[PD-04-016] Web GIS Programming

Abstract

Web GIS programming involves creating, extending, utilizing, Web GIS or web mapping solutions to solve specific problems, build complete applications, or consume or produce data and geospatial processing services. With the expansion of the internet and availability of Web GIS or Web mapping options, web GIS programming is becoming a commonly required skill set in many organizations. Web GIS programming is a type of software development that provides a means of handling internet, browser-based software application development tasks which require unique solutions to web GIS or web mapping problems. In addition, a number of Web GIS software options offer application programming interfaces (APIs) that provide a means by which developers can leverage the published data and processing services of others to build and customize applications through standardized interfaces with external web GIS software, data, and services. Web GIS programming applies to mobile as well as desktop application development. A browser typically runs software applications by submitting Hypertext Transfer Protocol (HTTP) or Hypertext Transfer Protocol Secure (HTTPS) requests to a server hosting resources the application user wishes to access available through a Uniform Resource Locator (URL), and the server replies by providing resources or performing functions requested by the user. This entry reviews the fundamentals of web GIS programming, accompanying the Web Mapping and other entries in the Programming and Development section, the Web GIS entry in the Computing Platforms section, and the User Interface and User Experience (UI/UX) Design entry in the Cartography and Visualization section (Sack, 2017; Quinn, 2018; Roth, 2017).

Keywords: application, CSS, debug, deploy, development, HTML, JavaScript, programming, test, tier, web mapping. web services

Author & citation

Swift, J., and Goldberg, D. (2019). Web GIS Programming. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2019 Edition), John P. Wilson (ed). DOI: 10.22224/gistbok/2019.1.5

Explanation

1. Definitions

application development: writing of computer software code for use in desktop software applications, internet browsers, via application programmer interfaces, through virtual or cloud-based computing environments, or on mobile devices

application programming interface (API): a set of programming components that communicate with each other and can be used to develop applications using a particular computer language (Quinn, 2018)

big data: unstructured or structured data volumes that are so large that they are difficult to gather store, analyze, manage and disseminate using traditional technologies (Li et al., 2016)

client: computer hardware, such as a device, that accesses a service provided by a server, or a client can be a software application installed and run on computers such as laptops and hand-held devices, such as web GIS applications that include programming

fat client: provides user functionality independently of a server, such as through a browser, also referred to as rich, thick, or heavy (client)
thin client: relies on a server for functionality, such as computations, also referred to as zero, slim or lean (client)

development environment: a protected, often temporary, environment specifically used to perform initial or ongoing application development before the system is tested or released for use

framework: a set of tools written to help other developers, usually modularizing a language so that others can use the tools to more easily build sophisticated applications

Hypertext Transfer Protocol (HTTP): functions as a request response protocol in client-server models, and can be extended by Hypertext Transfer Protocol Secure (HTTPS) for safe or protected communication over a computer network

integrated development environment (IDE): a software application designed to help developers write code. IDEs typically include a source code editor, compiler or interpreter, a debugger and other tools to facilitate coding, all within a single user interface

library: a collection of shared programming code, such as files, programs, routines scripts or functions that can be used to facilitate the development of other software programs and applications

mashup: a new data source, website, or web service created by combining existing data or services from multiple providers

network: digital telecommunications technologies, including software, cable hardware, and wireless media, used to exchange data and other resources between computers using connections

object-oriented (OOP): a programming paradigm designed to handle large, complex software systems, which allows developers to create containers or sections for data that can be manipulated independently within a given software program, so programs can be written and executed in parts. Objects can be created by programmers to represent real-world nouns and can be given methods (programmed procedures) that describe the desired behavior of the object

open source: developers can access, use and alter the software source code as needed

copyright: open source code that requires licensing in order to change, use and distribute to others
copyleft: the rule that when redistributing programs, restrictions cannot be added to deny others the four freedoms: to run, copy, distribute, study, change and improve source code

programming: implementing an algorithm using a programming language for execution by a computer

compiled language: a computer programming language in which human-readable source code is translated into machine code to be executed

proprietary: commercially available, copyrighted software that requires licensing for limited copying, change, use, and/or distribution, and typically the software source code is not released

optimization: in the context of web GIS programming, the process of setting up or scaling a high-performance server and network configuration to support intended usage of applications

production environment: a duplicate of a staging environment specifically used to deploy a finalized application for use after testing is complete

REST: representational state transfer (REST), a software architectural style that provides rules for creating web services that can be used for 2-D and 3-D data transfer, geoprocessing, and delivery to web GIS programming projects (Fielding, 2000). Web services conforming to this style are referred to as Restful.

Restful web service: a Restful web service makes requests to a given resource's typically accessed through a Uniform Resource Locator (URL) and receives a response formatted in a web scripting language such as Hypertext Markup Language (HTML), Extensible Markup Language (XML), or Javascript Object Notation (JSON)

scalability: ability of software and hardware architectures to handle increases in user demand

scripting language: an interpreted computer language that supports scripts, which are programs that execute in a run-time environment, often used for automating multiple tasks

staging environment: a duplicate of a production environment specifically used to test application before they are released for use

user-centered design: an iterative testing process involving the programmer and the end users that seeks to determine the needs and the intended audience when creating the user interface for a web GIS programming project (Roth et al., 2015)

Uniform Resource Locator (URL): a reference to a resource that describes its location on a computer network, beginning with HTTP:// or HTTPS://, named and defined by the resource owner

web application framework: a software framework such as ASP.NET or Ruby on Rails, created to help developers build web applications such as web API's and web services and

often providing libraries for simplifying programming tasks such as accessing and configuring data

web service: a service provided by one device such as a server to another device such as client through the internet, often providing access to remote data or computational capabilities

2. What is Web GIS Programming?

With the expansion of the internet and availability of Web GIS or Web mapping options, web GIS programming is becoming a commonly required skill set in many organizations. Web GIS programming is a type of software development that provides a means of handling internet, browser-based software application development tasks which require unique solutions to web GIS or web mapping problems. Web GIS programming applies to mobile as well as desktop application development. A browser typically runs software applications by submitting **Hypertext Transfer Protocol** (HTTP) or **Hypertext Transfer Protocol Secure** (HTTPS) requests to a server hosting resources the application user wishes to access available through a **Uniform Resource Locator** (URL), and the server replies by providing resources or performing functions requested by the user. This entry reviews the fundamentals of web GIS programming, accompanying the Web Mapping and other entries in the Programming and Development section, the Web GIS entry in the Computing Platforms section, and the User Interface and User Experience (UI/UX) Design entry in the Cartography and Visualization section (Sack, 2017; Quinn, 2018; Roth, 2017).

Web GIS programming involves creating, extending, utilizing, Web GIS or web mapping solutions to solve specific problems, build complete applications, or consume or produce data and geospatial processing services. In addition, a number of Web GIS software options offer **application programming interfaces** (APIs) that provide a means by which developers can leverage the published data and processing services of others to build and customize applications through standardized interfaces with external web GIS software, data, and services. Sack (2017) provides a list of commonly used open source and proprietary software APIs used in web GIS programming projects focused on **mashups**. Among the most popular are the Google Maps API, Mapbox GL JS, OpenLayers, Esri ArcGIS API for Javascript, ArcGIS **REST** API, ArcGIS Python API, ArcGIS Runtime SDKs (software development kits), and Leaflet. For example, the Esri GeoServices REST specification provides a standard way for clients to communicate with servers through the REST architecture using Restful web services and URLs (Esri, 2010). In web GIS programming, such hosted Web GIS software libraries and other **scripting language libraries** such as JQuery provide globally available, reliable collections of modular code that can be used by developers to simplify writing programs. Open Layers and Leaflet are examples of open source libraries used to create web map mashups.

3. Guidance for Web GIS Programmers

As with any type of software development, web GIS programming involves taking a big idea and breaking it into separate components and steps, defining the tasks required to build the application, and choosing the appropriate Web GIS platform, software, and supporting

computer language to build the given application. When choosing web GIS software and supporting programming options, the following questions offer project planning guidance for the developer:

1. What are the functional requirements of the web GIS programming project?
2. Should the Web GIS system architecture be client-side or server-side?
3. What are the application performance requirements, constraints, or limitations?
4. What is the level of end-user interactivity anticipated within the application?
5. Are there specific browsers that the program must work in, or must it work in all browsers?
6. Is the application intended to only work in browsers on desktops or server-based Web GIS software, or should it also work on tablets and mobile devices?
7. Will data be provided to the user, or gathered from the user, or both?
8. What are the sizes of the spatial datasets to be displayed or utilized in the web GIS application, and can the chosen data storage format meet the performance needs of the system, meaning a fast response from the web map?
9. What are the locations of the spatial datasets to be displayed in the web maps, and how will they be accessed?
10. Is the data stored in a database on the server hosting the web GIS program, or being streamed as a web service to the web map from another server?
11. What are the web GIS and supporting programming languages that can be supported by both the application developer, and those who requested the application, if maintenance of the application will be turned over after completion?

The following sections provide general information to serve as a guide in answering the questions above directly pertaining to web GIS programming.

4. Web GIS Programming Fundamentals

The main components of a web GIS programming project are illustrated in Figure 1, including examples of common application (app) development activities that can involve programming in one or more computer programming languages.

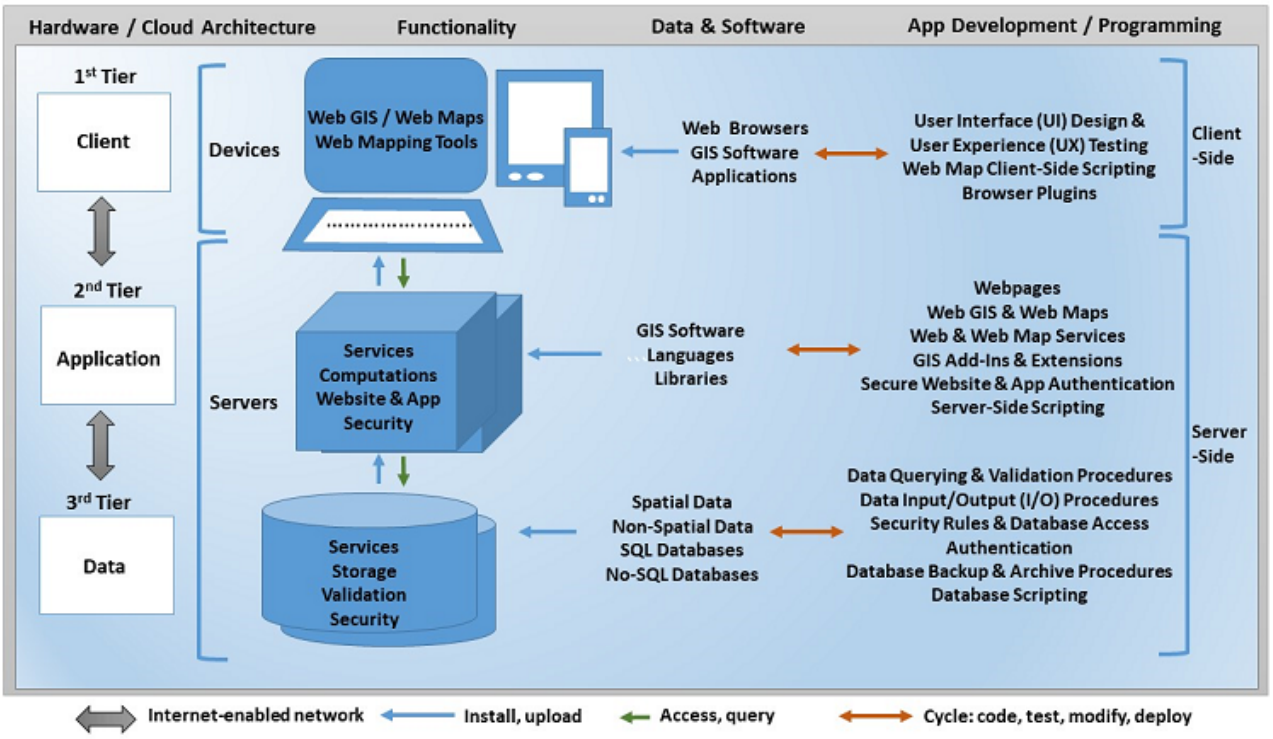

Figure 1. Main components of a web GIS programming (development) environment. Source: authors.

A web GIS programming environment typically runs on a local or remote computer. Programming activities commonly include writing code, extending or customizing existing software, repeated testing, improvement, and maintenance of web GIS applications. A web GIS programming environment requires internet connectivity and network access to devices intended to deploy the application to users. Developers typically use **integrated development environments** (IDEs) and **web application frameworks** to develop web pages and web GIS applications. Network access between the tiers is accomplished via standard internet protocols, with communication transmitted through hardwired (Ethernet) or wireless (wifi or cellular) connections, usually broadband in all cases. Networks must have bandwidth capable of supporting all phases of web GIS application development, testing, and deployment. A **development environment** must also include direct access to software such as GIS, Web GIS, computer languages, and to the data to be utilized in the web GIS application.

4.1 System Architecture

A multi-tier system architecture is commonly set up or utilized by the programmer, or developer, to support a web GIS programming project. This architecture includes one or more **clients**, application servers, and data servers. Client-server architecture is described in detail in Sack (2017), and web GIS architectures and GIS client applications in Quinn (2018). The client is the programmed web GIS application that runs on an end user's device and executes the functionality of the web GIS application. The client sends and receives requests to and from a server, often through a browser, a stand-alone software application, or an add-on or plugin to another software installed on the client. The application server typically has more computing resources than the client and manages computations and

resources such as data storage and internet traffic over the networks that connect to it. The server communicates with the data server, storing and retrieving the information collected or requested by the client.

In a multi-tier system, each tier usually executes on separate computers, which allows for better performance than another common configuration, a 2-tier client-server architecture. The client is often referred to as thin versus fat. A **thin client** depends heavily on a server for functionality, while a **fat client** operates more of the computation on the client device. A thin client model is considered more straightforward to manage because the hardware and software resources required of end users is lower.

Software programmers often use separate computers for web GIS application development, staging, and production, illustrated in Figure 2. Web GIS application development most often includes writing computer code in a **compiled** or **scripting language** or modifying existing web GIS programming code. **Staging** refers to setting up an environment for testing applications before they are released for use. A **production environment** is the final deployment system supporting websites and web GIS applications accessible to end users. Development and staging environments are required to support developers in building, debugging and testing web GIS components, creating scalable applications, functionality, websites, and databases, and constructing security rules. Staging and production environments are normally complete duplicates of the development environment with additional or different data and server security precautions intended to minimize the production environment's vulnerability, such as unauthorized use.

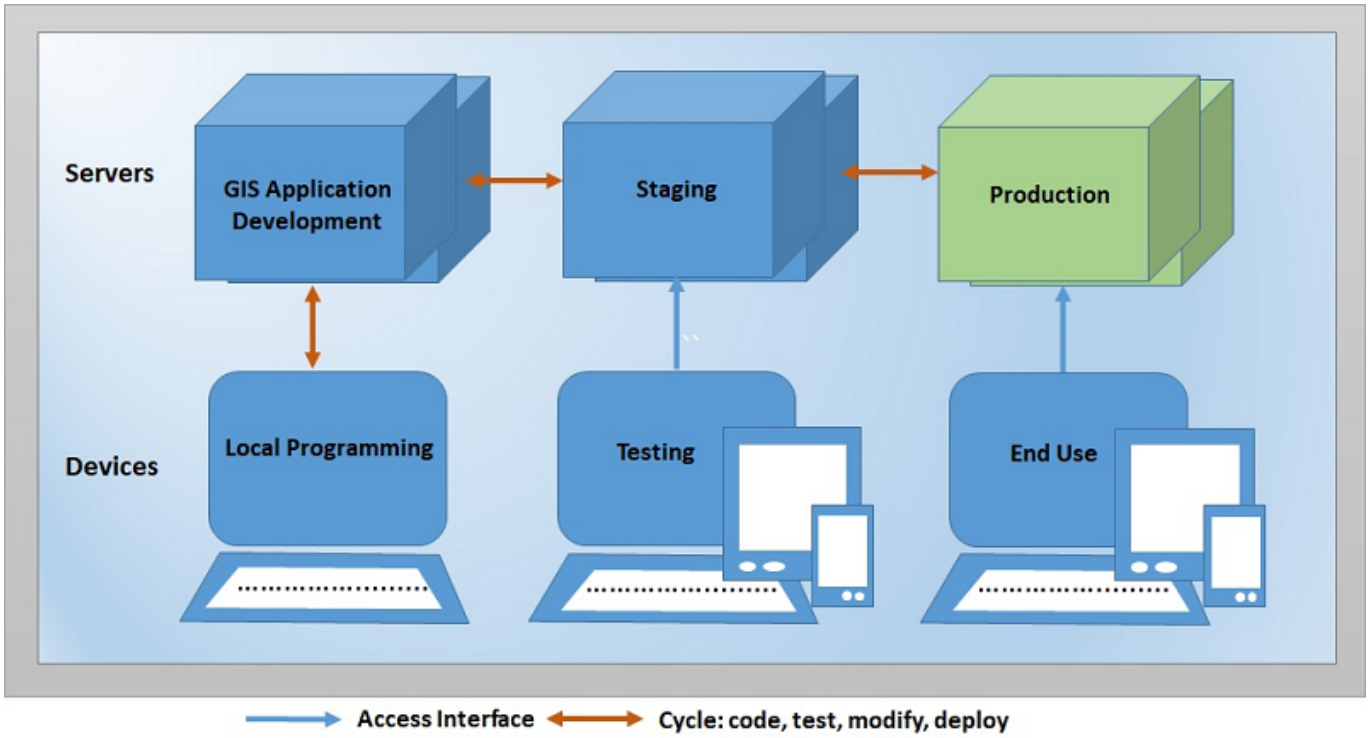

Figure 2. The main components of Web GIS application development, staging, and production. Source: authors.

4.2 Application performance requirements, design, documentation, constraints, limitations and testing

When planning a web GIS programming project, requirements may be provided by the intended user, such as by a regulatory authority, or through study of the intended user community. Ideally, a user scenario study is carried out to determine the needs of the target user audience to produce a storyboard that describes how intended users will interact with the application. Performance requirements also include determining whether the audience prefers browsers or stand-alone applications, as well as what devices (PC, Mac, laptop, tablet, mobile phone) the application should run on. Optimizing application performance for multiple browsers versus standalone applications on multiple devices and operating systems requires different time investments in testing, bug fixing, and maintenance. User requirements guide the choice of platform, software, and programming languages, which in turn have inherent constraints and limitations.

An estimate of the potential number of users is critical in planning the required infrastructure, such as servers and data storage. Both 2-tier and 3-tier architectures (Figure 1) can be scalable, meaning that if user demand increases, extra resources to handle the usage can be added to the architecture behind-the-scenes to support high-performance deployment. Common examples include adding redundant systems, load-balancing, additional computing power, data storage, or servers. Scalability in a web GIS programming implies that applications will be designed or even purposefully coded to dynamically handle website and web map functionality performance challenges. Dynamic computer performance refers to optimization of computer resources such as storage and processors, usually done through server-side operating system-level programming. This is different from programming web map functionality that is responsive to user interaction, such as enhancing the functionality of a web GIS program to dynamically update legends and symbols when a web map embedded within a website redraws (Roth, 2017).

The choice of local infrastructure versus cloud-based computing resources is an important consideration in any programming project. Cloud computing is a broad term for a model that enables on-demand network access to a shared pool of virtual resources such as servers, storage, and web services (Mell and Grance, 2011). Cloud computing vendors such as Amazon AWS and Microsoft Azure offer an enormous range of computing and data storage options. Some of the main advantages include speed of setup, ease of access and redundancy, and scalability of solutions for web GIS programming projects. The cost structure to host a web GIS programming project is an important consideration as commercial hosting platforms. Commercial hosting platforms are businesses designed to make profits thus generally charge fees on an ongoing basis, demanding a different budget model from onsite hardware acquired through one-time purchases.

Although there is a current trend toward blending the capabilities, goals, and software used in both web GIS and mobile GIS application development, web GIS programming and mobile GIS programming are usually treated as separate topics. This is due to significant differences in native operating system languages in desktop, web-based, and mobile device (i.e. Apple, Microsoft, and Android) development, and the large number of programming language, computer platform, and GIS software package options available as web GIS solutions that support programming in diverse hardware and software platforms.

One of the most important application performance constraints is the size of the spatial and

non-spatial data to be provided to users, or gathered from the users, through the web GIS application. The size of the data will guide the choice of system resources, such as data storage and processing capacity, and in particular the selection of the database or data store. In general, a **big data** set may consist of a data with significant variety, size (disk space required) and speed with which it must be handled by the application (Li et al., 2016). Examples of big data include social media such as Instagram data and satellite imagery. Data handling includes visualization in maps and tabular or text format, spatial or non-spatial analysis, querying, or collection. The size of the data as well as the nature of the data, in particular, if the data will reside as static files or as dynamically changing database records, is a critical determinant for application performance planning.

Where the data will reside, such as within the application development environment data server versus data that must be streamed through web services or geospatial web services from remote sources may significantly impact programming tasks and network **optimization**. Geospatial web services consist of images and data resources delivered real-time or near-real-time by a server. Near-real-time data is used when data processing time in minutes versus seconds in acceptable in an application. Sack (2017) provides a list of OGC geospatial web services' descriptions of standards which can be consumed through web GIS programs to publish raster map images, tilesets, vector data, space/time-varying visualizations, and geoprocessing services, all of which generally produce some output or response that is brought into a web GIS application.

A developer must also ensure that applicable copyrights for data and all software used and developed as part of a project are documented and transparent to other developers and end users (Buckley, 2010; Dempsey, 2012). Privacy, security, and confidentiality of application and end-user data are also important considerations in any web GIS programming project (Li et al., 2016; Richardson et al., 2015), particularly within some domains such as health, educational records, and finance. Sensitive data can be encrypted before being transmitted across a network, and it can be required that the receiver accept the encrypted data.

In addition, careful attention must be paid to the application UI design and intended UX (Roth, 2015 Roth et al., 2015, and 2017). Important goals of a user scenario study storyboard, or narrative, include determining the required functionality of the application that will best suit the intended audience. A web GIS programming project may require writing computer code to complete such tasks as creating a unique UI, streamlining database queries, adding stored procedures into a database, data capture or live (on-the-fly) data processing within the UI, augmenting an existing web GIS software functionality with a custom tool such as a widget, button or menu, and/or building a project-centric website in which to embed the Web GIS application. Similarly, data capture, or the gathering of data generated by users, must be accounted for in the database design. For example, a database designed as a Volunteered Geographic Information (VGI) (Quinn, 2018) application will require additional security measures to make sure that data provided by volunteers does not expose the system or other users to negative consequences.

User testing is of critical importance in creating a web GIS application, as this process determines the usability and utility of the application for the intended audience (Roth et al., 2015). User-centered design (UCD) principles should be applied to the creation and evolution of the user interface. In UCD, the application user interface is shared with a target audience for testing. The web GIS programmer gathers feedback from the users'

experiences, and then uses the feedback to improve the user interface. Multiple iterations of user testing can provide valuable input about the users' needs. This guidance facilitates the improvement of the application design through revisions to the utility, or client-side functional requirements, of the user interface. Ideally, such iterations lead to the enhanced usability of the web GIS programming project by the target audience.

Additionally, the web GIS programmer must provide developer-centric documentation for computer code from the beginning to the end of the project. Documentation can include comments within computer code, readme files with instructions for use, sharing, and further development of the code, and creation of metadata associated with any geospatial data provided with the code. User-centric documentation may include help content within the end user interface. Documentation is critical for long term use and maintenance of an application, and for sharing a project through a repository such as GitHub.

Version control or a software update tracking protocol should also be implemented to keep track of modifications to the project over time, especially if there is a team of programmers working on the project. Open source as well as proprietary software tools are available that are specifically geared toward application version control, and may also include tools for tracking issues such as bugs as well as iterative changes made to code.

Next, performance testing of web GIS applications using a staging environment includes checking the response time of all functionality in the application, to verify the efficiency of the system configuration. This is considered server-side functional requirement testing. Generally, a response time in tens or hundreds of milliseconds is expected for most web map functions. Correctness testing, also known sometimes as validation or unit testing depending on the granularity of the test, confirms that the responses provided by data and computational services match expected results. This is typically performed using a subset of input data for which output values are known in advance and compared to current system operational results. Both forms of testing can and should be automated programmatically, and are carried out by members of the development team or specific groups of selected end users (alpha and beta test groups).

Long-term application support and maintenance needs are also considered critical to successful project implementation. GIS and web GIS software, and server and client device operating systems versioning requires vigilance. Versioning of browsers will also have a critical impact on any application. In addition, client-side and server-side programming languages change versions just as frequently, though major changes in languages generally occur every three to five years. Versioning of software and hardware requires re-testing of production-ready applications. Regularly scheduled application maintenance and some allowance for unexpected updates should be included in the scope of any project for the anticipated lifespan of the application. Ideally, developers should foster a mindset to be able to "go with the flow" since this is simply part of the world of programming web GIS applications.

5. Web GIS Programming Languages

Web GIS programming involves using client-side programming languages for creating web pages and web GIS applications, and server languages for creating server-side applications. Several commonly used web mapping platforms and web GIS software options available for

implementing Web GIS projects are discussed in Sack (2017) and Quinn (2018), respectively. Client-side (Figure 1) languages frequently used in web GIS programming are listed in Table 1.

Table 1. Languages Typically Used Client-Side

Language	File Extension	Description
HTML	.html	Hypertext markup language: controls the structure and content of a web page.
CSS	.css	Cascading stylesheets: controls the look and feel of a web page.
JavaScript	.js	Scripting language used to write web GIS clients and to utilize or extend 3rd party APIs. Can be used to create 3rd party frameworks and libraries.

HTML, CSS, and JavaScript are scripting or front-end languages that work together in the browser, or the client, to control what is displayed and what happens on a webpage, including embedding a web map within a webpage. All of the processing of these languages is done on the client's computer. Libraries such as JQuery and Bootstrap provide **open source** toolkits that facilitate developing with scripting languages such as HTML, CSS, and JavaScript. One strength of these frameworks is that they work around bugs and inconsistencies between major browsers, so applications run smoothly across many browsers.

There are many compiled languages, and some scripting languages used in the spatial sciences and the geospatial industry that run on servers. These are often known as server-side, or back-end languages. A selection of the most common, general-purpose, object-oriented server-side programming languages used in web GIS programming that can also be used with many databases is listed in Table 2. Python, Java and ASP.NET, for example, can also be deployed as client-side languages.

Table 2. Languages Typically Used Server-Side

Language	File Extension	Description
ASP.NET	.aspx, .asmx, .ashx, .ascx	Developed by Microsoft as part of the .NET framework (Microsoft, 2018), specifically designed for development of dynamic web pages. Succeeded Microsoft's Active Server Pages (ASP).
C#	.cs	Developed by Microsoft as part of the .NET framework. C# is portable in that it can be used to develop web GIS software components that can be compiled then deployed on distributed servers.
Java	.java, .jar	Developed by Sun Microsystems then acquired by Oracle Corporation, typically used for development of client-server web applications. Java applications can run on any Java Virtual Machine (JVM) without recompilation of source code.
php	.php	Hypertext preprocessor; scripting language designed for web application development which can be embedded within HTML. PHP is executed by php runtime, an interpreter in the server, typically to create dynamic web page content or images.
Python	.py	Scripting language interpreted on the server at runtime that can be used to extend and customize desktop and cloud-based web GIS software (Rey, 2017).
Ruby	.rb	Web application development framework written in C that provides built-in structures for web pages, web services, and a database (Ruby, 2018). Ruby on Rails is a popular web framework written in Ruby. Example Ruby on Rails applications include GitHub, Basecamp, Airbnb, and Hulu (Rails, 2018).

The web application frameworks mentioned above can be used to develop applications

entirely separate from a GIS. Integration with geospatial software is voluntary and should be undertaken with clear objectives as to why GIS is needed. Every web GIS computer program is a set of instructions, a sequence of a few or many commands executed in a specific order, regardless of the computer language, platform (operating system) or goal of a specific application. The main difference between languages is primarily the rules used to assign meaning to variables or objects. Though it is beyond the scope of this entry to describe the web GIS programming languages noted above in detail, references and additional resources are provided.

References

Buckley, A. (2010, December 3). Using and citing Esri data [Blog]. Retrieved August 17, 2018.

Dempsey, C. (2012). Cite GIS Materials. GIS Lounge. Blog post, retrieved August 17, 2018.

Esri. (2010). Esri GeoServices REST Specification Version 1.0. An Esri White Paper, September 2020.

Fielding, R. T. (2000). Architectural Styles and the Design of Network-based Software Architectures. Unpublished PhD Dissertation. University of California, Irvine.

Li, S., Dragicevic, S., Castro, F. A., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A., & Cheng, T. (2016). Geospatial Big Data Handling Theory and Methods: A Review and Research Challenges. ISPRS Journal of Photogrammetry and Remote Sensing 115, 119–33.

Mell, P., & Grance, T. (2011). The NIST Definition of Cloud Computing. Recommendations of the National Institute of Standards and Technology. Special Publication 800-145. U.S. Department of Commerce.

Quinn, S. (2018). Web GIS. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2018 Edition), John P. Wilson (ed).

Rails (2018). Ruby on Rails. Retrieved August 17, 2018.

Rey, S. J. (2017). Python for GIS. The Geographic Information Science & Technology Body of Knowledge (3rd Quarter 2017 Edition), John P. Wilson (ed).

Richardson, D. B., Kwan, M.-P., Alter, G. & McKendry, J. E. (2015). Replication of Scientific Research: Addressing Geoprivacy, Confidentiality, and Data Sharing Challenges in Geospatial Research. Annals of GIS 21(2), 101–10.

Roth, R. E. (2015). Interactivity and Cartography: A Contemporary Perspective on User Interface and User Experience Design from Geospatial Professionals. Cartographica 50(2), 94-115.

Roth, R. E. (2017). User Interface and User Experience (UI/UX) Design. The Geographi Information Science & Technology Body of Knowledge (2nd Quarter 2017 Edition),

John P. Wilson (ed.).

Roth, R. E., Ross, K. S., & MacEachren, A. M. (2015). User-Centered Design for Interactive Maps: A Case Study in Crime Analysis. ISPRS International Journal of Geo-Information, 4(1), 262-301.

Sack, C. (2017). Web Mapping. The Geographic Information Science & Technology Body of Knowledge (4th Quarter 2017 Edition), John P. Wilson (ed.).