Architecture Plan for the Web Experience Toolkit (WxT) v5

Reference Implementation for the Design System

Objectives

- Provide a reference implementation for the Design System, making it easier to implement and to keep pace with changes
- Enable implementers to use only the components they need through a modular architecture
- Make WxT more future-proof, enabling the framework to continually evolve with minimal impact to implementers
- Make it easier to contribute by reducing the learning curve and technology requirements

Current Industry Landscape

- Greater focus on modularity and loose coupling of code
- Emphasis is on maintainability through intuitive organization of code
- Steady evolution of baseline technologies (HTML, CSS, JavaScript)
- Node.js, Angular and React reign today but new frameworks and approaches are always on the horizon (e.g., ReasonML)
  - Vue's small following continues to grow (especially in China).
  - Facebook released ReasonML in late 2017 with more open-source releases expected from Facebook in the coming years.

Architectural Scan of Web Frameworkds

Framework	Architecture	Development Requires
ReactJS	Component	NodeJS(0.9+), react, react-dom
VueJS	Custom-attribute	None, Webpack
WxT v4	Monolithic	NodeJS (0.8+)
Angular	Component	NodeJS (0.9+), angular-cli
Ember.JS	Component	NodeJS (0.8+), ember-cli

Monolithic Architecture

WxT v4

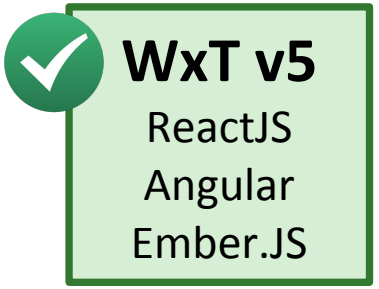
- Pros
  - All components are available in all cases
  - Rigid in design, minimizing HTML changes and downstream churn for implementers
  - Simple class-based approach to trigger features
  - All-inclusive bundle with theme and functionality
- Cons
  - Heavy footprint with little room for customization
  - All-inclusive bundle is prone to conflicts with applications and CMS systems
  - Component locations are unintuitive
  - Hard to debug the root cause because of the multi-layer complexity
  - Difficult to adapt to newer frameworks and approaches

Custom-Attribute Architecture

VueJS

- Pros
  - Maximum flexibility, enabling implementers to innovate by stacking features
  - Lighter footprint due to the focus on discrete functionality rather than full components
  - Normally requires less releases since discrete functionality is less prone to change
  - Facilitates rapid prototyping
- Cons
  - More difficult to scale as the complexity of components/features increases
  - Steeper learning curve for those who are not strong with HTML or JavaScript (e.g., application developers and CMS maintainers) due to the abstract approach
  - Increased maintenance effort for implementers since HTML markup is more likely to be impacted by component/feature updates

Component Architecture



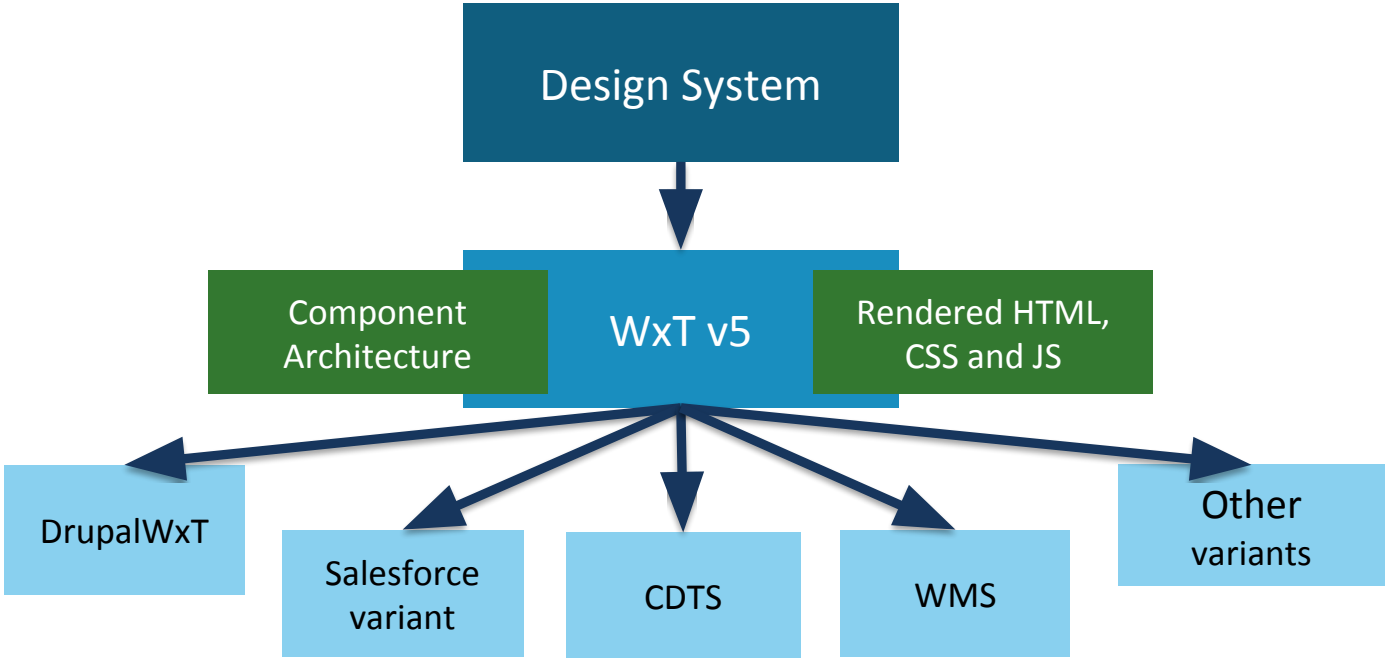
- Pros
  - Aligned with major frameworks such as Drupal, ReactJS, Angular and Salesforce
  - Shallower learning curve due to the more intuitive code structure
  - Easier to implement since components are divided up cleanly, enabling implementers to focus on only the components they need
  - Easier to debug and maintain due to less code complexity and duplication
  - Easier to leverage with JS specifications like CommonJS, AMD and ES6 Modules
- Cons
  - Usually requires loaders or dependency managers
  - Can be more challenging to implement code quality tools due to the modularity
  - A little less flexible when prototyping than custom-attribute architectures

How WxT v5 Supports the Design System

Specifications

Reference implementation

Concrete implementations



WxT v5 Architecture Details

- Component architecture implementing the AMD JS specification
  - Aligning with the architectures of CMSs and applications frameworks to simplify implementation
  - Ensuring compatibility with modern day build systems
  - Making it easier to customize the layout and design, including at the component level
- Focused on widely supported international standards (HTML, CSS and JS) to help future-proof the framework and to maximize compatibility
- Minimizing requirements for contributors, such as supporting lightweight browser-only development

WxT v5 Variants

- **Variant:** Concrete implementation of WxT v5 for a specific platform or framework (e.g., Drupal, Salesforce)
  - Makes it easier to implement and keep pace with WxT v5.0 and the Design System
- **WxT v5 goals for supporting variants:**
  - Minimize time and effort needed to build and maintain WxT v5 variants
  - Maximize flexibility and compatibility for variants by providing “component” and “rendered” implementation options
  - Include variant maintainers in evolving WxT v5 to ensure it continues to meet their needs

90 day roadmap

Architecture

* Proposed soft milestones

Oct

- Initial project setup
- Tools and build system integration
- Develop WxT v5 component architecture core

Nov

- 30% of components implemented
- Implementation review

Dec

- 60% of components implemented
- Alpha peer review

Jan

- 80% of components implemented
- Beta lockdown

Questions?