

## Re: Tkinter or wxpython?

---

*Source:* <http://coding.derkeiler.com/Archive/Python/comp.lang.python/2007-08/msg00718.html>

---

- *From:* "Chris Mellon" <[arkanes@xxxxxxxxxx](mailto:arkanes@xxxxxxxxxx)>
  - *Date:* Mon, 6 Aug 2007 12:43:08 -0500
- 

On 06 Aug 2007 09:44:15 -0700, Paul Rubin  
<[@nospam.invalid">http://phr.cx">@nospam.invalid](mailto:http://phr.cx)> wrote:

"Chris Mellon" <[arkanes@xxxxxxxxxx](mailto:arkanes@xxxxxxxxxx)> writes:

Might or might not matter for the application, especially  
considering  
that tkinter is part of the discussion.

The point is that you have no option with the browser – even Tkinter  
has platform theming support now.

Hmm, I don't know anything about that. I'm taking the view that for a  
lot of apps, the requirement is to just put some functionality on the  
screen, with slick visual appearance having little value or even  
negative value.

Define "functionality". From the rest of your posts, that seems to be  
limited to "press buttons" and "type small amounts of non-formatted  
text" on the interaction side and "display small amounts of simply  
formatted text" on the output side. I'm not denying that you can get  
by with this functionality in a large number of cases, but it's  
certainly not optimal in general. If it were, we'd all still be using  
VT100 terminals.

...Gmail uses ajax instead of a page load when you start  
typing a reply,

I see, the answer to what caused your problem is approximately "ajax  
is evil", or alternatively, the gmail app attempts slickness with a  
tool that doesn't support slickness that well. OK, I can accept that  
using browsers for slick interfaces has its problems, but the answer  
(a lot of the time, not always) is to just decide you don't need  
slickness.

Re: Tkinter or wxpython?

What you're calling "slickness" is what other people call "usable". Obviously it could be done without the ajax, but you lose features and usability. It's slower, because it takes a full page load, and I lose the context of the discussion.

File i/o and file system browsing are available from javascript if the user grants permission.

Which they won't (I don't even know how, from Firefox), so you can't rely on it working.

The application javascript pops a dialog asking for permission and the user clicks yes or no. If you can get them to install a desktop app (gah!!) then you can certainly get them to click yes in a browser. The permission mechanism is admittedly browser dependent.

My out of the box firefox install simply denies the operation and raises a javascript error.

But not of anything else. I've often wanted to drag & drop a file onto the file upload box in gmail, for example.

Well, ok, that's a slick feature that your app might need. None of mine have needed it so far.

How about something as simple as context menus?

Unnecessary slickness for many apps. I've never needed it.

How have you decided that you've never needed it? Have you ever worked with a UI designer or workflow expert? Or have you just never bothered to implement it and you never heard anything from your users? How closely have you worked with the users of your applications?

Multiple windows are evil most of the time,

Re: Tkinter or wxpython?

## Re: Tkinter or wxpython?

Multiple windows are the common case on the mac. They're not rare on other platforms as well. The fact that your windows are constrained to the browser and can't be torn off is a large limitation. No toolbars, no palettes, no modal dialogs (except on IE, or if you constrain them to the browser). Lots of unnecessary chrome on your extra windows, too (see below).

You can get rid of the chrome with javascript, but the extra windows are still usually evil and unnecessary. Just because they get used a lot doesn't change that. A heck of a lot of deployed interfaces, web or desktop, simply suck.

As phishing attacks become more common, browsing are restricting the ability to remove or modify chrome. Again, a good feature for a browser, not so much for an application platform.

More platforms to develop on and test with.

Compared to a desktop app? I don't think so.

Did you ever try counting? X browsers \* Y browser versions \* Z platforms. There are javascript and CSS bugs and differences between all of these.

If you can tell them to install a desktop app, you can alternatively tell them what browser to use. For example we use a complicated firefox-only browser app where I work, that relies heavily on canvas objects. But if you write your application with straightforward html you tend to have very few platform problems.

It depends on your target audience and deployment arena. The context is cross platform applications, so at least 2 browsers and at least 2 platforms are required. Don't forget browser versioning, too – IE 5 is different than IE 6 is different than IE 7. Even "straightforward" HTML has non-trivial cross browser differences.

From my own professional experience, it's not any easier to account for browser differences than it is for platform differences. It's getting better as more people jump on the web bandwagon and CSS and JS abstraction libraries show up,

I guess I see that stuff as mostly-evil and prefer straightforward

## Re: Tkinter or wxpython?

html.

You simply can't do much of anything beyond display text and basic crud with straightforward HTML without styling. If that is the sole limit of your UI needs, then bully for you but that's hardly the common case and I'd venture that you don't have the experience to support the rather sweeping claim you made.

I'm not talking about chrome and slickness. I'm talking about basic usability concerns like "will this work with a higher font size" and "will it reflow in a sensible way if I change the window size". The CSS box model works okay for text, it's not so good for widgets.

I just haven't had this problem with HTML interfaces. I've never even used CSS, though that makes me a bit of a lamer. An interface that needs to fill the whole screen with widgets is probably too complicated.

An application is not a book. Of course you fill the whole "screen" with widgets – your application is defined by the widgets that it has, and the amount of space those widgets take up is your screen space.

More than a few web apps end up writing their own layout engines in dynamic javascript. This is sad. While I'm talking about chrome, add "wasted screenspace due to browser chrome" to a limitation of web apps, again unless you write your own browser host. This is another example of a feature that makes a good browser (don't let arbitrary web pages mess with my web browser functionality) clashing with the desires of a good application (don't waste screen space with irrelevant functionality).

I guess the term I'd use to describe all these effects is "slick" and a heck of a lot of the time it's just not needed. Anyway you can pop a chromeless browser window with a simple javascript call.

Man, you should be in PR with the way you spin things. You can't implement anything except the most trivial of applications using only the simple, familiar web elements like HTML forms. Anything more complicated than that needs to be done with DHTML and probably AJAX, and those UI elements don't look anything like the familiar ones. You go in one breath from saying that you can implement dialogs in HTML to saying that the rich client is the \*less\* familiar of the interfaces?

## Re: Tkinter or wxpython?

I don't see the contradiction—with HTML you have just a few widgets that everyone understands and you get an interface that's almost always self-explanatory. Yes you could call those interfaces trivial, but the little secret that I'm trying to convey is that very often, trivial is all that's needed.

That's simply not true, and I don't think you can objectively justify it. Besides, you specifically claimed HTML dialogs as being a replacement for "real" ones, and that can't be done using trivial interfaces. It requires DHTML, sometimes ajax, and/or browser support.

If you'd said "if you're making something really simple that has limited rich UI or data entry needs, consider a web application instead" I wouldn't have posted. A web application is something you make because the deployment and access benefits outweigh the UI and integration limitations, not as your default "go to" whenever you create an application.

Well, I could back off to somewhere between the two. Like, "ask yourself whether your application really needs a rich gui or complex data entry features. If you can get by with a simple HTML interface, and a lot of the time you can, you'll probably have an easier time doing it that way and that should be the default".

I would challenge the assumption that, starting from zero, it is less complicated to write a web application, much less a "pretend" web app using a local web server, than a desktop application using Tkinter (ick) or wxPython or PyQt. I would also challenge the assumption that 'a lot of the time' you can get away with a simple HTML interface, at least by my personal definition of 'get away with'. I've worked a lot with data entry people, where the actual requirements in terms of slickness and flash are quite small. But "basic html" simply does not fit the bill. There is a real cost in terms of how fast and how accurately they can enter data in this environment.

Moreover, if you *\*don't\** need global access or zero-deployment (zero-deployment is always nice but most environments where you can reasonably force a specific browser and version also have IT infrastructure), then you should ask yourself if forcing your application in the web app model (I haven't even touched on the whole stateless HTTP being mapped to a stateful environment issue, or the need to manage the local web server) actually buys you anything. I simply do not buy the claim that HTML interfaces are necessarily simpler in any but the most trivial of cases, and even in those

## Re: Tkinter or wxpython?

super-trivial applications there are often hidden concerns that don't filter up to management.

I'd like an example you have of a desktop application that could have just as easily been a web application. Bonus points if it's not a CRUD screen, and double bonus if you don't just handwave away things like file browsing. Although there are benefits even for the crud screen – I've written screens and applications for data entry professionals and they almost always prefer the speed optimizations you can make in a client application.

I guess I've written 4 or 5 nontrivial desktop gui apps and ZERO of them had file browsing. One of them did need access to the pc's serial port, which means there had to be application code on the desktop, but the gui could have been a browser (talking to an application-embedded http server) if browsers were available in those days. Another one of them saved some state to a local file but it did that without file browsing, and could have instead saved the state on a remote server if it were written as a remote app.

Exactly how much time do you think using an HTML interface with an embedded server would have saved you, especially considering the careful care you have to take with access to the serial port? For the record, I've written over a dozen client applications, many of which I would consider trivial (but used features that would be expensive or impossible to implement in the browser) and probably twice that many web apps.

I'm not saying file browsing is never needed, just that from experimental observation, it's quite often not needed.

Plural of anecdote and all that. Make a survey of every application that you interact with and see how many of them need to interact with arbitrary files from the filesystem.

Again, it all depends on what you're trying to do. For data entry stuff you probably want the data on a remote server anyway, and you can do basic CRUD validation with fairly simple javascript. Maybe that departs from pure HTML but it's nothing like the ajax/dhtml madness that causes the problems you've described.

--

Re: Tkinter or wxpython?

CRUD with javascript is something I actually have a lot of experience with. Deficiencies in the data entry UI have real consequences because you get invalid data and slow data entry speeds. The auto-completing combo-box, for example, is simply impossible in a browser without quite complicated (and slow) DHTML and is a huge boon for data entry.

I'm not trying to claim that there are no benefits to web applications. But I often see people touting the web as a \*superior application platform\*, which is simply false, and as innately simpler to develop for, which is also false.

.