Drupal 8

THE VIDERITY APPROACH

VIDERITY

STRATEGIC, CREATIVE, TECHNICAL

Introduction

DRUPAL 8: THE VIDERITY APPROACH

Viderity focuses on designing the "Total User Experience" for Drupal sites, using a user-centered design approach

Traditionally, many companies employ a technology-driven approach that focuses on a "build" point of view. User-Centric Design shifts the focus to place the solution's users at the forefront of design to make the Total User Experience the priority.

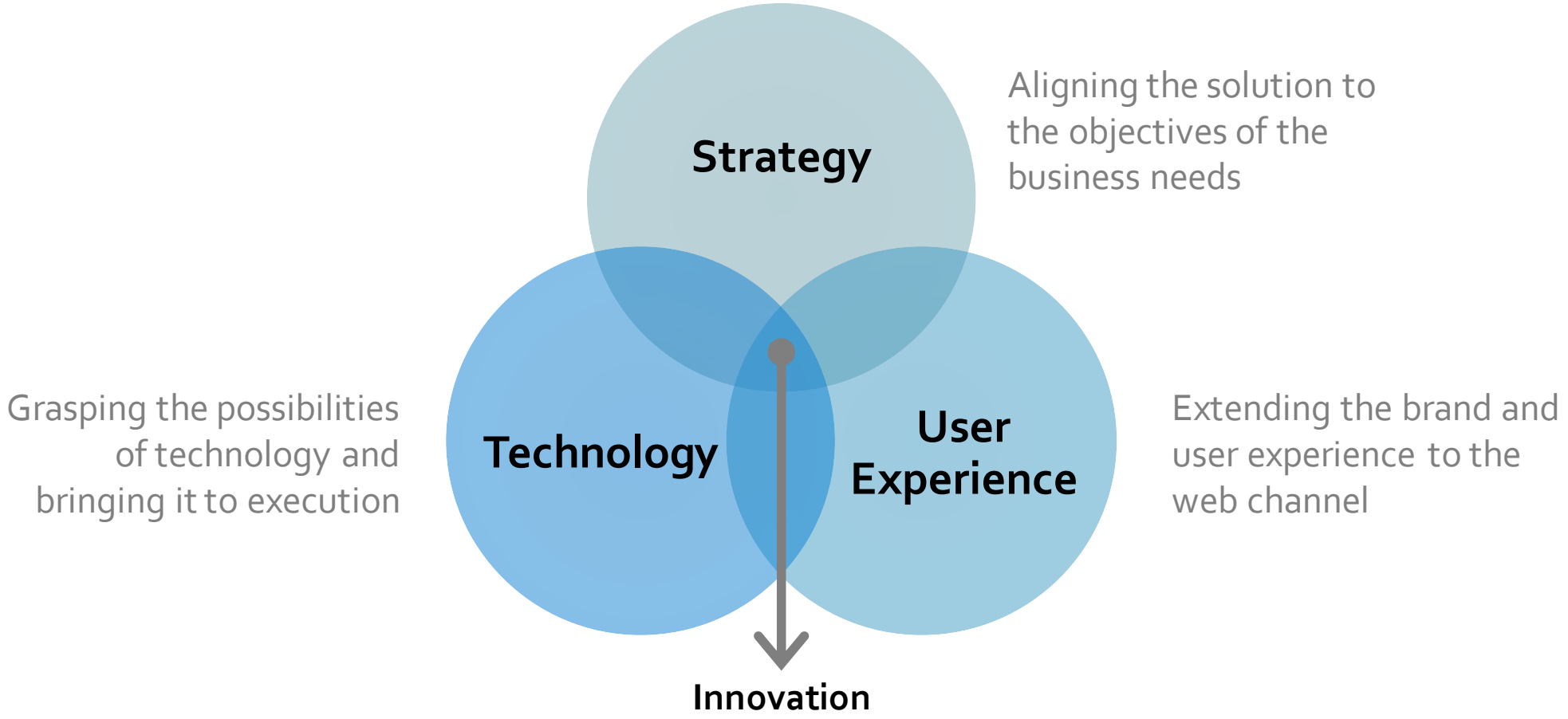
Tech-Driven Approach

- Technology driven
- Component focus
- Limited cross-discipline cooperation
- Focus on internal architecture
- No specialization in user experience
- Some competitive focus
- Development prior to user validation
- "Product Defect View" of quality
- Limited focus on user measurement

User-Driven Approach

- User driven
- Solutions focus
- Cross-disciplinary team work
- Focus on external design
- Specialization in user experience
- Focus on competition
- Develop only user-validated designs
- "User View" of quality
- Prime focus on user measurement

The artifacts produced through this approach drive both the front-end user experience as well as the back-end architecture



What does it mean to take a User-Centric Approach?

Project scope definition (business requirements documentation)

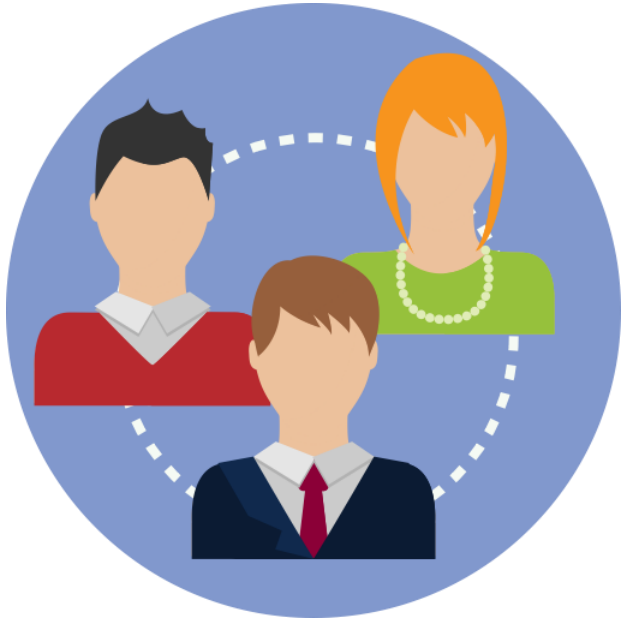
Audience definitions (user needs assessment/persona development)

User scenarios (identification of core tasks and use cases)

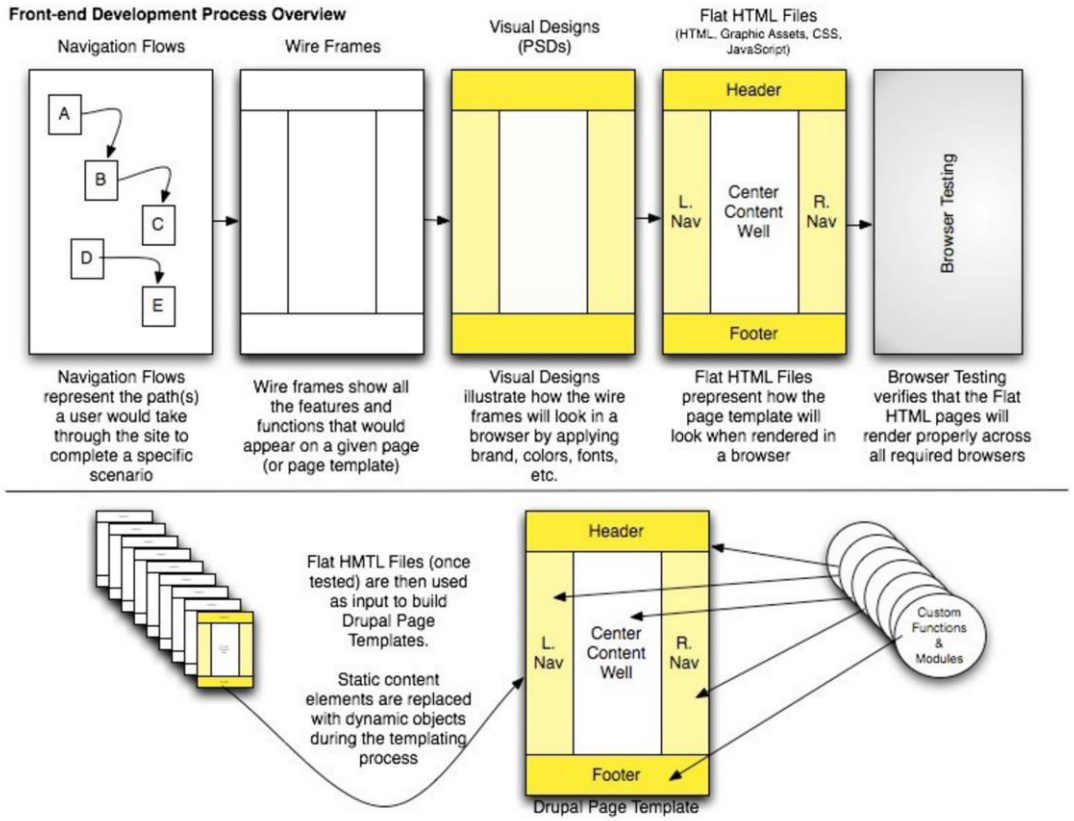
Feature matrix (mapping of functionality to user needs)

Core Deliverables:

- Navigation Models & Flows
- Wireframes
- Visual Designs
- Flat HTML
- Architecture Model
- Component Model
- Low-Level Design Document(s)



Put another way, by focusing on the user you are able to create a richer, more engaging user experience



How does Viderity use Drupal?

We have worked with many groups who have researched and elected to use Drupal due to its benefits in addressing customer needs as well as internal requirements.

Why use Drupal? Because it is:

- Fast
- Flexible
- Low Infrastructure Footprint
- Expandable
- Scalable
- Built upon a good foundation
- Low Total Cost of Ownership
- Strong support from the Open Source Community
- A lot of functionality is easy to "snap-in" and use

What does Viderity look for in a project when considering to use Drupal?

A business problem that has been defined.

- Both users and owners of the system's needs are well thought out

"Portal" like application where there is a need for more functionality than just a blog or simple one-way communication

- In those cases Word Press, or Moveable Type may be a better solution

Minimal connectivity to heavy or complex legacy systems

- Low technology impact implementation base (LAMP)

Target community using this system is usually in the 100-10,000 user range

- Think "Small / Medium business". Issues of (untested) scalability usually cause us to use differently scaled systems if the user base or complexity is larger.

Quick "module" based implementation or custom construction to change the personality of the system quickly.

What have been some of the challenges we have encountered?

Making the visual presentation "not so Drupal like".

- Most skins / templates we have seen are typical variations on the same look & feel
- Altering the navigation flow to better meet the needs of the user vs. what normally comes out of the box
- Working hard to avoid the "me too-ish" look

Creating an overall experience that meets the needs of the customer's brand without having to hack Drupal Core

Properly integrating true front-end "whizzyness" (e.g., AJAX, Flash, JavaScript) into the Model View Controller (MVC) model for Drupal

Properly taking into account the view and information needs of a particular customer type on the site

- Creating the User Experience through careful execution of Information Architecture and Visual Design techniques

What are some key estimating factors to consider?

Level of “stand alone” vs. “integration”

- If the site is not self-contained site and integrates with a back end system like SAP, the "module/portlet" and possible workflow are affected.

Level of functionality per user type

- The more "common" the user flow is to the base features of Drupal, the quicker things can get done.
- The more custom, well, the longer it would be to Information Architecture, Visual Design, and Application Development

Issues of scalability

- Number of simultaneous users
- Frequency of data changes
- Number of modules required to assemble a page of information

Lifecycle of data

- Is there going to be a lot of "turn on/ turn off" articles or need for long-term storage?

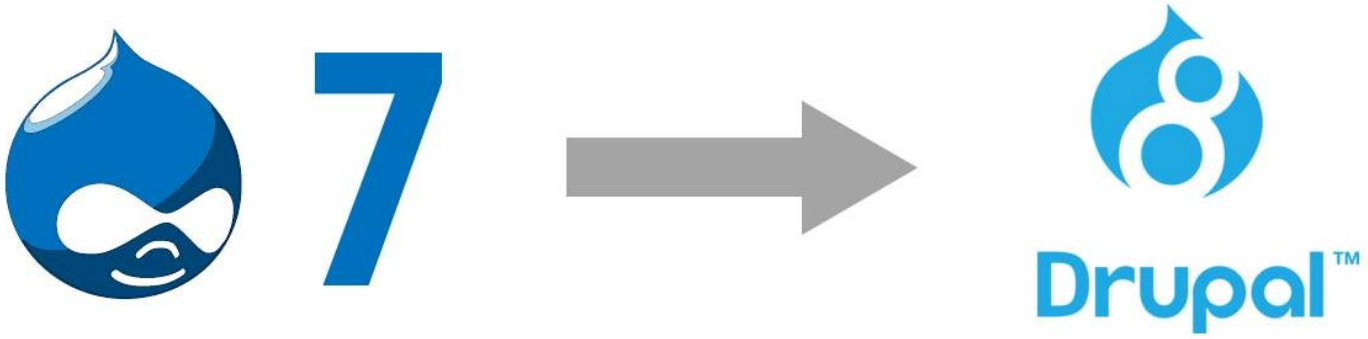
Migrating from Drupal 7 to Drupal 8

DRUPAL 8: THE VIDERITY APPROACH

Overview

The release of Drupal 8 in November 2015 brought new features and a brand new architecture to the Drupal foundation. Drupal 8 had a radical overhaul with the adoption of some components from Symfony, the new features from PHP 5 and 7 and new web development practices.

The Drupal architecture became a bleeding edge framework in the CMS community, and at the same time it brought a steep learning curve for everyone involved in the development process.



What Makes Drupal 8 so Different from Drupal 7?

Drupal 8 was reconstructed from the ground up, the major differences are:

- Drupal core is built on some components from Symfony
- The adoption of MVC and OOP
- More APIs
- The adoption of namespaces and PSR-4 for loading classes
- More entity types
- New data structure
- …many more

Learn more:

- The ultimate guide to Drupal

The Challenges

Since Drupal 8 has a brand new architecture, migrating data from a previous version of Drupal into a brand new Drupal 8 site brings the following challenges:

- **Configuration incompatibility**
  Drupal 7 core and contributed modules might use different configuration keys, database schema, global variables and data structure than Drupal 8.
- **No counterpart components and modules**
  As of December 2016, not all contributed modules have released a version for Drupal 8, and many of the modules already available are still in beta or alpha version.
- **Limitations in automated migration tools**
  Despite the number of tools available, due to the significant differences between configuration components, automated processes can't migrate an entire site on it's own. Some of the issues include: Missing upgrade paths, migrate modules in experimental stage, fail processes, and no patches available yet for know issues.

Even in the absence of errors, an auto-migrated D8 site is likely to have anomalies that a standard new D8 site does not.

Learn more:

- Brief overview, and history of automated upgrading to Drupal 8
- Migrate API overview
- Issues for Migrate

What Data is Migrated?

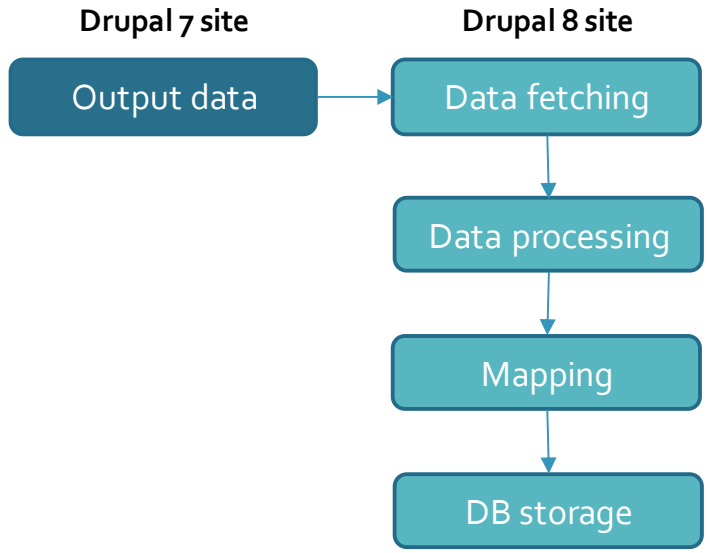
All of the data involved in a migration process includes:

- Settings and configuration
- Users
- Taxonomies
- Menus
- Entity Bundles
- Content Data
- Views

Migration Process

Every site migration would follow the same general process:

1. **Preparing the data source**
   This is where the data to be migrated into the new site gets prepared and becomes available for the migration tool.
2. **Data fetching**
   The new Drupal 8 site access the prepared data and pulls it into the migration system.
3. **Data processing**
   Any data that needs to be reformatted or restructured gets processed in this stage.
4. **Mapping**
   The new structured and processed data gets mapped into the fields and configuration keys for database storage.
5. **Database storage**
   The processed and mapped data is saved into the database.
6. **Testing**
   The data migrated into the Drupal 8 site is compared with the source site.



Tools Available for a Site Migration

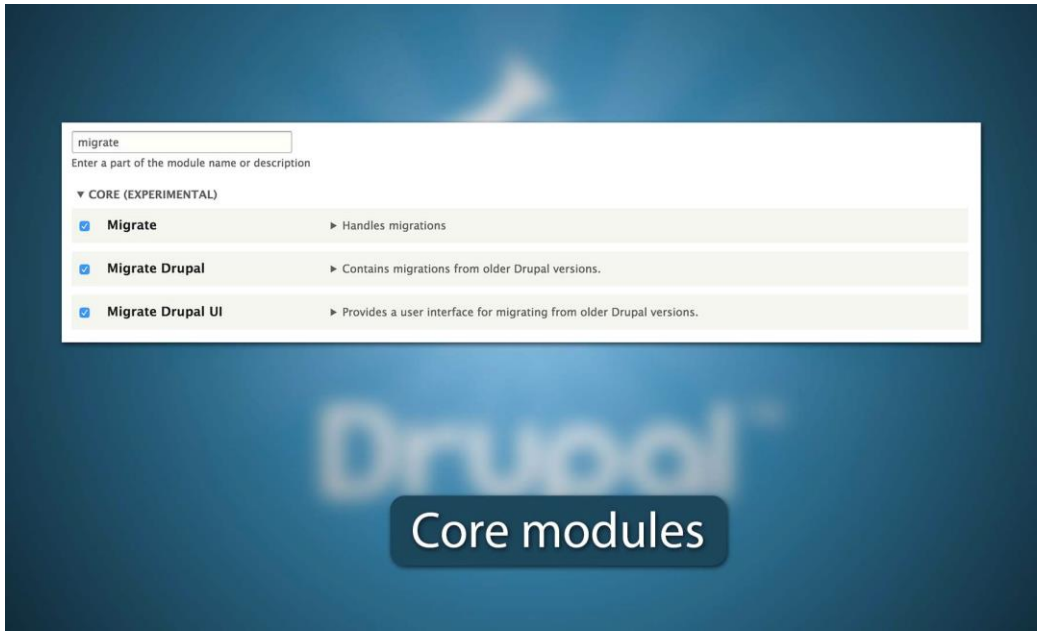
1. Core Modules

Drupal 8 comes with three core modules to perform a data migration:

- Migrate
- Migrate Drupal
- Migrate UI

The goal of these modules is to complete a migration from D6 / D7 to D8 and they include a UI to connect to the database and files location from the existing Drupal site.

These modules are still in the experimental phase, still throw errors and does not complete the migration process without inconsistencies.



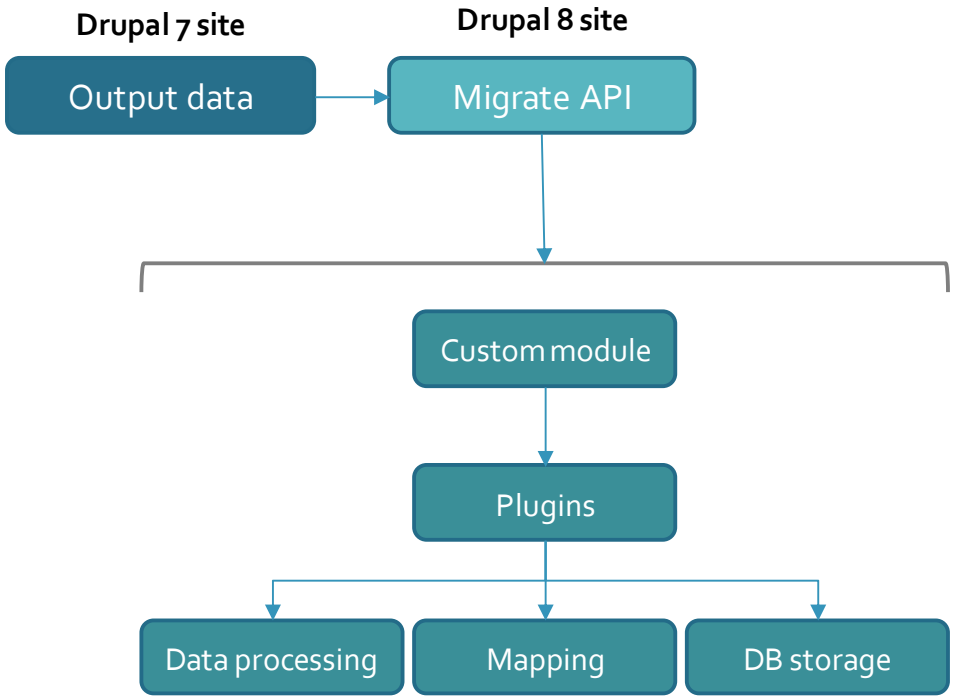
Learn more:

- Upgrade using the migration user interface

Tools Available for a Site Migration

2. Migrate API

Drupal 8 also offers a migration API that can be complemented and extended with contributed modules. This API offers a migration framework so developers can write custom modules that would implement custom plugins for data mapping and processing between the source and destination. The use of this API would require extensive programing and testing. It's implemented through Drush and it offers the option to rollback and keep track of data migrations.



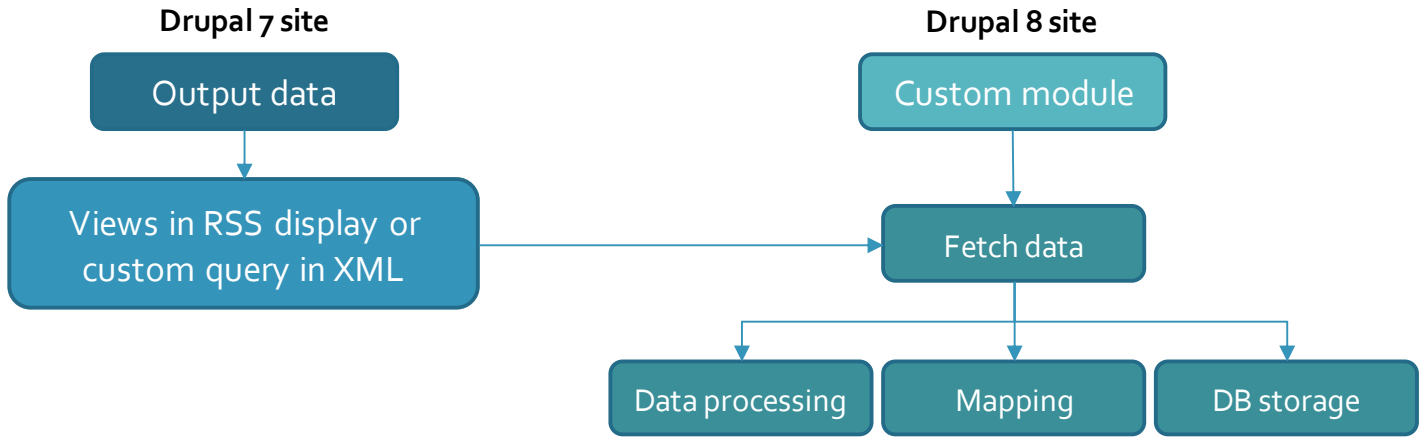
Learn more:

- Migrate API overview

Tools Available for a Site Migration

3. Custom Feeds and Modules

Using the power of views, RSS feeds and custom queries to output any data and deliver it in XML format. Later it would get processed by a custom module that would handle the data mapping and database storage.



Tools Available for a Site Migration

4. Contributed Modules

Other contributed modules extend the migrate API to integrate more tools in an UI or in the command line with Drush.

Some of those modules are:

- Migrate Tools
- Migrate Plus
- Migrate Manifest
- Migrate UI

Learn more:

- Brief overview, and history of automated upgrading to Drupal 8

Conclusion about Migration Process

As the Drupal community keeps moving forward in the upgrade and transition form D7 to D8, we are in a period of time where due to major differences between both systems, it will be necessary to create and implement custom scripts to successfully achieve a complete and consistent migration.

There is no a single tool that would perform a complete migration and the approach to solve the migration challenges should not be limited by the use of a single tool.

Theming

DRUPAL 8: THE VIDERITY APPROACH

Theming Advantages and Improvements in Drupal 8

Some of the most noteworthy advantages in developing a theme in Drupal 8 are:

- An extra layer of security with Twig
- Distribution of theme assets with libraries
- More possibilities to override system-generated output
- Integration of third-party jQuery plugins
- Bootstrap as a front-end framework
- Theme settings

An extra layer of security with Twig

Drupal 8 adapted Twig as its template engine. This engine removes the possibility to write functionality in the templates by not running executing PHP. This way the presentation layer is strictly for theming and all of the back-end functionality gets executed in the PHP files.

Other advantages of using Twig are:

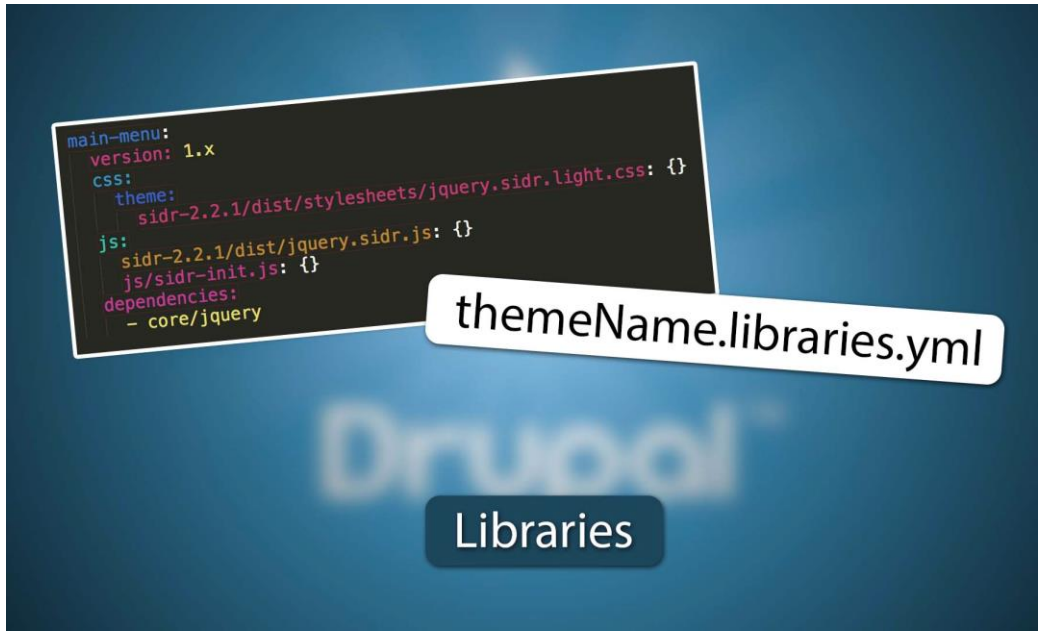
- Templates can be reused
- Template elements can be overridden
- We can add or remove more variables from preprocess functions
- It supports kint() to dump data
- Theme libraries can get loaded from templates
- Fast, secure and flexible

Learn more:

- Twig in Drupal 8
- Twig Official Documentation

Distribution of theme assets with libraries

Drupal 8 includes the possibility to break up and group the theme files into libraries. The main advantage is that the required libraries get loaded only when necessary as opposed to loading every theme file on all pages.



Learn more:

- Adding stylesheets and JavaScript to a Drupal 8 theme

More possibilities to override system-generated output

When the system-generated HTML markup is not what we need, we can always override it with preprocess functions in the .theme file and with the custom templates that would override a system template.

The new Drupal debug mode offers template name suggestions to easily override a system template.

```
<!-- THEME DEBUG -->
<!-- THEME HOOK: 'field' -->
<!-- FILE NAME SUGGESTIONS:
  * field--node--title--page.html.twig
  x field--node--title.html.twig
  * field--node--page.html.twig
  * field--title.html.twig
  * field--string.html.twig
  * field.html.twig
-->
<!-- BEGIN OUTPUT from 'core/themes/stable/templates/field/field--node--title.html.twig' -->
```

Template name suggestions

Learn more:

- Working with Twig Templates

Integration of third-party jQuery Plugins

Developing the UI in a Drupal environment is not limited to what Drupal offers.

Integrating the front-end jQuery plugins would allow us to create interactive UI components like:

- Carousels
- Galleries
- Mega menus
- Modal windows and an infinite number of UI components.

A new jQuery plugin can be encapsulated in a theme library and then get loaded only when a template requires it. This way would improve performance and speed by only loading the necessary files.

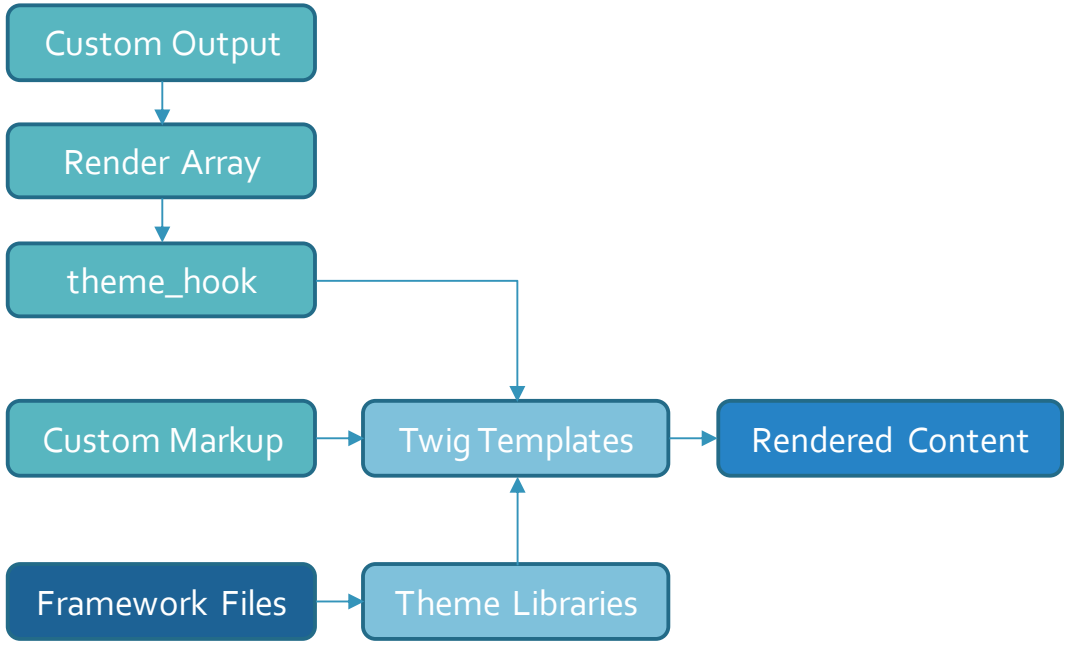
Learn more:

- Attaching libraries from templates

Integration of front-end frameworks

The presentation layer gets more flexible and granular with the adoption of the MVC pattern, Twig, and the theme libraries. That brings a more efficient way to integrate any other front-end framework or plugin. Drupal 8 handles these key components in the following way:

- **Page layouts**
  The entire layout can be handled with a HTML responsive framework like Bootsrap, Foundation , jQuery Mobile or any other. The framework markup is integrated in the Twig templates using the Drupal template system.
- **Framework files**
  The framework files are loaded only when needed by calling the theme library form custom templates.
- **Render arrays themed with custom Twig templates**
  Every custom output structured in a render array can be themed with a custom Twig template through a theme hook. The custom template can use any component form the front-end framework.



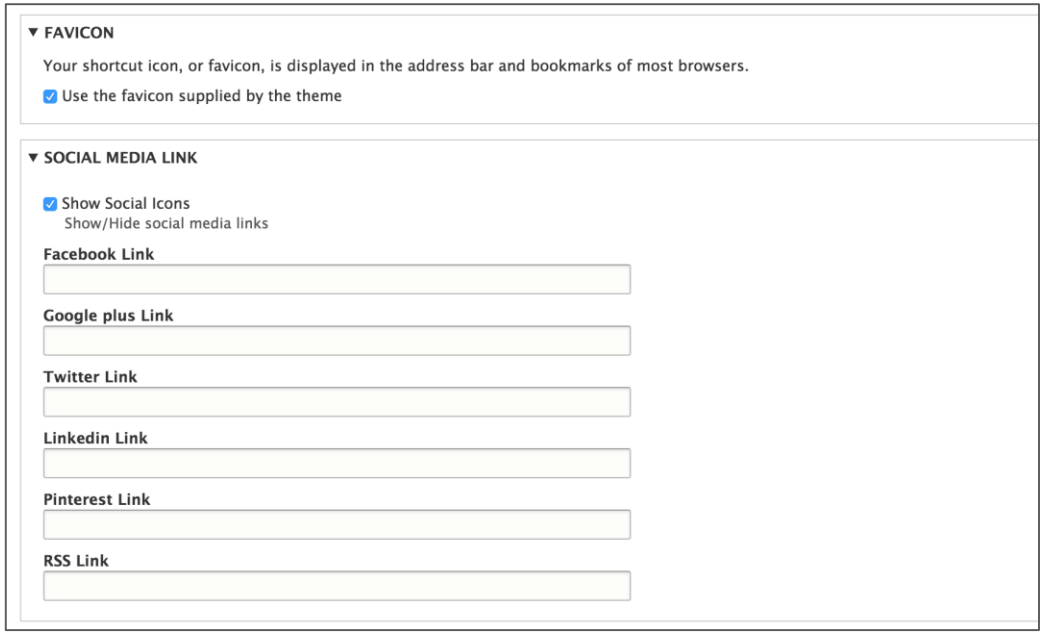
Learn more:

- Render Arrays

Theme Settings

Every Drupal theme has it's own page configuration at **admin/appearance/settings/themeName**

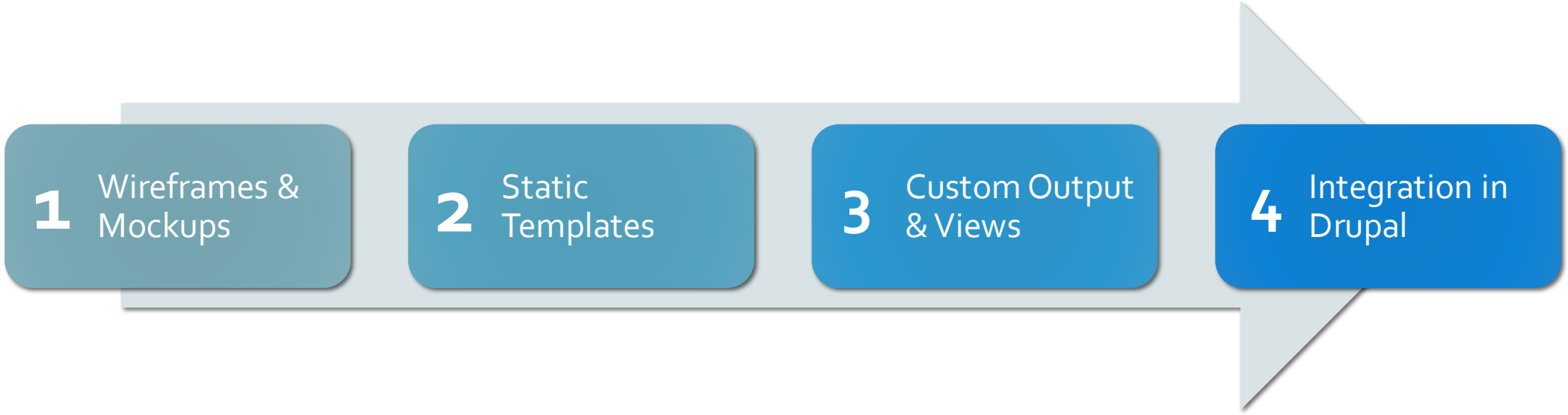
In case site administrators would need to change some theme configuration like background colors, page logos, favicon, etc., Drupal has a great API to easily create the configuration form and implement that user input across the theme.



The Drupal 8 Theming Process

The Drupal 8 theming workflow is similar to Drupal 7, but the implementation is significantly different with the introduction of these key elements:

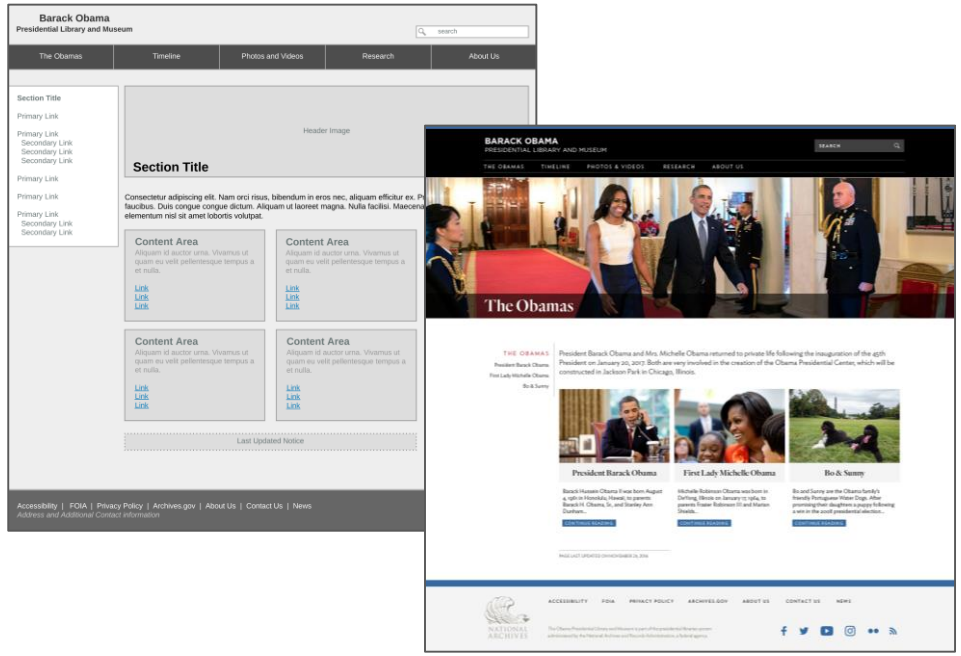
- Twig as a template engine
- Libraries for grouping theme assets
- Implementation of the MVC (Model-View-Controller) pattern.



1: Creation of Wireframes & Mockups

This is the stage where the desired UX and the user interface design come together. Designing the UI is not only about creating the look and feel of the general theme, but also the thoughtful design of intuitive usability.

- Development of creative concept
- Graphic design of page elements
  - Menus
  - Sidebars
  - Content areas
  - Footers
  - Etc.
- Design of general layouts and templates



2: Development of Static Templates

The page elements and layouts are developed using HTML 5, CSS 3 and JavaScript. The user interface is mobile responsive and cross browser compatible.

This part includes:

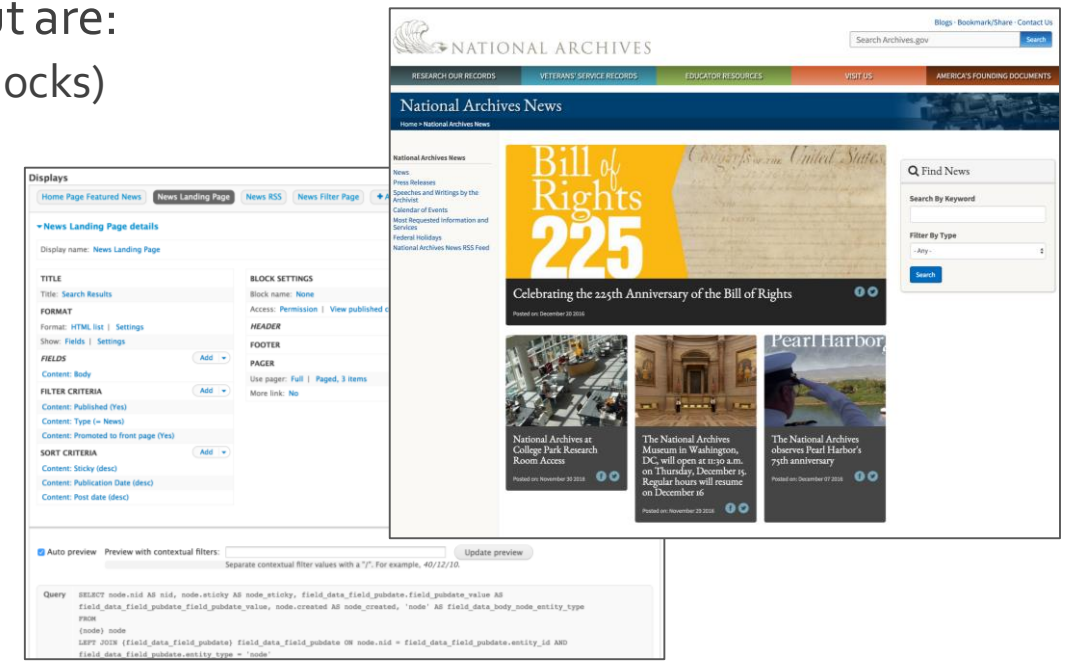
- Integration of a mobile responsive framework like Bootstrap
- Development of generic UI elements and general layouts
- Integration of jQuery plugins for components like: carousels, interactive galleries, modal windows, accordions, and other UI widgets.

3: Custom Output & Views

Theming a site also involves creating the necessary views and custom output. To achieve that, Drupal has a flexible system and API to generate custom views and output with custom queries.

The options we have to generate custom output are:

- Creation and configuration of views (pages and blocks)
- Configuration of content regions and blocks
- Preprocess functions
- Overriding system templates

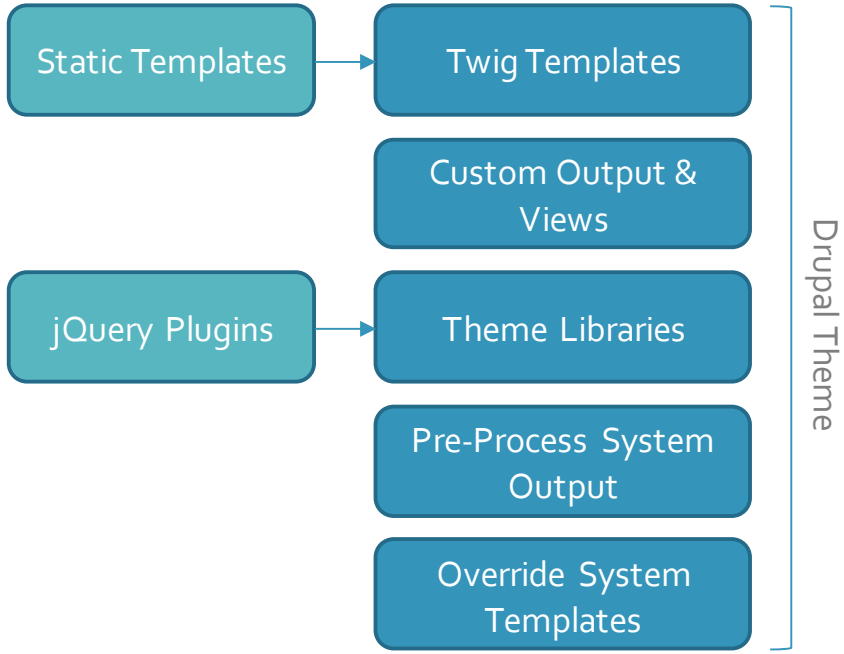


4. Integration in Drupal

The architecture in Drupal 8 includes the MVC design pattern, and the theming process takes place in a layer separated from PHP and the back-end.

The custom templates previously developed in HTML 5, CSS 3 and JavaScript get integrated in a Drupal theme. This integration involves:

- Adapting the HTML templates into Twig templates
- Creating the theme libraries for a better organization and distribution of the theme assets and third-party jQuery plugins
- Preprocess output in the .theme file to achieve custom markup in system-generated output
- Overriding system templates for custom HTML markup



Learn more:

- The Drupal 8 render pipeline

Conclusion about the Theming Process

In general, the theming process in Drupal 8 is very similar as in Drupal 7, but the implementation is different with the adoption of Twig, the MVC pattern, and the theme libraries. This new approach makes Drupal 8 more secure, faster and brings a better way to organize the theme assets.

Learn more:

- Theming Drupal 8

www.viderity.com