By mandate, the United States Census Bureau compiles and distributes data on the American population. Open data initiatives have made it possible for users to access and analyze data with simple web-based tools. A new method for requesting data from the Census Bureau is described here, along with two different mapping mashups. Using the technology described in this article, a simple web mapping interface could unlock vast amounts of available data for user exploration.

INTRODUCTION

The United States Census Bureau collects and maintains a large, widely-used collection of data on general demographic, social, housing, and economic characteristics. Recently, they made their data available through an Application Programming Interface (API) (US Census Bureau 2014). Their API is a mechanism to access Census data through a set of web technologies referred to as AJAX (Asynchronous JavaScript and XML), which facilitate the continuous sending and receiving of data between client and server environments (Powell 2008). The main advantage of using the API is that the Census data are available for use without the need to store them on a local computer: they remain in the cloud.

JavaScript is the main programming language used with APIs; most are completely written in JavaScript. Interpreted by the web browser, JavaScript is also the most widely used programming language for web development (Raasch 2013). As an interpreted computer language, there are no software packages that need to be installed, nor any special server-side setup. It is an efficient and simple solution for web-based application development and hosting.

The Esri JavaScript API (Esri 2014a) allows developers to use web services based on spatial data servers that implement their software. The Esri software packages and API are popular with government enterprise GIS systems. Data Driven Documents (D3) is another powerful set of JavaScript functions used to visualize large datasets. Both Esri and D3 provide JavaScript-based environments for creating mapping mashups, which are the pulling together of various online resources (Batty, et al. 2010).

Figure 1: Representation of AJAX and JavaScript consuming web resources and a mashup of the attribute and spatial data in the client.
Two mashup methods are demonstrated here for mapping Census data obtained from the Census Bureau API. In the first, the Esri JavaScript API is used to obtain a base-map from the US Census Bureau TIGERweb spatial data server (see Figure 1). In the second method, the base layer for D3 is provided in GeoJSON or TopoJSON format, which are compatible with JavaScript.

**BACKGROUND**

For mapping, most APIs are based on a technology called REST (Representational State Transfer) web services, which manipulate representations of web resources using a uniform set of stateless operations (Fielding 2000). One of the most popular map service architectures is the tiled web map service. This type of REST service involves a series of predefined static map tiles, produced for various scales, that populate the web map based on user interaction. There are additional REST mapping services, both raster and vector, that are the primary technology behind the plethora of Multi-Scale Panable maps available on the web (Peterson 2014). These mapping APIs provide the base layer of most map mashups.

There are alternatives for creating web map mashups. The D3 JavaScript Library can render a projected, SVG map element (base layer) that can be used to visualize spatial data (Cuesta 2013). D3 is designed for the creation of dynamic visualizations. In addition to mapping APIs, there are also APIs that revolve around accessing data. Data APIs focus on delivering specific queried data or streaming updated data. In a map mashup, these attribute (thematic) data sources, referred to as operational layers, are overlaid on top of the base layers.

**CENSUS DATA API**

As a product of the US Government, Census data have always been available free of charge. In the Internet era, there have been two main ways to access the hundreds of tables and thousands of spatial data elements derived from the Census. First, there are web-based query and extraction methods, the latest being the American Factfinder. Secondly, there is direct FTP access to the data that can then be processed locally within a relational database management system.

To promote access and innovation, the Census Bureau released an API in 2012 that would allow users to access data through properly formatted HTTP requests. This allows for AJAX methods to request and use data on-the-fly within a mashup environment. The Census API opens the door for alternative methods for developing mapping applications.

To make a data request, you must have a properly formatted HTTP string. This string begins with the Census API website, “api.census.gov/data.” Next, the dataset you are querying is specified (i.e., Census 1990, 2000, 2010 or ACS aggregate datasets). After this, you must provide your key, which is required to access the Census API and can be obtained by completing a short form at www.census.gov/developers/tos/key_request.html. Finally, the combination of variables and the spatial units are specified. Queries can be made directly in your web browser by pasting into your address bar the example requests listed in Example 1 (you will need to use your own key provided by the Census). A partial output of the first request is shown in Example 2: a two-dimensional array of data in the JSON (JavaScript Object Notation) format. The first row provides column names and subsequent rows contain the data values. The data in this array can then be mapped.

The Census API allows for up to 50 variables to be queried in a single request; a series of requests can unlock a vast amount of Census data for user mapping and analysis. In order to implement the Census API in a JavaScript mashup, the jQuery library is needed. jQuery is a free and widely adopted JavaScript library that has the built-in functions necessary for accomplishing common AJAX requests (Powell 2008).
Before the data can be mapped, they must first be properly formatted. The two-dimensional array has certain limitations and is more usable if it is reformatted as key-and-value paired objects. In other words, we need to restructure the data from an array of individual elements into an array of record-like objects. The function shown in Example 3 produces the output shown in Example 4.

In this format, the data can be accessed more efficiently for database operations within a coding environment. As a two-dimensional array, the data would have to be referenced as a numeric [row], [column] of table elements. After reformattting, data items can be referenced by name in a {key: value} pair and more easily mapped using JavaScript.

In order to map the Census data, we need to acquire a basemap layer. After this, we dynamically join the reformatted Census data to their corresponding spatial counterparts as attributes. The map symbology (such as a choropleth) is then made based on those joined Census attributes.

Mapping with the Esri JavaScript API

The Census’ TIGERweb spatial data platform is based on the Esri ArcGIS Server software, which serves Open Geospatial Consortium (OGC) compliant spatial data as web services. Since the data are OGC compliant, a mashup could be done by using the OpenLayers API, an open source JavaScript API used for web mapping and consuming spatial data services. However, since the TIGERweb services are natively using Esri software, it is simpler to use Esri’s freely-provided API.

To demonstrate the use of dual web services, the example here has two parts. First, the page is loaded with spatial
function getCensusData(){
    // jQuery AJAX function getting data from the Census API
    // and call a return function processing the Census JSON object in ‘data’ variable
    $.getJSON("http://api.census.gov/data/2010/sf1?key=[user key]&get=P0010001,P0050010,NAME&for=state:*", function(data){
        var keys = data[0]; // extract the first row of the returned 2d array that are the column headers
        var values = data; // copy the array
        values.splice(0,1); // delete the first row of headers in the copied array
        arrayCensus = [];// create a new array to store the formatted object outputs
        // nested loops combining the column header with appropriate values as {key:value} pair objects
        for(var i = 0 ; i < values.length; i++){
            var obj = {};
            for(var j = 0 ; j < values[i].length; j++){
                obj[keys[j]] = values[i][j];
            }
            arrayCensus.push(obj);
        }
    });
}

Example 3: Sample function that requests data using the Census API and then reformats the results into a usable array of objects that can be mapped.

[ { "P0010001" : "4779736" , "NAME" : "Alabama" , "state" : "01" },
  { "P0010001" : "710231" , "NAME" : "Alaska" , "state" : "02" },
  { "P0010001" : "6392017" , "NAME" : "Arizona" , "state" : "04" },
  { "P0010001" : "2915918" , "NAME" : "Arkansas" , "state" : "05" },
  { "P0010001" : "37253956" , "NAME" : "California" , "state" : "06" }, ...
]

Example 4: The reformatted Census API request, now an array of objects and ready to be used as a mashup with either the Esri API or D3.

data—in this case, data for the continental 48 states—derived from the TIGERweb web mapping services that is being consumed locally. In Figure 2, a feature has been selected to show the default popup window included with the Esri API. Second, there is a button marked “Map Census Data” in the top left corner of the page. Clicking it invokes the Census Data API AJAX request; when this button is pressed, the Census data are requested, reformatted, spatially joined, and symbolized. The resultant map is shown in Figure 3.

Example 5 shows the combination of Census and Esri code needed. Notice in Figure 3 that there is an additional
Figure 3: Map showing Total Hispanic Population (calculated CenData attribute) after Census data request, reformat, join, and symbolization.

Example 5: A mashup using both the Census and Esri JavaScript APIs to map census data. Continued next page.
attribute at the bottom named CenData in the popup window. The feature symbology is a gradient between two colors assigned to the minimum and maximum CenData attribute values (outlined in the code).

**MAPPING WITH THE D3 JAVASCRIPT LIBRARY**

The D3/Census Bureau mashup is essentially the same as above, with two exceptions: the D3 JavaScript library requires data defined in either the GeoJSON or TopoJSON formats, and D3 has its own AJAX functionality— the jQuery library is not needed. Example 6 outlines the process to render the map in Figure 4. To demonstrate dynamic field calculation with Census data, percent values are calculated on-the-fly.

D3 has more robust visualization capabilities because of its unique data handling functions. For example, the quantize classification utilized in Example 6 removes outliers in the data range that could skew the color values of the map symbology. Furthermore, we are using ColorBrewer (www.colorbrewer.org)'s 9-class “Reds” scheme, which is integrated into D3.
Figure 4: A mashup map produced with the Census Bureau API and the D3 JavaScript library displaying percent-minority population.

Example 6: Code to combine Census API with the D3 JavaScript library. Continued next page.
A variety of APIs and JavaScript can be used to access Census data for on-the-fly mapping. With some additional work, an interface that allows the user to specify Census attributes, color scheme, classification, and spatial units (i.e., state, county, tract, etc.) could be implemented. This would unlock the potential of the Census data for spatial analysis.

Furthermore, the capabilities of D3 could be incorporated into the Esri JavaScript API to enhance how the data is visualized. There are current examples of D3 being used with Esri API on the Esri developers website (Esri 2014b). Using the technology described in this article, a simple web mapping interface could permit users to more easily explore the vast amount of data available from the Census.

REFERENCES


