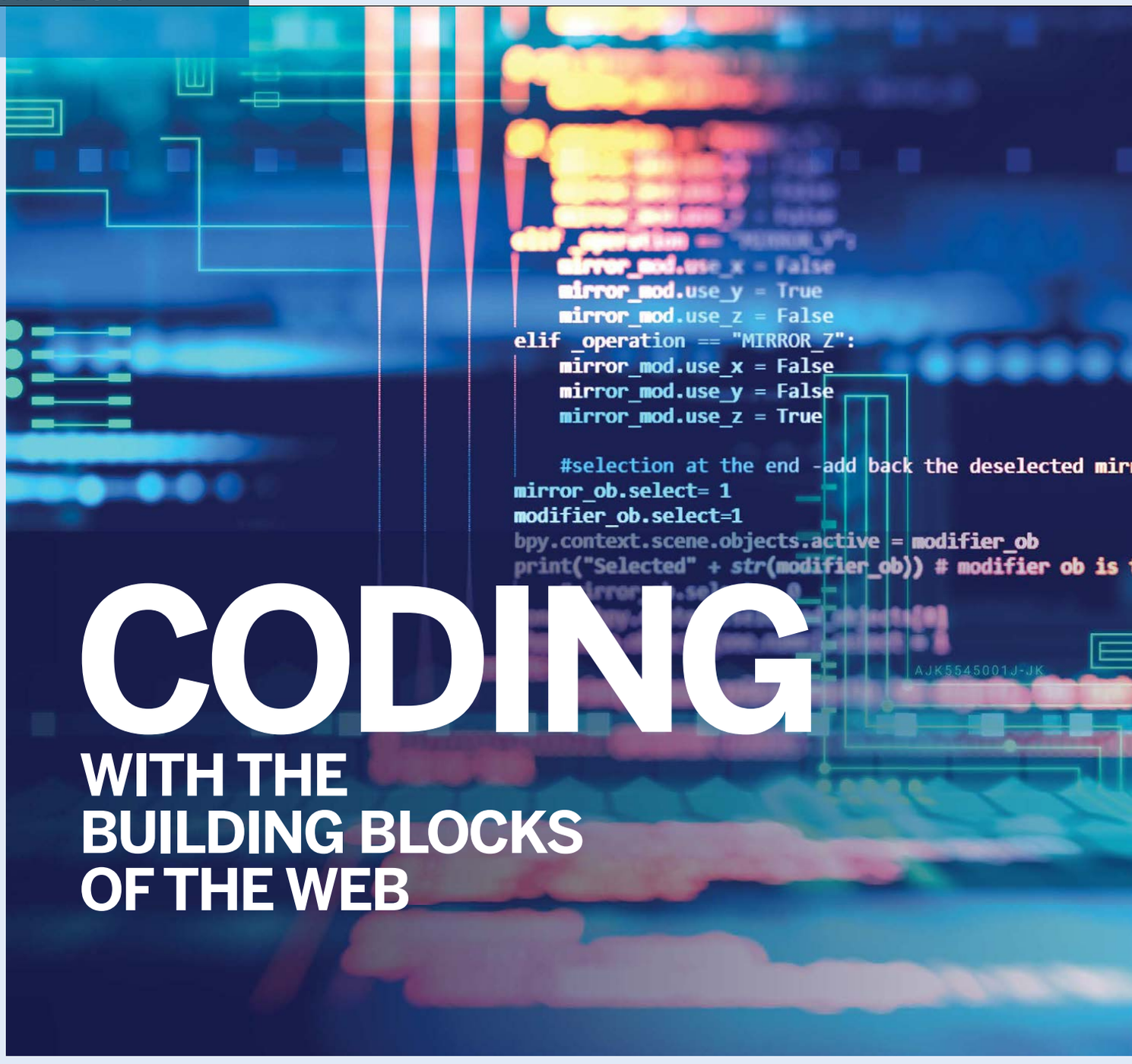


# CODING WITH THE BUILDING BLOCKS OF THE WEB

How law librarians can use JavaScript, jQuery, and JSON to create surprisingly powerful customizations and applications, while solving real-world problems on their own.

**BY NICK SZYDLOWSKI**

Should every librarian code? That debate has been going on for more than a decade, with no resolution in sight. It may be less controversial, though, to argue that librarians need to understand the foundational technologies that power the web. These technologies are the basis for public-facing catalogs, digital collections, and online resources, and increasingly for integrated library systems and other back-end systems as well. However, the web changes rapidly, and unless librarians update their skills and their understanding, they can quickly fall behind. Not even full-time developers can keep up with every new tool or approach that comes along, but by focusing on technologies that are powerful, established, and designed to integrate with existing systems, librarians can take advantage of their existing knowledge and expand their capabilities.

A basic understanding of HTML (Hypertext Markup Language) and CSS (Cascading Style Sheets) has been a part of many library and information school programs for years. While every webpage uses HTML, and pages that lack CSS are now as rare and antique as a Ford Model T, a third technology, also developed in the 1990s, has become nearly as ubiquitous. The internet research firm BuiltWith estimates that 94.1 percent of the million most trafficked sites on the web use JavaScript, and that one JavaScript library—jQuery—is now used on 88.8 percent of the top million sites.

The popularity of JavaScript libraries like jQuery, along with newer frameworks like Angular.js and React.js, has contributed significantly to the dramatic changes that the web has undergone. For instance, these frameworks have enabled the creation of webpages that look and feel like desktop applications, while also fueling the popularity of the JavaScript-friendly data format JSON. Moreover, because jQuery is so well-established, a wealth of documentation and open source code exists that can help librarians who are not full-time web developers create surprisingly powerful customizations and applications.

The popularity of JavaScript libraries like jQuery, along with newer frameworks like Angular.js and React.js, has contributed significantly to the dramatic changes that the web has undergone.

## JavaScript and jQuery

JavaScript is a scripting language, first deployed in 1995 as part of the Netscape Navigator web browser. It is frequently used as a client-side language, allowing website creators to add scripts to a page, which are then executed within the user's browser. jQuery is a JavaScript library, first released in 2006, that simplifies many of the basic tasks needed to create interactive webpages. Among its other features, jQuery provides standard methods for creating, removing, and manipulating the HTML elements that make up a webpage. These methods work consistently in different browsers, and the code to invoke them is simpler and easier to read and write than code that performs the same functions without jQuery. Libraries like jQuery contain a collection of reusable, pre-written functions, which can then be called by local scripts, allowing relatively simple local scripts to perform complex tasks. These libraries have become so popular that JavaScript written without using such libraries is commonly referred to (with derision or admiration, depending on the context), as "vanilla JavaScript."

For those who are interested in getting more comfortable with JavaScript, jQuery is a great place to start, because it leverages familiar CSS selectors like IDs and classes, some of which are likely already present in a site's code. For example, jQuery makes it easy to hide all of the links on a page that have a certain class:

```
jQuery('a.myClass').hide();
```

Many commonly used jQuery commands follow this formula: take some element on the page, as specified by the type of element (e.g., "a", a link) and/or another CSS selector like a class or ID, and then do something to those elements such as hide or show them, move them around the page, or replace their contents with something else entirely. By taking advantage of CSS selectors, jQuery makes it possible to add required functionality to existing web content, even without changing the underlying HTML. For instance, the popular LibGuides platform enables customization via JavaScript, allowing simple changes that can help libraries match the look and feel of their own sites on LibGuides.

Another advantage of jQuery is its large user base and the resulting profusion of open source code and plugins that extend its functionality. At Boston College Law Library, staff learned to use JavaScript by downloading simple code for common features like content carousels and word clouds, and then made minor modifications to fit local needs. Additionally, due to its popularity, many other powerful JavaScript libraries borrow jQuery's syntax. These include D3.js, a library for creating data visualizations that has become a go-to tool for digital humanities scholars and technologists. A basic knowledge of jQuery opens up a surprising number of possibilities.

## JSON

Librarians, especially those who work with metadata, are likely familiar with XML (eXtensible Markup Language), the data standard that powers an alphabet soup of web and library technologies, including MARCXML(MAchine-Readable Cataloging Markup Language), OAI-PMH (Open Archives Initiative Protocol for Metadata Harvesting), and RSS (Rich Site Summary). While XML isn't going anywhere, another human-readable data format has gained significant traction on the web: JSON (JavaScript Object Notation). While JSON can be used and interpreted by any language, it follows the syntax of JavaScript objects, meaning that it is already in a format that JavaScript expects and understands.

As a data format, JSON has some of the same advantages as XML. Both are

compact, but relatively easy for people to read and interpret. Both are hierarchical and can represent complex data models and information. As a simple example, shown below is a single row from a dataset used for an interactive tool displaying a list of student employment outcomes by region and state. The data is represented first in JSON, as used for the project, and then in XML.

Many web services make data available through application programming interfaces (APIs), and the output of these interfaces is very often available as JSON. Librarians accustomed to the relatively buttoned-up world of XML schemas, which standardize the way particular types of data are represented, may find the JSON environment to be more freewheeling, with data often provided in a structure that closely follows the logic of the application that produces it, rather than a standard or schema shared across applications.

```
{"Employer":"Southern Poverty Law
Center","State":"Alabama", "Region":"South", "Country":"United States"}
```

```
<employer>
<name>Southern Poverty Law Center</name>
<state>Alabama</state>
<region>South</region>
<country>United States</country>
</employer>
```

Some may be partial to one format or the other, but each has significant advantages for different tasks. XML benefits from a more developed set of related technologies for schema validation, transformation, and querying, as well as a rich set of established schemas. JSON is simple to parse and create using JavaScript, and a variety of JavaScript tools exist to display, manipulate, and visualize JSON data on a webpage. While JSON is not a replacement for the complex implementations of XML that many libraries rely on, it is an increasingly important data format due to the ubiquity of JavaScript on the web.

## Real-World Applications for Libraries

No matter how much or how little experience a person has, learning new skills, such as programming languages and programming libraries, represents a significant investment. However, it is also worth considering the value of gaining foundational skills that provide a platform for further development and growth, especially in the context of the extensive time many library staff spend mastering new vendor tools and interfaces. For staff who are constantly learning and adding new skills, it may make sense to strike a balance between skills with a shorter-term payoff, such as mastering the newest "cool tool," and more foundational skills that represent a longer-term investment.

Once staff start to develop these skills, it may be surprising how broad their applications are. Many libraries may find that they, or their parent institutions, need to create website features that allow users to interact with, and select from, a collection of data—whether that data represents new books or archived exams in the library, faculty or attorneys and their expertise, or the name and location of employers hiring students. Faster internet connections and powerful JavaScript libraries make it possible to build elements like these using primarily or entirely front-end, or client-side, tools. Rather than building a database back-end, the data may be stored in a Google Sheet and pulled onto the site using an open source tool designed for that purpose, such as Tabletop.js. Alternatively, tools exist to easily convert spreadsheet data to JSON, allowing for faster load times. Scripts using jQuery, or powerful jQuery plugins such as DataTables, can then manipulate the data and display it on the page. Lightweight methods like this are not necessarily appropriate for systems that work with large amounts of data or with data that cannot be made public, but they can be very useful for the types of web development problems that libraries are often expected to solve on their own, without the help of full-time web developers.

Additionally, a basic familiarity with JavaScript can prove useful in some surprising contexts. Libraries that use Google's office productivity products can expand the capabilities of spreadsheets and forms by adding scripts using Google Apps Script, a scripting language based on JavaScript. This can be useful for simple tasks such as creating a form that sends its results to an email list, or even more complex jobs like harvesting OAI-PMH data from an institutional repository into a spreadsheet. JavaScript libraries like jQuery can even be useful in a research or web archiving context, since it is possible, with some imagination, to use them to parse data from downloaded copies of unstructured or eccentrically structured webpages. That data might be useful for quantitative research based on the contents of a set of webpages, or as a source of automatically generated metadata for a web archiving project.

## Maintenance and Code Management

Library staff who write code may face questions from colleagues about how the tools and sites they create can be maintained over time. These questions may even take the form of categorical objections to these projects, on the basis that the project could not be maintained if the staff member who developed the project were to leave the library.

There are a variety of good practices that allow projects to be maintained and improved over time. Basic best practices, such as writing legible code

and including comments and other documentation to explain what the code does and how it works, are an important first step. Cross-training staff and creating opportunities for multiple staff members to develop skills with frequently used technologies decreases the risk that the library will be left without staff who can maintain existing projects.

Code management is another important safeguard. Many libraries, including some law libraries, maintain accounts on GitHub, a popular tool for sharing open source software. GitHub allows users to track the changes that are made to each project, and its workflow allows multiple staff members to work in parallel on the same project. New staff members can use an institution's GitHub account to find and understand the code a library has created, and even to see the history of a given project. (Learn more about GitHub at bit.ly/JF18GitHub.)

While practices like these should help to address specific concerns, it is also worth examining the context of these general objections. Staff development of technology skills may be an area where the notoriously risk-averse nature of libraries does real damage. If projects that respond to a pressing need are discouraged out of fear that future staff will lack the technical skills to maintain the project, the library misses an opportunity to develop a culture of technical competency, and the fear of inadequate staff skills is likely to become a self-fulfilling prophecy. On the other hand, creating a culture where staff are encouraged to develop and share the technical skills needed to do their jobs makes it easier to attract and retain staff who are willing and able to develop the required skills.

## Leveraging Existing Skills

If all of this seems daunting, it may be comforting to remember that even librarians with no coding experience at all may possess a wealth of complementary skills that provide a head start for many projects. Librarians who are comfortable working with

## JQUERY IN ACTION

This simple example shows a list of courses from Fusion, a federated search tool for prospective students, developed by Boston College Law Library staff. In this example, clicking on the name of the course once shows the course description; clicking again hides the description.



The jQuery code below assigns a function to the click event for the element containing the name of the course. That function toggles the class "open" for the element that was clicked, meaning "open" is removed if it is present, but added if it is not. The code then finds the description of that element and toggles it as well, using the slideToggle function to create a simple animation that slides the description up and down. These changes are highlighted in the HTML code below.

```
jQuery(".answer-tab").click( function() {
    jQuery(this).toggleClass("open").parent().find("td.description").slideToggle();
});
```

**Closed:**

```
<tr role="row" class="odd">
    <span class="answer-tab details-control">
        <td class="sorting_1">
            <a href="#">Administrative Law</a>
        </td>
    </span>
    <td class=" description" style="display: none;">This course will examine the legal framework for the
    work of administrative agencies. We will explore the sources of authority for agency action under
    the U.S. Constitution and will examine the accountability of agencies to the legislative and executive
    branches of government. This course is intended to introduce students to regulatory agencies in a
    variety of substantive fields of law, such as financial, environmental, healthcare, immigration, labor,
    to name a few.</td>
</tr>
```

**Open:**

```
<tr role="row" class="odd">
    <span class="answer-tab details-control open">
        <td class="sorting_1">
            <a href="#">Administrative Law</a>
        </td>
    </span>
    <td class=" description" style="display: block;">This course will examine the legal framework for
    the work of administrative agencies. We will explore the sources of authority for agency action under
    the U.S. Constitution and will examine the accountability of agencies to the legislative and executive
    branches of government. This course is intended to introduce students to regulatory agencies in a
    variety of substantive fields of law, such as financial, environmental, healthcare, immigration, labor,
    to name a few.</td>
</tr>
```

**You can see this code in action at bit.ly/JF18Code**.

metadata will find those skills useful for creating data models and formats required for many web development projects. Those who are accustomed to helping patrons use web resources may find this experience helpful when faced with basic interface design decisions. Identifying these existing skills may bring to light projects where a relatively small investment in new technical abilities can amplify the existing capabilities of library staff.

For most of us, it is neither realistic nor desirable to attempt to add the skills of full-time web developers to the already extensive competencies expected of professional librarians. Instead, it makes sense to focus on developing technological skills in areas that complement the existing skills and interests of library staff, and that address genuine needs for the library and its users. Narrowing the scope of what needs to be learned right away—moving away from "I want to learn JavaScript" and toward "I want to learn JavaScript to solve this specific problem"—provides a quick return for the library while also helping staff identify specific code and patterns that are helpful in their context.

## Libraries, Codes, and the Future

JavaScript, jQuery, and JSON have become essential building blocks of the web, and an understanding of what they are and how they are used may be helpful for achieving the high level of technical and information literacy to which librarians aspire. Additionally, they are mature technologies that are relatively accessible to novice coders due to the large amount of open source code, documentation, and advice available on the open web.

Beyond any specific technologies, law libraries may benefit from embracing a culture in which staff are encouraged to invest in foundational technological skills. While many law libraries embrace new technology, there can sometimes be more emphasis on learning consumer-level tools, in the form of new vendor products and features, and not enough emphasis on the type of fundamental skills that can be re-used project after project. While this article has focused on JavaScript and related foundational technologies of today's web, the same argument could be made for time spent mastering other powerful and possibly more accessible tools like spreadsheet formulas or regular expressions. Other scripting languages, such as Python, which is popular among librarians who work with metadata, may prove a better investment in a less web-oriented context.

The possible uses for these mature, stable technologies are everywhere, but it takes a willingness to invest staff time to find and identify these opportunities. ■

## CSS SELECTORS

In CSS, selectors indicate which elements on the page a given style applies to. In jQuery, CSS selectors can be used to further manipulate those items. CSS selectors are based on simple patterns, but the patterns can be combined in order to create more complex selectors. There are many types of CSS selectors; the following table shows four examples.

| TYPE OF SELECTOR | EXAMPLE | EXPLANATION |
|---|---|---|
| Element | h3 | Selects all <h3> elements (level three headers) on the page. |
| Class | .headline | Selects all elements with the class "headline." |
| ID | #footer | Selects the element with the ID "footer." |
| nth-child | :nth-child(1) | Selects any element which is the first child of its parent element. |

The CSS selector h3.headline:nth-child(1) would select all <h3> elements with the class "headline" that are the first child of their parent element. This is what that element might look like in a page:

```
<div class="headline-wrapper>
        <h3 class="headline>This is the Headline Text</h3>
</div>
```

**AALL2go EXTRAS**

Watch the 2015 AALL Annual Meeting program "So You Want to Be a Web Services Librarian?" at bit.ly/AM15Web.

Watch the 2015 AALL Annual Meeting program "Enough to Be Dangerous: 00000110 Things Every Librarian Needs to Know About Coding," at bit.ly/AM15Coding.

**NICK SZYDLOWSKI**
**DIGITAL INITIATIVES & SCHOLARLY COMMUNICATIONS LIBRARIAN**
Boston College Law Library
Newton, MA
nick.szydlowski@bc.edu