A Linked GeoData Map for Enabling Information Access

Open-File Report 2017–1150

U.S. Department of the Interior
U.S. Geological Survey

A Linked GeoData Map for Enabling Information Access

By Logan J. Powell and Dalia E. Varanka

Open-File Report 2017–1150

U.S. Department of the Interior
U.S. Geological Survey

U.S. Department of the Interior
RYAN K. ZINKE, Secretary

U.S. Geological Survey
William H. Werkheiser, Deputy Director
exercising the authority of the Director

U.S. Geological Survey, Reston, Virginia: 2018

Suggested citation:
Powell, L.J., and Varanka, D.E., 2018, A linked GeoData map for enabling information access: U.S. Geological Survey Open–File Report 2017–1150, 6 p, https://doi.org/10.3133/ofr20171150.

ISSN 2331-1258 (online)

Contents

Figures

Abbreviations

D3	Data Driven Documents
GeoJSON	geographic-oriented JSON
GNIS	Geographic Names Information System
GNIS–LD	Linked GeoData version of the GNIS
GSW	Geospatial Semantic Web
HTML	Hyper Text Markup Language
JSON	JavaScript Object Notation
JSON–LD	JSON format for linked data
MKB	map knowledge base
RDF	Resource Description Framework
SPARQL	SPARQL Protocol and RDF Query Language
SVG	Scalable Vector Graphics
Turtle	Terse RDF Triple Language
URI	Uniform Resource Identifier

A Linked GeoData Map for Enabling Information Access

By Logan J. Powell[1] and Dalia E. Varanka[2]

Overview

The Geospatial Semantic Web (GSW) is an emerging technology that uses the Internet for more effective knowledge engineering and information extraction. Among the aims of the GSW are to structure the semantic specifications of data to reduce ambiguity and to link those data more efficiently. The data are stored as triples, the basic data unit in graph databases, which are similar to the vector data model of geographic information systems (GIS); that is, a node-edge-node model that forms a graph of semantically related information. The GSW is supported by emerging technologies such as linked geospatial data, described below, that enable it to store and manage geographical data that require new cartographic methods for visualization. This report describes a map that can interact with linked geospatial data using a simulation of a data query approach called the browsable graph to find information that is semantically related to a subject of interest, visualized using the Data Driven Documents (D3) library. Such a semantically enabled map functions as a map knowledge base (MKB) (Varanka and Usery, 2017).

A MKB differs from a database in an important way. The central element of a triple, alternatively called the edge or property, is composed of a logic formalization that structures the relation between the first and third parts, the nodes or objects. Node-edge-node represents the graphic form of the triple, and the subject-property-object terms represent the data structure. Object classes connect to build a federated graph, similar to a network in visual form. Because the triple property is a logical statement (a predicate), the data graph represents logical propositions or assertions accepted to be true about the subject matter. These logical formalizations can be manipulated to calculate new triples, representing inferred logical assertions, from the existing data.

To demonstrate a MKB system, a technical proof-of-concept is developed that uses geographically attributed Resource Description Framework (RDF) serializations of linked data for mapping. The proof-of-concept focuses on accessing triple data from visual elements of a geographic map as the interface to the MKB. The map interface is embedded with other essential functions such as SPARQL Protocol and RDF Query Language (SPARQL) data query endpoint services and reasoning capabilities of Apache Marmotta (Apache Software Foundation, 2017). An RDF database of the Geographic Names Information System (GNIS), which contains official names of domestic feature in the United States, was linked to a county data layer from The National Map of the U.S. Geological Survey. The county data are part of a broader Government Units theme offered to the public as Esri shapefiles. The shapefile used to draw the map itself was converted to a geographic-oriented JavaScript Object Notation (JSON) (GeoJSON) format and linked through various properties with a linked geodata version of the GNIS database called "GNIS–LD" (Butler and others, 2016; B. Regalia and others, University of California-Santa Barbara, written commun., 2017). The GNIS–LD files originated in Terse RDF Triple Language (Turtle) format but were converted to a JSON format specialized in linked data, "JSON–LD" (Beckett and Berners-Lee, 2011; Sorny and others, 2014). The GNIS–LD database is composed of roughly three predominant triple data graphs: Features, Names, and History. The graphs include a set of namespace prefixes used by each of the attributes. Predefining the prefixes made the conversion to the JSON–LD format simple to complete because Turtle and JSON–LD are variant specifications of the basic RDF concept.

To convert a shapefile into GeoJSON format to capture the geospatial coordinate geometry objects, an online converter, Mapshaper, was used (Bloch, 2013). To convert the Turtle files, a custom converter written in Java reconstructs the files by parsing each grouping of attributes belonging to one subject and pasting the data into a new file that follows the syntax of JSON–LD. Additionally, the Features file contained its own set of geometries, which was exported into a separate JSON–LD file along with its elevation value to form a fourth file, named "features-geo.json." Extracted data from external files can be represented in HyperText Markup Language (HTML) path objects. The goal was to import multiple JSON–LD files using this approach.

[1]A contractor: Rolla, Missouri. Work done under contract to the U.S. Geological Survey.

[2]U.S. Geological Survey.

Linking Data for Mapping

To link the data contained in the JSON–LD files as a way to provide a MKB, an HTML file uses the D3 library paired with code written in JavaScript. In this example, the program accesses the D3's geo functions to create a frame of reference centered upon Missouri, a randomly chosen case study, in an Albers Equal-Area Conic projection (Bostock, 2015). This is done by combining the `.path()` and `.projection()` functions with a heavily enlarged scale, allowing the code to correctly plot any received coordinates in a Scalable Vector Graphics (SVG) window. After setting the SVG window as a canvas for the map, the program opens the GeoJSON file that originated from The National Map and begins creating the counties. This is done by assigning the data for each county to their own path object and then plotting each of the coordinates embedded in the data as shown below:

```
svg.selectAll(“path”)
 .data(json.features)
 .enter()
 .append(“path”)
 .attr(“d”, path)
 .style(“stroke-width”, .5)
 .style(“stroke”,“rgb(50,50,50)”)
 .style(“stroke-opacity”, .6)
```

The stylistic elements of the data are applied using Cascading Style Sheets (Atkins and others, 2017), some of which is included in the above code (fig. 1). As part of the visualization, the “STATE_NAME” value from The National Map for each dataset differentiates counties by State, with each State distinguished by differing colors. To add clarification on interaction with the map, some text is added below the SVG window to provide an explanation on how to use it.

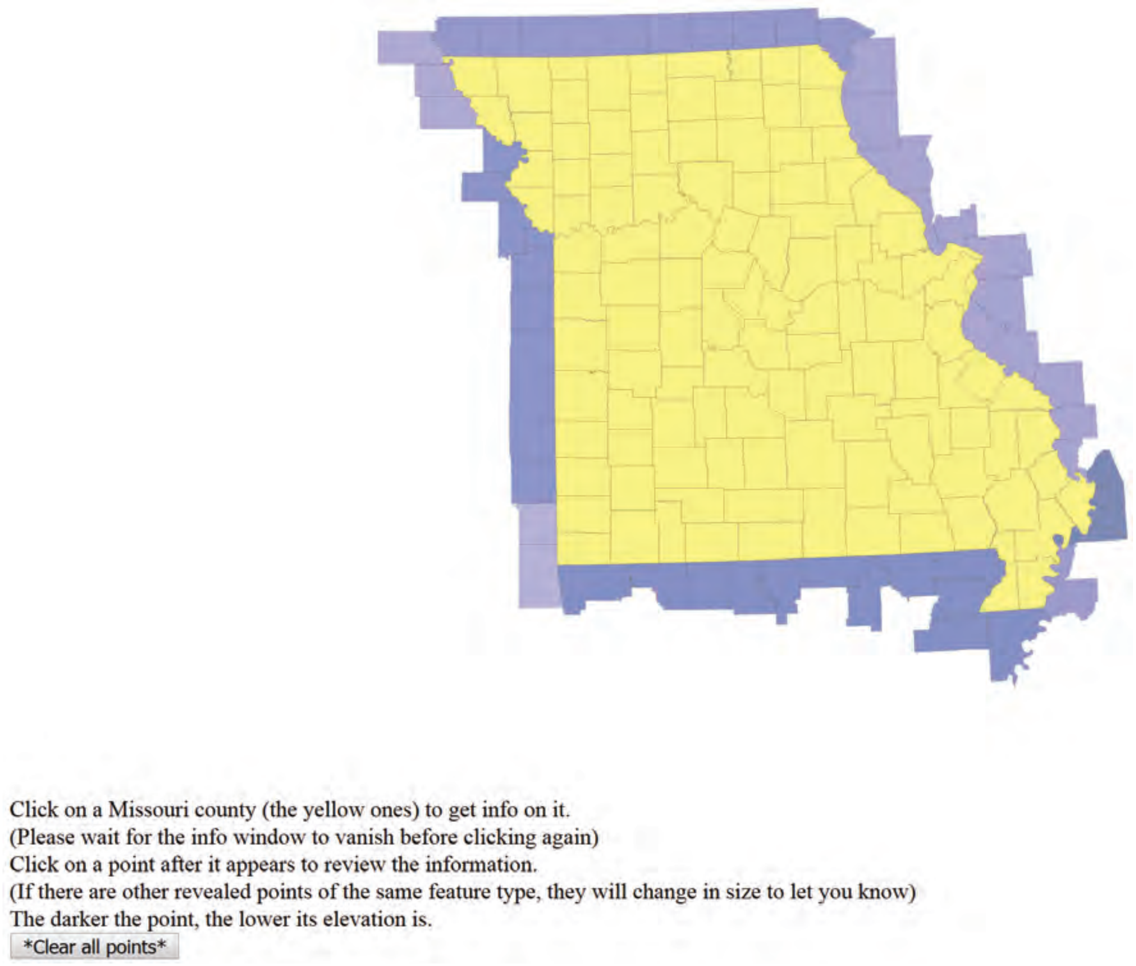

Click on a Missouri county (the yellow ones) to get info on it.
(Please wait for the info window to vanish before clicking again)
Click on a point after it appears to review the information.
(If there are other revealed points of the same feature type, they will change in size to let you know)
The darker the point, the lower its elevation is.

Figure 1. Hyper Text Markup Language (HTML) base map for geospatial triple data queries with no current interactions.

Once the map is created, JavaScript's event listeners, functions that wait for a certain user-initiated event to happen before running, are used to run a variety of custom functions triggered by the cursor hovering or clicking on the MKB; for example, a county's outline is highlighted and enlarged when the cursor hovers over it due to the `.mouseover()` listener. Upon moving off of the county, the outline reverts back to its original appearance by means of the `.mouseout()` listener. Stylistic changes like this are improved by D3's `.transition()` function, which provides a smooth change between the object's initial and later appearance. The `.onclick()` function takes the selected county and begins accessing the other JSON–LD files to link an arbitrary set of data relevant to it. The function does this by parsing one of the .ttl file triple resources for enough information to successfully link with the mapped GeoJSON.

This application program (app) executes a browseable graph query approach, which is different than a query started by SPARQL request. The process is started by generating a random row number within the range of the "features.json" file. Using this index as a way to refer back to the arbitrary "features.json" element, the program compares the element's county value with that of the currently selected county. If the county value matches the currently selected county, the program cycles through all the elements in the "history.json" and "names.json" files to find any matching data. Once all counterparts have been found, the JavaScript sends the collected values of that instance to a window that displays the collected data organized as a point object array. If the county values do not match the currently selected county, the program generates a new random number and tries again; however, because the JSON–LD files were filtered to only contain data for a single State (in this case, Missouri), the program first checks to see if the selected county is in that State.

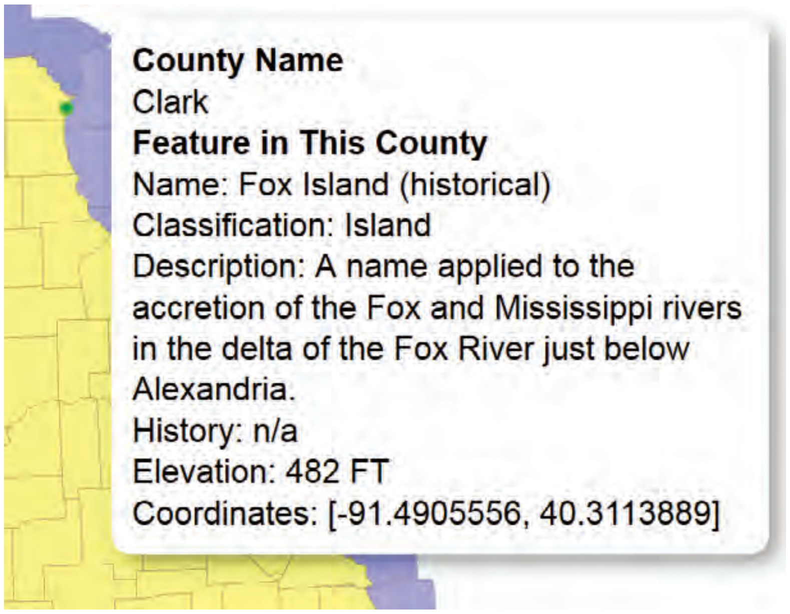

County Name
Clark
Feature in This County
Name: Fox Island (historical)
Classification: Island
Description: A name applied to the accretion of the Fox and Mississippi rivers in the delta of the Fox River just below Alexandria.
History: n/a
Elevation: 482 FT
Coordinates: [-91.4905556, 40.3113889]

Figure 2. The `.onclick()` function for counties. Here the user has clicked on Knox County, as shown with the red outline and named in the data window. Also visible here is a small green point inside the county, depicting the location of the shown feature.

All files are easily linkable because each contains the same value for their primary key with the exception of the "features-geo.json" file. In searching for the data for the arbitrary feature, the program stores certain values from each JSON–LD file in variables for sending to the data display window and combining into a point-coordinate object if applicable. These include the following:

- County name, from the "features.json" file
- Name of the feature, from the "names.json" file
- Classification of the feature, from the "features.json" file
- Description of the feature, from the "history.json" file
- History of the feature, from the "history.json" file

The program then checks to see if this feature has a coordinate pair to fetch. If there is not a pair to fetch, the information-gathering code ends; however, if the "features.json" file property "hasGeometry" is not null, the program sends a message to the data window to indicate that the coordinates and elevation values are still loading because both still need to be retrieved.

Cycling through the "features-geo.json" file as done with the previous loops, the program retrieves the elevation information, made up of the numeric value and units, and coordinates of the matching feature to store and display. Once the Linked Data Map program has gathered all the data for this feature and confirmed that there are corresponding coordinates, the program combines all the data into a point object. All point objects are stored in an array that is then appended to a circle object, which is plotted at the coordinates recently retrieved; however, if this point already exists in the currently plotted points, the "new" point is neither replotted nor added to the dataset array. Once completed, the county's `.onclick()` function is finished and ready to be repeated for the next time a county is clicked.

All the data that relate to a plotted feature are stored in the circle object itself. This enables an efficient recall of collected information relevant to a particular point through a new `.onclick()` function triggered by clicking on a plotted point. The new `.onclick()` behaves similar to how the county `.onclick()` function was able to pull the name of the selected county from the county path object itself; the new `.onclick()` function for points simply extracts all the data that were contained inside the selected point object that was used as the dataset for its production. This information is then sent directly to the data display window, with a slight modification to the window's title to convey that these data relate to an already plotted feature; furthermore, this allows for another data-linking opportunity (for example, when a point is selected, all other found points of the same feature type grow in size to indicate that they are similar to the selected structure).

Graphic Presentation

Managing the visual and triple data was challenging for several reasons. An interesting limitation of D3 is that only one set of `.transition()` functions can be active at a time. This does not restrict any of the data linking, but it does interfere with data presentation. One way this manifested is in the process of making the points appear on the map. Originally, the `.transition()` that was used made dots appear in red with an opacity of zero, effectively turning them invisible. The dots would then shrink to an appropriate size as well as change color to a shade of green dependent on the elevation value while becoming visible with the use of another `.transition()`. This had a conflict with the `.transition()` implemented in the `.mouseover()` function for the points. The new `.transition()` causes the point that the cursor hovers over to increase in size and generates a border around it, abruptly ending any other `.transition()` that was originally running, preventing any color change. A small glitch occurs if a point, as it is plotted on the map, is close enough to the mouse that it occludes the cursor from the county. This causes the program to consider the point as being hovered over, forcing the code to start a new `.transition()`, interrupting the old `.transition()` going from red to green. The result is to have a red dot suddenly among all the other points with no way of being fixed other than removing all points from the dataset. A similar situation can occur if, in the point removal process, a user quickly tries to trigger the `.mouseover()` function for a point as it begins to fade away. This will cause the point to stay on the map with an unusual color because it triggers a `.transition()`. Additionally, if for example three points are not removed because of this interruption, the next three points that would have been generated instead have their data appended to the existing points rather than generating the appropriate points for the three new features. The first of these two issues is solved by having the circle objects transition into the map starting as their respective shade of green. The second of these, however, is solved by requiring the user to reactivate the removal process to clear the map of all old points.

Another minor issue arises from the nature of the hover or `.mouseover()` and `.mouseout()` functions. The data display window was originally written to vanish immediately on the `.mouseout()` function. The problem that arises is that when a point is generated, there is a chance it will appear under the cursor as previously mentioned. The program interprets this as having moved away from the recently selected county, prompting the `.mouseout()` function for counties, which causes the window to vanish only moments after it appeared. The same thing can happen if the user accidentally moves off the county before viewing the data. To address this issue, a function that gives a one second delay before the window disappears has been implemented. This delay fixes the accidental problem because the user has a chance to return the cursor back to the selected county and provides a moment to view the information before it goes away. The user will often need to move the mouse around on the county for the program to recognize that the cursor has returned to the selected county, which will cause the window to reappear. Another work-around to this problem is that the stored information in the points provides a second chance to view the data; however, this means that the user must wait for the window to hide itself before selecting a new county or point, otherwise, the window's delay will cause it to disappear before the new data are presented.

One other challenge was with the original JSON format, particularly with getting the data in a JSON format to be compatible with the triplestore software Marmotta and the D3 library. The original versions of the JSON files worked well with D3 but were not compatible with Marmotta. The first version of a replacement JSON form, RDF+JSON, was proposed as an alternative because Marmotta can recognize and use it without problem. Unfortunately, it did not work well with D3 and written code at first because the data are identified by their Uniform Resource Identifier (URI), which all have "//", making the rest of the line a comment. Also, the URIs include dots that would easily confuse the program when accessing the data through dot notation, the method originally used in the code. The solution was to use bracket notation when accessing attributes. This means rather than using a dot to separate each part, such as

```
    json.hist[i].properties.http://data.usgs.gov/lod/gnis/ontology/description.
value
```

the code uses brackets, as in

```
    json['hist']['i']['properties']['http://data.usgs.gov/lod/gnis/ontology/
description']['value']
```

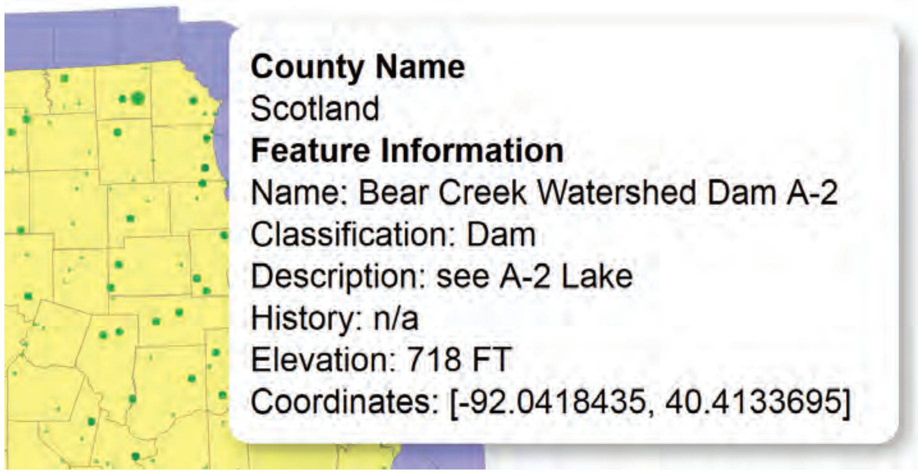
Using bracket notation not only resolved the URI problem, but also resolved the primary problem JSON–LD had with its attributes having the "@" sign, such as "@id" and "@type." With the code now using bracket notation, JSON–LD was revisited as a usable and preferable JSON structure.

It is interesting to note that the RDF+JSON format, which was temporarily implemented, used the identifier for each instance as the label, meaning there was no way to arbitrarily call upon an instance. This would not be problematic outside of the proof-of-concept because a proper query is looking for more precise results, but it did make this program difficult to write. Fortunately, the switch to JSON–LD cleared up this temporary setback because the dataset is contained in an array called "@graph," and each instance is identifiable by calling upon "@id" rather than by the unique identifier itself. Using bracket notation, the code can successfully navigate through the contents of the Marmotta compatible JSON–LD files, as shown below with the "history.json" file:

```
    json['@graph'][i]['@id'] json['@graph'][i]['http://data.usgs.gov/lod/gnis/
ontology/description']['@value'] json['@graph'][i]['http://data.usgs.gov/lod/
gnis/ontology/history']['@value']
```

Conclusions

The outcome of the application successfully shows that linked data are viable for mapping, depicted by screen shots of the map results shown in figure 3. By modifying the point sizes to correspond to Linked GeoData properties, the figure introduces the possibility that visual design elements can be developed that use and align with semantic technology, such as linked data, for semi-automated transfer of information and meaning.



County Name
Scotland
Feature Information
Name: Bear Creek Watershed Dam A-2
Classification: Dam
Description: see A-2 Lake
History: n/a
Elevation: 718 FT
Coordinates: [-92.0418435, 40.4133695]

Figure 3. The map after discovering several points. The `.onclick()` function for points is shown here; the selected point is enlarged and outlined in bright green, while all matching features, in this case all cemeteries, are also slightly larger than the other points. Also notable is the varying colors of the points, which is dependent on their respective elevations.

This demonstration of linking data to a visual map advances geographical information science towards more robust user-data interactions with geospatial applications of semantic technology. Among the advantages that these interactions will offer are that features can have any number of properties, so more precise semantics are available to users with specific queries in mind. Also, the categorical nature of the RDF data model allows metadata to be embedded with data instances, while still leveraging the power of visualization.

References Cited

Apache Software Foundation, 2017, Apache Marmotta: The Apache Software Foundation software, accessed August 29, 2017, at https://marmotta.apache.org/.

Atkins, Tab, Jr., Etemad, E.J., and Rivoal, Florian, 2017, CSS snapshopt 2017—W3C working group note, 31 January 2017: World Wide Web Consortium (W3C) web page, accessed December 15, 2017, at http://www.w3.org/TR/css-2017/.

Beckett, David, and Berners-Lee, Tim, 2011, Turtle—Terse RDF triple language: World Wide Web Consortium (W3C) web page, accessed August 29, 2017, at http://www.w3.org/TeamSubmission/turtle/.

Bloch, Matthew, 2013, Mapshaper: Mozilla, accessed August 29, 2017, at http://mapshaper.org/.

Bostock, Mike, 2015, D3, Data-driven documents: JavaScript library, accessed August 29, 2017, at https://d3js.org/.

Butler, H., Daly, M., Doyle, A., Gillies, S., Hagen, S., and Schaub, T., 2016, The GeoJSON format: Internet Engineering Task Force web page, accessed August 29, 2017, at https://tools.ietf.org/html/rfc7946.

Sorny, Manu; Longley, Dave; Kellogg, Gregg; Lanthaler, Markus; and Lindstöm, Niklas, 2014, JSON–LD 1.0—A JSON-based serialization for linked data: World Wide Web Consortium (W3C) web page, accessed October 16, 2017, at http://www.w3.org/TR/json-ld/.

For more information about this publication, contact
Director, USGS National Geospatial Technical Operations Center
1400 Independence Road
Rolla, MO 65401
(573) 308–3500

For additional information visit https://ngtoc.usgs.gov

Publishing support provided by the
Rolla Publishing Service Center

Powell and Varanka—A Linked GeoData Map for Enabling Information Access—Open-File Report 2017–1150

ISSN 2331-1258 (online)
https://doi.org/10.3133/ofr20171150