Why learn Perl, Python, Ruby if the company is using C++, C# or Java as the application language? [closed]

Asked 16 years, 3 months ago Modified 13 years, 2 months ago

Viewed 33k times

67

Closed. This question needs to be more focused. It is not currently accepting answers.

Want to improve this question? Update the question so it focuses on one problem only by editing this post.

Closed 11 years ago.

Improve this question

I wonder why would a C++, C#, Java developer want to learn a dynamic language?

Assuming the company won't switch its main development language from C++/C#/Java to a dynamic one what use is there for a dynamic language?

What helper tasks can be done by the dynamic languages faster or better after only a few days of learning than with the static language that you have been using for several years?

Update

After seeing the first few responses it is clear that there are two issues. My main interest would be something that is justifiable to the employer as an expense. That is, I am looking for justifications for the employer to finance the learning of a dynamic language. Aside from the obvious that the employee will have broader view, the employers are usually looking for some "real" benefit.

c# java python ruby perl

Share

Improve this question

Follow

edited May 20, 2010 at 8:15

community wiki
5 revs, 2 users 96%
szabgab

11 If a question has dozens of upvotes just leave the damn thing open. Clearly it's useful to someone. Isn't that the whole point of this website. Stop just reflexively closing things. – Ethan Aug 11, 2013 at 19:37

35 Answers

Sorted by: Highest score (default)

1 2 Next

A lot of times some quick task comes up that isn't part of the main software you are developing. Sometimes the task is one off ie compare this file to the database and let me know the differences. It is a lot easier to do text parsing in Perl/Ruby/Python than it is in Java or C# (partially because it is a lot easier to use regular expressions). It will probably take a lot less time to parse the text file using Perl/Ruby/Python (or maybe even vbscript *cringe* and then load it into the database than it would to create a Java/C# program to do it or to do it by hand.

Also, due to the ease at which most of the dynamic languages parse text, they are great for code generation. Sure your final project must be in C#/Java/Transact SQL but instead of cutting and pasting 100 times, finding errors, and cutting and pasting another 100 times it is often (but not always) easier just to use a code generator.

A recent example at work is we needed to get data from one accounting system into our accounting system. The system has an import format, but the old system had a completely different format (fixed width although some things had to be matched). The task is not to create a program to migrate the data over and over again. It is to shove the data into our system and then maintain it there going forward. So even though we are a C# and SQL Server shop, I used Python to convert the data into the format that could be imported by our application. Ultimately it doesn't matter that I used python, it matters

that the data is in the system. My boss was pretty impressed.

Where I often see the dynamic languages used for is testing. It is much easier to create a Python/Perl/Ruby program to link to a web service and throw some data against it than it is to create the equivalent Java program. You can also use python to hit against command line programs, generate a ton of garbage (but still valid) test data, etc.. quite easily.

The other thing that dynamic languages are big on is code generation. Creating the C#/C++/Java code. Some examples follow:

The first code generation task I often see is people using dynamic languages to maintain constants in the system. Instead of hand coding a bunch of enums, a dynamic language can be used to fairly easily parse a text file and create the Java/C# code with the enums.

SQL is a whole other ball game but often you get better performance by cut and pasting 100 times instead of trying to do a function (due to caching of execution plans or putting complicated logic in a function causing you to go row by row instead of in a set). In fact it is quite useful to use the table definition to create certain stored procedures automatically.

It is always better to get buy in for a code generator. But even if you don't, is it more fun to spend time cutting/pasting or is it more fun to create a

Perl/Python/Ruby script once and then have that generate the code? If it takes you hours to hand code something but less time to create a code generator, then even if you use it once you have saved time and hence money. If it takes you longer to create a code generator than it takes to hand code once but you know you will have to update the code more than once, it may still make sense. If it takes you 2 hours to hand code, 4 hours to do the generator but you know you'll have to hand code equivalent work another 5 or 6 times than it is obviously better to create the generator.

Also some things are easier with dynamic languages than Java/C#/C/C++. In particular regular expressions come to mind. If you start using regular expressions in Perl and realize their value, you may suddenly start making use of the Java regular expression library if you haven't before. If you have then there may be something else.

I will leave you with one last example of a task that would have been great for a dynamic language. My work mate had to take a directory full of files and burn them to various cd's for various customers. There were a few customers but a lot of files and you had to look in them to see what they were. He did this task by hand....A Java/C# program would have saved time, but for one time and with all the development overhead it isn't worth it. However slapping something together in Perl/Python/Ruby probably would have been worth it. He spent several hours doing it. It would have taken less than one to create the Python script to inspect each file,

match which customer it goes to, and then move the file to the appropriate place.....Again, not part of the standard job. But the task came up as a one off. Is it better to do it yourself, spend the larger amount of time to make Java/C# do the task, or spend a much smaller amount of time doing it in Python/Perl/Ruby. If you are using C or C++ the point is even more dramatic due to the extra concerns of programming in C or C++ (pointers, no array bounds checking, etc.).

Share Improve this answer Follow

edited Sep 27, 2011 at 13:54

community wiki
4 revs, 3 users 92%
Cervo

I've actually recently come across the exact problem you're describing. I was given a PDF and needed to convert the contents to an XML syntax. I was able to do it in PHP in less than an hour, and I ensured that there were no typos. Far easier than copying and pasting. – cwallenpoole Sep 17, 2009 at 14:39

I feel like this chart is relevant to your answer – Kris Welsh Jul 21, 2013 at 23:45

21

Let me turn your question on its head by asking what use it is to an American English speaker to learn another language?

The languages we speak (and those we program in) inform the way we think. This can happen on a fundamental level, such as c++ versus javascript versus lisp, or on an implementation level, in which a ruby construct provides a eureka moment for a solution in your "real job."

Speaking of your real job, if the market goes south and your employer decides to "right size" you, how do you think you'll stack up against a guy who is flexible because he's written software in tens of languages, instead of your limited exposure? All things being equal, I think the answer is clear.

Finally, you program for a living because you love programming... right?

Share Improve this answer Follow

answered Sep 17, 2008 at 15:18

community wiki
Pete Michaud

6 true 30% of the time... – Haoest Sep 17, 2008 at 20:17

14

I don't think anyone has mentioned this yet. Learning a new language can be fun! Surely that's a good enough reason to try something new.

Share Improve this answer

answered Sep 18, 2008 at 1:46

Follow

community wiki
epochwolf

1 Agreed. I was programming almost exclusively in C# before one day seeing a Perl tutorial and trying it. I could not believe how fun it was! – neo2862 Aug 30, 2009 at 11:46

I tried Python from a Java point of view. It is just awesome :-) – Martin Ueding Apr 7, 2011 at 18:08

9

I primarily program in Java and C# but use dynamic languages (ruby/perl) to support smoother deployment, kicking off OS tasks, automated reporting, some log parsing, etc.

After a short time learning and experimenting with ruby or perl you should be able to write some regex manipulating scripts that can alter data formats or grab information from logs. An example of a small ruby/perl script that could be written quickly would be a script to parse a very large log file and report out only a few events of interest in either a human readable format or a csv format.

Also, having experience with a variety of different programming languages should help you think of new ways to tackle problems in more structured languages like Java, C++, and C#.

Share Improve this answer answered Sep 17, 2008 at 15:24

Follow

community wiki
Alex B

7

One big reason to learn Perl or Ruby is to help you automate any complicated tasks that you have to do over and over.

Or if you have to analyse contents of log files and you need more mungeing than available using grep, sed, etc.

Also using other languages, e.g. Ruby, that don't have much "setup cost" will let you quickly prototype ideas before implementing them in C++, Java, etc.

HTH

cheers,

Rob

Share Improve this answer

answered Sep 17, 2008 at 15:31

Follow

community wiki
Rob Wells

Even though my workplace is all c++ or embedded c, I use Ruby for exactly this kind of stuff. Log file processing, quick prototyping, test stub stand-ins for the other end of a TCP interface, etc. – AShelly Sep 17, 2008 at 18:30

5

Do you expect to work for this company forever? If you're ever out on the job market, pehaps some prospective employers will be aware of the Python paradox.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:18

community wiki
Chris Upchurch

+1 for the link on the Python Paradox. – Eric O. Lebigot May 20, 2010 at 11:25

5

A good hockey player plays where the puck is. A great hockey player plays where the puck is going to be. - Wayne Gretzky

Our industry is always changing. No language can be mainstream forever. To me Java, C++, .Net is where the puck is right now. And python, ruby, perl is where the puck is going to be. Decide for yourself if you wanna be good or great!

Share Improve this answer Follow

answered Sep 18, 2008 at 13:43

community wiki
liangzan

You left comment in 2008 year, now is 2013, and puck is still on Java, C# side. – svlada Mar 21, 2013 at 19:28

5

Paul Graham posted an article several years ago about why Python programmers made better Java programmers. (http://www.paulgraham.com/pypar.html)

Basically, regardless of whether the new language is relevant to the company's current methodology, learning a new language means learning new ideas. Someone who is willing to learn a language that isn't considered "business class" means that he is interested in programming, beyond just earning a paycheck.

To quote Paul's site:

> And people don't learn Python because it will get them a job; they learn it because they genuinely like to program and aren't satisfied with the languages they already know.
>
> Which makes them exactly the kind of programmers companies should want to hire. Hence what, for lack of a better name, I'll call the

> Python paradox: if a company chooses to write its software in a comparatively esoteric language, they'll be able to hire better programmers, because they'll attract only those who cared enough to learn it. And for programmers the paradox is even more pronounced: the language to learn, if you want to get a good job, is a language that people don't learn merely to get a job.

If an employer was willing to pay for the cost of learning a new language, chances are the people who volunteered to learn (assuming it wasn't a mandatory class) would be the same people to are already on the "fast track".

Share Improve this answer Follow

answered Sep 22, 2008 at 7:33

community wiki
crystalattice

4

When I first learned Python, I worked for a Java shop. Occasionally I'd have to do serious text-processing tasks which were much easier to do with quick Python scripts than Java programs. For example, if I had to parse a complex CSV file and figure out which of its rows corresponded to rows in our Oracle database, this was much easier to do with Python than Java.

More than that, I found that learning Python made me a much better Java programmer; having learned many of the same concepts in another language I feel that I understand those concepts much better. And as for what makes Python easier than Java, you might check out this question: Java -> Python?

Share Improve this answer Follow

edited May 23, 2017 at 12:18

community wiki
2 revs
Eli Courtwright

3

Edit: I wrote this before reading the update to the original question. See my other answer for a better answer to the updated question. I will leave this as is as a warning against being the fastest gun in the west =)

Over a decade ago, when I was learning the ways of the Computer, the Old Wise Men With Beards explained how C and C++ are the tools of the industry. No one used Pascal and only the foolhardy would risk their companies with assembler.

And of course, **no one would even mention the awful slow ugly thing called Java**. It will not be a tool for serious business.

So. Um. Replace the languages in the above story and perhaps you can predict the future. Perhaps you can't.

Point is, Java will not be the Last Programming Language ever and also you will most likely switch employers as well. The future is charging at you 24 hours per day. Be prepared.

Learning new languages is good for you. Also, in some cases it can give you bragging rights for a long time. My first university course was in Scheme. So when people talk to me about the new *language du jour*, my response is something like "First-class functions? That's so last century."

And of course, you get **more stuff done** with a high-level language.

edited Sep 17, 2008 at 17:33

community wiki
2 revs
Antti Rasinen

2

Learning a new language is a long-term process. In a couple of days you'll learn the basics, yes. But! As you probably know, the real practical applicability of any language is tied to the standard library and other available components. Learning how to use the efficiently requires a lot of hands-on experience.

Perhaps the only immediate short-term benefit is that developers learn to distinguish the nails that need a

Python/Perl/Ruby -hammer. And, if they are any good, they can then study some more (online, perhaps!) and become real experts.

The long-term benefits are easier to imagine:

1. The employee becomes a better developer. Better developer => better quality. We are living in a knowledge economy these days. It's wiser to invest in those brains that already work for you.
2. It is easier to adapt when the next big language emerges. It is very likely that the NBL will have many of the features present in today's scripting languages: first-class functions, closures, streams/generators, etc.
3. New market possibilities and ability to respond more quickly. Even if you are not writing Python, *other people are*. Your clients? Another vendor in the project? Perhaps a critical component was written in some other language? It will cost money and time, if you do not have people who can understand the code and interface with it.
4. Recruitment. If your company has a reputation of teaching new and interesting stuff to people, it will be easier to recruit the top people. *Everyone* is doing Java/C#/C++. It is not a very effective way to differentiate yourself in the job market.

answered Sep 17, 2008 at 17:59

community wiki
Antti Rasinen

2

Towards answering the updated question, its a chicken/egg problem. The best way to justify an expense is to show how it reduces a cost somewhere else, so you may need to spend some extra/personal time to learn something first to build some kind of functional prototype.

Show your boss a demo like "hey, i did this thing, and it saves me this much time [or better yet, this much $$], imagine if everyone could use this how much money we would save"

and then after they agree, explain how it is some other technology and that it is worth the expense to get more training, and training for others on how to do it better.

Share Improve this answer Follow

answered Sep 17, 2008 at 20:12

community wiki
John Gardner

1

I have often found that learning another language, especially a dynamically typed language, can teach you things about other languages and make you an overall better programmer. Learning ruby, for example, will teach you Object Oriented programming in ways Java wont,

and vice versa. All in all, I believe that it is better to be a well rounded programmer than stuck in a single language. It makes you more valuable to the companies/clients you work for.

Share Improve this answer

Follow

answered Sep 17, 2008 at 15:20

community wiki
Mike Farmer

1

check out the answers to this thead:

https://stackoverflow.com/questions/76364/what-is-the-single-most-effective-thing-you-did-to-improve-your-programming-ski#84112

Learning new languages is about keeping an open mind and learning new ways of doing things.

Share Improve this answer

Follow

edited May 23, 2017 at 11:54

community wiki
2 revs
Jean

Im not sure if this is what you are looking for, but we write our main application with Java at the small company I work for, but have used python to write smaller scripts

1

quickly. Backup software, temporary scripts to manipulate data and push out results. It just seems easier sometimes to sit down with python and write a quick script than mess with classes and stuff in java.

Temp scripts that aren't going to stick around don't need a lot of design time wasted on them.

And I am lazy, but it is good to just learn as much as you can of course and see what features exist in other languages. Knowing more never hurts you in future career changes :)

Share Improve this answer Follow

answered Sep 17, 2008 at 15:23

community wiki
Arthur Thomas

1

It's all about broadening your horizons as a developer. If you limit yourself to only strong-typed languages, you may not end up the best programmer you could.

As for tasks, Python/Lua/Ruby/Perl are great for small simple tasks, like finding some files and renaming them. They also work great when paired with a framework (e.g. Rails, Django, Lua for Windows) for developing simple apps quickly. Hell, 37Signals is based on creating simple yet very useful apps in Ruby on Rails.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:24

community wiki
Ed Schwehm

1

They're useful for the "Quick Hack" that is for plugging a gap in your main language for a quick (and potentially dirty) fix faster than it would take to develop the same in your main language. An example: a simple script in perl to go through a large text file and replace all instances of an email address with another is trivial with an amount of time taken in the 10 minute range. Hacking a console app together to do the same in your main language would take multiples of that.

You also have the benefit that exposing yourself to additional languages broadens your abilities and learning to attack problems from a different languages perspective can be as valuable as the language itself.

Finally, scripting languages are very useful in the realm of extension. Take LUA as an example. You can bolt a lua interpreter into your app with very little overhead and you now have a way to create rich scripting functionality that can be exposed to end users or altered and distributed quickly without requiring a rebuild of the entire app. This is used to great effect in many games most notably World of Warcraft.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:25

community wiki
Wolfwyrd

1

Personally I work on a Java app, but I couldn't get by without perl for some supporting scripts.

I've got scripts to quickly flip what db I'm pointing at, scripts to run build scripts, scripts to scrape data & compare stuff.

Sure I *could* do all that with java, or maybe shell scripts (I've got some of those too), but who wants to compile a class (making sure the classpath is set right etc) when you just need something quick and dirty. Knowing a scripting language can remove 90% of those boring/repetitive manual tasks.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:25

community wiki
shelfoo

1

Learning something with a flexible OOP system, like Lisp or Perl (see Moose), will allow you to better expand and understand your thoughts on software engineering. Ideally, every language has some unique facet (whether it

be CLOS or some other technique) that enhances, extends and grows your abilities as a programmer.

Share Improve this answer Follow

answered Sep 17, 2008 at 16:42

community wiki
jshirley

1

If all you have is a hammer, every problem begins to look like a nail.

There are times when having a screwdriver or pair of pliers makes a complicated problem trivial.

Nobody asks contractors, carpenters, etc, "Why learn to use a screwdriver if i already have a hammer?". Really good contractors/carpenters have tons of tools and know how to use them well. All programmers should be doing the same thing, learning to use new tools and use them well.

> But before we use any power tools, lets take a moment to talk about shop safety. Be sure to read, understand, and follow all the safety rules that come with your power tools. Doing so will greatly reduce the risk of personal injury. And remember this: there is no more important rule than to wear these: safety glasses -- Norm

Share Improve this answer

Follow

answered Sep 17, 2008 at 17:41

community wiki
John Gardner

1

I think the main benefits of dynamic languages can be boiled down to

1. Rapid development
2. Glue

The short design-code-test cycle time makes dynamic languages ideal for prototyping, tools, and quick & dirty one-off scripts. IMHO, the latter two can make a huge impact on a programmer's productivity. It amazes me how many people trudge through things manually instead of whipping up a tool to do it for them. I think it's because they don't have something like Perl in their toolbox.

The ability to interface with just about anything (other programs or languages, databases, etc.) makes it easy to reuse existing work and automate tasks that would otherwise need to be done manually.

Share Improve this answer Follow

answered Sep 17, 2008 at 19:25

community wiki
Michael Carman

1

Don't tell your employer that you want to learn Ruby. Tell him you want to learn about the state-of-the-art in web framework technologies. it just happens that the hottest ones are Django and Ruby on Rails.

Share Improve this answer Follow

answered Sep 19, 2008 at 0:20

community wiki
ykaganovich

1

I have found the more that I play with Ruby, the better I understand C#.

1) As you switch between these languages that each of them has their own constructs and philosophies behind the problems that they try to solve. This will help you when finding the **right tool for the job** or the **domain of a problem**.

2) The **role of the compiler** (or interpreter for some languages) becomes more prominent. Why is Ruby's type system differ from the .Net/C# system? What problems do each of these solve? You'll find yourself understanding at a lower level the constructs of the compiler and its influence on the language

3) Switching between Ruby and C# really helped me to understand **Design Patterns** better. I really suggest implementing common design patterns in a language like C# and then in a language like Ruby. It often helped me see through some of the compiler ceremony to the philosophy of a particular pattern.

4) A different **community**. C#, Java, Ruby, Python, etc all have different communities that can help engage your

abilities. It is a great way to take your craft to the next level.

5) Last, but not least, because new languages are fun :)

Share Improve this answer Follow

answered May 20, 2010 at 4:07

community wiki
Jimmy Lyke

1

Given the increasing focus to running dynamic languages (da-vinci vm etc.) on the JVM and the increasing number of dynamic languages that do run on it (JRuby, Grrovy, Jython) I think the usecases are just increasing. Some of the scenarios I found really benifited are

1. **Prototyping**- use RoR or Grails to build quick prototypes with advantage of being able to runn it on the standard app server and (maybe) reuse existing services etc.
2. **Testing**- right unit tests much much faster in dynamic languages
3. **Performance/automation test scripting**- some of these tools are starting to allow the use standard dynamic language of choice to write the test scripts instead of proprietary script languages. Side benefit might be to the able to reuse some unit test code you've already written.

Share Improve this answer Follow

edited Aug 2, 2011 at 5:39

community wiki
2 revs, 2 users 86%
mataal

0

Philosophical issues aside, I know that I have gotten value from writing quick-and-dirty Ruby scripts to solve brute-force problems that Java was just too big for. Last year I had three separate directory structures that were all more-or-less the same, but with lots of differences among the files (the client hadn't heard of version control and I'll leave the rest to your imagination).

It would have taken a great deal of overhead to write an analyzer in Java, but in Ruby I had one working in about 40 minutes.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:20

community wiki
Jim Kiley

0

Often, dynamc languages (especially python and lua) are embedded in programs to add a more plugin-like functionality and because they are high-level languages

that make it easy to add certain behavior, where a low/mid-level language is not needed.

Lua specificially lacks all the low-level system calls because it was designed for easeof-use to add functionality within the program, not as a general programming language.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:22

community wiki
Christian P.

0

You should also consider learning a functional programming language like Scala. It has many of the advantages of Ruby, including a concise syntax, and powerful features like closures. But it compiles to Java class files and and integrate seamlessly into a Java stack, which may make it much easier for your employer to swallow.

Scala isn't dynamically typed, but its "implicit conversion" feature gives many, perhaps even all of the benefits of dynamic typing, while retaining many of the advantages of static typing.

Share Improve this answer Follow

answered Sep 17, 2008 at 15:36

community wiki
sanity

0

Dynamic languages are fantastic for prototyping ideas. Often for performance reasons they won't work for permanent solutions or products. *But*, with languages like Python, which allow you to embed standard C/C++/Java inside them or visa versa, you can speed up the really critical bits but leave it glued together with the flexibility of a dynamic language.

...and so you get the best of both worlds. If you need to justify this in terms of why more people should learn these languages, just point out much faster you can develop the same software and how much more robust the solution is (because debugging/fixing problems in dynamic languages is in my experience, considerably easier!).

Share Improve this answer Follow

answered Sep 17, 2008 at 17:48

community wiki
Jon Cage

0

Knowing grep and ruby made it possible to narrow down a problem, and verify the fix for, an issue involving tons of java exceptions on some production servers. Because I threw the solution together in ruby, it was done (designed, implemented, tested, run, bug-fixed, re-run, enhanced,

results analyzed) in an afternoon instead of a couple of days. I could have solved the same problem using an all-java solution or a C# solution, but it most likely would have taken me longer.

Having dynamic language expertise also sometimes leads you to simpler solutions in less dynamic languages. In ruby, perl or python, you just intuitively reach for associative arrays (hashes, dictionaries, whatever word you want to use) for the smallest things, where you might be tempted to create a complex class hierarchy in a statically typed language when the problem doesn't necessarily demand it.

Plus you can plug in most scripting languages into most runtimes. So it doesn't have to be either/or.

Share Improve this answer Follow

answered Sep 17, 2008 at 17:58

community wiki
JasonTrue

0

The "real benefit" that an employer could see is a better programmer who can implement solutions faster; however, you will not be able to provide any hard numbers to justify the expense and an employer will most likely have you work on what makes money now as opposed to having you work on things that make the future better.

The only time you can get training on the employer's dime, is when they perceive a need for it and it's cheaper than hiring a new person who already has that skill-set.

Share Improve this answer

Follow

answered Sep 17, 2008 at 18:01

community wiki
Dan

1 2 Next