

# What is the worst programming language you ever worked with?

## [closed]

Asked 15 years, 6 months ago    Modified 2 years, 5 months ago

Viewed 176k times

44  
votes



As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, [visit the help center](#) for guidance.

Closed 15 years ago.



**Locked.** This question and its answers are [locked](#) because the question is off-topic but has historical significance. It is not currently accepting new answers or interactions.

If you have an interesting story to share, **please post an answer**, but do not abuse this question for bashing a language.

We are programmers, and our primary tool is the programming language we use.

While there is a lot of discussion about the best one, **I'd like to hear your stories about the worst programming languages you ever worked with** and I'd like to know exactly what annoyed you.

I'd like to collect this stories partly to avoid common pitfalls while designing a language (especially a DSL) and partly to avoid quirky languages in the future in general.

---

*This question is **not subjective**. If a language supports only single character identifiers (see [my own answer](#)) this is **bad in a non-debatable way**.*

---

## EDIT

Some people have raised concerns that this question attracts trolls. Wading through all your answers made one thing clear. The large majority of answers is appropriate, useful and well written.

## UPDATE 2009-07-01 19:15 GMT

The language overview is now complete, covering **103 different languages** from 102 answers. I decided to be lax about what counts as a programming language and included anything reasonable. Thank you [David](#) for your comments on this.

Here are all programming languages covered so far  
(alphabetical order, linked with answer, new entries in bold):

[ABAP](#), [all 20th century languages](#), [all drag and drop languages](#), [all proprietary languages](#), [APF](#), [APL](#) <sup>(1)</sup>, [AS400](#), [Authorware](#), [Autohotkey](#), [BancaStar](#), [BASIC](#), [Bourne Shell](#), [Brainfuck](#), **[C++](#)**, [Centura Team Developer](#), [Cobol](#) <sup>(1)</sup>, [Cold Fusion](#), [Coldfusion](#), [CRM114](#), [Crystal Syntax](#), [CSS](#), [Dataflex 2.3](#), [DB/c DX](#), [dbase II](#), [DCL](#), [Delphi IDE](#), [Doors](#), [DXL](#), [DOS batch](#) <sup>(1)</sup>, [Excel Macro language](#), [FileMaker](#), [FOCUS](#), [Forth](#), [FORTRAN](#), [FORTRAN 77](#), [HTML](#), [Illustra web blade](#), **[Informix 4th Generation Language](#)**, [Informix Universal Server web blade](#), [INTERCAL](#), [Java](#), [JavaScript](#) <sup>(1)</sup>, [JCL](#) <sup>(1)</sup>, [karol](#), [LabTalk](#), [Labview](#), [Lingo](#), [LISP](#), [Logo](#), [LOLCODE](#), [LotusScript](#), [m4](#), [Magic II](#), [Makefiles](#), [MapBasic](#), [MaxScript](#), [Meditech Magic](#), [MEL](#), [mIRC Script](#), [MS Access](#), [MUMPS](#), [Oberon](#), [object extensions to C](#), [Objective-C](#), [OPS5](#), [Oz](#), [Perl](#) <sup>(1)</sup>, [PHP](#), **[PL/SQL](#)**, [PowerDynamo](#), [PROGRESS 4GL](#), [prova](#), [PS-FOCUS](#), [Python](#), [Regular Expressions](#), **[RPG](#)**, [RPG II](#), [Scheme](#), [ScriptMaker](#), [sendmail.conf](#), [Smalltalk](#), [Smalltalk](#), [SNOBOL](#), [SpeedScript](#), [Sybase PowerBuilder](#), [Symbian C++](#), [System RPL](#), [TCL](#), [TECO](#), [The Visual Software Environment](#), [Tiny praat](#), [TransCAD](#), [troff](#), [uBasic](#), [VB6](#) <sup>(1)</sup>, [VBScript](#) <sup>(1)</sup>, [VDF4](#), **[Vimscript](#)**, [Visual Basic](#) <sup>(1)</sup>, [Visual C++](#), [Visual Foxpro](#), [VSE](#), [Webspeed](#), [XSLT](#)

The answers covering 80386 assembler, VB6 and VBScript have been removed.

Share

edited May 23, 2017 at 12:10

community wiki

59 revs, 13 users 69%

Ludwig Weinzierl

---

2 Also check out this question:

[stackoverflow.com/questions/282329/...](https://stackoverflow.com/questions/282329/...)

– Cameron MacFarland Jun 7, 2009 at 15:24

---

42 I'm shocked to see this re-opened. Stack Overflow is not a discussion site, and this question is exceedingly subjective. While we certainly might agree on certain characteristics common to "bad" languages (such as the single-char identifier aspect that Ludwig points out), there's far more potential for the sort of bitter bashing and idle reminiscing seen in Emil H's VB answer. – Shog9 Jun 7, 2009 at 16:02

---

I'm locking this due to a **lot** of negative feedback from lots of different users. If you feel strongly that we should keep this alive, please take the discussion to meta.stackoverflow.com

– Marc Gravell Jul 1, 2009 at 20:58

---

Comments disabled on deleted / locked posts / reviews

100 Answers

Sorted by:

Highest score (default)



1

2

3

4

Next

212 PHP (In no particular order)



- **Inconsistent function names and argument orders**
  - Because there are a zillion functions, each one of which seems to use a different naming convention and argument order. "Lets see... is it foo\_bar or foobar or fooBar... and is it [needle](#), [haystack](#) or [haystack, needle](#)?" The PHP string functions are a perfect example of this. Half of them use [str\\_foo](#) and the other half use [strfoo](#).
- **Non-standard date format characters**
  - Take `j` for example
    - In [UNIX](#) (which, by the way, is what everyone else uses as a guide for date string formats) `%j` returns the day of the year with leading zeros.
    - In PHP's [date](#) function `j` returns the day of the month without leading zeros.
- **Still No Support for Apache 2.0 MPM**
  - [It's not recommended.](#)
  - Why isn't this supported? "When you make the underlying framework more complex by not having completely separate execution threads, completely separate memory segments and a strong sandbox for each request to play in, feet of clay are introduced into PHP's system." [Link](#) So... it's not supported 'cause it makes things harder? 'Cause only the things that are easy are worth doing right? (To be fair, as [Emil H](#) pointed out, this is generally attributed to bad 3rd-party

libs not being thread-safe, whereas the core of PHP is.)

- **No native Unicode support**

- [Native Unicode support is slated for PHP6](#)
- I'm sure glad that we haven't lived in a global environment where we might have need to speak to people in other languages for the past, oh [18 years](#). *Oh wait*. (To be fair, the fact that *everything* doesn't use Unicode in this day and age really annoys me. My point is I shouldn't have to do any extra work to make Unicode happen. This isn't *only* a PHP problem.)

I have other beefs with the language. These are just some. Jeff Atwood has an [old post](#) about why PHP sucks. He also says it doesn't matter. I don't agree but there we are.

Share

[edited Nov 17, 2023 at 20:38](#)

community wiki  
[7 revs, 2 users 96%](#)  
[baudtack](#)

---

101 XSLT.

votes



- XSLT is baffling, to begin with. The metaphor is completely different from anything else I know.



- The thing was designed by a committee so deep in angle brackets that it comes off as a bizarre frankenstein.
- The weird incantations required to specify the output format.
- The built-in, invisible rules.
- The odd bolt-on stuff, like scripts.
- The dependency on XPath.
- The tools support has been pretty slim, until lately. Debugging XSLT in the early days was an exercise in navigating in complete darkness. The tools change that but, still XSLT tops my list.

XSLT is weird enough that most people just ignore it. If you must use it, you need an XSLT Shaman to give you the magic incantations to make things go.

Share

[edited Jun 9, 2009 at 12:17](#)

community wiki

[4 revs](#)

[Cheeso](#)

---

I agree. Started using oXygen 10. Changes everything  
– user34411 Jun 9, 2009 at 3:30

---

- 2 XSLT is powerful enough to get some very complex stuff done, but it's done entirely without your knowledge of how it worked to begin with. That, and it's pretty processor heavy

thanks to frequent recursion. – [Robert K](#) Jun 10, 2009 at 21:38

---

21 XSLT is a sort of functional programming language. I don't find it to be that painful. – [Brian](#) Jun 23, 2009 at 16:42


---

1 I am an XSLT shaman :) More seriously. almost all bullets are actually advantages. 1. "The metaphor is completely different from anything else I know". If you're a baby and your parents listened to this you'd still be a baby. 2. "The thing was designed by a committee so deep in angle brackets that it comes off as a bizarre frankenstein". Then talk with a shrink about this psychological problem. – [Dimitre Novatchev](#) Apr 24, 2010 at 19:12

---

5 3. "The odd bolt-on stuff, like scripts". There is no such thing in XSLT. 4. "The dependency on XPath". C'mon, this is XSLT's greatest strength that makes it so powerful. Have you ever heard that XPath is not designed as a standalone language and must be hosted? 5. "The tools support has been pretty slim, until lately. Debugging XSLT in the early days was an exercise in navigating in complete darkness. The tools change that but, still XSLT tops my list". You're missing the state of the art here: Oxygen, VS 2005 +, the 10 year old XSelector. And *my* XPathVisualizer from year 2000 :) – [Dimitre Novatchev](#) Apr 24, 2010 at 19:16

---

96  
votes  


DOS Batch files. Not sure if this qualifies as programming language at all. It's not that you can't solve your problems, but if you are used to `bash` ...



Just my two cents.

Share

[edited Jan 3, 2012 at 21:40](#)



community wiki  
14 revs, 3 users 92%  
bbuser

---

19 variables + if + goto == Turing complete – [guns](#) Jun 7, 2009 at 19:38

---

I come from a windows background, but after using Ubuntu for a month... OMG. I keep trying to "ls" but it doesn't work :( Anyway, I think bash was sort of ill-conceived too, even if it is more powerful. "esac", "fi"? Seriously? What the hell is up with that? – [mpen](#) Jul 6, 2009 at 5:06

---

Mark: The designer of the Bourne shell probably thought of ALGOL as a nice language :) – [Joey](#) Oct 1, 2009 at 17:28

---

2 Yes, if I had to pick a language that maximises `installedBaseSize * terribleness`, it would be DOS batch files. *Of course* you use the `FOR` command to get a file's full path! – [j\\_random\\_hacker](#) Jun 1, 2010 at 13:54

---

+1 I completelt agree. I just spent 7 hours tryingg to monkeypunch some functionality out of a batch script to launch Portable msysgit. The only thing worse than batch scripting... Batch scripting mixed with bash scripting. I know where Bourne scripting got the 'bash' name from because after I was finished I felt like bashing my head against a wall. Why can't we kill scripting languages once and for all, we have python now. – [Evan Plaice](#) Jan 25, 2012 at 4:59

---

---

91 Not sure if its a true language, but I hate Makefiles.

votes



Makefiles have meaningful differences between space and TAB, so even if two lines appear identical, they do not run the same.

Make also relies on a complex set of implicit rules for many languages, which are difficult to learn, but then are frequently overridden by the make file.

A Makefile system is typically spread over many, many files, across many directories. With virtually no scoping or abstraction, a change to a make file several directories away can prevent my source from building. Yet the error message is invariably a compilation error, not a meaningful error about make, or the makefiles.

Any environment I've worked in that uses makefiles successfully has a full-time Make expert. And all this to shave a few minutes off compilation??

Share

[edited Nov 25, 2011 at 17:50](#)

community wiki

[2 revs](#)

[abelenky](#)

---

11 "Any environment [...] that uses makefiles successfully has a full-time Make expert" See Recursive Make Considered Harmful to fix this. Make is dense, but does not need to be *that* awful. The scary part is that the WS sensitivity was indentified as a bug when there were dozens of users, and left in *because* there were dozens of users... – [dmckee](#) --- [ex-moderator kitten](#)  
Jun 7, 2009 at 20:19

---

7 I'll take make over ant any day of the week. Use a decent editor that is make-aware and the tab thing is mostly a non-

issue (for example, emacs does a decent job of catching whitespace problems) – [Bryan Oakley](#) Jun 8, 2009 at 3:19

---

5 Martin York: Recognize that most programmers want to PROGRAM, not spend significant amounts of time and knowledge on a build-system. Most of us expect the build-system to "just work". After all, we're not asking much of it, just compile and link together some source-files; little more. Make massively overshoot its target niche. – [abelenky](#) Jun 8, 2009 at 19:31

---

5 you think make is bad, pair it with automake, and you've got a diabolical global badness I refuse to attempt to debug. – [Kent Fredric](#) Jun 9, 2009 at 3:04

---

4 make is so problematic that there are a gazillion tools out there whose only purpose is to create and maintain makefiles: automake, qmake, cmake, Jam, etc. – [Adam Rosenfield](#) Jun 28, 2009 at 21:51

---

---

79 votes

The worse language I've ever seen come from the tool praat, which is a good audio analysis tool. It does a pretty good job until you use the script language. *sigh* bad memories.



## Tiny praat script tutorial for beginners

•

### Function call

We've listed at least 3 different function calling syntax :

- **The regular one**

```
string = selected("Strings")
```

Nothing special here, you assign to the variable string the result of the selected function. Not really scary... yet.

- **The "I'm invoking some GUI command with parameters"**

```
Create Strings as file list... liste
```

```
'path$'/'type$'
```

As you can see, the function name start at "Create" and finish with the "...". The command "Create Strings as file list" is the text displayed on a button or a menu (I'm to scared to check) on praat. This command take 2 parameters liste and an expression. I'm going to look deeper in the expression `'path$'/'type$'`

Hmm. Yep. No spaces. If spaces were introduced, it would be separate arguments. As you can imagine, parenthesis don't work. At this point of the description I would like to point out the suffix of the variable names. I won't develop it in this paragraph, I'm just teasing.

- **The "Oh, but I want to get the result of the GUI command in my variable"**

```
nolifftt = Get number of strings
```

Yes we can see a pattern here, long and weird function name, it must be a GUI calling. But there's no '...' so no parameters. I don't want to see what the parser looks like.

- 

## The incredible type system

(AKA Haskell and OCaml, praat is coming to you)

- **Simple natives types**

```
windowname$ = left$(line$, length(line$)-4)
```

So, what's going on there? It's now time to look at the convention and types of expression, so here we got :

- `left$ :: (String, Int) -> String`
- `length :: (String) -> Int`
- `windowname$ :: String`
- `line$ :: String`

As you can see, variable name and function names are suffixed with their type or return type. If their suffix is a '\$', then it return a string or is a string. If there is nothing it's a number. I can see the point of prefixing the type to a variable to ease implementation, but to suffix, no sorry, I can't

- **Array type**

To show the array type, let me introduce a 'tiny' loop :

```
for i from 1 to 4
  Select... time time
  bandwidth'i'$ = Get bandwidth... i
  forhertz'i'$ = Get formant... i
endfor
```

We got `i` which is a number and... (no it's not a function)

```
bandwidth'i'$
```

What it does is create string variables : `bandwidth1$`, `bandwidth2$`, `bandwidth3$`, `bandwidth4$` and give them values. As you can expect, you can't create two dimensional array this way, you must do something like that : `band2D__'i'__'j'$`

- **The special string invocation**

```
outline$ =  
"'time'@F'i':'forhertznum'Hz, 'bandnum'Hz,  
'spec' 'newline$'  
outline$ >> 'outfile$'
```

Strings are weirdly (at least) handled in the language. the `"` is used to call the value of a variable inside the global `""` string. This is `_weird_`. It goes against all the convention built into many languages from bash to PHP passing by the powershell. And look, it even got redirection. Don't be fooled, it doesn't work like in your beloved shell. No you have to get the variable value with the `"`

- **Da Wonderderderful execution model**

I'm going to put the final touch to this wonderderderful presentation by talking to you about the execution model. So as in every procedural

languages you got instruction executed from top to bottom, there is the variables and the praat GUI. That is you code everything on the praat gui, you invoke commands written on menu/buttons.

The main window of praat contain a list of items which can be :

- files
- list of files (created by a function with a wonderderful long long name)
- Spectrogramm
- Strings (don't ask)

So if you want to perform operation on a given file, you must select the file in the list programmatically and then push the different buttons to take some actions. If you wanted to pass parameters to a GUI action, you have to follow the GUI layout of the form for your arguments, for example "To Spectrogram... 0.005

5000 0.002 20 Gaussian

" is like that because it follows this layout:

Needless to say, my nightmares are filled with praat scripts dancing around me and shouting "DEBUG MEEEE!!".

More information at the [praat](#) site, under the well-named section "easy programmable scripting language"

Share

edited Jul 16, 2022 at 15:36

- 
- 1 the\_drow > Oh yes, and I didn't even speak about the GUI description part, to allow the creation of "easy to use" forms. There's also the how to fill the "Very Long Function Name..." parameters which had to respect the layout of the form "from left to right, top to bottom". – [Raoul Supercopter](#) Jun 7, 2009 at 19:34
- 
- 1 +1 Wow ! In the description on the "Controlling the user" (?) command, if this : "This example uses several tricks. A useful one is seen with number\_of\_channels: this is at the same time the value that is passed to optionMenu (and therefore determines the setting of the "Number of channels" menu when the window appears) and the name of the variable in which the user's chosen value of "Number of channels" is stored (because "number\_of\_channels" is what you get by replacing the spaces in "Number of channels" with underscores and turning its first letter to lower case." – [Sylvain Rodrigue](#) Jun 7, 2009 at 20:25
- 
- 1 Actually appending \$ to variables and functions which are/return strings was in Spectrum Basic. Assembly subroutines with CHR\$(), anyone? – [Anton Tykhyy](#) Jun 7, 2009 at 20:29
- 
- 1 This reminds me a little of VBA actually. VBA shares a lot of its problems with your language, but its real problem comes from its mainstreamness. – [Alexandre C.](#) Aug 1, 2011 at 11:24
- 

---

74 Well since this question refuses to die and since the OP did prod me into answering...

votes







I humbly proffer for your consideration **Authorware** (AW) as the worst language it is possible to create. (*n.b. I'm going off recollection here, it's been ~6 years since I used AW, which of course means there's a number of awful things I can't even remember*)

[the horror, the horror](http://img.brothersoft.com/screenshots/softimage/a/adobe_authorware-67096-1.jpeg)

[http://img.brothersoft.com/screenshots/softimage/a/adobe\\_authorware-67096-1.jpeg](http://img.brothersoft.com/screenshots/softimage/a/adobe_authorware-67096-1.jpeg)

Let's start with the fact that it's a **Macromedia product** (-10 points), a **proprietary language** (-50 more) primarily intended for creating e-learning software and moreover software that could be created by *non-programmers and programmers alike* implemented as an iconic language AND a text language (-100).

Now if that last statement didn't scare you then you haven't had to fix WYSIWYG generated code before (hello Dreamweaver and Frontpage devs!), but the salient point is that AW had a library of about 12 or so elements which could be dragged into a flow. Like "Page" elements, Animations, IFELSE, and **GOTO** (-100). Of course removing objects from the flow created any number of broken connections and artifacts which the IDE had variable levels of success coping with. Naturally the built in **wizards** (-10) were a major source of these.

Fortunately you could always step into a code view, and eventually you'd have to because with a limited set of iconic elements some things just weren't possible otherwise. The language itself was **based on TUTOR** (-50) - a candidate

for worst language itself if only it had the ambition and scope to reach the depths AW would strive for - about which wikipedia says:

...the TUTOR language was not easy to learn. In fact, it was even suggested that several years of experience with the language would be required before programmers could build programs worth keeping.

An excellent foundation then, which was built upon in the years before the rise of the internet with exactly nothing. Absolutely **no form of data structure beyond an array** (-100), certainly **no sugar** (real men don't use switch statements?) (-10), and a large splash of **syntactic vinegar** ("--" was the comment indicator so no decrement operator for you!) (-10). Language reference documentation was provided in paper or **zip file** formats (-100), but at least you had the support of the developer run usegroup and could quickly establish the solution to your problem was to use the DLL or SWF importing features of AW to enable you to do the actual coding in a real language.

AW was driven by a flow (with necessary PAUSE commands) and therefore has all the attendant problems of a **linear rather than event based** system (-50), and despite the outright marketing lies of the documentation it was **not object oriented** (-50) either. *All* code reuse was achieved through GOTO. **No scope, lots of globals** (-50).

It's not the language's fault directly, but obviously **no source control integration was possible**, and certainly no TDD, documentation generation or any other add-on tool you might like.

Of course Macromedia met the challenge of the internet head on with a stubborn refusal to engage for years, eventually producing the **buggy, hard to use, security nightmare which is Shockwave** (-100) to essentially serialise desktop versions of the software through a **required plugin** (-10). AS HTML rose so did AW stagnate, still persisting with it's shockwave delivery even in the face of IEEE SCORM *javascript* standards.

Ultimately after years of begging and promises Macromedia announced a radical new version of AW in development to address these issues, and a few years later offshored the development and then cancelled the project. Although of course **Macromedia are still selling it** (EVIL BONUS -500).

If anything else needs to be said, **this is the language which allows spaces in variable names** (-10000).

If you ever want to experience true pain, try reading somebody else's uncommented hungarian notation in a language which isn't case sensitive and allows variable name spaces.

---

*Total Annakata Arbitrary Score (AAS): -11300*

*Adjusted for personal experience: OutOfRangeException*

(apologies for length, but it was cathartic)

Share

[edited Jun 30, 2009 at 14:56](#)

community wiki

[2 revs](#)

[annakata](#)

---

Awesome, +1. I work with a proprietary language that I like very much for its sparseness and clarity. But the one convention it permits that can make code difficult to read is... spaces in variable names. – user447688 Jun 18, 2009 at 12:03

---

10 You can always tell how badly someone is scarred by how elegantly they write the language's epitaph – [Mike Robinson](#) Jun 30, 2009 at 13:44

---

1 Ah, TUTOR. I learned TUTOR in while waiting for my WATFOR IV Express jobs to complete on a 360/195 (?). Perhaps even more horrible/unforgivable was that I actually *taught* TUTOR at UICC for a year. The only upside was that at 2AM I had a CDC Dual Cyber 6600 with 64MB of *core memory* (check it out, youngsters: [en.wikipedia.org/wiki/Magnetic\\_core\\_memory](http://en.wikipedia.org/wiki/Magnetic_core_memory)) almost all to myself. And we had 512x512 graphics ... in 1973. And, yes, the language sucked planetoids -- giant, slime-dripping, radioactive planetoids. – [Peter Rowell](#) Dec 23, 2009 at 20:31

---

Authorware was just the successor to Authority which we actually used in high-school for a bit. In general 5GLs are just silly. – [Synetech](#) Feb 26, 2010 at 17:30

---

You know, I agree with that BUT I would love a language that allowed spaces in variable names. I have having to use camel case or underscores, very unpretty – anon271334 Mar 14, 2010 at 11:53

---

---

69

votes



Seriously: Perl. It's just a pain in the ass to code with for beginners and even for semi-professionals which work with perl on a daily basis. I can constantly see my colleagues struggle with the language, building the worst scripts, like 2000 lines with no regard of any well accepted coding standard. It's the worst mess i've ever seen in

programming. Now, you can always say, that those people are bad in coding (despite the fact that some of them have used perl for a lot of years, now), but the language just encourages all that freaking shit that makes me scream when i have to read a script by some other guy.

Share

[edited Jan 3, 2012 at 21:46](#)

community wiki

[2 revs, 2 users 77%](#)

[buster](#)

---

9 The problem is not that Perl is bad, it's that it's easy enough to use that your colleagues get themselves into serious trouble. I suspect they'd screw themselves over in any language.

– [David Thornley](#) Jun 8, 2009 at 16:02

---

7 @Andrei: I rank PHP ahead of Perl because it at least has a reasonable way of passing parameters to functions - I think having to deal with `$_[n]` (or doing repeated shifts) is crazy. I finally gave up on using Perl for new development when I wanted to pass 2 arrays to one function and had to spend the better part of a day reading & experimenting to do what I would do easily and in seconds in most other languages I've used.

– [PTBNL](#) Jun 8, 2009 at 16:26

---

5 @David: I disagree. I usually try hard to make my programs understandable by others, but when I wrote a 3000 line suite of Perl programs, I found that even I had great difficulty maintaining it myself if I hadn't looked at it in a while. I think Python is very easy to use, probably easier than Perl, but I don't see the problems with it that I do with Perl ... or many people leaving Python for Perl, although I know several (including myself) who've dropped Perl for Python. – [PTBNL](#)  
Jun 8, 2009 at 16:32

---

4 @PTBNL, wow, you wrote 3000 line suites of perl without knowing how to use an array-reference? `foo(\@bar , \@baz )` ?  
– [Kent Fredric](#) Jun 9, 2009 at 3:27

---

4 @PTBNL, what is the aversion to this style: `sub foo { my ($name, $age, %other) = @_ ; }` You must not have looked at any decent Perl code before you got started. There is a lot that can be learned with a cursory visit to a newsgroup like `comp.lang.perl.misc`. Or CPAN. A well written Perl program is like the curves on a beautiful woman. – [xcramps](#) Jun 18, 2009 at 10:21

---

43

votes



MS Access Visual Basic for Applications (VBA) was also pretty bad. Access was bad altogether in that it forced you down a weak paradigm and was deceptively simple to get started, but a nightmare to finish.

Share

[edited Jun 9, 2009 at 2:11](#)

community wiki

[2 revs](#)

[John MacIntyre](#)

---

@Andrei so you've said twice so far. If you dislike PHP so much, make your own answer rather than spamming comments. – [Neil Aitken](#) Jun 9, 2009 at 12:31

---

Why is that answer so low? VB6 > VBA, yet VB6 is almost at the top. – [Meta-Knight](#) Jun 9, 2009 at 13:29

---

2 @Meta-Knight-Don't forget it's not really a 'which language is worse', but 'which language is most popularly bad'.  
– [John MacIntyre](#) Jun 9, 2009 at 18:48

---

1 Agreed that the MS Access dialect is just...whacked. Let's get queried data with `DoCmd` but edit it with a recordset that requires a different API? What were they smoking?  
– [Avery Payne](#) Jul 6, 2010 at 0:16

---

---

## 40 No answer about Cobol yet? :O

votes



Share

answered [Jun 7, 2009 at 15:46](#)



community wiki  
[Joril](#)

---

See my post elsewhere on MetLife English Language, a *precursor* to Cobol that our Y2K remediation company worked on. – [Jim Ferrans](#) Jun 7, 2009 at 20:06

---

13 Actually COBOL is quite a decent language. It was created and standardised before type checking was thought of. But it was the one of the first languages to have built in type checking and automatic type conversions within a program but no type checking on subroutine calls. It is still one of the very few languages that can do decimal arithmetic properly and its



handling of structures was way ahead of its time. Modern COBOL is a pretty neat language, they made it fully OO with the addition of only three reserved words, and, it supports pointers and pointer arithmetic. – [James Anderson](#) Jun 8, 2009 at 2:37

---

Does modern COBOL support functions in the sense of something you can put into an expression? I found the lack of those incredibly frustrating, way back when. – [David Thornley](#) Jun 8, 2009 at 15:57

---

If I remember correctly, we didn't really *use* expressions "back in the day", except in the COMPUTE command. There wasn't any place to put a function call. – [John Saunders](#) Jun 9, 2009 at 0:16

---

**11** @John Saunders: other languages described here are just bad. COBOL is beautiful, elaborate, baroque in its terribleness. – user447688 Jun 9, 2009 at 12:27

---

40  
votes



Old-skool BASICs with line numbers would be my choice. When you had no space between line numbers to add new lines, you had to run a renumber utility, which caused you to lose any mental anchors you had to what was where.

As a result, you ended up squeezing in too many statements on a single line (separated by colons), or you did a goto or gosub somewhere else to do the work you couldn't cram in.

Share

[edited Jun 7, 2009 at 19:17](#)

- 
- 61 There was a renumbering utility? Man, that would have saved me some work back when I was 11. :) – [rtperson](#) Jun 7, 2009 at 15:46
- 
- 3 Some BASIC's had a renum command. For others the process could be truly atrocious--save your BASIC file out as ASCII, run the renumber tool on it as data, then reimport your new ASCII file as code. – [Nosredna](#) Jun 7, 2009 at 15:59
- 
- 19 Heh... I just used multiples of 100 for each line number, leaving plenty of room for inserting new lines... Then again, I moved on to better languages after cutting my teeth on BASIC. What a shock it was indeed to see other, similarly-impractical aspects of the language preserved in VB years later! That said, I still consider old-school BASIC a reasonably good language for beginners - it's simple, easy to use interactively, and line numbers actually do provide a reasonable abstraction for the program counter on the CPU - jumping from learning BASIC to learning an assembly language works well. – [Shog9](#) Jun 7, 2009 at 16:09
- 
- 4 Anyone here remember programming BASIC on Apple computers and using all the wonderful Beagle Bros utilities? – [Barry Brown](#) Jun 7, 2009 at 18:02
- 
- 7 Standard practice was to increment line numbers by 10, so you had room to insert a few lines. If you need to insert just one line, you added 5. If you found you needed to insert another later, you add two or three, and start to get nervous... – [T.E.D.](#) Jun 9, 2009 at 12:39
-



I worked in it for a couple years, but have done a complete brain dump since then. All I can really remember was no documentation (at my location) and cryptic commands.

It was horrible. Horrible! HORRIBLE!!!

Share

answered [Jun 8, 2009 at 16:54](#)

community wiki  
[dna123](#)

- 
- 2 that language WTF's hard. It deserves a high ranking, with PHP. People call Perl line noise, ye gods, if perl is line noise, that code is pure bitrot. – [Kent Fredric](#) Jun 9, 2009 at 3:00
- 
- 5 In defense of MUMPS: It was awesome for its time. As one of the first ANSI standardized languages, the documentation was great. It was cryptic only if you used the one-letter abbreviations for commands (which everybody did). For example, you could write SET X=5 or S X=5 instead. PERSISTENT MULTI-DIMENSIONAL ASSOCIATIVE ARRAYS! Example: Storing phone numbers. SET ^PHONE("Joe Blow","home") = "867-5309" Awesome. There was also regex-like pattern matching built in. Don't get me wrong, it had its weaknesses, but there are many ways in which it still hasn't been trumped, even 40 years later. – [I. J. Kennedy](#) Jun 28, 2009 at 17:36
- 
- 2 I'm shocked that no one linked to this The Daily WTF classic. [thedailywtf.com/Articles/A\\_Case\\_of\\_the\\_MUMPS.aspx](http://thedailywtf.com/Articles/A_Case_of_the_MUMPS.aspx) – [Yahel](#) Mar 8, 2011 at 18:39
-

37

votes



There are just two kinds of languages: the ones everybody complains about and the ones nobody uses.

Bjarne Stroustrup

Share

answered [Jun 28, 2009 at 18:30](#)

community wiki

[Dinah](#)

- 
- 4 +1. Doesn't quite answer the question but very true and relevant. I would say "make this a comment" but then it would get buried among the close-vs-reopen comments. – [finnw](#) Aug 7, 2010 at 11:44
- 

34

votes



I haven't yet worked with many languages and deal mostly with scripting languages; out of these **VBScript** is the one I like least. Although it has some handy features, some things really piss me off:

- Object assignments are made using the `set` keyword:

```
Set foo = Nothing
```

Omitting `set` is one of the most common causes of run-time errors.

- No such thing as structured exception handling. Error checking is like this:

```
On Error Resume Next

' Do something

If Err.Number <> 0
    ' Handle error
    Err.Clear
End If

' And so on
```

- Enclosing the procedure call parameters in parentheses requires using the `call` keyword:

```
Call Foo (a, b)
```

- Its English-like syntax is way too verbose. (I'm a fan of curly braces.)
- Logical operators are long-circuit. If you need to test a compound condition where the subsequent condition relies on the success of the previous one, you need to put conditions into separate `If` statements.
- Lack of parameterized class constructors.
- To wrap a statement into several lines, you have to use an underscore:

```
str = "Hello, " & _
      "world!"
```

- Lack of multiline comments.
-

**Edit:** found this article: [The Flangy Guide to Hating VBScript](#). The author sums up his complaints as "VBS isn't Python" :)

Share

edited Jul 10, 2009 at 14:05

answered Jun 7, 2009 at 15:56



Helen

96.9k ● 17 ● 272 ● 338

---

9 The error-handling aspect was the worst. Debugging a large ASP app built on VBS (crammed full of ON ERROR RESUME NEXT functions of course) was pure insanity. – [Shog9](#) Jun 7, 2009 at 16:14

---

3 I personally find the difference between Functions and Subs to be particularly problematic. Why should calling subs need no parentheses (nor allow them), but functions do, and call changes the whole sub paradigm. I hated it. Error handling is painful too (especially in classic ASP, where it's not wise to just exit). – [C. Ross](#) Oct 4, 2009 at 1:08

---

The points you raise are the same as VBA (or any old dialect of VB). I write thousands of lines of those stuff for a living on a daily basis (hopefully I do interesting stuff too) and I must admit that 1) yeah error handling sucks ass, you can feel the longjmp behind and 2) the "object or with block variable not set" is difficult to understand, and is **runtime**, whereas those things can be checked beforehand. – [Alexandre C.](#) Aug 1, 2011 at 11:34

---

votes



The annotations are confusing, using brackets to call methods still does not compute in my brain, and what is worse is that all of the library functions from C are called using the standard operators in C, -> and ., and it seems like the only company that is driving this language is Apple.

I admit I have only used the language when programming for the iPhone (and looking into programming for OS X), but it feels as if C++ were merely forked, adding in annotations and forcing the implementation and the header files to be separate would make much more sense.

Share

answered Jun 7, 2009 at 16:04

community wiki  
John Bellone

---

**36** Objective C was created while C++ was still in its infancy (as C with Classes) and there is not really any relationship between them. Obj-C took the Smalltalk way of doing OO; C++ stuck closer to Simula. The syntax takes a while to get used to, but if you keep an open mind there is a lot of power in the language too (I wrote up one technique I used here: [metatechnology.blogspot.com/2009/04/...](http://metatechnology.blogspot.com/2009/04/...)). – [philsquared](#) Jun 7, 2009 at 16:53

---

**11** I love Objective-C. It's a very nice blend of hands-on-hardware C and higher-level dynamic language. The syntax takes time, but the named parameters are great for readability. – [zoul](#) Jun 7, 2009 at 17:38

---

**1** I used the original 1980's Objective-C from StepStone (Brad Cox's) company, and I loved it, it was a huge advance over C

and actually a very simple extension to C. Moving to C++ a year or two later was quite painful, due to its complexity and its immature and incompatible implementations. – [Jim Ferrans](#)

Jun 7, 2009 at 20:05

---

3 I like C, and I like dynamic languages, but I find it difficult to like Objective-C. I expected to love it, but I ended up just finding it difficult to read. – [Nosredna](#) Jun 8, 2009 at 2:25

---

6 It's just that the more I use Python, the more I like it. The more I use Objective-C, the less I like it. – [Nosredna](#) Jun 11, 2009 at 0:15

---

---

28 votes PROGRESS 4GL (apparently now known as "[OpenEdge Advanced Business Language](#)").



PROGRESS is both a language and a database system.



The whole language is designed to make it easy to write crappy green-screen data-entry screens. (So start by imagining how well this translates to Windows.) Anything fancier than that, whether pretty screens, program logic, or batch processing... not so much.

I last used version 7, back in the late '90s, so it's vaguely possible that some of this is out-of-date, but I wouldn't bet on it.

- It was originally designed for text-mode data-entry screens, so on Windows, all screen coordinates are in "character" units, which are some weird number of pixels wide and a different number of pixels high. But of course they default to a *proportional* font, so the number of "character units" doesn't correspond to the



actual number of *characters* that will fit in a given space.

- No classes or objects.
- No language support for arrays or dynamic memory allocation. If you want something resembling an array, you create a temporary in-memory database table, define its schema, and then get a cursor on it. (I saw a bit of code from a later version, where they actually built and shipped a primitive object-oriented system on top of these in-memory tables. Scary.)
- ISAM database access is built in. (But not SQL. Who needs it?) If you want to increment the `Counter` field in the current record in the `State` table, you just say `State.Counter = State.Counter + 1`. Which isn't so bad, except...
- When you use a table directly in code, then behind the scenes, they create something resembling an invisible, magic local variable to hold the current cursor position in that table. They *guess* at which containing block this cursor will be scoped to. If you're not careful, your cursor will vanish when you exit a block, and reset itself later, with no warning. Or you'll start working with a table and find that you're not starting at the first record, because you're reusing the cursor from some other block (or even your own, because your scope was expanded when you didn't expect it).
- Transactions operate on these wild-guess scopes. Are we having fun yet?

- Everything can be abbreviated. For some of the offensively long keywords, this might not seem so bad at first. But if you have a variable named `Index`, you can refer to it as `Index` or as `Ind` or even as `I`. (Typos can have very interesting results.) And if you want to access a database field, not only can you abbreviate the field name, but you don't even have to qualify it with the table name; *they'll guess the table too*. For truly frightening results, combine this with:
- Unless otherwise specified, they assume *everything* is a database access. If you access a variable you haven't declared yet (or, more likely, if you mistype the variable name), there's no compiler error: instead, it goes looking for a database field with that name... or a field that abbreviates to that name.

The guessing is the worst. Between the abbreviations and the field-by-default, you could get some nasty stuff if you weren't careful. (Forgot to declare `I` as a local variable before using it as a loop variable? No problem, we'll just randomly pick a table, grab its current record, and completely trash an arbitrarily-chosen field whose name starts with `I`!)

Then add in the fact that an accidental field-by-default access could change the scope it guessed for your tables, thus breaking some completely unrelated piece of code. Fun, yes?

They also have a reporting system built into the language, but I have apparently repressed all memories of it.

When I got another job working with Netscape LiveWire (an ill-fated attempt at server-side JavaScript) and classic ASP (VBScript), I was in heaven.

Share

answered [Jun 7, 2009 at 17:27](#)

community wiki

[Joe White](#)

---

I don't know if vote this up or down. I worked with PROGRESS and although the language is not the best I started to feel certain sympathy to it – [victor hugo](#) [Jun 8, 2009 at 17:01](#)

---

---

27 The worst language? BancStar, hands down.

votes



3,000 predefined variables, all numbered, all global. No variable declaration, no initialization. Half of them, scattered over the range, reserved for system use, but you can use them at your peril. A hundred or so are automatically filled in as a result of various operations, and no list of which ones those are. They all fit in 38k bytes, and there is no protection whatsoever for buffer overflow. The system will cheerfully let users put 20 bytes in a ten byte field if you declared the length of an input field incorrectly. The effects are unpredictable, to say the least.

This is a language that will let you declare a calculated gosub or goto; due to its limitations, this is frequently necessary. Conditionals can be declared forward or

reverse. Picture an "If" statement that terminates 20 lines before it begins.

The return stack is very shallow, (20 Gosubs or so) and since a user's press of any function key kicks off a different subroutine, you can overrun the stack easily. The designers thoughtfully included a "Clear Gosubs" command to nuke the stack completely in order to fix that problem and to make sure you would never know exactly what the program would do next.

There is much more. Tens of thousands of lines of this Lovecraftian horror.

Share

answered [Jun 8, 2009 at 3:11](#)

community wiki  
[R Ubben](#)

---

2 [geocities.com/connorbd/tarpit/bancstar.html](http://geocities.com/connorbd/tarpit/bancstar.html) I see what you mean... And they trusted this abomination with people's money?! – [cheduardo](#) Jun 16, 2009 at 11:17

---

@cheduardo: seeing your link makes me sad. How much (useful?) information was lost along with geocities?  
– [Esteban Küber](#) Jan 7, 2010 at 3:33

---

Probably quite a bit, but the Wayback Machine is pretty useful.  
– [R Ubben](#) Jan 20, 2010 at 14:01

---

From wikipedia : BANCStar resembles an esoteric programming language; so much so that it has sometimes been mistaken for a joke language. – [fingerprint211b](#) Jun 14, 2010 at 13:58

---

- 1 @voyager there is a mirror of that geocities site here: [oocities.org/connorbd/tarpit/bancstar.html](http://oocities.org/connorbd/tarpit/bancstar.html) – [Colin Pickard](#) May 25, 2011 at 11:28
- 

23

votes



The [.bat files scripting language](#) on DOS/Windows. God only knows how un-powerful is this one, specially if you compare it to the Unix shell languages (that aren't so powerful either, but way better nonetheless).

Just try to concatenate two strings or make a for loop. Nah.

Share

answered [Jun 7, 2009 at 19:24](#)

community wiki  
[friol](#)

- 
- 5 Unix shell programming is much more powerful and infinitely more sane than .bat/.cmd programming. – [Cheeso](#) Jun 8, 2009 at 16:40
- 

- 6 Concatenating strings is easy: %VAR1%%VAR2%. And there's a for loop: [robvanderwoude.com/ntfor.php](http://robvanderwoude.com/ntfor.php). The syntax is ugly as hell, but don't claim it can't be done when it clearly can. – [Joe White](#) Jun 9, 2009 at 1:10
- 

- 1 "Concatenating two (or three, I don't remember) strings leaves a whitespace hole between them" - Don't know where you get that idea, but it's still wrong. We do this in our build scripts and

it works just fine. I just double-checked. "echo  
%SystemDrive%%SystemDrive%%SystemDrive%" -> "C:C:C:"  
– [Joe White](#) Jun 10, 2009 at 13:10

---

6 Duplicate of this answer higher up:  
[stackoverflow.com/questions/961942/...](http://stackoverflow.com/questions/961942/...) – [Jonik](#) Jun 17, 2009  
at 12:40

---

2 Most people encounter superfluous whitespace when they  
actually inset it into the commands. The set command doesn't  
need whitespace surrouncing the equals sign, so "set a = foo"  
clearly sets a to " foo". No surprise here and probably a major  
reason why people are think they got whitespace that didn't  
belong there. – [Joey](#) Jul 1, 2009 at 13:54

---

---

## 23 [VSE, The Visual Software Environment.](#)

votes



This is a language that a prof of mine ([Dr. Henry Ledgard](#))  
tried to sell us on back in undergrad/grad school. (I don't  
feel bad about giving his name because, as far as I can tell,  
he's still a big proponent and would welcome the chance to  
convince some folks it's the best thing since sliced bread).  
When describing it to people, my best analogy is that it's  
sort of a bastard child of FORTRAN and COBOL, with  
some extra bad thrown in. From the [only really accessible  
folder](#) I've found with this material (there's lots more in there  
that I'm not going to link specifically here):

- [VSE Overview](#) (pdf)
- [Chapter 3: The VSE Language](#) (pdf) (Not really an  
overview of the language at all)
- [Appendix: On Strings and Characters](#) (pdf)

- [The Software Survivors](#) (pdf) (Fevered ramblings attempting to justify this turd)

VSE is built around what they call "The Separation Principle". The idea is that Data and Behavior must be completely segregated. Imagine C's requirement that all variables/data must be declared at the beginning of the function, except now move that declaration into a separate file that other functions can use as well. When other functions use it, they're using the same data, not a local copy of data with the same layout.

Why do things this way? We learn that from *The Software Survivors* that **Variable Scope Rules Are Hard**. I'd include a quote but, like most fools, it takes these guys forever to say anything. Search that PDF for "Quagmire Of Scope" and you'll discover some true enlightenment.

They go on to claim that this somehow makes it more suitable for multi-proc environments because it more closely models the underlying hardware implementation. Riiight.

Another choice theme that comes up frequently:

```
INCREMENT DAY COUNT BY 7 (or DAY COUNT =  
DAY COUNT + 7)  
DECREMENT TOTAL LOSS BY GROUND_LOSS  
ADD 100.3 TO TOTAL LOSS(LINK_POINTER)  
SET AIRCRAFT STATE TO ON_THE_GROUND  
PERCENT BUSY = (TOTAL BUSY CALLS *  
100)/TOTAL CALLS
```

Although not earthshaking, the style of arithmetic reflects ordinary usage, i.e., anyone can read and understand it - without knowing a programming language. In fact, VisiSoft arithmetic is virtually identical to FORTRAN, including embedded complex arithmetic. **This puts programmers concerned with their professional status and corresponding job security ill at ease.**

Ummm, not that concerned at all, really. One of the key selling points that Bill Cave uses to try to sell VSE is the democratization of programming so that business people don't need to indenture themselves to programmers who use crazy, arcane tools for the sole purpose of job security. He leverages this irrational fear to sell his tool. (And it works-- the federal gov't is his biggest customer). I counted 17 uses of the phrase "job security" in the document.

Examples:

- ... and fit only for those desiring artificial **job security**.
- More false **job security**?
- Is **job security** dependent upon ensuring the other guy can't figure out what was done?
- Is **job security** dependent upon complex code...?



- One of the strongest forces affecting the acceptance of new technology is the perception of one's **job security**.

He uses this paranoia to drive wedge between the managers holding the purse strings and the technical people who have the knowledge to recognize VSE for the turd that it is. This is how he squeezes it into companies-- "Your technical people are only saying it sucks because they're afraid it will make them obsolete!"

---

## A few additional choice quotes from the overview documentation:

Another consequence of this approach is that **data is mapped into memory on a "What You See Is What You Get" basis**, and maintained throughout. This allows users to move a complete structure as a string of characters into a template that describes each individual field. Multiple templates can be redefined for a given storage area. Unlike C and other languages, **substructures can be moved without the problems of misalignment due to word boundary alignment standards**.

Now, I don't know about you, but I know that a WYSIWYG approach to memory layout is at the *top* of my priority list when it comes to language choice! Basically, they ignore alignment issues because only old languages that were

designed in the '60's and '70's care about word alignment. Or something like that. The reasoning is bogus. It made so little sense to me that I proceeded to forget it almost immediately.

**There are no user-defined types in VSE.** This is a far-reaching decision that greatly simplifies the language. The gain from a practical point of view is also great. VSE allows the designer and programmer to organize a program along the same lines as a physical system being modeled. VSE allows structures to be built in an easy-to-read, logical attribute hierarchy.

Awesome! User-defined types are lame. Why would I want something like an `InputMessage` object when I can have:

```
LINKS_IN_USE INTEGER
INPUT_MESSAGE
  1 ORIGIN          INTEGER
  1 DESTINATION     INTEGER
  1 MESSAGE
    2 MESSAGE_HEADER CHAR 10
    2 MESSAGE_BODY   CHAR 24
    2 MESSAGE_TRAILER CHAR 10
  1 ARRIVAL_TIME    INTEGER
  1 DURATION         INTEGER
  1 TYPE            CHAR 5

OUTPUT_MESSAGE CHARACTER 50
```

You might look at that and think, "Oh, that's pretty nicely formatted, if a bit old-school." Old-school is right.

Whitespace is significant-- *very* significant. And redundant! The `1`'s must be in column 3. The `1` indicates that it's at the first level of the hierarchy. The Symbol name must be in column 5. You hierarchies are limited to a depth of 9.

Well, ok, but is that so awful? Just wait:

It is well known that for reading text, use of conventional upper/lower case is more readable. **VSE uses all upper case (except for comments).** Why? The literature in psychology is based on prose. Programs, simply, are not prose. Programs are more like math, accounting, tables. Program fonts (usually Courier) are almost universally fixed-pitch, and for good reason – vertical alignment among related lines of code. **Programs in upper case are nicely readable, and, after a time, much better in our opinion**

Nothing like enforcing your opinion at the language level! That's right, you cannot use any lower case in VSE unless it's in a comment. Just keep your CAPSLOCK on, it's gonna be stuck there for a while.

VSE subprocedures are called processes. This code sample contains three processes:

```
PROCESS_MUSIC
  EXECUTE INITIALIZE_THE_SCENE
  EXECUTE PROCESS_PANEL_WIDGET

INITIALIZE_THE_SCENE
```

```

      SET TEST_BUTTON PANEL_BUTTON_STATUS TO ON
      MOVE ' ' TO TEST_INPUT PANEL_INPUT_TEXT
      DISPLAY PANEL PANEL_MUSIC

PROCESS_PANEL_WIDGET
      ACCEPT PANEL PANEL_MUSIC

*** CHECK FOR BUTTON CLICK
      IF RTG_PANEL_WIDGET_NAME IS EQUAL TO
      'TEST_BUTTON'
          MOVE 'I LIKE THE BEATLES!' TO TEST_INPUT
      PANEL_INPUT_TEXT.
      DISPLAY PANEL PANEL_MUSIC

```

All caps as expected. After all, that's easier to read. Note the whitespace. It's significant again. All process names must start in column 0. The initial level of instructions must start on column 4. Deeper levels must be indented exactly 3 spaces. This isn't a big deal, though, because you aren't allowed to do things like nest conditionals. You want a nested conditional? Well just make another process and call it. And note the *delicious* COBOL-esque syntax!

You want loops? Easy:

```

EXECUTE NEXT_CALL
EXECUTE NEXT_CALL 5 TIMES
EXECUTE NEXT_CALL TOTAL CALL TIMES

EXECUTE NEXT_CALL      UNTIL NO LINES ARE AVAILABLE
EXECUTE NEXT_CALL      UNTIL CALLS_ANSWERED ARE EQUAL
TO CALLS_WAITING
EXECUTE READ_MESSAGE UNTIL LEAD_CHARACTER IS A
DELIMITER

```

Ugh.

community wiki

5 revs

Greg D

- 
- 2 The problem with **The Software Survivors** is that it assumes that all programming tasks are as easy as writing enterprise software - clearly he needs to be asked to implement his own C compiler, operating system, graphics subsystem, window manager, and web browser. – [new123456](#) Apr 3, 2011 at 22:32
- 

---

22 Here is the contribution to my own question:

votes

## Origin LabTalk



My all-time favourite in this regard is [Origin LabTalk](#). In LabTalk the maximum length of a string variable identifier is **one** character. That is, there are only 26 string variables at all. Even worse, some of them are used by Origin itself, and it is not clear which ones.

From the manual:

LabTalk uses the % notation to define a string variable. A legal string variable name must be a % character followed by a single alphabetic character (a letter from A to Z). String variable names are

caseinsensitive. Of all the 26 string variables that exist, Origin itself uses 14.

## Doors DXL

For me the second worst in my opinion is [Doors DXL](#). Programming languages can be divided into two groups: Those with manual memory management (e.g. delete, free) and those with a garbage collector. Some languages offer both, but DXL is probably the only language in the world that supports neither. OK, to be honest this is only true for strings, but hey, strings aren't exactly the most rarely used data type in requirements engineering software.

The consequence is that memory used by a string can never be reclaimed and DOORS DXL leaks like sieve.

There are countless other quirks in DXL, just to name a few:

- [DXL function syntax](#)
- [DXL arrays](#)

Share

edited May 23, 2017 at 12:26



Community Bot

1 ● 1

answered Jun 7, 2009 at 14:23



Ludwig Weinzierl

16.6k ● 10 ● 46 ● 49

---

1 "Of all the 26 string variables that exist, Origin itself uses 14."  
Just reading this sentence from manual is scary. How could the  
creators of the language not realize this is utterly stupid?  
– [ya23](#) Jun 9, 2009 at 10:57

---

1 +1 Doors is number 12 at [www.dreckstool.de](http://www.dreckstool.de) (List of worst  
software ever) – [bbuser](#) Jun 24, 2009 at 10:35

---

---

## 21 Cold Fusion

votes



I guess it's good for designers but as a programmer I  
always felt like one hand was tied behind my back.

Share

answered [Jun 10, 2009 at 13:06](#)

community wiki  
[Ken Burkhardt](#)

---

5 +1. It's not good for designers. Or programmers. Or dB types. I  
have met many who worked on it and no one who liked it.  
Including me. – [kmarsh](#) Jul 1, 2009 at 13:14

---

---

## 20

votes



The worst two languages I've worked with were APL, which  
is relatively well known for languages of its age, and TECO,  
the language in which the original Emacs was written. Both  
are notable for their terse, inscrutable syntax.

APL is an array processing language; it's extremely  
powerful, but nearly impossible to read, since every

character is an operator, and many don't appear on standard keyboards.

TECO had a similar look, and for a similar reason. Most characters are operators, and this special purpose language was devoted to editing text files. It was a little better, since it used the standard character set. And it did have the ability to define functions, which was what gave life to emacs--people wrote macros, and only invoked those after a while. But figuring out what a program did or writing a new one was quite a challenge.

Share

answered Jun 7, 2009 at 16:35



PanCrit

2,718 ● 2 ● 18 ● 16

---

+1 for APL, but -1 for TECO. OMG, TECO-haters? Really? your hating on the first programmable video editing language, ever?  
– [RBarryYoung](#) Jun 10, 2009 at 2:35

---

Never used teco, so can't pass judgement on it, but there's an anecdote in the jargon file about how hackers would mash random keys and challenge another hacker to explain what the result would produce when entered into teco. frightful!  
– [SingleNegationElimination](#) Jun 10, 2009 at 6:49

---

Some dialects of APL not only use characters that do not appear on standard keyboards, their visual representation cannot be drawn on standard screens. (OK, I thought it was funny...) – [j\\_random\\_hacker](#) Jun 1, 2010 at 14:40

---

type your name into teco and see what happens... – [blabla999](#)  
Aug 24, 2010 at 17:21

---



19

- [LOLCODE](#):

votes



```
HAI
CAN HAS STDIO?
VISIBLE "HAI WORLD!"
KTHXBYE
```

Seriously, the worst programming language ever is that of Makefiles. Totally incomprehensible, tabs have a syntactic meaning and not even a debugger to find out what's going on.

Share

answered [Jun 7, 2009 at 16:26](#)

community wiki  
[Tobias](#)

- 
- 7 any environment where the behaviour hinges on the particulars of the *whitespace* has to be at the top of the list! – [JustJeff](#) Jun 7, 2009 at 17:55

---

I LOVE LOLCODE! Its obviously not practical, but its great to mess around with on the side or for jokes when your dev team needs a little brevity. -1 for hatein on LOLCODE, +1 for makefiles...guess that makes it a draw. – [jrista](#) Jun 7, 2009 at 20:38

---

@JustJeff, to be fair, Make treats space as different than tab, which is a serious problem... – [Brian Postow](#) Jun 9, 2009 at 14:06

- 
- 2 Oh, man. About two years ago, I tried to suggest a "funny" syntax addition (since the grammar was missing an entire

genre of lolcat sentences!!1), and it got vetoed because (I kid you not) that functionality already existed with different syntax. I still can't wrap my head around people taking that "language" seriously. – [ojrac](#) Jun 11, 2009 at 2:36

---

18  
votes



I'm not sure if you meant to include scripting languages, but I've seen TCL (which is also annoying), but... the **mIRC scripting language** annoys me to no end.

Because of some oversight in the parsing, it's whitespace significant when it's not supposed to be. Conditional statements will sometimes be executed when they're supposed to be skipped because of this. Opening a block statement cannot be done on a separate line, etc.

Other than that it's just full of messy, inconsistent syntax that was probably designed that way to make very basic stuff easy, but at the same time makes anything a little more complex barely readable.

I lost most of my mIRC scripts, or I could have probably found some good examples of what a horrible mess it forces you to create :(

Share

answered [Jun 7, 2009 at 18:52](#)

community wiki  
[Thorarin](#)

- 
- 1 This this this this oh geez this. I've had to try to maintain a 3000-line mIRC script before; it's utterly horrible to work with.  
– [Rob Howard](#) Jun 8, 2009 at 22:31
- 

- 5 mIRC scripting is hell. Imagine a programming language where you MUST use stupid hacks in order to process strings with whitespace in them. Since there are no string literals, it's not possible to store a string with a space at the end.  
– [Vladimir Panteleev](#) Jun 9, 2009 at 14:01
- 

---

## 18 Regular expressions

votes



It's a write only language, and it's hard to verify if it works correctly for the right inputs.

Share

answered [Jun 7, 2009 at 19:08](#)

community wiki

[o3wəJ](#)

---

Extended regexes can be commented, but not every engine supports them. – [user60401](#) Jun 8, 2009 at 15:54

---

- 6 Its only a write only language to people who never learned to write regex. – [Kent Fredric](#) Jun 9, 2009 at 2:55
- 

- 9 Regex definitely merits heavy unit testing, but it's far more powerful than it is confusing. – [ojrac](#) Jun 11, 2009 at 3:26
- 

---

## 17 Visual Foxpro

votes



Share

answered Jun 7, 2009 at 14:20



[Arnis Lapsa](#)

47.5k ● 29 ● 118 ● 196

- 
- 4 Can you elaborate a bit on why? – [Emil H](#) Jun 7, 2009 at 14:27
- 
- 4 Well, there are things you could do in Foxpro (for DOS!) in one line of code that still you cant do in 5 lines of code in c#.  
– [Binoj Antony](#) Jun 8, 2009 at 7:52
- 
- 2 I half agree and half disagree. VFP tried to port the event model of the DOS product to Windows which was an epic fail IMO. That said, for a long time VFP was my Swiss army knife for doing database tasks. It was one of the first products that allowed one to easily interoperate among multiple databases.  
– [Jeff Leonard](#) Jun 8, 2009 at 21:44
- 
- 2 VFP sucks mostly because it promotes billions of amateur " I can write an accounting package too" software which is basically 2 parts: 1) The most awful unusable accounting package interface known to man, 2) nasty databases with 4 tables at 150 columns each full of junk – [Kent Fredric](#) Jun 9, 2009 at 3:06
- 
- 3 VFP is not that bad. Learning the syntax is a little difficult, but overall it's pretty good for a language. Shame MS dropped support for it. Lots of code to convert to another platform now.  
– [dna123](#) Jun 9, 2009 at 19:18
- 

17 I can't belive nobody has said this one:

votes



**LotusScript**

I thinks is far worst than php at least.

Is not about the language itself which follows a syntax similar to Visual Basic, is the fact that it seem to have a lot of functions for extremely unuseful things that you will never (or one in a million times) use, but lack support for things you will use everyday.

I don't remember any concrete example but they were like:

"Ok, I have an event to check whether the mouse pointer is in the upper corner of the form and I don't have an double click event for the Form!!?? WTF??"

Share

answered [Jun 8, 2009 at 11:57](#)

community wiki  
[Jorge Córdoba](#)

---

It's definitely worst then PHP and worst think on LS is debugger. – [MicTech](#) Jun 9, 2009 at 12:30

---

Totally - though not surprised it wasn't mentioned - how many LotusScript developers are there out there (especially among SO users) – [Anthony Rizk](#) Jun 9, 2009 at 13:47

- 
- 1 The scripting in Lotus Notes was terrible. The events you could programme were terribly weak. Each line was compiled on loss of focus and if it failed to compile it would be deleted. So you want to write half a line of code and the copy and paste a long variable name in? You are halfway through writing a line of code and you need to look up the documentation? Why would you want to do these things? – [Gordon Guthrie](#) Jun 9, 2009 at 13:57
-

Annoyingly it has received little attention in the last 3 major releases and continues to lack many needed features. IBM has concentrated on making Notes some sort of Java Eclipse container and ignored making Notes itself a more capable development environment. – [Martlark](#) Jun 14, 2009 at 2:28

---

Glad someone else finds LotusScript baffling, even for someone familiar with other scripting languages. – [Lunatik](#) Jul 1, 2009 at 11:03

---

17  
votes



Twice I've had to work in 'languages' where you drag-n-dropped modules onto the page and linked them together with lines to show data flow. (One claimed to be a RDBMs, and the other a general purpose data acquisition and number crunching language.)

Just thinking of it makes me want to throttle someone. Or puke. Or both.

Worse, neither exposed a text language that you could hack directly.

Share

[edited Jun 9, 2009 at 13:32](#)

community wiki  
[3 revs](#)  
[dmckee](#)

---

Yahoo Pipes?? ::click:: ::click:: ::click:: ::shudder::  
– [dmckee](#) --- [ex-moderator kitten](#) Jun 9, 2009 at 13:31

---

One of them was, indeed, an early version of Labview for the Mac. People assure me that it is better now. I haven't bothered to find out... – [dmckee](#) --- [ex-moderator kitten](#) Jun 9, 2009 at 17:59

---

Yes. Another example is DIS a horrible idea from a company called Metaphore – [EvilTeach](#) Jul 1, 2009 at 17:56

---

---

**17** I find myself avoid having to use **VBScript/Visual Basic 6** the most.  
votes



I use primarily C++, javascript, Java for most tasks and dabble in ruby, scala, erlang, python, assembler, perl when the need arises.

I, like most other reasonably minded polyglots/programmers, strongly feel that you have to use the right tool for the job - this requires you to understand your domain and to understand your tools.

My issue with VBscript and VB6 is when I use them to script windows or office applications (the right domain for them) - i find myself struggling with the language (they fall short of being the right tool).

VBScript's lack of easy to use native data structures (such as associative containers/maps) and other quirks (such as set for assignment to objects) is a needless and frustrating annoyance, especially for a scripting language. Contrast it with Javascript (which i now use to program wscript/cscript windows and do activex automation scripts) which is far more expressive. While there are certain things that work

better with vbscript (such as passing arrays back and forth from COM objects is slightly easier, although it is easier to pass event handlers into COM components with jscript), I am still surprised by the amount of coders that still use vbscript to script windows - I bet if they wrote the same program in both languages they would find that jscript works with you much more than vbscript, because of jscript's native hash data types and closures.

Vb6/VBA, though a little better than vbscript in general, still has many similar issues where (for their domain) they require much more boiler plate to do simple tasks than what I would like and have seen in other scripting languages.

Share

[edited Dec 10, 2011 at 16:36](#)

community wiki  
[3 revs, 2 users 98%](#)  
[Faisal Vali](#)

---

VB6 is not a scripting language. It's a complete language, including support for classes and interfaces. Granted, it's not OO (no inheritance of behavior, only has interface inheritance), but it's really not that bad as a language. OTOH, there are a large number of really bad pieces of code written in it, and only a small number of good pieces of code. – [John Saunders](#) Jun 9, 2009 at 0:25

---

I never said VB6 is only a scripting language (what exactly defines a pure scripting language is another discussion ;) I said that I have used VB6 to script/macro-control office applications.



And as far as being a complete language (Turing complete), even assembler is a complete language - it's just not very expressive - and in my experience, based on programming idioms i am used to, VB6 could be more expressive and less annoying. But there are many people who feel the same about C++ and for some reason (that may be worth investigating more thoroughly someday) I don't ;) – [Faisal Vali](#) Jun 11, 2009 at 4:14

---

I gotta disagree, I can't think of a single thing I have ever wanted to do with VBA and not been able to. That kind of read like, "I don't like VBA because I can't figure out how to do stuff:)) NOI, as a C++ guy I'm sure you can bury me any day of the week. But still VBA can do a ton in very few lines that take other languages 10 times as many. – [Oorang](#) Jun 11, 2009 at 6:37

---

- 1 @Oorang: I'm sorry, I didn't mean for it to read like that. I was able to complete my task in VBA (a calendar that takes different peoples vacation days and all the national holidays, and designates when they are on call in as fair a manner as possible) but having done something similar in javascript (you can manipulate office applications via jsript), i enjoyed doing it less in VBA. What I meant for it to read like is: I don't like VBA because the programming idioms i like to use are less well supported. But I realize that is in the eye of the beholder :) – [Faisal Vali](#) Jun 11, 2009 at 13:26
- 

16  
votes



In 25+ years of computer programming, by far the worst thing I've ever experienced was a derivative of MUMPS called Meditech Magic. It's much more evil than PHP could ever hope to be.

It doesn't even use '=' for assignment! 100^b assigns a value of 100 to b and is read as "100 goes to b". Basically,

this language invented its own syntax from top to bottom.  
So no matter how many programming languages you know,  
Magic will be a complete mystery to you.

Here is 100 bottles of beer on the wall written in this  
abomination of a language:

```
BEERV1.1,  
100^b,T("")^#,DO{b'<1 NN(b,"bottle"_IF{b=1 " "; "s  
"}_"of beer on the wall")^#,  
N(b,"bottle"_IF{b=1 "  
"; "s "}_"of beer!")^#,  
N("You take one down,  
pass it around,")^#,b-1^b,  
N(b,"bottle"_IF{b=1 "  
"; "s "}_"of beer on the wall!")^#},  
END;
```

Share

answered [Jun 9, 2009 at 13:46](#)

community wiki  
[17 of 26](#)

---

Ah! \$T was my first language; I have fond memories of it,  
despite its warts ;p – [wkf](#) Jun 10, 2009 at 21:05

---

My office has an introductory guide to Meditech Magic. There's  
an example program that makes sandwiches. – [csj](#) Mar 1, 2010  
at 20:29

---

---

**16** votes    TCL. It only compiles code right before it executes, so it's  
possible that if your code never went down branch A while



testing, and one day, in the field it goes down branch A, it could have a SYNTAX ERROR!

Share

edited Jun 14, 2009 at 1:23

community wiki

2 revs, 2 users 67%

[dicroce](#)

- 
- 14 This is true of most interpretive languages and not just Tcl. And while it's true, in practice it's not nearly as bad as you make it sound. Certainly, I wouldn't use an interpreted language for hyper-critical applications but for the vast majority of uses of this type of language it's not that big of a deal and can be virtually eliminated with proper testing. – [Bryan Oakley](#) Jun 7, 2009 at 16:03
- 
- 2 Exactly what other interpreted languages is it true for? I can't think of one. Most scripting languages completely parse the scripts before executing them. – [Emil H](#) Jun 7, 2009 at 16:19
- 
- 3 It's not up to the language--it's up to the implementation. JavaScript--probably the most-used scripting language in the world--is parsed in different ways depending on the browser or app that uses it. – [Nosredna](#) Jun 7, 2009 at 16:26
- 
- 1 Even if the interpreter parses the whole script file before executing it, 'include' statements still affect which other script files will get parsed, and that is determined at run-time. So the above statements are true even of such languages - it is possible that code path A in a certain file never gets parsed during testing because that file isn't included. – [thomasrutter](#) Jun 9, 2009 at 2:43
- 

TCL is very non-algol-like. In many ways this is bad, but in some ways, since it is so much simpler, this is good. It's very

easy to understand exactly what a bit of code will do, simply because the parser is so stupid. All things combined, it about breaks even. – [SingleNegationElimination](#) Jun 10, 2009 at 6:59

---

---

1

2

3

4

Next