Good post.

> I've never had an organization system except keeping them in folders with about 200 files in each, making new ones as needed.

Yeah I do the same. Most of my pictures are anime-related so I have a rough hierarchy of "show > character > emotion" but then I end up with a bunch of "misc" things that I chuck in their own folder (either as a subfolder of "show" or root level). I try to use the filename as a form pseudo-tagging by just using some keywords concatenated together, so I can fuzzy-match on them.

You mentioned adding tags via exif, this isn't a good solution for 2 reasons: first as you mentioned exif isn't really standardized (png support is poor), and second it doesn't really integrate with the filesystem so you still need something on top to parse it and build an index.

One better option is to do this at a filesystem level. HFS (and presumably APFS?) for instance support aribtrary extended attributes you can set on files. OSX actually does support "tags" as a first class feature in its Finder, and it's implemented in this way – each file has a list of tags stored in the kMDItemUserTags xatr of the file, and there's an index built on top of it. The issue is that this is very file-system specific, so it's not cross-platform. And I personally don't use tags because it's kind of clunky.

If you want a really forward-thinking file-system that flips the traditional hierarchical paradigm, take a look at WinFS. One of the sad few things that was discarded when MS abandoned Longhorn.

> As the internet became more important, it started to subsume functionality that used to belong to host machines.

One annoying this is that the browser doesn't necessarily _have_ to take away power from the client. Browsers used to support WebSQL which allowed browsers to directly map into and interact with sqlite database stored on the client. This could make the browser truly an extension of the desktop, allowing browser and non-browser apps to seamlessly operate on the same data. But then they crippled it and decided to go with IndexedDb which is basically a glorified key-value store, kills any potential to do powerful things, and isn't 1:1 with the host storage format. This effectively silo'd off the web from "desktop apps". (*)

As for innovation on desktop, I don't have much hope for the future. Forget about introducing new ideas, we still have ideas from QNX and Plan9 that were already explored decades ago and still haven't been introduced into modern OSs. Even the few powerful ideas that were implemented are left to languish, and will probably be removed soon (**)

(*) Ironically Chrome is the only one that ended up implementing websql. I believe the argument from W3C for rejecting it was that it's too tightly coupled to SQLite, which feels like bullshit to me – they're OK with drm standards, ok with effectively making the broader Internet browser-only by going ahead with HTTP/3, but they decide to draw the

line at adopting a rock-solid, extensible, open DB?

(**) My candidate for "most powerful but underrated feature" in modern OSs (well in OSX at least) is AppleScript, not so much the language itself (which is atrocious) but the fact that what underpins it is effectively an IPC mechanism that makes automation a first-class citizen on the desktop. The unix people go on about the power of the command line, how it makes stdin/stdout a common interface that unifies different applications; unfortunately the extent of "unification" stops at CLI apps, since GUI applications get left out. AppleScript closed that gap, providing a common interface to allow GUI applications to expose endpoints. As an example, a music player might offer endpoints to get the currently playing track, get all tracks in a playlist, etc. Additionally, events are also network transparent, so you could simply send the same request to a different machine to get the currently playing track on a different computer.

This effectively meant that every GUI application could be interacted with in a CLI-esque manner, and moreover a developer got the CLI-counterpart "for free" assuming they structured their program right and bothered to declare supported events. The principle is simple enough, and you could probably cobble something like this together even today on Linuxes, but without adoption it'd be dead in the water. The neat part is Apple made this a first-class part of the platform & development stack, integrating this to a surprising extent on 1st party apps, and many 3rd party developers also added support for this relatively obscure feature.

Of course then the whole infrastructure they built up slowly bitrotted over the course of OS updates, support for it in 1st party apps broke, they never bothered to expose a library for the underlying apple-event infrastructure to languages other than their shitty AppleScript until it was too late, and they'll probably remove it entirely at some point in the future.