DESIGNING WEB BASED MAPS

By

WILLIAM J. BEAVER

MASTER OF SCIENCE
GEOGRAPHIC INFORMATION SYSTEMS TECHNOLOGY
FINAL PROJECT

THE UNIVERSITY OF ARIZONA

2014

To Randy Sue for putting up with me.

ACKNOWLEDGMENTS

I would like to thank Dr. Larry Stevens and Jeri Ledbetter at Springs Stewardship Institute for allowing me to do this project. I would also like to thank all the people in the MSGIST program for their knowledge and help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AMD	Asynchronous Module Definition
API	Application Programming Interface
CZML	Cesium Markup Language
DLCC	Desert Landscape Conservation Cooperative
DOM	Domain Object Model
Exif	Exchangeable Image File Formats
GIS	Geographical Information Science
GPS	Global Positioning Systems
GPX	GPS Exchange Format
IT	Information Technology
JSON	JavaScript Object Notation
KML	Keyhole Markup Language
MCV	Model Controller View
OOP	Object Oriented Programming
REST	Representational State Transfer
SQL	Structured Query Language
SSI	Springs Stewardship Institute
SVG	Scalable Vector Graphics
UTC	Coordinated Universal Time
XML	Extensible Markup Language

ABSTRACT

Application design is a broad concept that uses design at three levels: layout and style; user interface and experience; and code design patterns. In this paper the author describes these design concepts with respect to web applications, in particular web mapping applications. Web mapping design concepts are used to develop the application called SeepApp for the Springs Stewardship Institute. SeepApp allows a user to verify existing springs and to add newly found springs to a map. The application includes the ability to annotate photos and video and to upload spatial tracks associated with the springs. The application reads jpeg image metadata (Exif data) from the photos to capture UTC (Coordinated Universal Time) date/time, GPS coordinates, and orientation of the camera. The photos are tiled and a drawing program allows for markup of large resolution photos. The project accomplishes the following tasks: development of a working web application; a template which others can use; and a body of working knowledge which can be applied to future projects. All source code for this project is available at https://github.com/wjbeaver/Hide-Seep. The application currently resides at http://overtexplorations.com/seepApp.html.

INTRODUCTION

Developing web/mobile applications for GIS is much the same as any IT project. There are, however, important differences that are coded into the statement: “Spatial is Special” (Anselin 1989, 1-9, Haining 2009, 1-18). This “specialness” encompasses data structures, statistical analysis, computation, and human cognition; the way we think and move in space. With these differences in mind, the scope of this project is to describe the workflow for the development of a GIS web/mobile application and to use this workflow to create the application. Application development will be guided by three organizing principles; design patterns; information design; and spatial cognition.

Design Patterns

During the 1990s several important thinkers came to prominence with their speculations about how computers and humans would interact. These people and their ideas have had a tremendous, often hidden impact on programmers, programming paradigms and the present status of computation in our society. One of these thinkers is Brenda Laurel, who has worked in interactive media since 1976 at Atari, Apple, and Sun in addition to several companies she founded. She is currently an adjunct professor of Computer Science at U. C. Santa Cruz (Laurel 2014). One of her most influential books is *Computers as Theatre* (Laurel 1995).

In *Computers as Theatre*, Laurel uses the simile of a theatrical production to describe the components of human interaction with a computer. In doing this she references an original source, Aristotle's *Poetics* (Aristotle 1954). In particular, she analyses the four causes: Formal cause; Material cause; Efficient cause; and End cause. She translates these ideas into four causes of human-computer usage:

- Functionality or actions: Programs differ from a script in that programs are intrinsically non-linear. Humans interact with programs in ways which result in a collaboration between humans and the program.
- Application: What actions the program carries out.
- Representation: Actions are represented both internally by the code and externally by what the user sees and does on the screen.
- Agency: “a bundle of functionality that performs some task for a person, either in real time or asynchronously” (Laurel 1995, 46)

Laurel spends a great deal of the book talking about agency and the idea of the holodeck, particularly how it was used in the TV show Star Trek: Voyager (Roddenberry 1995). Advances in agency have taken a bad rap recently with hidden and often unwelcome tasks acting on a person rather than for a person. Much of the underlying structure of agency is in place with wireless, web services, and cloud computing, and just the action of interacting with a map on the web is an example of agency that has become commonplace. Routing is another excellent example of agency, especially when the query is voice activated and the routing directions are spoken back in real time. As for the holodeck, the underlying structure is just starting to appear in the form of game and 3D graphics on the web.

In software development, the idea of design patterns (Gamma et al. 1994) encompasses much of what Laurel talks about. First defined by the architect Christopher Alexander in *A Pattern Language* (Alexander, Ishikawa, and Silverstein 1977), these patterns are reusable solutions to problems of Application, Action, Representation, and Agency.

Design patterns are usually associated with Object-Oriented Programming (OOP) and design patterns are usually built on top of the OOP paradigm. One popular global design pattern is the Model/View/Controller pattern (Chauhan 2014) which is used by Apple products (Apple Computer 2014) and variations of this pattern are also used in Android products (Sokolova, Lemercier, and Garcia 2013). The model/view/controller pattern separates the data and all the software associated with the data which is called the model from the view, which is the visualization of the model. A user manipulates the view which interacts with the model through a controller.

Two other programming paradigms have importance for GIS. These are data-driven programming and functional programming. These two paradigms don't replace OOP, but complement it, providing insights into problems where OOP has difficulties.

In data-driven programming the model is the data with part of the controller being a parser of the data flow. This idea is important in game development (White 2013). In Web GIS, service requests for base maps and layers result in a JSON or XML string that is turned into a working 2D map. Google Earth uses KML as a data-driven request to map onto the surface of a globe (Schubotz and Harth 2012). Cesium (AnalyticalGraphicsInc 2014), a web-based open source global mapping API, uses a JSON format called CZML (AnalyticalGraphicsInc 2013) which is intrinsically temporal and 3D and which can be streamed like audio or video.

Functional programming is based on the idea that functions should only pass other functions as parameters and that data is represented by the state of the function at completion. This paradigm is naturally recursive and one that has support in abstract mathematics (Hughes 1990). Functional programming is the basis of languages like Scala (Odersky 2014) but functional elements are available in languages important to GIS such as Python, R, and JavaScript. Functional programming is useful in workflow problems and is one reason for the popularity of Python in the sciences. Of course, Python is basic to workflow programming in GIS.

Mapping Patterns

The following patterns are *ad hoc* (Peterson 2011) and are not specific to any type of programming solution.

Analysis

In a way, reading and understanding a good map is a form of analysis. Basic interactions like panning, zooming, changing layers and base maps, rotation, etc. are forms of analysis.

Exploratory Data Analysis

The user can explore a map based on queries of attributes, topologies or raster algebra. Presentation of results is very important.

Model Driven Analysis

Much like exploratory data analysis, except the user is presented with a theoretical model and requests information based on this model. An example of this is routing. Again, presentation of results is important.

Exploratory Learning

A learner is presented with an interactive map and uses it to learn some feature of GIS or about a subject that uses GIS. This map may or may not be embedded into some course management system like D2L or Moodle. An example of this is in GISTutor (GISTutor 2014a). PostgreSQL is an enterprise database and

PostGIS is it's GIS extension. The tutorial presents a map with some data and allows the user to change the display based on queries to the database. The purpose is to learn how to write SQL queries to a PostGIS geodatabase (GISTutor 2014b).

Modification

The ability of a user to add, delete, or edit geospatial locations, boundaries and attributes of an existing layer. These actions are embedded within what is called a transaction, the layer itself doesn't change until the user and the program both approve of the changes.

Annotation

A user can add content to a map through specific annotation layers. This can include:

- A specific feature with attributes
- Saved results of an analysis
- Commentary
- Photographs
- Video

Photos can be geocoded, the date and time encoded in UTC format, and markup in the form of comments and SVG graphics can be added. Video can also be geocoded, time coded, and related to a line track. In addition, markup such as captioning and drawings can be added.

OOP Design Patterns in JavaScript

Module Pattern

In modern JavaScript all useful code is written in conjunction with an API, a code library, or a coding framework. An API is usually an interface to a set of web services while a library is a set of functions or classes the separates the code from the underlying DOM. As different browsers have differently structures DOMs, the written code doesn't have to worry about this, it can interact with the DOM through the library. This is the essence of modularity, the DOM can change depending upon the browser used and hopefully the code will not break. Some JavaScript libraries have been extended with OOP design pattern functionality and class modules that can extend the DOM. These extended libraries are considered frameworks. To extend the modular concept further it would be nice to be able to swap out APIs, libraries or frameworks without having to change the calling code. Modular code has little or no global variables and modules at the highest level can be added and removed whenever needed.

Modularity is something that can be strived for but is hard to achieve. Different GIS and other APIs use different underlying libraries or frameworks and it can be impossible to separate them. The ESRI JavaScript API is built out of Dojo, it is an extension of the framework and it is impossible to write code using this API without using Dojo. Dojo uses the Asynchronous Module Definition (AMD), a module pattern for wrapping namespaces of called modules within blocks of Dojo code. This form of modularity allows for other libraries to be called but this form of modularity is *within* the Dojo framework (Voytyshyn 2014). OpenLayers more closely adheres to the module design pattern allowing the use of any underlying library and specific bindings to the Dojo framework (Asantiago 2012).

Model Controller View Pattern

In JavaScript (Osmoni 2012), the view is the underlying browser DOM. The DOM is loaded using HTML and modified using a DOM library or framework. This modification of the DOM is the controller. In Apple devices the MCV pattern is strictly enforced but has not been observed in JavaScript mapping. In the ESRI JavaScript API much of the model (feature classes and their attributes) is bound to the map through Dojo widgets. This helps in coding quick and dirty maps and can be worked around.

Transaction Pattern

A MCV pattern allows for an easy transaction pattern between the model and a set of controllers. The controllers are given a temporary subset of the model which they pass back when the transaction successfully occurs. At this point the model is changed. In JavaScript maps transaction patterns are the correct way to work with attributes and annotations.

Promise Deferred Pattern

An application's map can have multiple feature classes coming from one or more different servers. Querying each feature requires one asynchronous call to the web service. Asynchronous calls use callbacks, functions that are called when the server responds. For a single asynchronous call this is fine, but what if the program wanted to handle the results only after multiple queries complete? There is no guarantee of what order the calls will be returned. The promise deferred pattern allows chaining of callback functions, a promise to handle the queries is deferred until all the queries return. This very useful pattern in built into Dojo (Forbes 2013).

Design patterns are a great way to conceptualize a project. Whether for general organization or using specific abstract design patterns related to a specific programming concept or language, design patterns are important to the development process (Gordillo et al. 1999). If the pattern is hard-coded into the software like the MVC pattern in Apple products or the promise deferred pattern in Dojo then the pattern can be particularly useful. If not coded in, the pattern becomes more of a template and a suggestion and a pure pattern is not always possible.

Information Design

A major problem in computer interface design has been the low resolution of computer screens compared to paper. Resolution in both mediums can be measured in dpi, dots per inch. For computers this is often called ppi or pixels per inch. A quality printed product has a dpi of around 1200 dpi. For years, the resolution of a computer screen was 72 dpi, a potential information content of only 1/17th of a printed page. Today, most displays are in the 150 ppi range with some topping 300 ppi, the resolution of most copy machines (Wikipedia 2014). Unfortunately, many of the higher resolution displays are for tiny screens. This has led to a lessening of information content on viewing screens. An early critic of this tendency has been Edward Tufte. One of his several visually stunning books is *Envisioning Information* (Tufte 1990).

In *Envisioning Information*, Tufte looks at the problem of presenting information in a visual form. He illustrates both successful and unsuccessful attempts throughout history and shows ways to visualize a great deal of data in a format that can be grasped without loss of complexity or the addition of clutter. Tufte sees a document in a holistic manner, as more than the sum of its parts. The necessity of specialized computer programs has led to a fracturing of documents into their components with the loss of how these components are placed. Web design has been struggling with this problem with mixed results. Tufte sees display of data as intrinsically multi-variant, single variable relationships can have little meaning and can show a false causality. The world, being complex, requires visualization of complex data.

Tufte has an excellent critique and a good starting and ending place for looking at any application. Other important sources for information design are: Stephen Few (Few 2009) who has written extensively about design in business graphics, Yuri Engelhardt (Engelhardt 2002) who has created a library of information graphic patterns, Colin Ware (Ware 2008) who has studied ways to present complex oceanographic information, and Ian Muehlenhaus (Muehlenhaus 2013) who has recently written a book on Web based cartography.

Information design is important and the ease and ability to display complex information on the web has improved greatly over the years.

Spatial Cognition

In a review by Neil Burgess (Burgess 2008), he states that, at a coarse level, there are two parallel systems or modules of spatial cognition which have their own frames of reference and learning rules. These systems are egocentric and allocentric. Egocentric refers to a relationship between the viewer and objects in view. Thus the frame of reference follows the gaze of the viewer. Allocentric refers the relationship between objects and the frame of reference in terms of certain landmarks - particular objects that stand out for the viewer. Of particular interest are experiments in which individuals looking at objects on a table had better recall of the objects when they walked around the table as opposed to the table rotating while their view didn't change.

So how does this relate to web mapping? On a large display a map is basically vertical, a wall map. On a hand-held device this is different, the map is basically horizontal. These perhaps represent significant differences in the way maps are presented, used, and understood. The topic of landmarks is fascinating. One thing this suggests is that people who have actually visited a location are best knowledgeable about important landmarks for navigating that location. This is the philosophy behind such projects as OpenStreetMap (OpenStreetMap contributors 2014).

Open Source vs Proprietary

There is an important tension between proprietary and Open Source software (Stallman 1985). Proprietary allows huge resources to be allocated to a problem while Open Source allows for tinkering and the development of new generations of programmers. In addition, Open Source can solve problems often overlooked by large corporations and, by distributing talent, can often result in a superior product. Thus Open Source and proprietary software both compete and complement each other. I will be using both proprietary and open source software.

Conclusion

Design concepts are important criteria in which to help navigate the development of any programming project. Design concepts are not rigid. The ultimate goal of any project is completion using a certain set of software tools and a preconceived functionality.

METHODS

This application is designed for the Springs Stewardship Institute (SSI) (Stevens and Ledbetter 2014) in Flagstaff, Arizona.

Springs are ecosystems of heightened management attention, serving as hydrogeologic windows into aquifers, as critical water supplies, as keystone ecosystems, and as refugia for rare or unique species, remarkable paleontological repositories, and focal points of human cultural and development. However, springs have yet to receive substantial attention or protection from water or natural resource managers or policy makers: little significant attention has been paid to springs ecosystems in any major review of national water resources status in the past two decades. This lack of recognition is partially due to the inherently complex and multidisciplinary nature of springs ecological research, the lack of a lexicon with which to describe different types of springs, jealous guarding of springs as domestic and agricultural water sources, and lack of legislative protection. Improving springs stewardship requires assessment, planning, implementation, and monitoring, all of which are best when based on rigorous, scientific inventory (Stevens, Springer, and Ledbetter 2010).

Application Specifications

The purpose of this application is to allow a user to annotate existing springs or to add information and annotation about newly discovered springs. There are two reasons for this. First, it might provide current information about existing springs including location: current coordinates could be incorrect, and water flow, condition, etc. Second, photographs, video, and video tracking provide researchers with important help if they need to return to a spring to do an in-depth survey. Annotation includes photographs, video, and the appropriate geo and temporal referencing and markup. Since the application is located as part of an open web site, some security features are necessary. Because of the sensitive nature of springs, the results of adding information to a layer are hidden after the user leaves the application. An admin login allows an administrator to view the complete annotation layer and to be able to delete any inappropriate additions. The map consists of a topological base map, a layer extent polygon of Desert Landscape Conservation Cooperative (DLCC) (Desert Landscape Conservation Cooperative 2012) publicly published springs in the southwest, and a layer of these springs as points with their attributes. These two layers have been published through SSI's REST services. There are four annotation layers, one point layer for the springs themselves, one point layer for photographs, a point layer for single georeferenced video and a set of layers: polypoint, line, and a table for Geographical Exchange Format (GPX) tracks which may or may not include video. In addition, a table of names relates to the different layers. The geospatial layers and the accompanying tables and relationships are in a geodatabase, either SQL or PostgreSQL because this is required for adding, editing, and deleting elements of a layer. The annotation layers are either served by the SSI ESRI ArcServer (ESRI 2014a) or personal GeoServer (GeoServer 2014a) installation. Both geoservers allow queries as part of the REST call (ESRI 2014b, GeoServer 2014b). The author has visited a few unverified existing springs and collected some data. These are preloaded and available for the regular user. Once an administrator logs in, all the annotation data will be available. The application uses a proxy page to talk to the geoserver as this prevents both cross browser scripting and allows for the layer set to be protected by a hidden password.

Figures

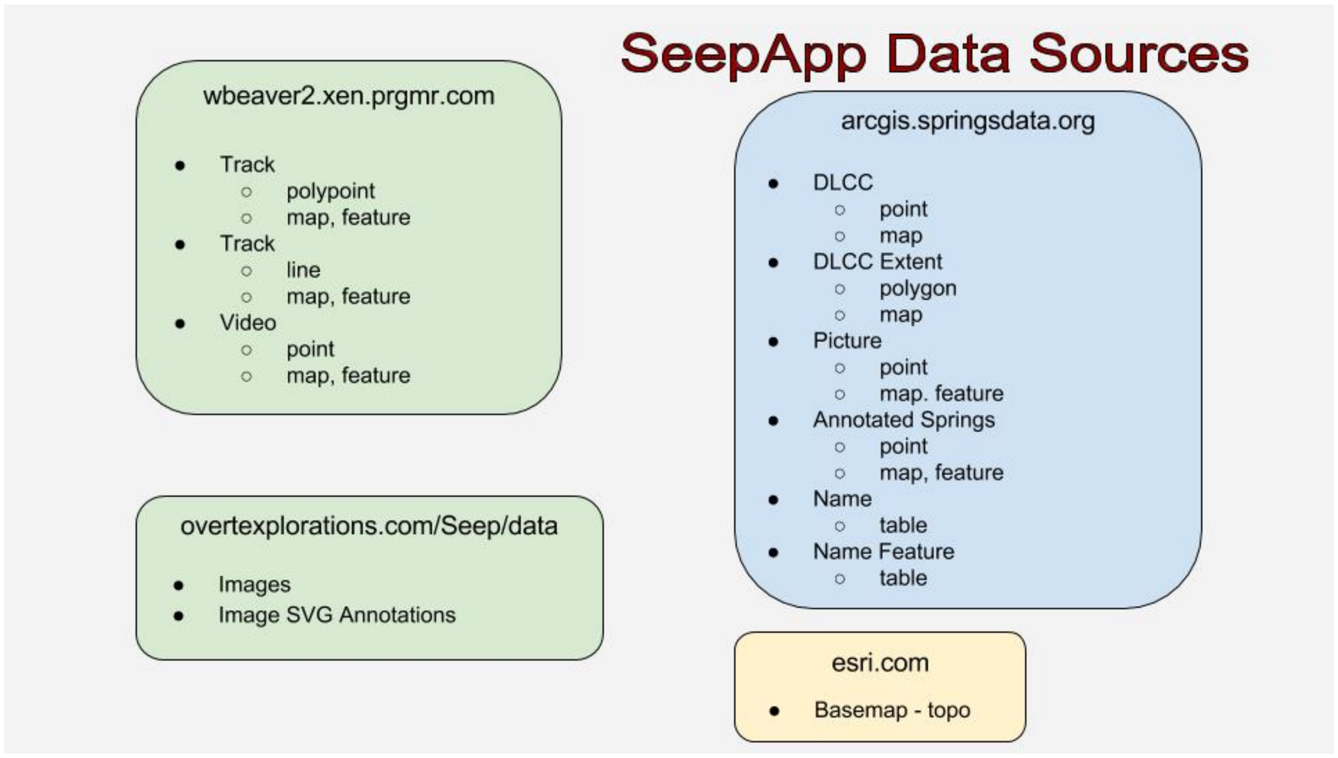

Figure 1. List of Data Sources

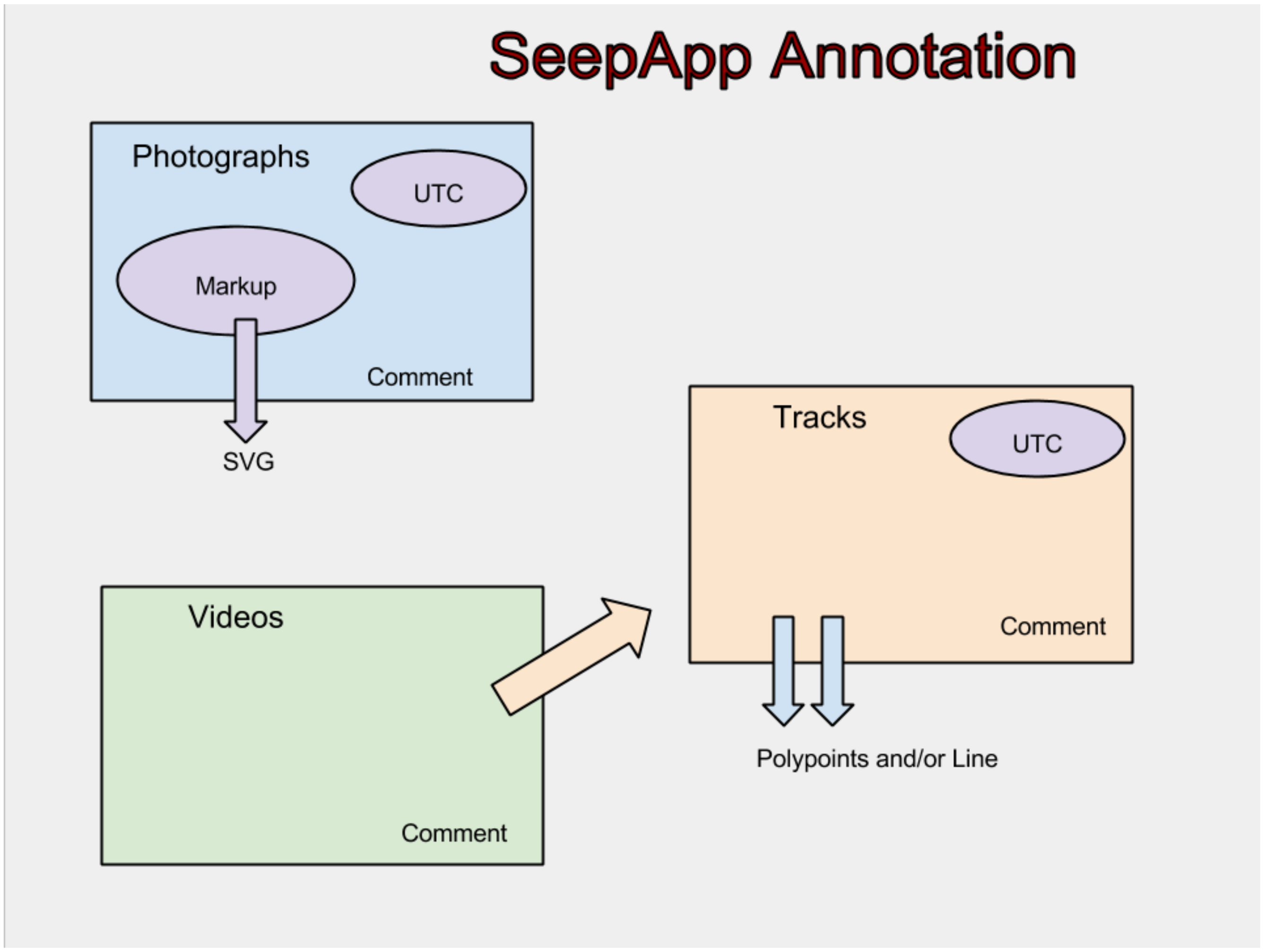

Figure 2. Annotation using photographs, videos, and tracks

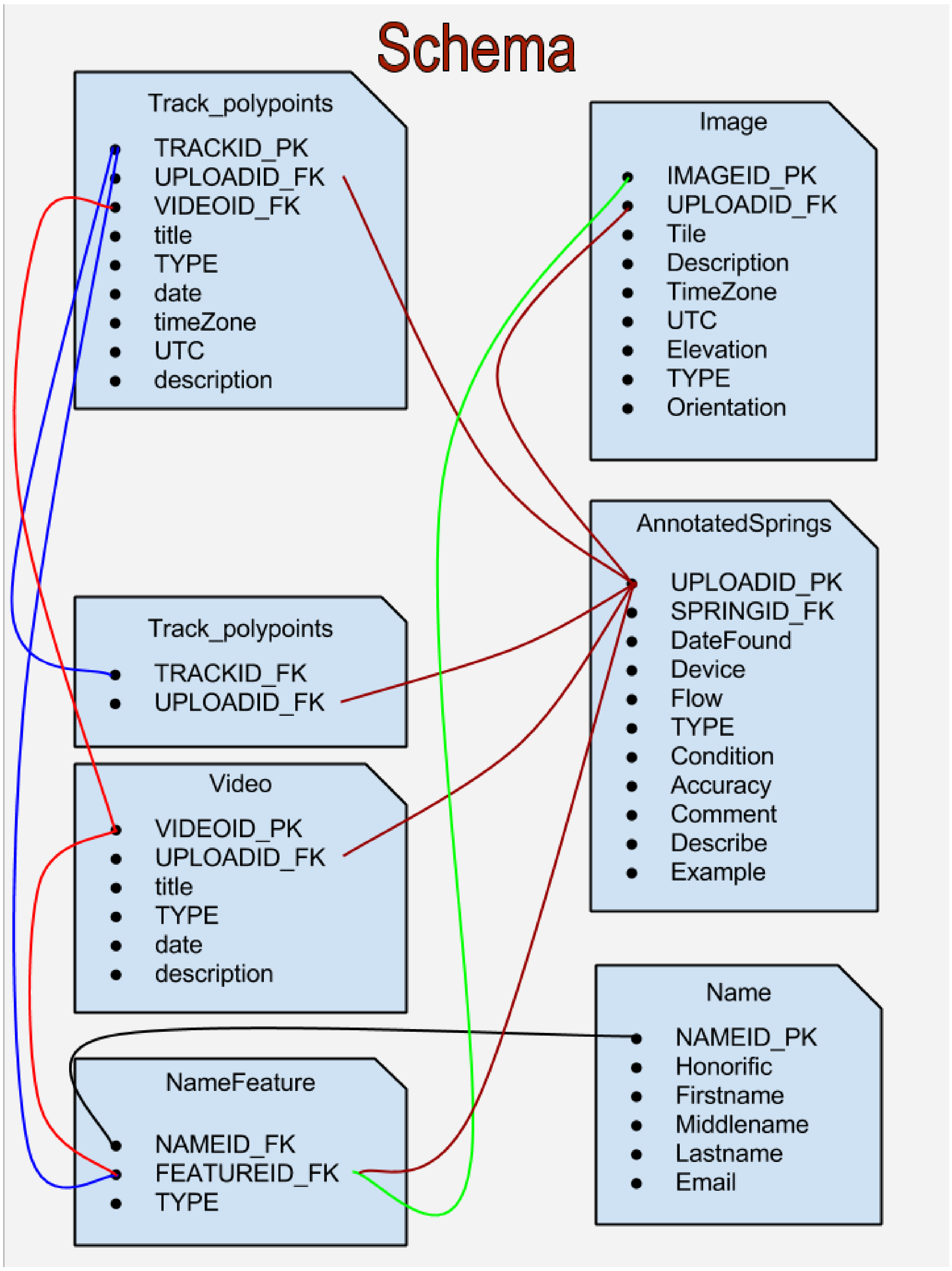

Figure 3. Schema of the data source attributes and their relationships

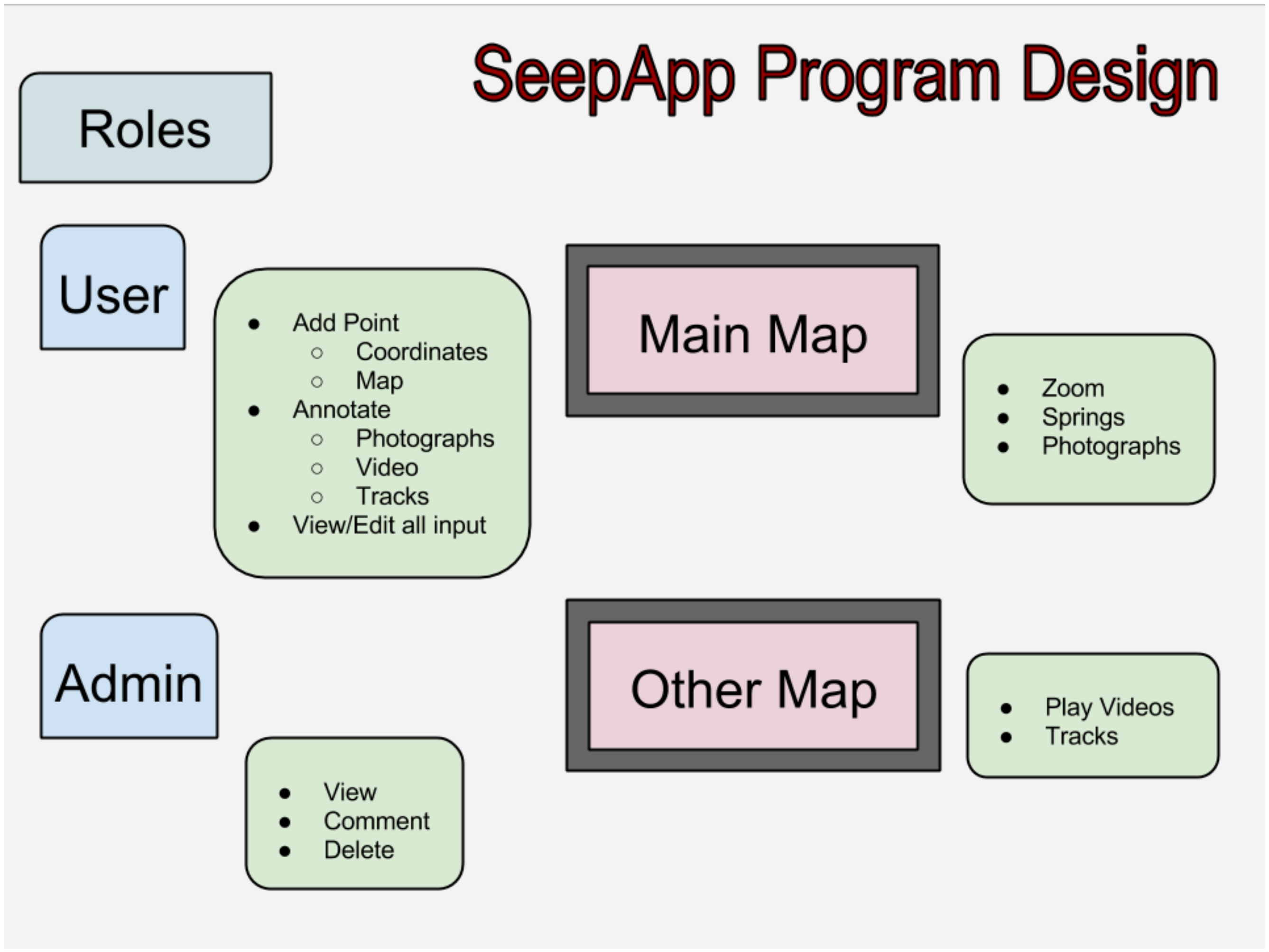

Figure 4. Basic program design

Programs, APIs, and Libraries Used

JavaScript and some PHP are used as programming languages. The ESRI API (ESRI 2014c) and the OpenLayers API (OpenLayers 2014) are used for the mapping. Both APIs use the Dojo framework (Dojo 2014). Photographs are stored on a server where sized copies and tiles of each photo are generated for display and annotation purposes. Markup of a photo is stored with the photo in JSON format. Video markup is stored on the server but the user is required to post their videos on YouTube (YouTube 2014) so the application only has to deal with one video format. Additional JavaScript libraries are used. Some work within the Dojo framework while those that don't run as a separate page within an iFrame tag to filter out incompatibilities. The additional JavaScript libraries are as follows:

- panojs.js (handles tiled photos) (Fedorov 2014)
- fabric.js (photo markup) (Zaytsev, Kienzle, and Bogazzi 2014)
- video.js (html5 video) (video.js 2014)
- MediaElement.js (YouTube html5 interface) (Dyer 2014)

Additional libraries and programs installed on the server:

- imgcnv (photo tiler) (Kvilekval et al. 2010)
- GDAL (Geospatial Data Abstraction Library)(GDAL 2014)
- proxy.php (ESRI proxy)(ESRI 2014d)

One of the problems with this use of blended API's is that one ends up with a great many conflicting CSS style sheets. These have to be manipulated into a consistent look and feel. The application uses Paletton (Paletton 2014) for style colors and experiments with CSS gradients using Ultimate CSS Gradient Generator (iosart labs 2014) and Subtle Patterns (Mo 2014).

A user survey will ask user testers a set of questions about the application.

Full code and other documentation for this project is posted on GitHub (GitHub 2014).

CONCLUSION

Deliverables

The application is located at: http://overtexplorations.com/seepApp.html. The Github source is at https://github.com/wjbeaver/Hide-Seep. The source is free to any who wish to use or modify it. Reference of original author material is appreciated.

Design Usage

Style

CSS styles in this program came from multiple sources because of the different APIs, frameworks, and libraries used. In some cases the styles were embedded within the HTML elements with no way to change them except to modify the code. In all other cases, existing styles were overwritten using application style sheets. The application uses colors from the Paletton color designer (Paletton 2014) and uses patterns from Subtle Patterns (Mo 2014).

Map Design

The goal of the mapping application is annotation of existing springs with the ability to add new springs. Dates, names of participants, and several structured and general questions are asked. There is the ability to download photos and tracks and to link to video. Photo EXIF metadata is read and time and date stamps are converted into UTC format. If available, GPS and orientation data are collected. Photos are tiled and sized for viewing. In addition, photos can be marked-up using a drawing module.

Design Patterns

Dojo widgets are used throughout the application. Widgets are design patterns that mix HTML templates with module functionality. This combines View and Controller patterns into a reusable module. In addition, widgets can be placed inside other widgets. All forms in the application are widgets along with widgets for floating attribute bars, name handling, and UTC handling.

All application form widgets use a transaction pattern that separates new attribute input from the feature attributes until an Add/Update button is selected.

Modularity is used whenever possible. In some cases, incompatible modules are isolated with iFrame tags.

User Survey

Question	Answered	Yes	No	% Yes
Do you think the application takes too long in loading?				
Did the map ever not complete loading by showing dark grey areas or out of focus areas when zooming?				
Did the application ever quit working?				
Do you think the application needs more instructions?				
Does the application fulfill its purpose?				

1-1 Template of User Survey

The survey resides at http://goo.gl/forms/Bf8XYK3uxI and will be open until beta testing is complete. No results were obtained as there are not enough participants at publication date.

The author has visited several springs in the region around Greer, Arizona and around Prescott, Arizona. Gathering data from these springs allows for a view of the application’s functionality in light of actual use. A spring in Prescott and a spring in

Greer have been marked as examples. This means that the two springs will appear on a new map and a user will be allowed to view but not edit the contents.

Photograph Exif Data

Source	**Date/Time**	**Coordinates**	**Elevation**	**Bearings**
Sony α 2000 SLR Camera	yes	no	no	no
IPad	yes	Wi-Fi GPS only	Wi-Fi GPS only	Wi-Fi GPS only
IPhone 5	yes	yes	yes	yes
Samsung Galaxy	yes	yes	yes	no

2-1 Photographic Sources of Exif Data

The table above shows the ability of various photographic sources to output Exif data. The IPad is the only device that outputs all information but Wi-Fi GPS only works in cities with special Wi-Fsi hotspots. True GPS for an IPad is several hundred dollars more. For the IPhone, the GPS stamp only works when there is a phone service connection. In addition, comparing elevation numbers between the three sources and a Garmin Trek-10 gives values that vary by 50 meters.

Issues

Modularity is a fine idea but difficult in practice. Part of modern programming is stitching together APIs, frameworks, and libraries from wildly different sources and programming types. Attempts to help rectify problems of modularity in JavaScript have resulted in several different solutions that are in themselves incompatible! One of the goals of modularity is for the program code to contain no global variables. The application does use a few global variables.

Dojo is an important framework for web mapping especially since ESRI uses it as the basis of its JavaScript API. It is also used with OpenLayers. Dojo is unfortunately not easy to learn. The framework has changed greatly the last few years and the documentation is just beginning to catch up. Knowing how to create and use widgets is highly useful for a web mapping developer.

Each module imported from an outside source contains its own CSS files along with many inline styles. The styles are specific to each module and often conflicting and bland. It is best when an application has its own style. The best way to do this is to clone all the files and modify them rather than expand on the CSS hierarchy with additional files. This can entail thousands of lines. For this application, files that modify the existing CSS styles were created.

ESRIs JavaScript API is free for use but the code isn't open source. Access to the code source can be important for a complex application. Most of the source can be viewed with a browser JavaScript debugger. All code found contains scary legal warnings and some important modules cannot be viewed. One such module, the Attribute Inspector, is an important component for adding and editing attributes of features. It is essentially a black box, which is OK if it is documented correctly. The Attribute Inspector is particularly powerful as it can adjust to all the Domains, SubTypes and Relationships associated with each attribute. Lacking is the ability to match custom widgets with certain attributes. Another is the apparent inability use inheritance with the widget. Because of these limitations and because the author wanted practice with creating widgets, the Attribute Inspector was partially reverse engineered and several different widgets created for the specialized functionality of the different layers.

Future Improvements

Any programming project is a continuing work in progress. Usage means the users require more functionality and more obscure problems are found and need to be solved. The program could use better handling of GPX data including a GPX reader that would route waypoints and tracks to particular annotation features. Problems with modularity need to be addressed and global variables phased out. Questions about Dojo need to be answered. Will Dojo 1.10 break the program? It probably will not. Will Dojo 2.0 break the program? It probably will. It would be interesting to add geologic and hydrologic layers to the application.

LIST OF REFERENCES

Alexander, Christopher, Sara Ishikawa, and Murray Silverstein. 1977. *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure)*. Oxford University Press.

AnalyticalGraphicsInc. 2013. “Cesium Language (CZML) Guide.” *GitHub*. https://github.com/AnalyticalGraphicsInc/cesium.

AnalyticalGraphicsInc. 2014. “Cesium.” *GitHub*. https://github.com/AnalyticalGraphicsInc/cesium.

Anselin, Luc. 1989. “What Is Special about Spatial Data?: Alternative Perspectives on Spatial Data Analysis.” National Center for Geographic Information and Analysis Santa Barbara, CA, 1-9.

Apple Computer. 2014. “View Controller Programming Guide for iOS: About View Controllers.” https://developer.apple.com/library/ios/featuredarticles/viewcontrollerpgforiphoneos/Introduction/Introduction.html.

Aristotle. 1954. “The Poetics.” In *Rhetoric and Poetics of Aristotle*, edited by Freidrich Solmsen, translated by Ingram Bywater. New York: The Modern Library.

Asantiago. 2012. “Dojo + OpenLayers = New Challenges | A Curious Animal.” http://acuriousanimal.com/blog/2012/01/23/dojo-openlayers-new-challenges/.

Burgess, N. 2008. “Spatial Cognition and the Brain.” *Annals of the New York Academy of Sciences* 1124 (1): 77–97. doi:10.1196/annals.1440.002.

Chauhan, Shailendra. 2014. “Understanding MVC, MVP and MVVM Design Patterns.” http://www.dotnet-tricks.com/Tutorial/designpatterns/2FMM060314-Understanding-MVC,-MVP-and-MVVM-Design-Patterns.html.

Desert Landscape Conservation Cooperative. 2012. “DLCC 2012 Science Projects: Mapping Springs and Seeps and Aquatic Habitat in the Desert LCC.” http://www.usbr.gov/dlcc/science/2012/Mapping-Springs-and-Seeps-and-Aquatic-Habitat-in-the-Desert-LCC.cfm.

Dojo. 2014. “Unbeatable JavaScript Tools - The Dojo Toolkit.” http://dojotoolkit.org/.

Dyer, John. 2014. “MediaElement.js - HTML5 Video Player and Audio Player with Flash and Silverlight Shims.” http://mediaelementjs.com/examples/?name=youtube.

Engelhardt, Yuri. 2002. *The Language of Graphics: A Framework for the Analysis of Syntax and Meaning in Maps, Charts and Diagrams*. Amsterdam; Amsterdam: Institute for Logic, Language and Computation, Universiteit van Amsterdam.

ESRI. 2014a. “ArcGIS for Server | GIS Web Server Software | Web Map Server.” http://www.esri.com/software/arcgis/arcgisserver.

ESRI. 2014b. “ArcGIS 10.1 for Server (Windows) - Filtering Features Using the layerDefs Parameter in WMS Requests.” http://resources.arcgis.com/en/help/main/10.1/index.html#//015400000604000000.

ESRI. 2014c. “ArcGIS API for JavaScript.” https://developers.arcgis.com/javascript/.

ESRI. 2014d. “Resource-Proxy.” https://github.com/Esri/resource-proxy/tree/master/PHP.

Fedorov, Dimitry V. 2014. “PanoJS3 - Pure JavaScript Viewer for Huge Images.” http://www.dimin.net/software/panojs/.

Few, Stephen. 2009. “Introduction to Geographical Data Visualization.” *Visual Business Intelligence Newsletter*, 1–11.

Forbes, Brian. 2013. “Dojo Deferreds and Promises - The Dojo Toolkit.” https://dojotoolkit.org/documentation/tutorials/1.9/promises/.

Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. 1994. *Design Patterns: Elements of Reusable Object-Oriented Software*. 1 edition. Reading, Mass: Addison-Wesley Professional.

GDAL. 2014. “GDAL - Geospatial Data Abstraction Library.” http://www.gdal.org/index.html.

GeoServer. 2014a. “OpenGeo Suite User Manual — OpenGeo Suite 4.1.1 User Manual.” http://suite.opengeo.org/opengeo-docs/.

GeoServer. 2014b. “Filtering in GeoServer — GeoServerUser Manual.” http://suite.opengeo.org/4.1/geoserver/filter/.

GISTutor. 2014a. “GIS Tutorials for Beginner, Intermediate, and Advanced GIS Users.” http://www.gistutor.com/.

GISTutor. 2014b. “PostGIS Query Viewer.” http://www.gistutor.com/demos/postgis/postgis_query_viewer.php.

GitHub. 2014. “Build Software Better, Together.” *GitHub*. Accessed July 3. https://github.com.

Gordillo, Silvia, Federico Balaguer, Catalina Mostaccio, and Fernando Das Neves. 1999. “Developing GIS Applications with Objects: A Design Patterns Approach.” *GeoInformatica* 3 (1): 7–32. doi:10.1023/A:1009809511770.

Haining, Robert. 2009. “The Special Nature of Spatial Data.” *The Sage Handbook of Spatial Analysis. Sage, Thousand Oaks, CA*, 1–18.

Hughes, John. 1990. “Why Functional Programming Matters.” In *Research Topics in Functional Programming”*, edited by D Turner, 17–42. Addison-Wesley.

iosart labs. 2014. “Ultimate CSS Gradient Generator - ColorZilla.com.” http://colorzilla.com/gradient-editor/.

Kvilekval, Kristian, Dmitry Fedorov, Boguslaw Obara, Ambuj Singh, and B. S. Manjunath. 2010. “Bisque: A Platform for Bioimage Analysis and Management.” *Bioinformatics* 26 (4): 544–52.

Laurel, Brenda. 1991. *Computers as Theatre*. Addison-Wesley.

Laurel, Brenda. 2014. “Brenda Laurel Bio.” . http://tauzero.com/Brenda_Laurel/BrendaBio.html.

Mo, Atle. 2014. “Subtle Patterns.” http://subtlepatterns.com

Muehlenhaus, Ian. 2013. Web Cartography: Map Design for Interactive and Mobile Devices. CRC Press.

Odersky, Martin. 2014. “The Scala Programming Language.” http://www.scala-lang.org/.

OpenLayers. 2014. “OpenLayers: Home.” http://openlayers.org/.

OpenStreetMap contributors. 2014. “OpenStreetMap.” http://www.openstreetmap.org/#map=5/51.500/-0.100.

Osmoni, Adi. 2012. *Learning Javascript Design Patterns*. http://www.slideshare.net/anhtuanbk/learning-javascript-designpatterns

Paletton. 2014. “Paletton - The Color Scheme Designer - SeepApp Colors.” http://paletton.com/#uid=75u0u0k1BlL00YR0isZ6ah6ngjB.

Peterson, Gretchen. 2011. “Map Design Patterns? - General Cartography.” *CartoTalk*. December 16. http://www.cartotalk.com/index.php?showtopic=7800.

Roddenberry, Gene. 1995. “Star Trek: Voyager.” *StarTrek.com*. 2014. http://www.startrek.com/page/star-trek-voyager.

Schubotz, René, and Andreas Harth. 2012. “Towards Networked Linked Data-Driven Web3D Applications.”

Sokolova, Karina, Marc Lemercier, and Ludovic Garcia. 2013. “Android Passive MVC: A Novel Architecture Model for the Android Application Development.” In *PATTERNS 2013, The Fifth International Conferences on Pervasive Patterns and Applications*, 7–

12.
http://www.thinkmind.org/index.php?view=article&articleid=patterns_2013_1_20_70039.

Stallman, Richard. 1985. “GNU Manifesto.” https://www.gnu.org/gnu/manifesto.html.

Stevens, Lawrence E., Abraham E. Springer, and Jeri D. Ledbetter. 2010. “Inventory and Monitoring Protocols for Springs Ecosystems.”

Stevens, Lawrence E., and Jeri D. Ledbetter. 2014. “Springs Stewardship Institute.” *Springs Stewardship Institute*. http://springstewardship.org/.

Tufte, Edward R. 1990. Envisioning Information. Cheshire, Conn.: Graphics Press.

video.js. 2014. “HTML5 Video Player | Video.js.” http://www.videojs.com/.

Voytyshyn, Volodymyr. 2014. “Modern JavaScript Applications: Design Patterns.” http://www.slideshare.net/VolodymyrVoytyshyn/modern-java-script-applications?related=1.

Ware, Colin. 2008. *Visual Thinking: For Design*. 1 edition. Burlington, MA: Morgan Kaufmann.

White, Walter. 2013. “Data Driven Design.” http://www.cs.cornell.edu/courses/cs3152/2013sp/index.php.

Wikipedia. 2014. “List of Displays by Pixel Density.” *Wikipedia, the Free Encyclopedia*. n.wikipedia.org/w/index.php?title=List_of_displays_by_pixel_density.

YouTube. 2014. “YouTube.” https://www.youtube.com/.

Zaytsev, Juriy, Stefan Kienzle, and Andrea Bogazzi. 2014. “Fabric.js Javascript Canvas Library.” http://fabricjs.com.