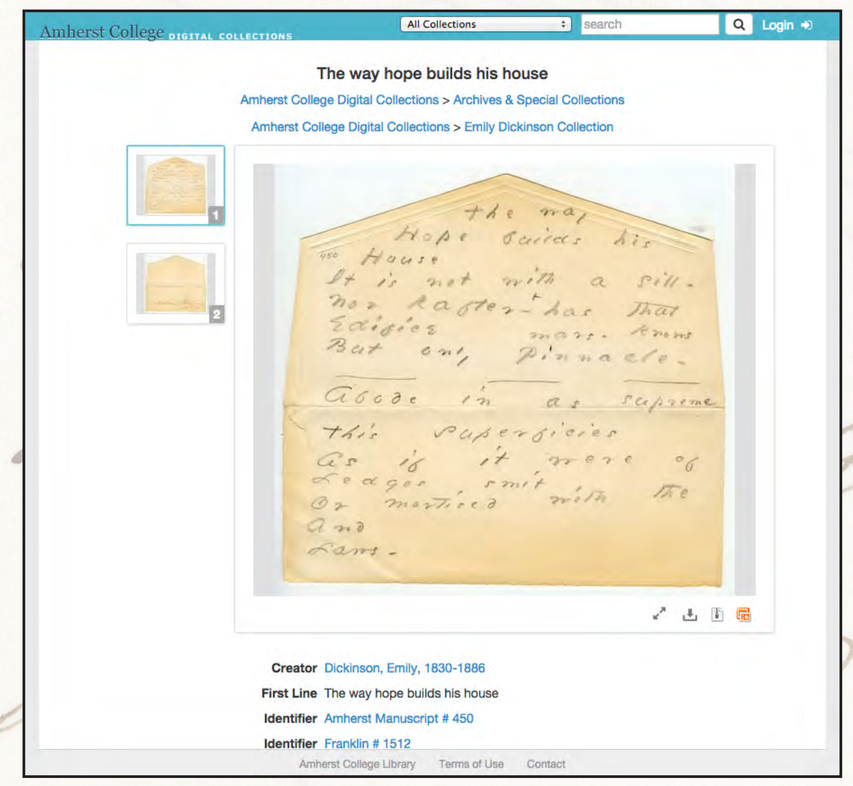
Emily Dickinson on the iPad
digital collections at Amherst College



At Amherst College, as we began designing the user interface for our digital repository, a key goal was to make the search interface extremely fast and interactive. In addition, we wanted the search queries to follow a clean and easily understandable URL pattern. Thus, we began to make extensive use of the HTML5 pushState() feature. This also allowed us to not break the expected behavior of the browser's back button for users. Initially, we used only jQuery for this, but it quickly became much easier to organize the code into a MV* JavaScript framework. For two years now, we have used Backbone.js, and it has worked exceedingly well for our development.

In order to make the site responsive to different device sizes, we are using Twitter's Bootstrap library. Bootstrap is very easy to work with, so the challenge was to make a Bootstrap-based design not look like every other Bootstrap-based site on the web. Also, we wanted a more semantic layout of the HTML, free of a profusion of Bootstrap css classes. So we have made significant use of Bootstrap's mixins with custom Less code.

In the mobile context, the site works basically the same as in the desktop context, though the layout is somewhat simplified. And depending on the device orientation, certain elements will appear in different locations. There are only a small number of features in the desktop version that are not available to smaller devices, notably certain drag and drop functions, though mobile device users are able to navigate high-resolution images with multi-finger gestures in ways that are not possible with desktop browsers.

A single page app

Initially, only portions of the site functioned as a single page app, notably the search interface. This was mostly because we wanted to ensure that the content was being properly indexed by search engines. Once we solved that issue and in order to better support mobile devices while also making the entire site feel more responsive, we moved the entire front-end into a single page app. The app uses Backbone.js as a MV* framework.

What this means is that when a user lands on the site (any page), that page is generated completely on the server side, but when the user navigates to other pages, data is requested over a RESTful API. That data is either JSON or binary content (images or other media). That is, the site is really just a big API.

This also makes navigation really fast and efficient.

Challenges

With more of the application written in client-side JavaScript, the two most significant challenges involved keeping the size of the main JavaScript file small (i.e. initial page load) while also supporting search engine indexing.

By using some standard compression and minification techniques along with asynchronous loading in the browser, the initial page load is kept to a minimum (under 1 second).

The SEO issue was far more complicated to address. Effectively, if a user starts on page A and then browses to page B, the HTML of page B should be isomorphic to what a crawler would retrieve by requesting page B directly. To make this easier, the same HTML templates for the various javascript views are used both by the browser when navigating through the site and by the server when generating an individual page. This templating system uses Mustache, which is compatible with a broad range of languages, and doesn't tie the backend to any particular language or execution framework.

Want to know more?

Talk to Aaron Coburn <acoburn@amherst.edu>,
Kelcy Shepherd <kshepherd@amherst.edu> or
Anita Rao <abrao@amherst.edu>

We're hiring!

Using Hydra

The user interface for acdc.amherst.edu does not use Hydra directly. The "interface" is really just a RESTful API, and it is written in Node.js. Behind Node.js (and a caching layer that uses Riak), there is still Fedora. At present, we are working on a Hydra head that will allow our librarians to edit the content in the system. By making use of Fedora's JMS (messaging) system, integrating the two systems is not overly difficult. And for that we use Apache Camel.

acdc.amherst.edu