Drupal CMS Cheatsheet

A comprehensive Drupal CMS cheatsheet covering essential concepts, modules, theming, and development practices.

Core Concepts & Architecture

Drupal Core Components

Nodes	The fundamental content entity in Drupal, representing articles, pages, and other content types.
Taxonomy	A system for classifying and organizing content using vocabularies and terms (categories, tags).
Blocks	Reusable content elements that can be placed in different regions of a website (navigation menus, sidebars).
Users	Registered users with defined roles and permissions to access and manage content.
Modules	Extend Drupal's functionality by adding new features, content types, or integrations.
Themes	Control the look and feel of a Drupal website using templates, CSS, and JavaScript.

Key Drupal Directories

`/core`	Contains Drupal core files.
`/modules`	Stores contributed and custom modules.
`/themes`	Holds contributed and custom themes.
`/sites/default/files`	Default location for uploaded files.
`/vendor`	Stores composer dependencies.

Content Entity API

Drupal's Entity API provides a standardized way to manage content, allowing for creating, reading, updating, and deleting (CRUD) operations on various entities like nodes, users, and taxonomy terms.

Key concepts:

- **Entity Type:** Defines the type of content (e.g., node, user).
- **Bundle:** A subtype of an entity type (e.g., article, page for nodes).
- **Fields:** Data attached to entities (e.g., title, body, image).

Module Development

Basic Module Structure

A Drupal module typically consists of the following files:

- `module_name.info.yml`: Contains metadata about the module (name, description, dependencies).
- `module_name.module`: Main PHP file where you define hooks and custom functions.
- `module_name.routing.yml`: Defines routes for accessing module-defined pages.
- `module_name.permissions.yml`: Defines custom permissions for the module.

Common Hooks

`hook_node_insert(Drupal\Core\Entity\EntityInterface $entity)`	Called after a node is created.
`hook_node_update(Drupal\Core\Entity\EntityInterface $entity)`	Called after a node is updated.
`hook_node_delete(Drupal\Core\Entity\EntityInterface $entity)`	Called after a node is deleted.
`hook_form_alter(array &$form, Drupal\Core\Form\FormStateInterface $form_state, $form_id)`	Allows altering existing forms.
`hook_menu_link_defaults_alter(&$links)`	Alter menu links

Creating a Simple Module

1. Create a directory for your module (`/modules/custom/my_module`).
2. Create `my_module.info.yml` with module metadata.
3. Create `my_module.module` to implement hooks and logic.
4. Enable the module in the Drupal admin interface (`/admin/modules`).

Theming in Drupal

Theme Structure

A Drupal theme typically includes:

- `theme_name.info.yml`: Metadata about the theme (name, description, regions).
- `theme_name.libraries.yml`: Defines CSS and JavaScript libraries.
- `theme_name.theme`: PHP file for theme-specific functions and preprocess hooks.
- `templates/`: Directory containing Twig templates for rendering content.

Twig Templating

`{{ content }}`	Renders the main content of a page.
`{{ page.sidebar }}`	Renders content in the sidebar region.
`{{ node.title }}`	Displays the title of a node.
`{{ attributes.addClass('my-class') }}`	Adds a CSS class to an element.
`{{ kint(variable) }}`	Debugging tool to output variable information (requires Devel module).

Regions

Regions are defined in the theme's `.info.yml` file and represent areas on a page where blocks can be placed (e.g., header, sidebar, footer).

Example:

```
regions:
  header: 'Header'
  sidebar: 'Sidebar'
  content: 'Content'
  footer: 'Footer'
```

Drush & Drupal Console

Drush Commands

`drush cr`	Clear all Drupal caches.
`drush en module_name`	Enable a module.
`drush dis module_name`	Disable a module.
`drush updb`	Run database updates (after module updates or core updates).
`drush cim`	Import configuration from the `sync` directory.
`drush cex`	Export current configuration to the `sync` directory.
`drush sqlq "SELECT * FROM node;"`	Execute a SQL query.

Drupal Console Commands

`drupal generate:module`	Generate a new module with boilerplate code.
`drupal generate:theme`	Generate a new theme with basic files.
`drupal chain --file=command.yml`	Execute chain commands.
`drupal --help`	Display help and usage information.