

## Forum:[Hack Forums](#)

Thread: [\[Your Opinion\] - Programming Frameworks](#)

**Uefi** - April 8, 2017, 9:15 pm

There are various frameworks for almost every stable and commonly used language spanning from web-development to desktop environments.

Many frameworks claim they should be adopted for:

- Stability
- Security
- Improved Syntax
- Low-level optimization
- Portability
- Community
- Configurability

I personally find that:

Desktop frameworks (Such as Qt) have porting issues, are unstable for implementation into an enterprise environment, lack a sufficient community, or improperly manage memory.

Web-Development frameworks (Such as Laravel) are insecure out of the box, have arbitrary syntaxes, and require masses of configuration to ensure the features guaranteed by the framework developers are implemented to which when not done consistent to the platform it's running on open security holes and result in inefficient processing.

Curious as to HF's thoughts on frameworks.

Do you use them?

Why?

Pros/Cons over the original language.

**Owl** - April 9, 2017, 2:58 am

I am relatively new to web frameworks and my only experience is with them 4 days of Angular, so I will speak from this.

The framework is a huge learning curve, partly because of syntax (though I disagree with calling it arbitrary, just 'new'). My main draw is its cross-platform capabilities. Porting may not always be stable and will rarely trump a native deployment in terms of performance. However, the idea of porting is to save time and money in startup and in maintenance - luxuries that small parties don't always have.

This is the opposite of large scale enterprises, so it comes down to needs.

**Uefi** - April 9, 2017, 5:52 am

(Apr 9, 2017 02:58 AM)Orka Wrote: [\[CITE\]](#) I am relatively new to web frameworks and my only experience is with them 4 days of Angular, so I will speak from this.

The framework is a huge learning curve, partly because of syntax (though I disagree with calling it arbitrary, just 'new'). My main draw is its cross-platform capabilities. Porting may not always be stable and will rarely trump a native deployment in terms of performance. However, the idea of

porting is to save time and money in startup and in maintenance - luxuries that small parties don't always have.

This is the opposite of large scale enterprises, so it comes down to needs.

So you're saying for smaller projects or new developers who need their program quickly multi-platformed portability is the way to go?

I totally agree and it's more than understandable, not everyone has multiple languages at their disposal in multiple operating systems or the time to program to code a native application for each.

**givevalue** - April 9, 2017, 7:01 am

> Do you use them?

yes

> Why?

Because they make development so much faster. Why create my own user account system when I can just use a library?

Specifically user accounts. Why use an untested and maybe security holed code you wrote yourself when you can use a library used by thousands of other devs that have read the code and tested it with penn tests?

**Uefi** - April 9, 2017, 7:11 am

(Apr 9, 2017 07:01 AM)givevalue Wrote: [\[CITE\]](#) > Do you use them?

yes

> Why?

Because they make development so much faster. Why create my own user account system when I can just use a library?

Specifically user accounts. Why use an untested and maybe security holed code you wrote yourself when you can use a library used by thousands of other devs that have read the code and tested it with penn tests?

But this is what object oriented programming is for, oop isn't just for desktop environment. If you are an incompetent programmer go open source and have people criticize your code. Transferring to a framework because of lack of skill in a programming language is becoming a goto; it's also the reason why frameworks have a bad reputation.

After you preform standard checking, registering a user comes down to a single line of code similar to:

PHP Code

```
<?php
$user->register($name, $username, $email, $pass)
```

**Owl** - April 9, 2017, 8:56 am

(Apr 9, 2017 05:52 AM)Uefi Wrote: [\[CITE\]](#) So you're saying for smaller projects or new developers who need their program quickly multi-platformed portability is the way to go?

I totally agree and it's more than understandable, not everyone has multiple languages at their disposal in multiple operating systems or the time to program to code a native application for each. Yes that's what I mean. Developing for individual platforms can require a completely different language stack (take Apple and their Swift/ObjC for example).

From your other response, I didn't fully understand your thread's question but you have a valid point. There are frameworks/libraries for absolutely everything. It just seems to be a trend going on, particularly in web dev.

Uefi - April 9, 2017, 9:12 am

(Apr 9, 2017 08:56 AM)Orka Wrote: [\[CITE\]](#) Yes that's what I mean. Developing for individual platforms can require a completely different language stack (take Apple and their Swift/ObjC for example).

From your other response, I didn't fully understand your thread's question but you have a valid point. There are frameworks/libraries for absolutely everything. It just seems to be a trend going on, particularly in web dev.

Swift is very clean and built specifically for iOS development and the IDEs for Swift are very professionally built. Apple's move with swift was an intelligent one, Visual Studio developers should honestly be envious. So when referring to the bad of frameworks Swift is far from within that group although I'm not a fan of it's syntax I don't even consider it a framework. I believe Qt has Apple/Android portability (don't quote me on this) but when it can't even port properly between linux and windows I honestly don't want to waste my time making a mobile application only to have it not port. From learning syntax, memory management, to new IDEs, frameworks that promise portability have failed me endlessly, I've given up on them and just code native. WinAPI & POSIX is a pain to covert between so I wanted cross-platform solution but it's simply a joke and so are all the upcoming frameworks.

As for web-development I see a lot of people using Angular, Node, and Laravel. Simply for curiosity purposes I took a look at Laravel, there is more configuration required for a secure setup than lines of code. It's highly confusing for new developers which is what most people using frameworks are not to deride professional programmers who utilize frameworks; these new frameworks are excepting more than just rudimentary coding experience. People who adopted node are more the type to adopt this newfound rise in frameworks being 'innovative' however, I just don't see any of these adapting to an enterprise environment and they're too hard to be securely used by newcomers to the programming world.

g4143 - April 9, 2017, 9:13 am

I looked into frameworks which have covered everything from web dev to cloud computing and it's the only way to keep current.

My current framework du jour is Elm's and it's fantastic.

Uefi - April 9, 2017, 9:13 am

(Apr 9, 2017 09:13 AM)g4143 Wrote: [\[CITE\]](#) I looked into frameworks which have covered everything from web dev to cloud computing and it's the only way to keep current.

Read post above. Particularly the last sentence or two.

g4143 - April 9, 2017, 9:22 am

(Apr 9, 2017 09:13 AM)Uefi Wrote: [\[CITE\]](#) Read post above. Particularly the last sentence or two. Frameworks are not there to make the rookies job easier. Frameworks are there to make the professionals job easier.

Uefi - April 9, 2017, 9:31 am

(Apr 9, 2017 09:22 AM)g4143 Wrote: [\[CITE\]](#) Frameworks are not there to make the rookies job easier. Frameworks are there to make the professionals job easier.

But the feasibility of using them in any professional environment is simply non-existent. If we need to prototype we'll use something such as java and later port it native. Frameworks on their own are sometimes so abstract it's to the point of learning a new language. Also I seem them more marketed towards dilettantes than professional programmers.

Owl - April 9, 2017, 10:07 am

(Apr 9, 2017 09:12 AM)Uefi Wrote: [\[CITE\]](#) Swift is very clean and built specifically for iOS development and the IDEs for Swift are very professionally built. Apple's move with swift was an intelligent one, Visual Studio developers should honestly be envious. So when referring to the bad of frameworks Swift is far from within that group although I'm not a fan of it's syntax I don't even consider it a framework. I believe Qt has Apple/Android portability (don't quote me on this) but when it can't even port properly between linux and windows I honestly don't want to waste my time making a mobile application only to have it not port. From learning syntax, memory management, to new IDEs, frameworks that promise portability have failed me endlessly, I've given up on them and just code native. WinAPI & POSIX is a pain to convert between so I wanted cross-platform solution but it's simply a joke and so are all the upcoming frameworks.

As for web-development I see a lot of people using Angular, Node, and Laravel. Simply for curiosity purposes I took a look at Laravel, there is more configuration required for a secure setup than lines of code. It's highly confusing for new developers which is what most people using frameworks are not to deride professional programmers who utilize frameworks; these new frameworks are excepting more than just rudimentary coding experience. People who adopted node are more the type to adopt this newfound rise in frameworks being 'innovative' however, I just don't see any of these adapting to an enterprise environment and they're too hard to be securely used by newcomers to the programming world.

Swift works differently from Android development (with Java) and can only be developed on a Mac, or OSX specifically. This was the reasoning behind my example, that prerequisites and learning required can make it more difficult to access.

Uefi - April 9, 2017, 10:14 am

(Apr 9, 2017 10:07 AM)Orka Wrote: [\[CITE\]](#) Swift works differently from Android development (with Java) and can only be developed on a Mac, or OSX specifically. This was the reasoning behind my example, that prerequisites and learning required can make it more difficult to access.

Proprietary!

Yes, the need for OSX is one of it's downfalls, there is one for windows somewhere but it is the most unstable IDE I've ever used. An Apple developer's license also brings a cost of 99\$ last time I checked. Android is 15 or so EU. Developing for android is easier because it utilizes a common language (Java as you previously mentioned) and just throws a SDK at it, but Swift is stable and is used in almost every modern app; objective-c (which is the alternative) is obsolete at this point so by most developers Swift is considered a replacement language entirely rather than a framework. Frameworks aspire to be swift.

g4143 - April 9, 2017, 11:09 am

(Apr 9, 2017 09:31 AM)Uefi Wrote: [\[CITE\]](#) But the feasibility of using them in any professional environment is simply non-existent. If we need to prototype we'll use something such as java and later port it native. Frameworks on their own are sometimes so abstract it's to the point of learning a new language. Also I seem them more marketed towards dilettantes than professional programmers.

Not sure what your point is or maybe we have different ideas about what a framework is...

**Malicious Intent** - April 9, 2017, 11:15 am

I recently started to mess around with Qt and i have to say its great. Takes time to get hang of but its a great framework. Not sure why u say its not multiplatform. I use it on windows and linux without any issue. In a fact you can simply copy project and rebuild it on different platform. Unless obviously you are using os specific libraries in your code. All in all it saves me a lot of time.

**Uefi** - April 9, 2017, 11:19 am

(Apr 9, 2017 11:09 AM)g4143 Wrote: [\[CITE\]](#) Not sure what your point is or maybe we have different ideas about what a framework is...

To put it simply, Swift has a monopoly on usage in Apple app development, for this reason it is considered it's own language by it's community.

"Swift is a new programming language for iOS, macOS, watchOS, and tvOS apps that builds on the best of C and Objective-C, without the constraints of C compatibility." - Apple Developer Documentation

Source: [Apple Developer Docs](#)

(Apr 9, 2017 11:15 AM)Malicious Intent Wrote: [\[CITE\]](#) I recently started to mess around with Qt and i have to say its great. Takes time to get hang of but its a great framework. Not sure why u say its not multiplatform. I use it on windows and linux without any issue. In a fact you can simply copy project and rebuild it on different platform. Unless obviously you are using os specific libraries in your code. All in all it saves me a lot of time.

Qt is the only framework who's syntax I've liked so far, as for portability issues, they may stem from different stages of advancement into the language. I made a rather large program that involved cryptography, was multithreaded, used asynchronous sockets, and had a very nice thread pooling system all to which had porting issues; after an error was amended another showed up upon compilation in the destined os recursively.

**SIM-1725686** - April 9, 2017, 2:31 pm

(Apr 9, 2017 07:11 AM)Uefi Wrote: [\[CITE\]](#) it's also the reason why frameworks have a bad reputation. Not sure what industry you're in where frameworks have a bad reputation. The reason frameworks exist is to make development [DRY](#), or to prevent redundancy by reusing common functionality.

A major advantage of frameworks, specifically open-source/community-powered ones is that they provide patches for things you would never think of/solve on your own. Security is a major one - I'd rather run a production server on software that has been publicly pentested and thoroughly documented, than some in-house code written by someone without security training.

If you're a beginner, frameworks will confuse you. But if you are already accustomed to a language and are looking to go professional, frameworks greatly increase productivity when used properly. Just make sure that if you're using a framework, you know the language well enough to do things manually if the need arises.

I just see no need to entirely avoid them.

**Uefi** - April 9, 2017, 3:53 pm

(Apr 9, 2017 02:31 PM)regiostech Wrote: [\[CITE\]](#) Not sure what industry you're in where frameworks have a bad reputation. The reason frameworks exist is to make development [DRY](#), or to prevent redundancy by reusing common functionality.

A major advantage of frameworks, specifically open-source/community-powered ones is that they provide patches for things you would never think of/solve on your own. Security is a major one - I'd rather run a production server on software that has been publicly pentested and thoroughly

documented, than some in-house code written by someone without security training.

If you're a beginner, frameworks will confuse you. But if you are already accustomed to a language and are looking to go professional, frameworks greatly increase productivity when used properly. Just make sure that if you're using a framework, you know the language well enough to do things manually if the need arises.

I just see no need to entirely avoid them.

Frameworks are marketed towards beginners for ease of use. Take jQuery for example, the javascript syntax and processes are rather advanced thus jQuery was born. I've worked several years in the field and have created rather complex programs enterprise ready, never have I heard of competent programmers using frameworks to go 'professional' if you are accustomed to a language, write secure code, and have been in a real world enterprise environment then you are a professional programmer, using a framework does not prop one upon this plateau, experience does.

Quote:they provide patches for things you would never think of/solve on your own

I'm not trying in anyway to discredit you in any way but provide me an example.

**g4143** - April 9, 2017, 7:17 pm

(Apr 9, 2017 03:53 PM)Uefi Wrote: [\[CITE\]](#) ...Take jQuery for example, the javascript syntax and processes are rather advanced thus jQuery was born....

No and no. jQuery was born because JavaScript has dangerous components and odd components and inconsistent components.

How would that reasoning explain Elm? Elm compiles to JavaScript like jQuery but its syntax and ideas are more complex than JavaScript.

**Uefi** - April 10, 2017, 1:26 am

(Apr 9, 2017 07:17 PM)g4143 Wrote: [\[CITE\]](#) No and no. jQuery was born because JavaScript has dangerous components and odd components and inconsistent components.

How would that reasoning explain Elm? Elm compiles to JavaScript like jQuery but its syntax and ideas are more complex than JavaScript.

I may just take your word on this one however every web-developer I've asked rookie or expert, said they opted jQuery over JS because it's simply easier to use. By born I was referencing having a wide-spread community of developers becoming the common language it is today, not the best wording on my end I'm afraid.

(Apr 9, 2017 07:17 PM)g4143 Wrote: [\[CITE\]](#) No and no. jQuery was born because JavaScript has dangerous components and odd components and inconsistent components.

How would that reasoning explain Elm? Elm compiles to JavaScript like jQuery but its syntax and ideas are more complex than JavaScript.

I've never heard of Elm and this is probably why, however you just instantiated my point that jQuery was adopted mostly for it's ease of use? Unless I'm missing something completely.

**g4143** - April 10, 2017, 3:49 am

(Apr 10, 2017 01:26 AM)Uefi Wrote: [\[CITE\]](#) I've never heard of Elm and this is probably why, however you just instantiated my point that jQuery was adopted mostly for it's ease of use? Unless I'm missing something completely.

No its was adopted because it jettisoned the dangerous and odd parts of JavaScript and only presented the nicer parts of JavaScript in a nice wrapper.

**Kickupx** - April 10, 2017, 9:16 am

To my 4 year experience programming frameworks usually just come into my way and they slow down the website insanely much. Most pages does not have much dynamic content and it is often easier to just include the DB library and query stuff. So what I usually do for most websites is that I write an custom compiler (in C# usually) which then generates the HTML based on the files in one or more directories. This is then the files I upload to my server.

With that I get 300 ms request time. Down from 300 ms to just answer which is what I had with Laravel.

Also I feel lot's more flexible because I own almost everything. So if something is fucked up I can fix it because I know how it works.

**Uefi** - April 10, 2017, 12:01 pm

(Apr 10, 2017 03:49 AM)g4143 Wrote: [\[CITE\]](#) No its was adopted because it jettisoned the dangerous and odd parts of JavaScript and only presented the nicer parts of JavaScript in a nice wrapper.

Just did some research and you are correct, thank you for being constructive and sticking to your knowledge, truly appreciated.

(Apr 10, 2017 09:16 AM)Kickupx Wrote: [\[CITE\]](#) To my 4 year experience programming frameworks usually just come into my way and they slow down the website insanely much. Most pages does not have much dynamic content and it is often easier to just include the DB library and query stuff. So what I usually do for most websites is that I write an custom compiler (in C# usually) which then generates the HTML based on the files in one or more directories. This is then the files I upload to my server.

With that I get 300 ms request time. Down from 300 ms to just answer which is what I had with Laravel.

Also I feel lot's more flexible because I own almost everything. So if something is fucked up I can fix it because I know how it works.

No framework is going to process faster than the native if you know what you're doing. As for dramatic slow down when using laravel this is interesting, I wonder what developers do when given a laravel task requiring to serve large amounts of requests. Have you tested on higher spec machinery? Under the assumption your laravel isn't flawed code-wise one can easily deduce that this is a cpu taxing issue.

**Uefi** - April 16, 2017, 2:58 pm

Still looking for more feedback, don't be discouraged by the terminology being thrown around.

**g4143** - April 16, 2017, 7:29 pm

(Apr 10, 2017 12:01 PM)Uefi Wrote: [\[CITE\]](#) ...

No framework is going to process faster than the native if you know what you're doing...

Flat out wrong. A framework has nothing that makes it inherently slower than hand coding the solution. For most cases a framework will produce faster code and maintainable code as compared to recreating the functionality by hand.

To think that re-creating the functionality present in a framework is constructive or advantageous is foolhardy. Just think if that reasoning was taken to the extreme. Do you think we should write our own custom operating system and GUI and networking system for each program we write? The resulting operating system + program would be minuscule as compared to what we have today but the progress of software would slow down to a crawl. There is a reason we embrace code reuse.

Uefi - April 16, 2017, 7:42 pm

Do you know what a framework is?

(Apr 16, 2017 07:29 PM)g4143 Wrote: [\[CITE\]](#) Do you think we should write our own custom operating system and GUI and networking system for each program we write?

I fail to see how I insinuated this, please elaborate so I may provide a constructive counter claim to the rest of your argument.