

# PROGRAMMING LANGUAGES

## The State of the Art

### Contents:

- The Problem <=> The Future
- Taxonomy of Languages
  - Assembly, System, Application
  - Run-Time Environment
- Design Issues
- Editors and IDEs
- Data Structures
- GUIs

### The Problem

VFP works fine, but within 10 years, for one reason or another, NONE of us will be writing applications in it. So what else is out there? Wikipedia lists several hundred programming languages, 25 beginning with the letter 'F' alone. We won't cover them all!

### Taxonomy of Languages

#### Ancient History:

- Assembly Languages -> System Languages
- Algol, FORTRAN, COBOL, PL/I
- (generated machine code, no 'run-time' package, limited portability)

#### Recent History:

- System Languages: Algol, C, C++, Java
- Application Languages: Basic, Lisp, VFP
- Run-time libraries: C, C++, Basic, Pascal, VFP
- 'Virtual machines': JavaScript, C#, VB, Ruby, etc

#### Current:

- Application Languages: Javascript, Perl, PHP, Python, Ruby, Tcl

#### Portable run-time environments:

- JVM - Java Virtual Machine, by Sun Microsystems (open source)
- .NET - Microsoft (proprietary)

#### Terminology:

- 'Application Language' = 'Scripting Language' = 'Dynamic Language'
- 'Virtual Machine' = 'Run-Time Environment'
- Microsoft usually invents its own terminology!

## The VM Wars:

Name	VFP	JAVA (Open Source)	.NET (Microsoft)
Platform	Windows	Unix, Windows, OS X	Windows only
Run-time VM	VFP dll (C++)	JVM (JRE) (C++, Assembler)	.NET (C++)
System Language	VFP	Java	C++
Application languages	VFP	Javascript, Python, Ruby, Perl, PHP, Tcl	C#, VB, Visual J#, JScript, VBScript, IronPython, IronRuby
IDEs	VFP	JavaBeans, Eclipse, JBuilder, Komodo	Visual Studio, SharpDevelop

## Where We Are:

Ousterhout's Dichotomy:

System vs Scripting languages

Scripting aims to produce software which provides services to the user (e.g. web pages) whereas systems programming aims to produce software which provides services to the computer hardware (e.g. disk defragmenter).

<http://home.pacbell.net/ouster/scripting.html>

PHP, JavaScript, Ruby, Perl, Python and Tcl Today - The State of the Scripting Universe

<http://www.cio.com/article/446829>

Scripting Languages are descendants of JCL and .BAT files

Javascript is the most used language today. (over 70% of developers!)

Usage started in browsers and Web apps, but is spreading rapidly to all applications.

## Current Scripting Languages:

**Javascript** (developed by Netscape)

Originally designed for client-side Web scripting. 'ECMAScript' is int'l standard.

SpiderMonkey is a JavaScript engine written in C. It is used in Mozilla Firefox.

Rhino is an open-source implementation of JavaScript written entirely in Java. It is typically embedded into Java applications to provide scripting to end users.

JScript is Microsoft version, using .NET.

**Perl** 'Parable of the Pearl' from gospel of Matthew

Started 1987 as Unix scripting language for reports. Derived broadly from C.  
"Perl is a language for getting your job done." No official specification.  
Implemented as a C interpreter on Unix and Windows.

**PHP** 'Personal Home Page'

Free, cross-platform interpreter. Runs in Web server, creating dynamic Web pages.  
Often embedded in HTML. Evolved from CGI ca 1995.  
From PHP4 (2000) compiles byte code for Zend engine.

**Python** 'Monte Python's Flying Circus'

Orig late '80s by Guido Van Rossum (BDFL). Version 3.0 in 2008.  
'CPython' runs on Python Virtual Machine (written in C). Windows, Unix, Mac.  
Jython compiles to the JVM, can use Java class libraries.  
IronPython Python for .NET runtime. (i.e., scripting for C# applications)

**Ruby** (birthstone for July)

Syntax from Perl, semantics from Smalltalk. Heavily object oriented.  
Up to V 1.8, runs on MRI interpreter (slow). V 1.9 will implement YARV  
(Yet Another Runtime Virtual machine).  
Ruby on Rails Open source Web app framework for Ruby.  
DHH (David Heinemeier Hanson) and the Rails trademark fuss.  
Iron Ruby Ruby for .NET (in process)  
JRuby Ruby for JRE, embeddable in any Java application.

**Ajax** (Asynchronous Javascript And XML)

Mainly for Web apps that run in browsers.

**Tcl** (originally from "Tool Command Language")

Created by John Ousterhout in 1990.  
Used extensively on embedded systems platforms, also used for CGI scripting.  
The combination of Tcl and the GUI toolkit Tk is referred to as **Tcl/Tk**.

**Visual J#** (Microsoft, 1995)

Java for .NET, no longer in development as of 2009.

**VBScript** (Microsoft, 1996)

Shipped with Windows 98 on. Used in Internet Explorer, ASP, and WSH.  
Now in ASP.NET, being replaced by JScript.

**JScript** (Microsoft, ....)

JavaScript for IE and .NET.

## Language Design Issues:

Variable Typing - Static or Dynamic? Strong or Weak?

(Steve Yegge on Who Cares?)

Static vs Dynamic

Java and C vs VFP and Python

Strong vs Weak:

a = '12'; b = 5; c = a + b      Does this work?

Scope - Static or Dynamic?

Static (lexical) scoping - in the code      Algol, C

Dynamic scoping - on the stack      VFP

"Static scoping also makes it much easier to make modular code and reason about it, since its binding structure can be understood in isolation. In contrast, dynamic scope forces the programmer to anticipate all possible dynamic contexts in which the module's code may be invoked. "

## IDEs

<[en.wikipedia.org/wiki/Comparison\\_of\\_integrated\\_development\\_environments](http://en.wikipedia.org/wiki/Comparison_of_integrated_development_environments)>

Editors are no longer language specific. Most have syntax coloring, auto-indent, and a compiler interface for multiple languages.

Full IDEs typically include an Editor, compiler interface, GUI toolkit, debugger, and other tools.

IDEs add convenience, but also add to the learning curve.

**Easy Eclipse** (free) is released for Expert Java, Desktop Java, Server Java, Mobile Java, Plugin Warrior, LAMP, PHP, Ruby/Ruby on Rails, Python and C/C++.

## Databases

VFP:            Native structure, others thru ODBC.

JVM:           thru JDBC.

.NET:          thru ODBC

ODBC:          Microsoft, 1992

Driver manager + database drivers. Called via SQL.

JDBC:          Java version of ODBC, included in JVM.

JDBC-ODBC Bridge: a JDBC driver which calls an ODBC driver to connect to a target database.

## GUIs

VFP: Native, included in IDE

JVM: several, depending on language

Swing, Wx, JFormDesigner, JBuilder, Eclipse Visual Editor

.NET: Windows Forms Designer

Unix: Gnome, KDE, X?

Windows: Windows Forms Designer, (VFP Forms Designer, VB Forms Designer)

Java: Swing, X, FormLayout

Python: Tkinter, WxPython, XGUI, MacPython, IronPython Studio (28 names!)

Where are GUI definitions stored? Code (Swing, Wx, NetBeans) or metadata (VFP)

## SO WHAT?

**The choice is not between Windows and Unix, it's between .NET and JVM.**

**The trend for applications is toward Dynamic Languages.**

System languages are too complex and take too long.

**Database interface will be SQL - period!**

1. Do you want to stay in Microsoft? Do you have a choice?  
Or would you prefer to be cross-platform?
2. Would you like to use Open Source tools?  
(Who pays for your tools, you or your employer?)
3. Would you prefer a full IDE or do you like to mix and match?  
Each language offers different options.

My take: If you're an employee of a stable company which is committed to Microsoft platforms, why fight it - .NET is your environment. If you're an independent, or likely to be changing jobs, then committing yourself to Microsoft looks a lot riskier than the more portable choices.