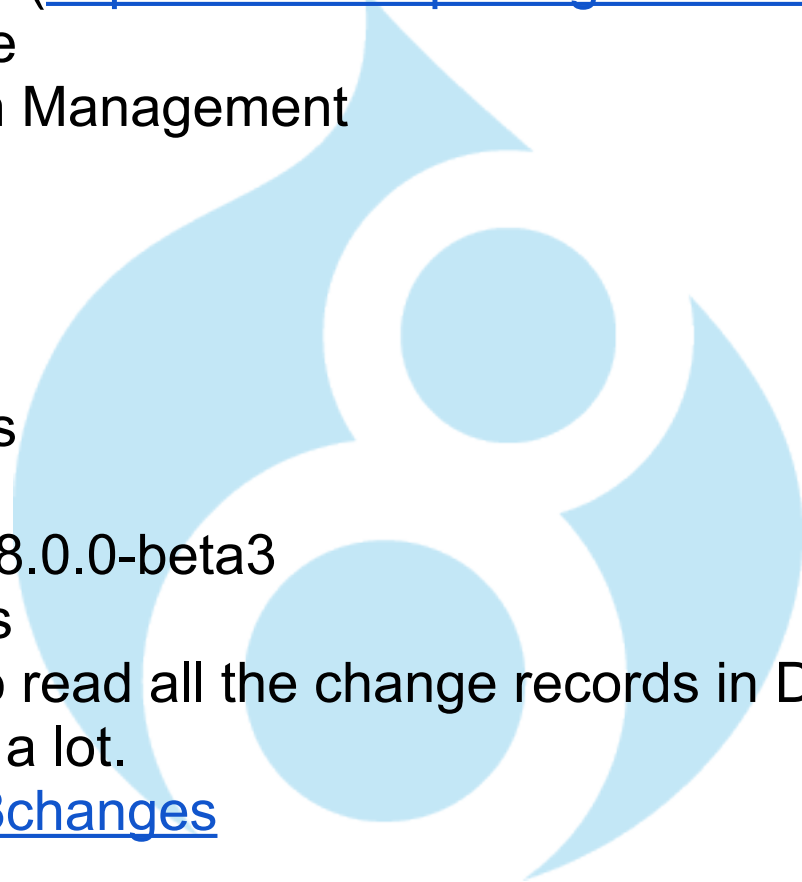
Owned a small Drupal shop

Senior Developer @ Verizon Wireless



Drupal 8 resources

- Official Initiatives (<https://www.drupal.org/node/2107085>)
 - Views in Core
 - Configuration Management
 - HTML5
 - Layouts
 - Mobile
 - Multilingual
 - Web Services
- Beta released
 - Currently on 8.0.0-beta3
- Change Records
 - If you want to read all the change records in D8, make some tea there are a lot.
 - <http://bit.ly/d8changes>



Site building (ie the UI)

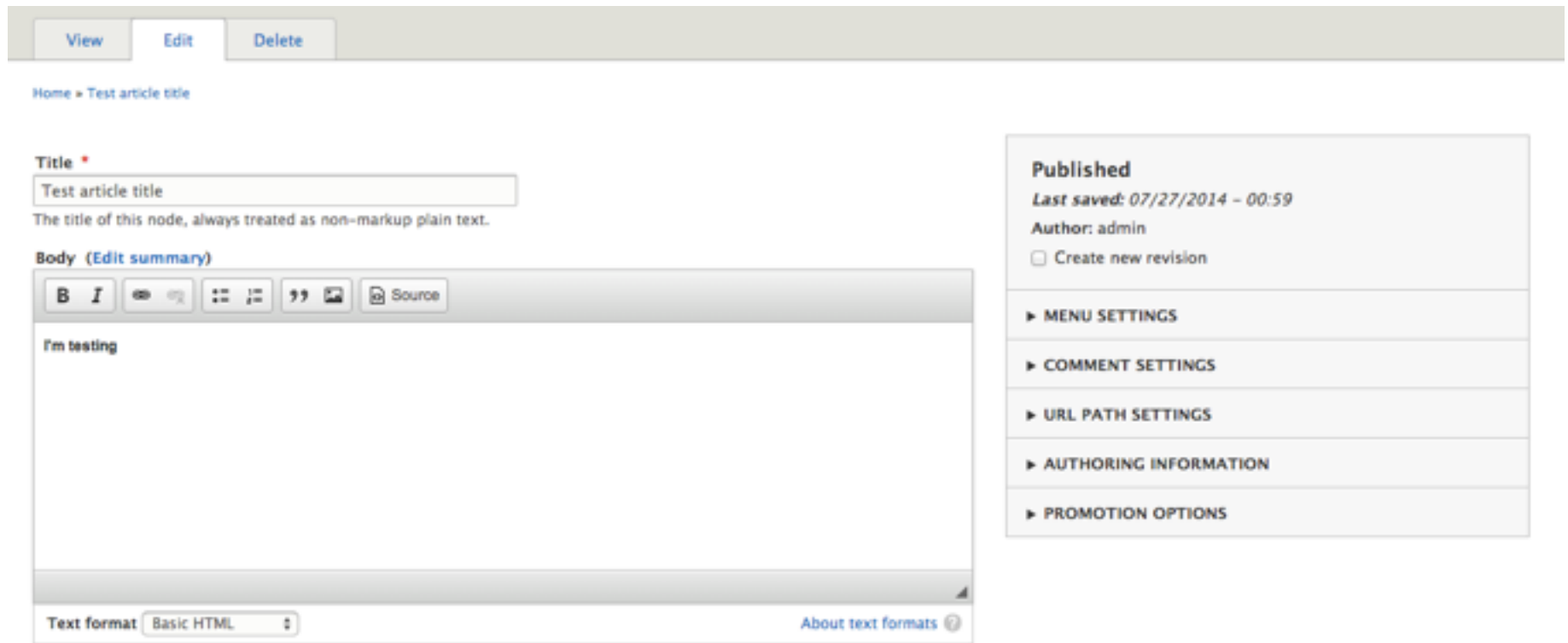
In-place editing

- ie the Edit module (Change record: <https://www.drupal.org/node/1872284>)
- Users with appropriate access can edit fields without clicking the edit tab



New node edit page

- No More Overlay
- 2 column screen
- Accessibility Initiative improved the editing process



The screenshot displays the Drupal 8 node edit page with a 2-column layout. At the top, there are three buttons: "View", "Edit", and "Delete". Below this is a breadcrumb trail: "Home » Test article title".

The main content area is divided into two columns. The left column contains the following elements:

- Title ***: A text input field containing "Test article title". Below it, a note states: "The title of this node, always treated as non-markup plain text."
- Body (Edit summary)**: A rich text editor with a toolbar containing icons for Bold (B), Italic (I), Bulleted list, Numbered list, Indent, Outdent, and Source. The editor contains the text "I'm testing".
- Text format**: A dropdown menu set to "Basic HTML". A link "About text formats ?" is located at the bottom right of the editor.

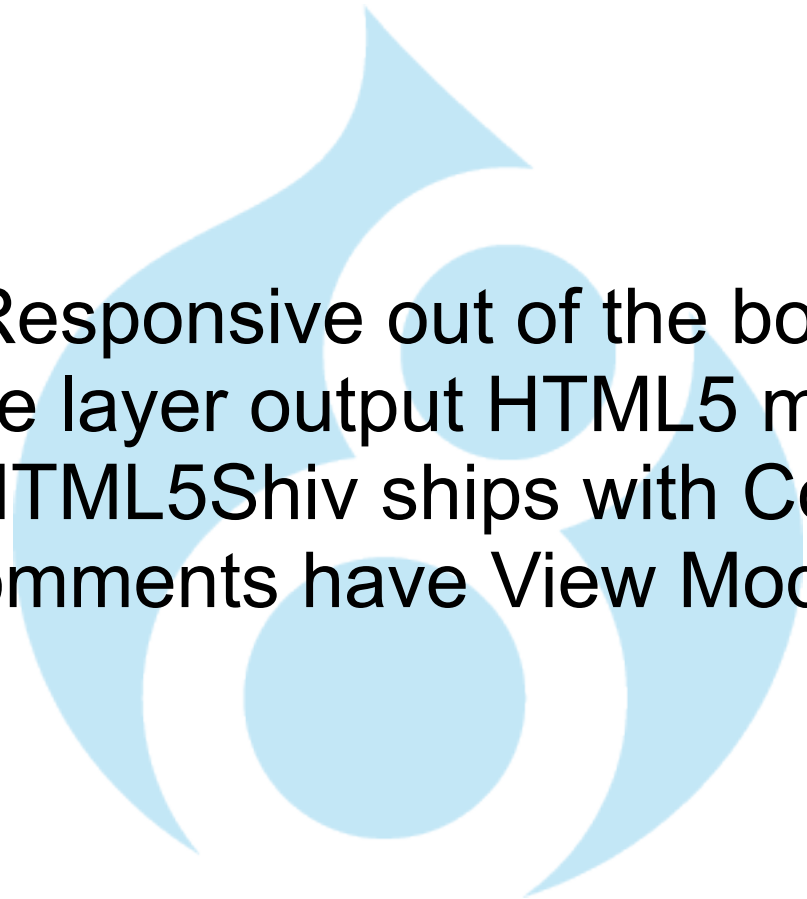
The right column contains a settings sidebar with the following sections:

- Published**: Shows "Last saved: 07/27/2014 - 00:59" and "Author: admin". There is a checkbox for "Create new revision" which is currently unchecked.
- MENU SETTINGS**: A section with a right-pointing arrow.
- COMMENT SETTINGS**: A section with a right-pointing arrow.
- URL PATH SETTINGS**: A section with a right-pointing arrow.
- AUTHORING INFORMATION**: A section with a right-pointing arrow.
- PROMOTION OPTIONS**: A section with a right-pointing arrow.



Other site building extras

Responsive out of the box
Theme layer output HTML5 markup
HTML5Shiv ships with Core
Comments have View Modes



Core modules added

Quickedit (In-place editing)
Editor (CKEditor)
Views
Migrate
Rest
Entity API
CTools (equivalent)
Config
Configuration Translation

History
Language
Content Translation
Responsive Image
(Picture)
Serialization
Tour
Diff
Breakpoint
Entity Cache



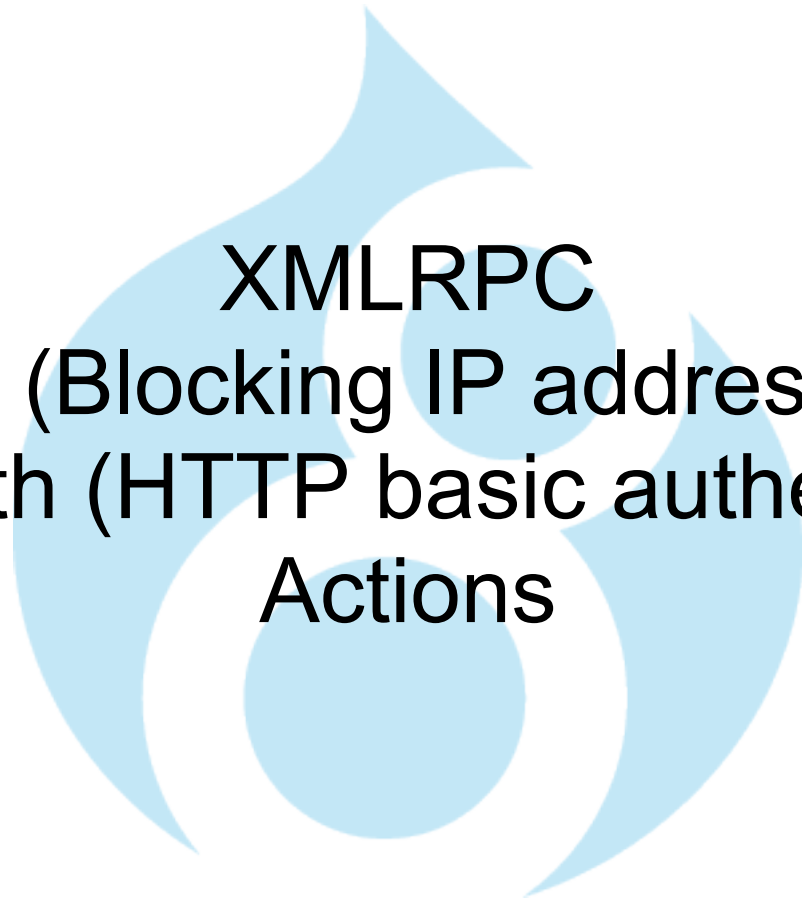
Module's created from core

XMLRPC

Ban (Blocking IP addresses)

Basic_auth (HTTP basic authentication)

Actions



Modules no longer in core



Blog
Profile
Garland (theme)
Trigger
Dashboard
OpenID
Poll
Php Filter
Overlay



Fields new in core

Entity Reference

Email

Number

Telephone

Date/Time

Link

Options (was List in D7)

Text

Menu Link (insert menu links from a field)



RESTful Services in core

Basically replace the output/input with XML/JSON/JSON-HAL in core. In other words Drupal can become headless and serve up content to web and/or mobile apps.

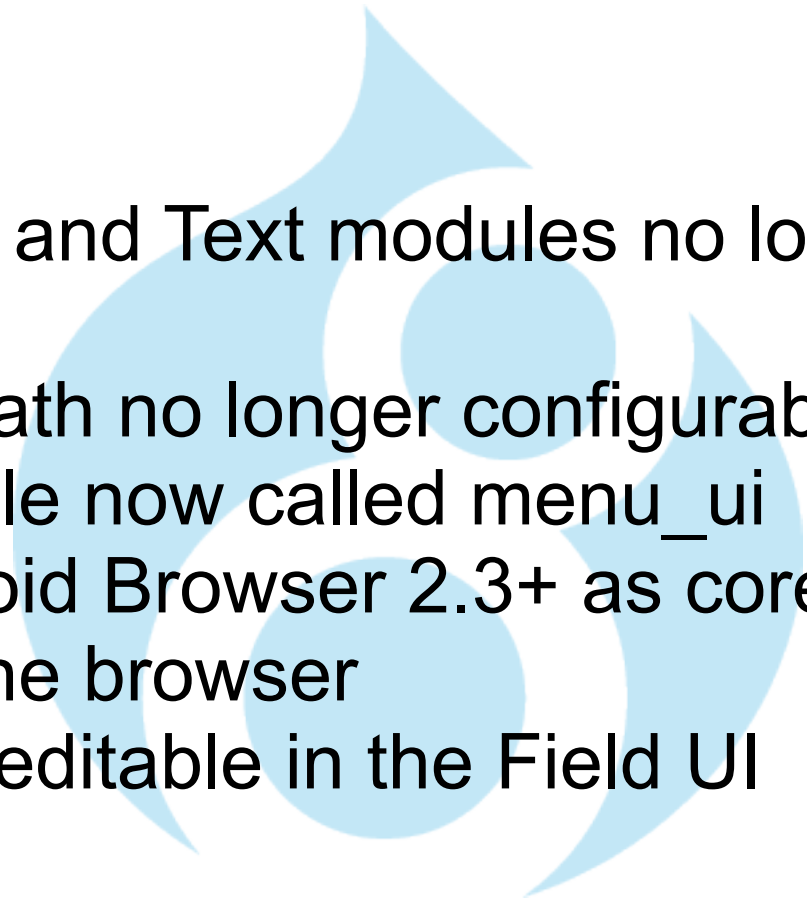
Modules:
Rest
Hal
Basic_auth
Serialization

Good Article: <http://drupalize.me/blog/201401/introduction-restful-web-services-drupal-8>



Fun little things

- Node, Filter and Text modules no longer required
- Public file path no longer configurable in the UI
- Menu module now called menu_ui
- IE9 +, Android Browser 2.3+ as core needs SVG support in the browser
- Field prefix editable in the Field UI



Theming: TWIG

Drupal 8's theme layer went through some pretty major changes, namely that it was completely replaced :)

With Drupal 8 TWIG from Sensiolabs (<http://twig.sensiolabs.org/>):

- **Fast:** Twig compiles templates down to plain optimized PHP code. The overhead compared to regular PHP code was reduced to the very minimum.
- **Secure:** Twig has a sandbox mode to evaluate untrusted template code. This allows Twig to be used as a template language for applications where users may modify the template design.
- **Flexible:** Twig is powered by a flexible lexer and parser. This allows the developer to define its own custom tags and filters, and create its own DSL (Domain Specific Language).



Theming: TWIG

- No more theme() functions in templates.
- No more preprocessors.
- No more HTML markup in module files (again, all markup come from templates).

Drupal Today

```
<?php echo $var ?>
```

TWIG

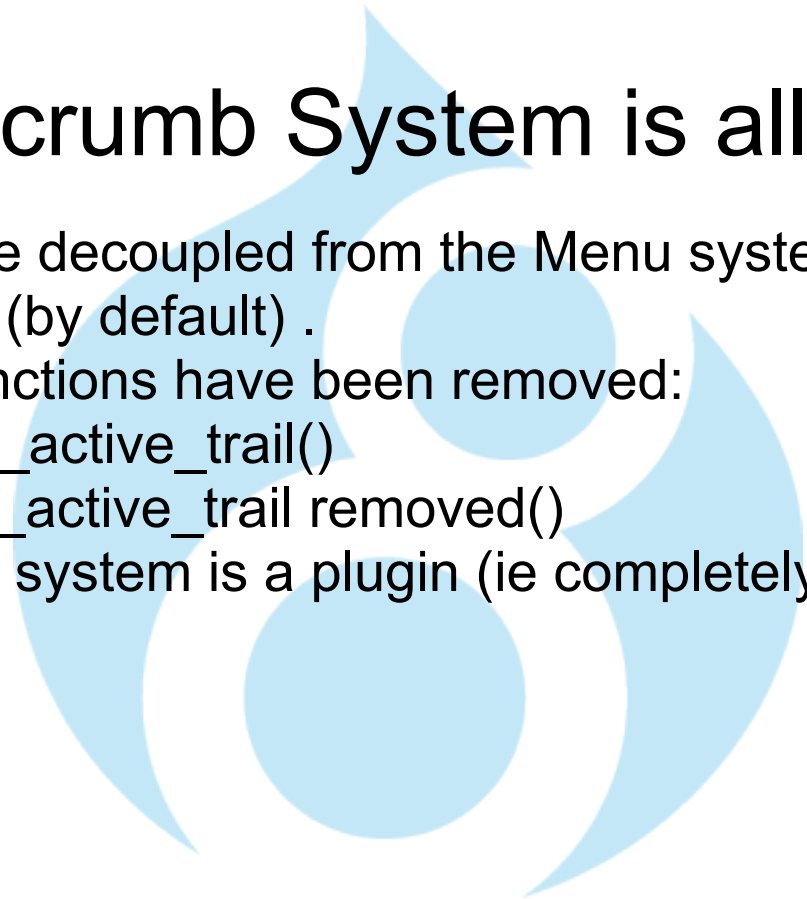
```
{{ var }}
```



Theming: Breadcrumbs

The Breadcrumb System is all new

- Breadcrumbs are decoupled from the Menu system and based on path instead (by default) .
- The following functions have been removed:
 - `menu_get_active_trail()`
 - `menu_set_active_trail` removed()
- The breadcrumb system is a plugin (ie completely customizable).



Theming: Other cool stuff

- Theme hook suggestions for View Modes
- New Base theme: Classy
- CSS classes being moved from preprocess to Twig templates
- Global theme variables got replaced by an ActiveTheme
- Attach libraries and files in render arrays, i.e. #attach:
 - No more:
 - drupal_add_css()
 - drupal_add_js()
 - drupal_add_library()
- drupal.base.css replaced by normalize.css
- CSS file system layout changed (<https://www.drupal.org/node/1887922>):
 - Base — CSS reset/normalize plus HTML element styling.
 - Layout — macro arrangement of a web page, including any grid



Module Dev



Module Dev:PHP basics

Learn Object Oriented Programming!

Suggested reading: PHP Objects, Patterns and Practice
by Matt Zandstra

PHP:

Minimum 5.4.2

PSR-4 for organizing and loading classes

PHP composer used to handle dependancies

The concept of the Drupal Kernel
Extends Symphony's HttpKernel



Module Dev: New web root file structure

Web Root:

```
Name
.editorconfig
.eslintignore
.eslintrc
.gitattributes
.htaccess
README.txt
composer.json
composer.lock
core
example.gitignore
index.php
modules
profiles
robots.txt
sites
themes
web.config
```

- new **Core** folder - guess what's stored here ;).
- Modules, profiles, themes and sites folders are for custom code
- .editorconfig file for IDE settings (ie 2 spaces in lieu of tab, etc...)
- composer config files
- .gitattributes file to prevent CLRF endings being committed to repo



Module Dev: CMI

CMI = **Configuration Management Initiative**

- translates to moving configuration from the database to the file system.

This will help with:

- Moving configuration between systems
 - ie Dev -> Staging -> Production
- Unify how configuration is stored among modules
- Allow developers to easily reset configuration to a previous state.
- Allow for human readable configuration, thanks to the use of YAML files for storage

@see <http://drupal.org/documentation/administer/config>



Module Dev: Entities

Entities are taking over with the build out of the Entity API.

Configuration v Content Entities:

- Configuration Entities - Entities that are stored in code, is CMI
- Content Entities - Stored in the data storage system (ie MYSQL, MariaDB, MongoDB, etc...)



Module Dev: Configuration entities

Configuration Entities live in code:

Stored using CMI

NOT “fieldable”

Example Configuration Entities:

Views

Content Types

Taxonomy Vocabularies

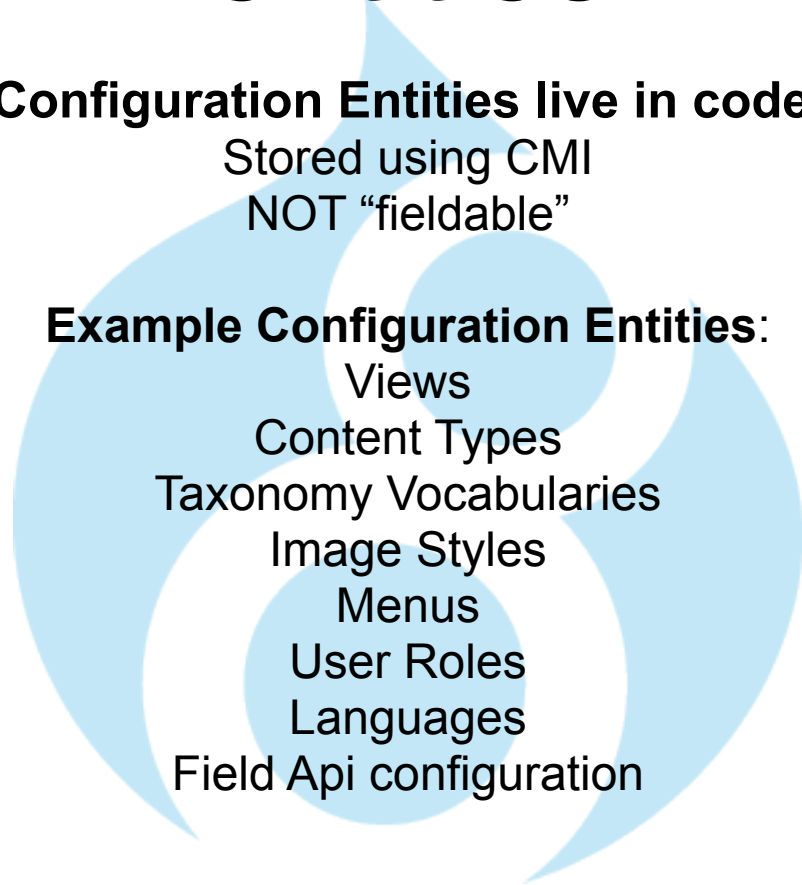
Image Styles

Menus

User Roles

Languages

Field Api configuration



Module Dev: Content entities

Content Entities are saved in the database:**

Nodes

Taxonomy

Files

Users

Comments

Aggregator Feeds and Feed Items

Contact Messages (these don't persist but are field able)

** I believe there are others :D



Module Dev: Entities cont.

More about Entities in Drupal 8

New APIs:

Entity Access API

Entity Translation API

Field Storage:

Fields are now bound to entities, in other words the same field cannot be used by two entities. This means you can name different field types the same thing on different entities.



Module Dev: EFQ

Entity Field Query was snuck into Drupal 7 at the last minute to support using MongoDB, and other SQL alternatives, as a data back-end. Entity Field Query is completely back-end agnostic, i.e. SQL, Document Storage engines, NoSQL systems, etc...

With Drupal 8 EFQ:

- Gets standardized to look more like the default DBTNG
- Move to the entity.query class
- Drop the distinction between fieldCondition and propertyCondition



Module Dev: Caching

New Cache API:

Two major themes here:

- New OOP syntax
 - `cache_get()` becomes `cache::get()`
- Cache Tags (<https://api.drupal.org/api/drupal/core!modules!system!core.api.php/group/cache/8#tags>)
 - Tagging system for cache records
 - `cache::invalidate('foo')` will delete all cache records “tagged” with the ‘foo’ tag. This happens regardless the cache ID, the cache bin etc...

Other:

- Sites and modules can implement their own policies for when requests and responses are cacheable.
- Bootstrap, config and discovery cache bins default to APCu

Note: content will move to being cached at the render layer, using the render array's #cache, this will



Module Dev: Plugins

Plugins: Small pieces of functionality that are swappable. Plugins that perform similar functionality are of the same plugin type.

Drupal contains many different plugins, of different types. For example, 'Field widget' is a plugin type, and each different field widget type is a plugin.

Plugins are defined by modules: a module may provide plugins of different types, and different modules may provide their own plugins of a particular type.

Example Plugins:

Field Types

CSS & JS preprocessors (aggregations/minification)

Actions

Path Alias Management

Field Formatters

Field Widgets

Image Operations (resize, scale, crop...)

Blocks



Module Dev: Services

Drupal 8 introduces the concept of services to decouple reusable functionality and makes these services pluggable and replaceable by registering them with a service container. As a developer, it is best practice to access any of the services provided by Drupal via the service container to ensure the decoupled nature of these systems is respected. The Symfony 2 documentation has a great introduction to services.

As a developer, services are used to perform operations like accessing the database or sending an e-mail.

Services are accessed using dependency injection which is a method of passing an object a service it depends on:

```
<?php
// Returns a Drupal\Core\Database\Connection object.
$connection = \Drupal::database();
$result = $connection->select('node', 'n')
  ->fields('n', array('nid'))
  ->execute();
?>
```



Module Dev: Services ex.

Example Services:

Database Connections

Entity Field Queries

Forms

String Translation

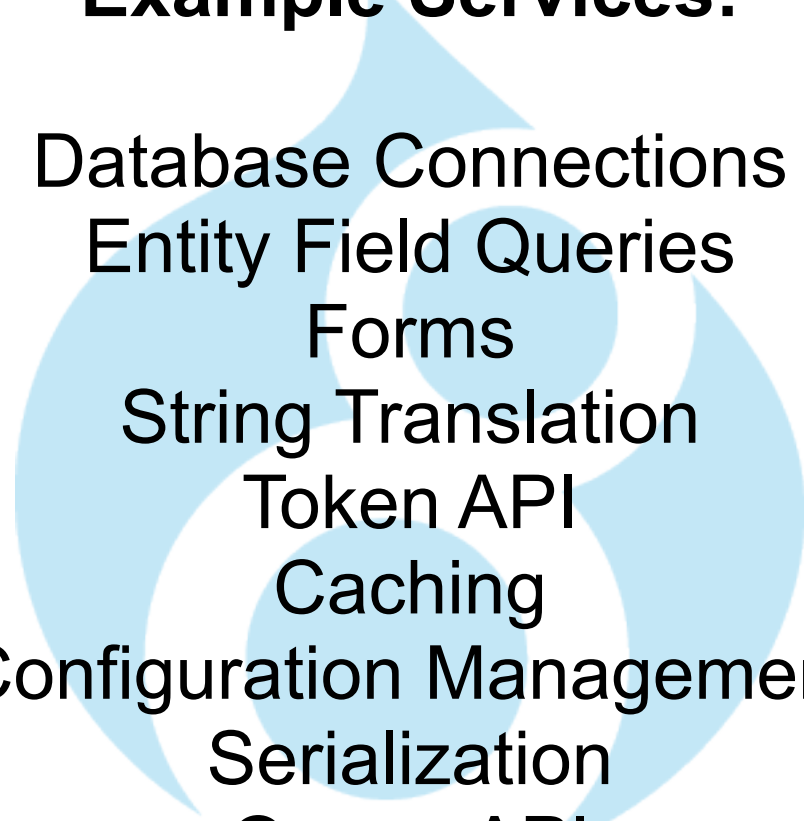
Token API

Caching

Configuration Management

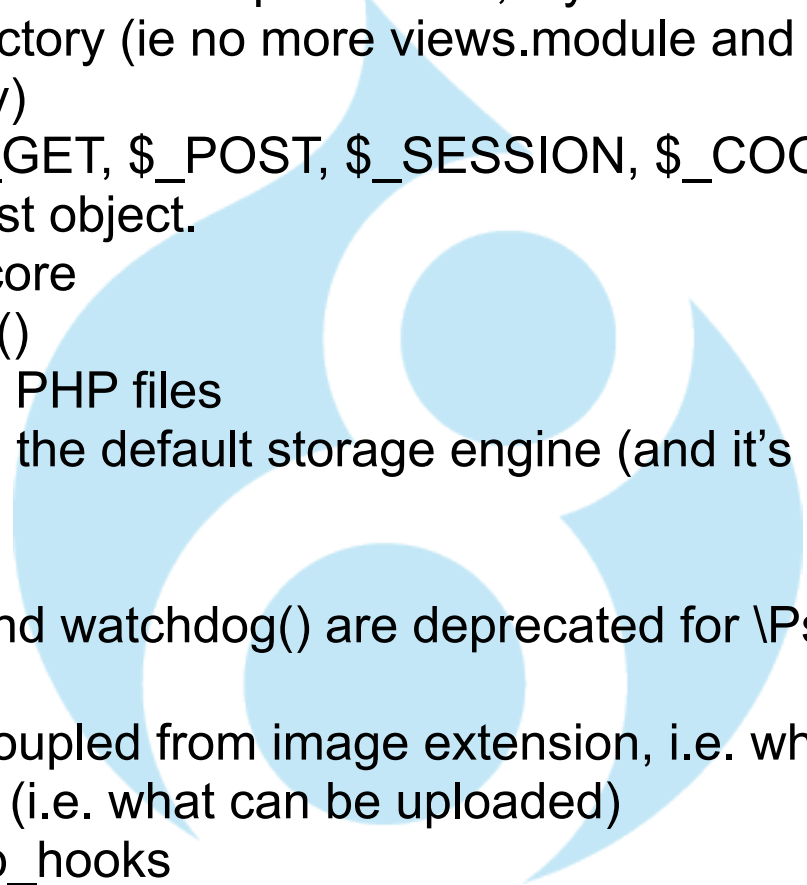
Serialization

Queue API



Module Dev: Little stuff

- You no longer need .module/.profile files, if you have them you can only have one in a directory (ie no more views.module and views)ui.module in the same directory)
- Super Globals (\$_GET, \$_POST, \$_SESSION, \$_COOKIE) replaced by Symfony Request object.
- UUID is finally in core
- no more hook_init()
- Modules can write PHP files
- MySQL InnoDB is the default storage engine (and it's fully supported and tested)
- no more arg()
- hook_watchdog and watchdog() are deprecated for \Psr\Log \LoggerInterface
- Image type is decoupled from image extension, i.e. when checking MIME types and support (i.e. what can be uploaded)
- No more bootstrap_hooks
- l() and url() are gone



3rd party libraries in core

jQuery @ 2.1
cookie, farbtastic, form, intrinsic, joyride, once
jQuery UI 1.10.2 (too many to list)
Guzzle
Symfony
TWIG
Underscore.js
Backbone.js
Zend_feed (parsing RSS/ATOM feeds)
PHPUnit
Modernizer (slightly customized)
DomReady
Matchmedia
Picturefill

