

# Why JavaScript rather than a standard browser virtual machine?

Asked 16 years, 3 months ago   Modified 4 years, 7 months ago

Viewed 18k times



**170**



Would it not make sense to support a set of languages (Java, Python, Ruby, etc.) by way of a standardized virtual machine hosted in the browser rather than requiring the use of a specialized language -- really, a specialized paradigm -- for client scripting only?

To clarify the suggestion, a web page would contain byte code instead of any higher-level language like JavaScript.

I understand the pragmatic reality that JavaScript is simply what we have to work with now due to evolutionary reasons, but I'm thinking more about the long term. With regard to backward compatibility, there's no reason that inline JavaScript could not be simultaneously supported for a period of time and of course JavaScript could be one of the languages supported by the browser virtual machine.

javascript

Share

edited Jun 9, 2016 at 9:55

Improve this question

Follow

community wiki  
9 revs, 3 users 100%  
[newdayrising](#)

- 
- 18 I don't know why this is getting voted down. I thought it was a good question! – [user4903](#) Sep 17, 2008 at 20:47
- 
- 11 Because it's more of a complaint than a question. – [Dustman](#) Sep 22, 2008 at 19:51
- 
- 6 It's due to the idea that javascript isn't a real language, or isn't as good as other languages. Neither of these have been true since the early days, yet the 'dirty' perception presently persists. – [Adam Davis](#) Oct 7, 2008 at 21:49
- 
- 45 Wow, I have never seen the SO community fail so completely. The only answers that even address the idea proposed here are all the way to the bottom, getting downvoted, while the answers needlessly "defending JS" are getting all the love. This question doesn't attack JS, it is merely advocating choice. It's simply saying: "whatever you may think of JS, wouldn't it be nice to be able to use some other languages as well if you prefer them?". What is wrong with you? – [Jordi](#) Dec 17, 2010 at 7:24
- 
- 4 This is a major problem with StackOverflow allowing for edits so far into the future after several answers have been provided. The original question asked is more relevant to the top answers, while the rest address the "new spirit" of the question after the edits. – [user4903](#) Dec 18, 2010 at 2:57
- 

32 Answers

Sorted by:

Highest score (default)



1

2

Next



29



Well, yes. Certainly if we had a time machine, going back and ensuring a lot of the Javascript features were designed differently would be a major pastime (that, and ensuring the people who designed IE's CSS engine never went into IT). But it's not going to happen, and we're stuck with it now.

I suspect, in time, it will become the "Machine language" for the web, with other better designed languages and APIs compile down to it (and cater for different runtime engine foibles).

I don't think, however, any of these "better designed languages" will be Java, Python or Ruby. Javascript is, despite the ability to be used elsewhere, a Web application scripting language. Given that use case, we can do better than any of those languages.

Share Improve this answer

edited Dec 17, 2010 at 13:07

Follow

community wiki

2 revs, 2 users 89%

Adam Wright

---

54 -1 for the IE CSS team remark. IE6 wasn't bad when it was released, it's main competitor was the buggiest piece of crap software that has ever been written. Person attacks, although sometimes fun, don't make the world a better place.  
– [erikkallen](#) Jan 16, 2010 at 14:52

---

- 5 Couldn't disagree with your assessment of JavaScript as "...a Web application scripting language..." less. It's a great, flexible language with a lot more applicability than that.  
– [T.J. Crowder](#) Mar 11, 2010 at 7:54
- 
- 2 @T.J. Crowder: ITYM "Couldn't disagree [...] more."?  
– [Christoffer Hammarström](#) Dec 17, 2010 at 9:29 ✎
- 
- 3 @Jaroslaw Szpilewski Shameless JS zealotism? Did you miscomment on this, thinking it another post? Also, for @erikkallen, the IE CSS team comment was what's commonly known as "a joke". – [Adam Wright](#) Dec 17, 2010 at 16:58
- 
- 2 Some "clairvoyance" in this answer: we now have CoffeeScript. – [andref](#) May 14, 2011 at 14:10
- 



21



I think JavaScript is a good language, but I would love to have a choice when developing client-side web applications. For legacy reasons we're stuck with JavaScript, but there are projects and ideas looking for changing that scenario:



1. [Google Native Client](#): technology for running native code in the browser.
2. [Emscripten](#): LLVM bytecode compiler to javascript. Allows LLVM languages to run in the browser.
3. Idea: .NET CLI in the browser, by the creator of Mono: <http://tirania.org/blog/archive/2010/May-03.html>

I think we will have JavaScript for a long time, but that will change sooner or later. There are so many developers

willing to use other languages in the browser.

Share Improve this answer

edited May 26, 2015 at 13:43


Follow

community wiki

2 revs, 2 users 95%

Manuel Ceron

---

LLVM could be an answer to all of this. There are already a big number of languages (Python, Ruby, even Java) with support for compiling to LLVM, and LLVM can compile to Javascript, so at the very least we could get automatic LLVM support in browsers simply by compiling to JS. Of course LLVM can't be optimized for all programming paradigms and specific languages, so performance won't be the same as 100% native, but Javascript's JITs/Interpreters, even taking into account recent advances, have always been slow relative to native anyway. – [Pepe Mandioca](#) Oct 12, 2015 at 2:14 

---



Answering the *question* - No, it would not make sense.

18



Currently the closest things we have to a multi-language VM are the JVM and the CLR. These aren't exactly lightweight beasts, and it would not make sense to try and embed something of this size and complexity in a browser.



Let's examine the idea that you could write a new, multilanguage VM that would be better than the existing solution.

- You're behind on stability.
- You're behind on complexity (way, way, behind because you're trying to generalize over multiple languages)
- You're behind on adoption

So, no, it doesn't make sense.

Remember, in order to support these languages you're going to have to strip down their APIs something fierce, chopping out any parts that don't make sense in the context of a browser script. There are a huge number of design decisions to be made here, and a huge opportunity for error.

In terms of functionality, we're probably only *really* working with the DOM anyway, so this is really an issue of syntax and language idiom, at which point it does make sense to ask, "Is this really worth it?"

Bearing in mind, the *only* thing we're talking about is client side scripting, because server side scripting is already available in whatever language you like. It's a relatively small programming arena and so the benefit of bringing multiple languages in is questionable.

What languages would it make sense to bring in?  
(Warning, subjective material follows)

Bringing in a language like C doesn't make sense because it's made for working with metal, and in a

browser there isn't much metal really available.

Bringing in a language like Java doesn't make sense because the best thing about it is the APIs anyway.

Bringing in a language like Ruby or Lisp doesn't make sense because JavaScript is a powerful dynamic language very close to Scheme.

Finally, what browser maker really wants to support DOM integration for multiple languages? Each implementation will have its own specific bugs. We've already walked through fire dealing with differences between MS Javascript and Mozilla Javascript and now we want to multiply that pain five or six-fold?

It doesn't make sense.

Share Improve this answer

answered [Dec 17, 2010 at 16:28](#)

Follow

community wiki  
[the happy moron](#)

---

Quite a subjective answer, but I didn't want to vote down as you raise some good points (and the original answer is more like discussion starter anyway). – [Gerbrand](#) Dec 18, 2010 at 13:52

---

- 2 I don't think the VM in browser is necessary to heavy. Of course it already exists as Silverlight and Applets. The latter failed to gain popularity, I think mostly because of bad timing and technical stupidities which are mostly resolved. Allowing any language between the script tag (with restrictions) would

be pretty cool, and certainly not impossible nor unpractical.

– [Gerbrand](#) Dec 18, 2010 at 13:55

---

- 2 Heaviness (MB)? Probably okay. Heaviness (complexity)? Way too heavy. Any multi-language VM you have, you will have language implementations sitting on top (eg. JRuby/IronRuby, Clojure, Jython/IronPython), etc. Either the JVM eats the complexity or the language implementers do.

– [the happy moron](#) Dec 20, 2010 at 20:57

---

It would take a staggering amount of work to re-implement multiple languages for multiple brand new platforms (IE/Firefox/Safari...). And languages change, too. "This page only visible in a Ruby 1.9 browser"? Please, no.

– [the happy moron](#) Dec 20, 2010 at 21:07

---

- 4 I don't think you're answering the question properly, you are only stating why you think other languages aren't suited for what javascript does in the browser at this moment. A universal bytecode suited for the web would be something other languages compile into, if that language is useful would be up to its creator not the web-bytecode, many languages already do this btw by compiling into javascript (ie, dart).

– [Timotheus](#) Jan 5, 2014 at 11:08

---



14



On Windows, you can register other languages with the Scripting Host and have them available to IE. For example VBScript is supported out of the box (though it has never gained much popularity as it is for most purposes even worse than JavaScript).



The Python win32 extensions allowed one to add Python to IE like this quite easily, but it wasn't really a good idea as Python is quite difficult to sandbox: many language



features expose enough implementation hooks to allow a supposedly-restricted application to break out.

It is a problem in general that the more complexity you add to a net-facing application like the browser, the greater likelihood of security problems. A bunch of new languages would certainly fit that description, and these are new languages that are also still developing fast.

JavaScript is an ugly language, but through careful use of a selective subset of features, and support from suitable object libraries, it can generally be made fairly tolerable. It seems incremental, practical additions to JavaScript are the only way web scripting is likely to move on.

Share Improve this answer

answered [Sep 17, 2008 at 20:40](#)

Follow

community wiki  
[bobince](#)



**12**

I would definitely welcome a standard language independent VM in browsers (I would prefer to code in a statically typed language).



(Technically) It's quite doable gradually: first one major browser supports it and server has the possibility to either send bytecode if current request is from compatible browser or translate the code to JavaScript and send plain-text JavaScript.

There already exist some experimental languages that compile to JavaScript, but having a defined VM would (maybe) allow for better performance.

I admit that the "standard" part would be quite tricky, though. Also there would be conflicts between language features (eg. static vs. dynamic typing) concerning the library (assuming the new thing would use same library). Therefore I don't think it's gonna happen (soon).

Share Improve this answer

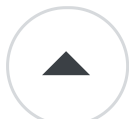
edited Dec 17, 2010 at 7:13

Follow

community wiki

2 revs

Aivar



10



If you feel like you are getting your hands dirty, then you have either been brainwashed, or are still feeling the after affects of the "DHTML years". JavaScript is very powerful, and is suited well for its purpose, which is to script interactivity client side. This is why JavaScript 2.0 got such a bad rap. I mean, why packages, interfaces, classes, and the like, when those are clearly aspects of server-side languages. JavaScript is just fine as a prototype-based language, without being full-blown object oriented.

If there is a lack of seamlessness to your applications because the server-side and client-side are not communicating well, then you might want to reconsider

how you architect your applications. I have worked with extremely robust Web sites and Web applications, and I have never once said, "Hmm, I really wish JavaScript could do (xyz)." If it could do that, then it wouldn't be JavaScript -- it would be ActionScript or AIR or Silverlight. I don't need that, and neither do most developers. Those are nice technologies, but they try to solve a problem with a technology, not a... well, a solution.

Share Improve this answer

edited Sep 17, 2008 at 20:48

Follow

community wiki

2 revs

user4903

---

13 How can you say, that JavaScript isn't full-blown object oriented? It's certainly one of the most object-oriented languages that I know of. Everything in JavaScript is an object - even functions. It's just that OOP in JavaScript doesn't look like OOP in many other languages.

– [Rene Saarsoo](#) Sep 18, 2008 at 20:23

---

2 OOP is not inherent to JavaScript. You don't have packages, interfaces, abstract classes, or method overloading built into the core, and you can't extend an object -- only an object's prototype, which makes it technically prototype-based, not OOP based. – user4903 Sep 19, 2008 at 16:14

---

3 Dead wrong on that one. "Prototype" is a Design Pattern (Gamma et. al., pp 117 - 126). As you'll recall Design Patterns revolve around common designs in Object Oriented Programming. And just because the language doesn't have the same features as other OOP languages doesn't mean it isn't OOP. – [Dustman](#) Sep 22, 2008 at 19:50

---

13 You're not dead wrong, I think the best way to put it is that JS is not class based OO, it's prototypal OO. – [Ruan Mendes](#) Dec 3, 2010 at 0:22

---

14 1. Javascript is fully OOP. OOP is about objects, not about classes. 2. You can extend an object in JavaScript, that's the whole point of Prototypal object oriented programming. 3. You're not answering the question, the question is not attacking JS, is just asking for choice. I think JS is a great language, but I would love to have other choices when programming in the browser. – [Manuel Ceron](#) Dec 17, 2010 at 9:44

---



7



I don't think that a standard web VM is that inconceivable. There are a number of ways you could introduce a new web VM standard gracefully and with full legacy support, as long as you ensure that any VM bytecode format you use can be quickly decompiled into javascript, and that the resulting output will be reasonably efficient (I would even go so far as to guess that a smart decompiler would probably generate better javascript than any javascript a human could produce themselves).

With this property, any web VM format could be easily decompiled either on the server (fast), on the client (slow, but possible in cases where you have limited control of

the server), or could be pre-generated and loaded dynamically by either the client or the server (fastest) for browsers that don't natively support the new standard.

Those browsers that DO natively support the new standard would benefit from increased speed of the runtime for web vm based apps. On top of that, if browsers base their legacy javascript engines on the web vm standard (i.e. parsing javascript into the web vm standard and then running it), then they don't have to manage two runtimes, but that's up to the browser vendor.

Share Improve this answer

answered Jul 5, 2011 at 17:31

Follow

community wiki

[Jeremy Bell](#)



6



While Javascript is the only well-supported scripting language you can control the page directly from, Flash has some very nice features for bigger programs. Lately it has a JIT and can also generate bytecode on the fly (check out [runtime expression evaluation](#) for an example where they use flash to compile user-input math expressions all the way to native binary). The Haxe language gives you static typing with inference and with the bytecode generation abilities you could implement almost any runtime system of your choice.

community wiki

[jjrv](#)

---

What am I missing with the question? It seems Flash would do exactly what he wants. We're not talking about another native language, he wants a VM, right? Good answer.

– [mwilcox](#) Dec 17, 2010 at 16:49

---



Quick update on this old question.

6



Everyone who affirmed that a "web page would contain byte code instead of any higher-level language like JavaScript" "won't happen".



June 2015 the [W3C](#) announced [WebAssembly](#) that is



a new portable, size- and load-time-efficient format suitable for compilation to the web.

This is still experimental, but there is already some prototypal implementation in Firefox nightly and Chrome Canary and there is already [some demonstration working](#).

Currently, WebAssembly is mostly designed to be produced from C/C++, however

as WebAssembly evolves it will support more languages than C/C++, and we hope that other compilers will support it as well.

I let you have a closer look at the [official page](#) of the project, it is truly exciting!

Share Improve this answer

answered Jun 9, 2016 at 9:53

Follow

community wiki  
[Quentin Roy](#)



this question resurfaces regularly. my stance on this is:

5

**A) wont happen and B) is already here.**



pardon, what? let me explain:



**ad A**



a VM is not just some sort of universal magical device. most VMs are optimized for a certain language and certain language features. take the JRE/Java (or LLVM): optimized for static typing, and there are definitely problems and downsides when implementing dynamic typing or other things java didn't support in the first place.

so, the "general multipurpose VM" that supports lots of language features (tail call optimization, static & dynamic

typing, foo bar boo, ...) would be colossal, hard to implement and probably harder to optimize to get good performance out of it. but i'm no language designer or vm guru, maybe i'm wrong: it's actually pretty easy, only nobody had the idea yet? hrm, hrm.

## ad B

already here: there may not be a bytecode compiler/vm, but you don't actually need one. afaik javascript is turing complete, so it should be possible to either:

1. create a translator from language X to javascript (e.g. coffeescript)
2. create a interpreter in javascript that interprets language X (e.g. [brainfuck](#)). yes, performance would be abysmal, but hey, can't have everything.

## ad C

what? there wasn't a point C in the first place!? because there isn't ... yet. google NACL. if anyone can do it, it's google. as soon google gets it working, your problems are solved. only, uh, it may never work, i don't know. the last time i read about it there were some unsolved security problems of the *really* tricky kind.

---

**apart from that:**



- javascript's been there since ~1995 = 15 years. still, browser implementations differ today (although at least it's not insufferable anymore). so, if you start something new yet, you might have a version working cross browser around 2035. at least a working subset. that only differs subtly. and needs compatibility libs and layers. no point in not trying to improve things though.
- also, what about readable source code? i know a lot of companies would prefer not to serve their code as "kind-of" open source. personally, i'm pretty happy i'm able to read the source if i suspect something fishy or want to learn from it. hooray for source code!

Share Improve this answer

answered [Dec 17, 2010 at 10:39](#)

Follow

community wiki  
[stefs](#)



Indeed. Silverlight is effectively just that - a client side .Net based VM.

4

Share Improve this answer

answered [Dec 17, 2010 at 11:25](#)



Follow



community wiki  
[redcalx](#)





4



There are some errors in your reasoning.

1. A standard virtual machine in a standard browser will never be standard. We have 4 browsers, and IE has conflicting interests with regard to 'standard'. The three others are evolving fast but adoption rate of new technologies is slow. What about browsers on phones, small devices, ...
2. The integration of JS in the different browsers and its past history leads you to under-estimating the power of JS. You pledge a standard, but disapprove JS because standard didn't work out in the early years.
3. As told by others, JS is not the same as AIR/.NET/... and the like. JS in its current incarnation perfectly fits its goals.

In the long term, Perl and Ruby could well replace javascript. Yet the adoption of those languages is slow and it is known that they will never take over JS.

Share Improve this answer

edited Aug 13, 2013 at 16:59

Follow

community wiki  
2 revs, 2 users 67%  
ydebilloez



How do you define best? Best for the browser, or best for the developer? (Plus ECMAScript is different than

3

Javascript, but that is a technicality.)



I find that JavaScript can be powerful and elegant at the same time. Unfortunately most developers I have met treat it like a necessary evil instead of a real programming language.



Some of the features I enjoy are:

- treating functions as first class citizens
- being able to add and remove functions to any object at any time (not useful much but mind blowing when it is)
- it is a dynamic language.

It's fun to deal with and it is established. Enjoy it while it is around because while it may not be the "best" for client scripting it is certainly pleasant.

I do agree it is frustrating when making dynamic pages because of browser incompatibilities, but that can be mitigated by UI libraries. That should not be held against JavaScript itself anymore than Swing should be held against Java.

Share Improve this answer

answered [Sep 17, 2008 at 20:08](#)

Follow

community wiki  
[Rontologist](#)

---

+1 - Lets not confuse language issues with browser interpretation of the code. – [JL](#). Dec 17, 2010 at 20:17

---

why should you defend JS, when he simply asked for a bytecode choice? – [Milind R](#) Feb 4, 2015 at 18:39

---



3



JavaScript is the browser's standard virtual machine. For instance, OCaml and Haskell now both have compilers that can output JavaScript. The limitation is not JavaScript the language, the limitation is the browser objects accessible via JavaScript, and the access control model used to ensure you can safely run JavaScript without compromising your machine. The current access controls are so poor, that JavaScript is only allowed very limited access to browser objects for safety reasons. The Harmony project is looking to fix that.

Share Improve this answer

answered [Dec 17, 2010 at 13:59](#)

Follow

community wiki  
[naasking](#)



3



It's a cool idea. Why not take it a step further?

- Write the HTML parser and layout engine (all the complicated bits in the browser, really) in the same VM language
- Publish the engine to the web



- Serve the page with a declaration of which layout engine to use, and its URL

Then we can add features to browsers without having to push new browsers out to every client - the relevant new bits would be loaded dynamically from the web. We could also publish new versions of HTML without all the ridiculous complexity of maintaining backwards compatibility with everything that's ever worked in a browser - compatibility is the responsibility of the page author. We also get to experiment with markup languages other than HTML. And, of course, we can write fancy JIT compilers into the engines, so that you can script your webpages in any language you want.

Share Improve this answer

answered [Dec 18, 2010 at 1:09](#)

Follow

community wiki  
[p-static](#)

---

I can't tell if you're kidding, but your idea is actually really cool. – [Milind R](#) Feb 4, 2015 at 18:45

---



I would welcome any language besides javascript as possible scripting language.

**3**



What would be cool is to use other languages then Javascript. Java would probably not be a great fit



between the tag but languages like Haskell, Clojure, Scala, Ruby, Groovy would be beneficial.

I came across Rubyscript some while ago ...

<http://almaer.com/blog/running-ruby-in-the-browser-via-script-type-text-ruby> and <http://code.google.com/p/ruby-in-browser/>

Still experimental and in progress, but looks promising.

For .Net I just found:

<http://www.silverlight.net/learn/dynamic-languages/> Just found the site out, but looks interesting too. Works even from my [Apple Mac](#).

Don't know how good the above work in providing an alternative for Javascript, but it looks pretty cool at first glance. Potentially, this would allow one to use any Java or .Net framework natively from the browser - within the browser's sandbox.

As for safety, if the language runs inside the JVM (or .Net engine for that matter), the VM will take care of security so we don't have to worry about that - at least not more than we should for anything that runs inside the browser.

Share Improve this answer

answered Dec 18, 2010 at 14:14

Follow

community wiki

[Gerbrand](#)

---



2

Probably, but to do so we'd need to get the major browsers to support them. IE support would be the hardest to get. JavaScript is used because it is the only thing you can count on being available.



Share Improve this answer

answered [Sep 17, 2008 at 19:03](#)



Follow



community wiki  
[Jeff Olhoeft](#)



2

The vast majority of the devs I've spoken to about ECMAScript et. al. end up admitting that the problem isn't the scripting language, it's the ridiculous HTML DOM that it exposes. Conflating the DOM and the scripting language is a common source of pain and frustration regarding ECMAScript. Also, don't forget, IIS can use JScript for server-side scripting, and things like Rhino allow you to build free-standing apps in ECMAScript. Try working in one of these environments with ECMAScript for a while, and see if your opinion changes.



This kind of despair has been going around for some time. I'd suggest you edit this to include, or repost with, specific issues. You may be pleasantly surprised by some of the relief you get.

A old site, but still a great place to start: [Douglas Crockford's site](#).

community wiki

[Dustman](#)

---

i'd be curious to hear more about why the "HTML DOM" is the pain point – [Alex Moore-Niemi](#) Mar 25, 2015 at 2:54

---

**2**

Well, we have already VBScript, don't we? Wait, only IE supports it!

Same for your nice idea of VM. What if I script my page using Lua, and your browser doesn't have the parser to convert it to bytecode? Of course, we could imagine a script tag accepting a file of bytecode, that even would be quite efficient.

But experience shows it is hard to bring something new to the Web: it would take years to adopt a radical new change like this. How many browsers support SVG or CSS3?

Beside, I don't see what you find "dirty" in JS. It can be ugly if coded by amateurs, propagating bad practice copied elsewhere, but masters shown it can be an elegant language too. A bit like Perl: often looks like an obfuscated language, but can be made perfectly readable.



Share Improve this answer

Follow

community wiki  
[PhiLho](#)



2

Check this out <http://www.visitmix.com/Labs/Gestalt/> - lets you use python or ruby, as long as the user has silverlight installed.



Share Improve this answer

answered [Dec 17, 2010 at 14:03](#)

Follow



community wiki  
[mcintyre321](#)

"as long as the user has silverlight installed." well, I can see a flaw in this one. – [Origamiguy](#) Dec 18, 2010 at 0:50

To be honest, it's probably easier to do that than getting Ruby embedded into ie6/7/8/9/ff/chrome/safari. Heck Chrome already includes flash, why not SL! – [mcintyre321](#) Dec 20, 2010 at 12:08



2

This is a very good question.



It's not the problem only in JS, as it is in the lack of good free IDEs for developing larger programs in JS. I know only one that is free: Eclipse. The other good one is Microsoft's Visual Studio, but not free.





Why would it be free? If web browser vendors want to replace desktop apps with online apps (and they want) then they have to give us, the programmers, good dev tools. You can't make 50,000 lines of JavaScript using a simple text editor, JSLint and built-in Google Chrome debugger. Unless you're a macohist.

When Borland made an IDE for Turbo Pascal 4.0 in 1987, it was a revolution in programming. 24 years have passed since. Shamefully, in the year 2011 many programmers still don't use code completion, syntax checking and proper debuggers. Probably because there are so few good IDEs.

It's in the interest of web browser vendors to make proper (FREE) tools for programmers if they want us to build applications with which they can fight Windows, Linux, MacOS, iOS, Symbian, etc.

Share Improve this answer

answered [Jan 3, 2011 at 12:42](#)

Follow

community wiki  
[user561168](#)

---

Visual studio is free, and you also have vs code, Atom, and other great IDEs, I think you just aren't looking hard enough  
– [GideonMax](#) May 7, 2020 at 15:04

---



1

Realistically, Javascript is the only language that any browsers will use for a long time, so while it would be very nice to use other languages, I can't see it happening.



This "standardised VM" you talk of would be very large and would need to be adopted by all major browsers, and most sites would just continue using Javascript anyway since it's more suited to websites than many other browsers.

You would have to sandbox each programming language in this VM and reduce the amount of access each language has to the system, requiring a lot of changes in the languages and removal or reimplementation of many features. Whereas Javascript already has this in mind, and has done a for a long time.

Share Improve this answer

answered [Sep 17, 2008 at 19:13](#)

Follow

community wiki  
[HappySmileMan](#)



1

Maybe you're looking for Google's Native Client.

Share Improve this answer

answered [Dec 17, 2010 at 5:54](#)

Follow



community wiki  
[Ebrahim Mohammadi](#)



1

In a sense, having a more expressive language like Javascript in the browser instead of something more general like Java bytecode has meant a more open web.



Share Improve this answer

answered [Dec 17, 2010 at 21:05](#)

Follow



community wiki

[Grav](#)



0

I think this is **not so easy** issue. We can say that we're stuck with JS, but is it really so bad with jQuery, Prototype, scriptaculous, MooTools, and all fantastic libraries?



Remember, JS is **lightweight**, even more so with V8, TraceMonkey, SquirrelFish - new Javascript engines used in modern browsers.



It is also **proved** - yeah, we know it has problems, but we have lots of these sorted out, like early security problems. Imaging allowing your browser to run Ruby code, or anything else. Security sandbox would have to be done for scratch. And you know what? Python folks already **failed** two times at it.

I think Javascript is going to be **revised and improved** over time, just like HTML and CSS is. The process may

be long, but not everything is possible in this world.

Share Improve this answer

answered Sep 17, 2008 at 19:08

Follow

community wiki

[Paweł Hajdan](#)

---

well, to my knowledge, every security check done for the JS sandbox can (and usually is) done on the byte code as checking permissions and such things by looking at a bunch of text is hard for a computer to do. sending byte code to the client instead of standard JS shouldn't change that

– [GideonMax](#) May 7, 2020 at 15:01

---



0



I don't think you "understand the pragmatic issue that JavaScript is simply what we have to work with now".

Actually it is very powerful language. You had your Java applet in browser for years, and where is it now?

Anyhow, you don't need to "get dirty" to work on client.

For example, try GWT.



Share Improve this answer

answered Oct 8, 2008 at 1:02

Follow

community wiki

[Marko Dumić](#)

---

... you mean...



0



Java and Java applet Flash and Adobe AIR etc..

In general, any RIA framework can fill your needs; but for every one there's a price to pay for using it ( ej. runtime available on browser or/and proprietary or/and less options than pure desktop )

[http://en.wikipedia.org/wiki/List\\_of\\_rich\\_internet\\_applications\\_frameworks](http://en.wikipedia.org/wiki/List_of_rich_internet_applications_frameworks)

For developing Web with any non-web language, you've GWT: develop Java, compile to Javascript

Share Improve this answer

answered Dec 17, 2010 at 12:25

Follow

community wiki  
[obesga](#)

---

1 Hence the suggestion to standardize a VM so it would be ubiquitous. Using JavaScript as a "machine language for the web" seems incredibly clunky and inefficient.

– [newdayrising](#) Jan 2, 2011 at 16:06

---

I think you misunderstand, the original poster is not suggesting that browsers should support other languages, he is suggesting that instead of compiling java to js, you would compile java into a VM. – [GideonMax](#) May 7, 2020 at 15:54

---



0

Because they all have VMs with bytecode interpreters already, and the bytecode is all different too. {Chakra(IE),



Firefox (SpiderMonkey), Safari (SquirrelFish),  
Opera(Carakan).



Sorry , I think Chrome (V8) compiles down to IA32  
machine code.



Share Improve this answer

answered [Dec 17, 2010 at 15:10](#)

Follow

community wiki  
[Stephen](#)



0

well, considering all browsers already use a VM, I don't  
think it will be that difficult to make a VM language for the  
web.

I think it would greatly help for a few reasons:



1. since the server compiles the code, the amount of data  
sent is smaller and the client doesn't waste time on



compiling the code.



2. since the server can compile the code in preparation  
and store it, unlike the client which tries to waste as little  
time quickly compiling the JS, it can make better code  
optimizations.

3. compiling a language to byte code is way easier than  
transpiling to JS.

as a final note (as someone already said in another  
comment), HTML and CSS compile down to a simpler  
language, not sure if it counts as byte code, but you could

also send compiled html and css from the server to the client which would reduce parse and fetch times

Share Improve this answer

answered [May 7, 2020 at 15:49](#)

Follow

community wiki  
[GideonMax](#)



-1



IMO, JavaScript, the language, is not the problem. JavaScript is actually quite an expressive and powerful language. I think it gets a bad rep because it's not got classical OO features, but for me the more I go with the prototypal groove, the more I like it.



The problem as I see it is the flaky and inconsistent implementations across the many browsers we are forced to support on the web. JavaScript libraries like jQuery go a long way towards mitigating that dirty feeling.

Share Improve this answer

answered [Sep 17, 2008 at 22:56](#)

Follow

community wiki  
[Andrew Hedges](#)

1

2

Next