SuperFriendly

Adventist.org Technical Approach Document

Infrastructure improvement requests

Technologically speaking, the Seventh-day Adventist Church has an infrastructure problem. This stunts the organization's ability to scale digitally and empower its subsidiaries and partners to communicate more broadly.

Addressing a handful of these pieces can facilitate significant digital change across the organization. The three specific areas we'll focus on in this short document are:

1. Consolidating the number of content management systems that currently exist
2. Distributing and reusing content across the organization (and the current inability thereof)
3. Existence and governance for a centralized design system

Addressing those in short order can act as a major technological investment and can help springboard the church in accomplishing its mission of evangelizing its beliefs throughout the world.

1. Consolidation

Simply put, there are too many content management systems in use across the Seventh-day Adventist organization. We've heard reports of Drupal, WordPress, Squarespace, NetAdventist, Typo3, SimpleUpdates, ModX, and many more. The first step is to incentivize an organizational movement toward one system.

Drupal is the likely candidate for its current use, organizational robustness, and enormous active community, but the final system choice still needs more research and discussion.

2. Content distribution and reuse

As an international organization, the SDA church can certainly make better digital use of its reach. Local news—like SDA pastors' involvement in quelling Baltimore riots—need better avenues for elevated to international levels. General Conference-sized issues—like legislation on female ordination, same-sex marriage, and many other important issues—need easy ways to get to the local level.

If the first step is consolidation of content management systems, the second is decoupling content from its views. So we're all on the same page, "a decoupled content store splits the content of a website from how it is displayed into multiple independent systems. Decoupled sites are the logical evolution of splitting content from templates in current CMSs" (via Lullabot). Most CMSs are setup to both manage content effectively in

addition to displaying it in a template, largely web-based ones. However, decoupling has a few great advantages:

- Moving to an API-based way of pulling in content paves the way for a methodology that transcends web-only platforms and can easily extends to native apps and other outputs.
- An API can be both private and public, allowing many developers—both within and outside the organization—to be able to build useful things with your data
- Decoupled systems train content producers to write content, agnostic of form, leading to less "dirty" and more orbital content that is modular and portable.

(You can read about more of the benefits of decoupled architecture or headless systems at Pantheon or Contentful.)

3. A shared pattern library

As you know, we're working together on creating the foundation for a pattern library. (At least we hope you know; if you don't, we should talk!) To boil it down, a pattern library or system is a collection of elements that you can use multiple times in multiple ways across a site or sites. The goal for this pattern library is that it becomes a great resource for anyone within the Seventh-day Adventist church who needs to quickly build a website that's tied to the organizations.

However, the success of achieving this goal comes with its own inevitable problem: you then have dozens—or hundreds, or thousands—of clones of this pattern library that no longer maintain a connection to the original. When you make updates to the original pattern library, none of the clones receive the updates unless its maintainer incorporates those updates manually.

The next evolution of this pattern library is to really fulfill the "living" part. The Lightning Design System—Salesforce' living design system—sees itself as a single source of truth for all Salesforce apps. As an example, their design tokens resource can output to multiple formats like Sass, JSON, Stylus and more through a tool called Theo. As the Lightning Design System team updates the product, they roll out to anyone using these design tokens. Lonely Planet built a Component API so that there was a standardized way of using the pattern library in a way that was still tied to the original. Again, as the original components get updated, the changes roll out to every instance in use.

We highly suggest building a way for original components to maintain a way with the versions of the elements in use.