The Pros and Cons of Web Programming

Ferenc Vajda
Computer and Automation Research Institute of the Hungarian
Academy of Sciences and the Technical University of Budapest
Budapest, P.O. Box 63, H-1518 Hungary

Abstract

The paper evaluates the most important Web programming languages (i.e. Perl, Unix Shell, C and C++, Java, Eiffel, Scripts). After overviewing the significant features, the advantages and limitations are compared. The suitability of different programming means for specific applications is also considered and discussed.

"There does not now, nor will there ever, exist a programming language in which it is least bit hard to write bad programs"

Flon's Axiom

1. Introduction

The World Wide Web or the Web for short comes to life when it is interactive. There are many ways to have some form of interactivity but most cases it involves programming of some type to process the user's request and provide the information he/she has asked for. Though the Web is only a few years old, it has matured at a very fast pace. Today, people expect more from sites that support Web pages, including text, graphics and other multimedia content. People can surf on the Web using a Web *browser program*, such as *Microsoft Internet Explorer or Netscape Navigator*. There are different categories or application areas for Web programming. Usually the main goal of Web programming is to *create dynamic Web pages*, which interact with the user in some way. A typical application of this category is processing user forms. Other Web programs do not interact with user directly. These programs are called *back-end programs*. A typical program of this kind is one that analyses the activity of a site.

There are many different application environments and programming means that can be used for Web programming. Each language in wide use for *applications* has advantages and disadvantages. A systematic comparison could help a Web programmer in choosing the right *language and environment* for the job. It means not only to find a solution which fits the job but it must fit to the implementer as well.

2. Comparison of Web programming means

There are many aspects, which should be considered. Web servers are based on the well-known *client server* principle and system architecture. The mechanism for safely transporting data from a client (e.g. a browser like Netscape Navigator) to a server is called Common Gateway Interface (CGI). [1]

Client requests are handled by the server using so-called *daemon* programs. The CGI is the most common approach to *client-initiated* programs which uses standardised methods and nearly all Web servers support it. On the other hand *client-side* programming uses applications that run in the client i.e. the browser. (Typical client-side programming languages are JavaScript and VBSscript, see later).

When dynamic elements are included in a Web page a method called *Server-Side Includes* (SSI) is ideal. A third mean is the *Application Programmer's Interface* (API) which is proprietary method based on a set of programming commands designed to run under a given server (as the Microsoft's IIS). The advantage of developing programs using API is that we can get faster solution than a CGI version but an important disadvantage is that APIs are not standardised.

A promising new version of the TCP/IP protocol is the Inter-ORB Protocol (IIOP) which is part of the Common Object Broker Architecture (CORBA). The CORBA specification [13] describes IIOP as the TCP/IP implementation of GIOP, the General Inter-ORB Protocol. GIOP defines a network protocol-independent set of messages, formats and data encoding that all Object Request Brokers (ORBs) must follow when communicating with each other.

Before discussing the advantages and disadvantages of the most important programming languages, a number of important categories are briefly summarised.

- *Language type:* Procedural vs. Object-Oriented (00). A part procedural, part object oriented language is called *Object-based* language. Examples of procedural languages used on the Web include Unix Shell script, C, Perl, examples of object-oriented languages are C++ and Java while examples of object-based languages used include VBScript and JavaScript.

- *Compilation method:* Interpreted vs. Compiled. (It is well known that compiled applications are generally considered more difficult to write but they tend to be faster. On the other hand, an interpreted program can be written and tested interactively but they usually run slower.) Examples of compiled languages used on the Web are C and Java while Perl, JavaScript, WBScript and Unix Shell script are included in the interpreted category.

- *Data Typing*: Strong vs. Loose. Strong data typing which requires the programmer to define the type used helps to improve program performance and reliability but requires more effort from the programmer. Loose (weak) data typing, on the other hand, provides automatic detection of the type of data being used but can cause programming errors when used carelessly and sometimes the finished program is less efficient. Examples of programming languages require strong data typing are C, C++ and Java while JavaScript, VBScript, Unix Shell scripts and Perl belong to the loose (weak) data-typing category.

It is hard to compare programming languages. In essence it can be said that it is not the programming languages but the programmers can be classified as good or bad.

For choosing the language of interest the following general criteria can be considered:

1. Criteria of the *language* itself. It includes
   - Programming efficiency (maintainability, expressiveness, writability, readability)
   - Simplicity (generality, orthogonality, uniformity)
   - Extensibility
   - Intended purpose
2. *Compiler* is also a very important factor when choosing programming languages. The compiler criteria include:
   - Efficiency (translation, execution, optimization)
   - Reliability
   - Portability (machine-independence)
3. The available *support tools* are also very important. As:
   - Compiler
   - Debugger
   - Documentation
   - Libraries
   - Integrated development environment

3. Web programming languages

To give a more comprehensive basis for comparison, selected features of the most commonly used Web programming languages are summarised.

3.1. Perl

Perl [2] [3] [4] is one of the most widely used language for Web applications. Originally it was developed for the Unix operating system but now it is available on all major operating systems including Windows, Macintosh, DOS, and OS/2.

Main reasons of Perl's popularity:

- Good *string management* capabilities
- Very many *free programs* and scripts available on the Internet
- *Interpreted* language (nearly identical interpreter on all platforms)
- *Easy to learn* (steep learning curve)

3.2. Unix Shell

On Unix servers it is natural using a Shell language to program Web-based applications. Unix Shells (the three most important ones are the sh, Korn and csh versions) provide for control via Shell scripts (Scripts are processed by the Unix operating system by a text-based command-line processor).

The most common Unix Shell used on Web servers is a (public domain) variant of sh called *bash.* Someone can assign variables, manipulate text, compare values, perform loops and activate programs on the server. (Functions from built-in Unix commands and utilities can also be used).

There are *negative features* as well:

- Slow *speed* (especially at complex programs)
- Strict *syntax*
- Programming *style* is moving to other environments therefore the available free support (e.g. example script) is less on the Web.

For quick and not very sophisticated application a Shell program is a perfect way because it is easy

- to *learn*
- to *write*
- to *test* (even interactively).

3.3. C and C++

C [5] is widely available and can be found on every modern platforms (Unix, Windows, MS DOS, OS/2, Macintosh). It is an ANSI standard language, which means almost platform-independent code.

C is favoured because it offers great

- *speed*
- *power*
- *flexibility*
- *portability.*

C provides a large *library* of routines for a variety of tasks (e.g. text manipulation, arithmetic, files reading and writing). Of central importance is the C *compiler,* but different commercial and non-commercial compilers are available.

C++, the language family *object-oriented* offspring has all the advantages and features of object oriented languages but it is more difficult to write programs in C++ than in C. To support program development some C++ variants as Visual C++ and Borland C++ provide an *IDE* (Integrated Development Environment) that integrates the compiler with an editor, an environment to compile a program directly from the editing.

3.4. Java

Java [6] is a *network-centric* computing platform. There are two forms of Java applications. Java *applets* [7][8] are small programs. They are embedded in Web pages and downloaded to and run within the browser on a client machine. (Similarly, *Active X* components are applets written for special platforms). In contrast, *Java applications* are full-fledged programs installed locally on a machine.

Java is usually referred as a portable or *cross-platform* language. This key feature is based on two key components, i.e. *bytecodes* and the *Java VM* (Virtual Machine). VM is a software-based machine i.e. just a software program which has been ported to nearly all of the predominant computing platforms in use today. The VM processes bytecodes, a *platform-independent instruction set.* Bytecode is translated, – on-the-fly – into the platform-specific machine code instructions executed by the host machine. This is the price of the portability and therefore Java programs run more slowly than programs compiled directly to machine code. However, because the bytecodes were designed for efficient translation to machine code, Java is significantly faster than other interpreted languages. (Recently several so-called *just-in time* compilers are available to compile the bytecode to platform-specific machine code). Java is an object-oriented language that is syntactically similar to C++ and provides the language support of usual object-oriented languages.

The *Java Chip* chipset is an alternative implementation of the virtual machine to allow Java programs to be run in small systems such as electronic devices. Typical examples are the Personal Java and Embedded Java platforms for consumer devices use in every day life (e.g. set-top boxes to connect to and download contents).

3.5. Visual Basic (VB)

Microsoft Visual Basic also can create Web-based applications. Its strength is in creating *Window-based* user interfaces, like dialogue boxes, quickly. Using native code compilation and visual creation of open ActiveX components (canned, industry standard elements) it can create high-performance applications. With up-to-date add-in tools VB applications can be converted to Java. (Note that while VBScript has "borrowed" the VB initials it lacks many of Visual Basic powerful features, e.g. file manipulation commands).

3.6. Eiffel

Eiffel is a pure object oriented language. It also includes a comprehensive approach to software construction: a method and an environment (ISE Eiffel). The *ISE Eiffel* compiler generates an internal form (bytecode) which can be interpreted directly but it can also be translated into other forms (e.g. the bytecode can be optimized and translated into C to take advantage of the presence of a C compiler).

3.7. Scripts

Scripts are small programs that interact with the browser and the HTML content of a page (In HTML the SCRIPT tag is used to enclose runable script, which is interpreted as executable code).

JavaScript [9] and *VBScript* [10], the two main solutions are strictly similar and supported by the Netscape Navigator and the Internet Explorer, respectively.

Scripts can be used for
- *tailoring* pages
- *mailing* interactive pages
- computer-aided *instructions*
- special *effects.*

JavaScript is based largely on C syntax while VBScript is based largely on the syntax of Visual Basic. Both are *object-based* and *interpreted* at execution. Objects (e.g. a document) are composed of so called *properties,* which are handled by so called *methods.* [9][10].

On the Web very many script *repositories* [11] [12] can be found where countless useful programs and scripts can be downloaded mainly either as *freeware* or *shareware.* Different programming languages can be combined to take advantage of the best each can offer.

4. Comparison

It is impossible to find or define a single comparison structure for comparing Web programming languages.

Therefore, we show different methods and point of views to demonstrate different comparison strategies.

4.1. Comparing a *single language's advantages and disadvantages,* its tradeoffs.
Example: Perl language tradeoffs. (Table 1)

Table 1: Perl language advantages and disadvantages.

Advantages	Disadvantages
Text manipulation (exceptional)	Syntax (overly flexible)
File management (exceptional)	Privacy (unenforced)
Compact code (for complex operations)	Exception mechanism (no built-in toy-catch-finally)
CGI scripting (easy)	Remote code execution (only via Penguin)
Data type management (convenient)	

4.2. Another possible consideration at comparison is the general *aspects of the user.* The data of Table 2 are based on this view.

Table 2: User aspects of languages' features

Attribute	Java	C++	Visual Basic
Security	A	F	F
Built-in network support	B	F	F
Language safety	A	D	C
Runtime performance	C/A	B	C
Compile time	B	D	B
Syntactical simplicity	B	B	B
Source portability	A	C	F
Industry acceptance	B	A	B

A : excellent   C : good   F : none
B : very good   D : moderate

4.3. It is worthwhile to compare the main features of *Object-Oriented languages* (Table 3).

Table 3: Comparison of Object-Oriented (OO) languages

Attribute	C++	Java	Eiffel	Smalltalk
Static typing	statically but support for "casts"	mostly statically but dynamic at generic structures	Statically	dynamically
Compilation technology	compiled	interpretation and "on-the-fly" compilation	Combination of interpretation and compilation	historically interpreter-based, partial compilation at current versions
Multiple inheritance	multiple but problematic	single with multiple interface facility	Multiple	single
Object orientation	hybrid	Pure	Pure	pure
Openness and interoperability	good with C	native methods	support for integration with C and C++	no standardized interface

4.4. Languages can be evaluated based on *pedagogic suitability* criteria, too. Without really judging the languages based on this kind of criteria here we consider only a few possible point of views.

Stability and standard: existence of an ANSI and/or ISO language standard and degree of not changing.

Simplicity: combination of the simplicity, regularity and brevity of the syntax.

Pedagogic support: availability and quality of support resources.

Cross curriculum: suitability of the language for use in different areas of the curriculum.

Learning curve: amount of syntactic overhead involved in producing an initial artifact.

Conceptual expressability: simplicity and regularity of the cognitive connections between syntax and semantics.

Hybrid or pure: support both procedural/imperative and object oriented styles of development or just for object oriented style.

4.5. Languages can be judged based on the *suitability to different tasks* (Table 4)

Table 4: Suitability of languages to different tasks.

Task	Unix Shell	C/C++	Perl	Java	Script
Form processing (mailing)	C	C	A	C	D
Form processing (shopping)	D	C	A	D	C
Database search (small)	C	B	B	C	C
Database search (large)	D	A	C	D	D
Visit counter (text)	B	B	B	D	D
Visit counter (graphic)	D	A	B	D	D
Dynamic page	B	C	A	B	A

A : excellent C : good B : very good D : moderate

The usual *programming* point-of views also should be considered, i. e.

- *Execution speed* (compiled language vs. interpreted language)
- *Ease of programming* (learning curve and matching complexity of language and task)
- *Support* (Built-in integrated support vs. third-party libraries)

5. Conclusions

Perhaps the largest obstacle facing a Web programmer is choosing the right language for the job. Not only the language must fit the job but it also must fit the programmer.

We have reviewed the major language choices available to Web programmers. The strength and weaknesses of these languages, the jobs they are best suited for were considered. The pros and cons of Web programming were discussed comparing the languages based on the criteria of the general aspects, user's points of views, pedagogic suitability.

The Web represents a new medium of information exchange based on globally networked, distributed and linked information space. The Web has an enormous impact on software development and distributions as well. However, these topics are beyond the scope of this paper.

References

[1] William E. Weinman: The CGI Book, New Riders Publishing, 1996

[2] Eric F. Johnson: Cross-platform Perl, IDG Books,1996

[3] Thomas Boutell: CGI Programming in C & Perl, Addison-Wesley Pub. 1996

[4] Achive Perl CGI Scripts, http://www.middlebury.edu/~otisg/

[5] Mark Felton: CGI: Internet Programming with C and C++, Prentice Hall, 1997

[6] Java, http://192.147.157.51:8000/galaxy/

[7] David Gulbrausen et al.: Creating Web Applets With Java, Sams Publishing, 1996

[8] Java Applets, http://www.javasoft.com/applets/

[9] Jason J. Manger: JavaScript Essential, Osborne McGraw-Hill, 1996

[10] Christopher J. Goddard et al.: VBscript Master's Handbook, Prima Publishing,1996

[11] Central Script Repository, http://www.selah.net/cgi.html/

[12] Yahoo! Internet-related pages, http://www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/

[13] CORBA 2.1 Specification http://www.omg.org/corba/corbiisp.hm/