Cross-platform native GUIs

{trade,pay}offs, {integra,distribu}tion

Zakariyya Mughal

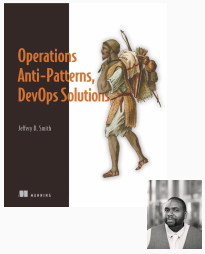
2021-06-10

The Perl and Raku Conference (In the Cloud) 2021

https://github.com/zmughal-biblio/talk-tprc2021cic-cross-platform-native-guis-20210610

Motivation

Operations Anti-Patterns, DevOps Solutions by Jeffery D. Smith



Working in Public by Nadia Eghbal

Power law distribution in FOSS contributions

The competition

Put on developer hat

Web applications are great

- Easy deployment (for developers) → Easy update (for users)
- Flexible declarative languages for *separation of content and presentation.*
  - HTML, CSS, MathML, SVG

I am now going to be slightly polemical to persuade, but don't worry, I'll get to the technical parts quickly. No fiddling with installers, no ABI compatibility issues. If this is your top priority, use a web application. These are useful outside the context of web applications. Note on recursive use of web standards:

You can embed an `iframe` and other HTML inside of SVG using `foreignObject`. But you can not easily render HTML to an HTML Canvas or WebGL — this is not allowed because it can be a security risk.

But there are drawbacks

- Web browsers are a sandbox for a reason.
- Platform + browser combination bugs.
- Polyfills/shims are... dirty

They are designed for dealing with the web and all the security concerns behind that. But sometimes one persons "security concerns" is another person's "yes, I really meant to do that Clippy". Go look on the bug trackers for browsers.

I want to do X, Y, & Z

- Lots of useful native code already exists that would need to reimplemented.
  - Yes, we have WebAssembly. And tooling like Emscripten.
  - But package management is incomplete. And packages are not well-tested.
- Accessing that code through a client-server architecture adds overhead for some applications.

For example, packages could have duplicate versions of libpng inside of them. Not to mention the bandwidth problems if everybody distributes packages like this.

There should be a lot of appreciation for system package maintainers for making sure that dependencies between packages work with all the downstream packages.

Computing APIs

KV store, e.g., SQLite (2000) (sql.js!), LevelDB (2011)

Initially browsers had very little access to computing resources, but now we have these (see table).

Other APIs at https://developer.mozilla.org/en-US/docs/Web/API, e.g., Web Speech API, Gamepad API.

But this is always going to be delayed from where computer hardware is. Feature support has to filter through the OS, compiler/user-space library, browser/standards unless hardware is designed specifically for the browser.

Creating standards is a lot of work. I'm certain that code to give access to native features can be done by simply binding to them (using IDL), but thinking about making this in a way so that most platforms can implement a common subset of that feature requires a lot of effort.

Fusion?

- **Problem** Limited control over memory, storage, battery, and other resources.
- **Solution** Embed a specific browser to provide a webview (Electron, NW.js, Chromium Embedded Framework, Apache Cordova).
- Gives a lot of control over how resources are passed into the webview part of the application, e.g., through custom scheme handlers.

How resources are managed is left up to the browser implementation. Much better than requiring a browser extension for using a web application.

Since all of these are based on Chromium, they can all be tested and controlled using the WebDriver protocol.

But certain platform + browser combination bugs will remain.

Put on user hat

User data goes… somewhere

Underappreciated values

- Privacy
- Data interoperability
- Future-proof (backups, format shifting)
- Integration with the desktop

During the course of gathering resources, several of my bookmarks are no longer available on the web or Internet Archive because they used JavaScript to retrieve their content.

Web apps make the web less machine-readable.

User data goes... somewhere

- Data interoperability can provide *separation of data and application*
- Possible solution: Solid

Solid Protocol

Editor's Draft, 2021-05-19

This version
https://solidproject.org/TR/protocol

Editors
Sarven Capadisli
Tim Berners-Lee
Ruben Verborgh
Kjetil Kjernsmo
Justin Bingham
Dmitri Zagidulin

Published
2020-12-16

Modified
2021-05-19

Repository
GitHub
Issues

Abstract

This document connects a set of specifications that, together, provide applications with secure and permissioned access to externally stored data in an interoperable way.

The Lost City of Interaction Design

Integration between applications is limited or sometimes broken.

- Clipboard
- Drag-and-drop

I saw a bug about copying SVG into the clipboard and different browsers implement this differently.

Widgets are often very non-desktop like

- Frameworks exist, e.g., qooxdoo
- Not every widget supports:
  - Jumping to items / incremental search
  - Multi-level undo
  - Binding all keyboard shortcuts

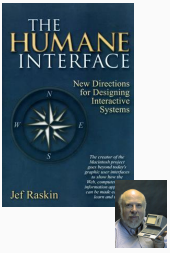
Put on carbon-based lifeform hat

The Humane Interface by Jef Raskin

- Visual system was made for object recognition, not reading.
- Latency:

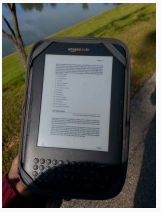
  input → render

  feedback loop



Terminal CLIs and TUIs are not enough.

Hardware



Intermission

C shanty

There once was a lib written in C
Installing it was not easy

Tried t'build on Win and go
But linker failed, Bill said no

Soon may the QA come
But not until all checks are done

The OS does segfault throw
The buffer did overflow

Is this what it means to ship it using Docker?

The technical part

Solutions on CPAN *today*

Cross-platform:

- IUP
- Prima
- Fl
- Tk
- Tkx
- Wx
- Gtk3

Other:

- Win32::GUI (non-cross-platform), Gtk2 (old)

GCstar - collections manager (port to Gtk3 done)

Shutter - screenshot application

Biodiverse - spatial analysis tool (for understanding climate impacts)

Gtk3

- Cross-platform
- Many language bindings provided through GObject Introspection.
- Glade interface builder
- Interactive debugger

```
export GTK_DEBUG='interactive'
my-gtk-application
```

Gtk3 on *

https://github.com/orbital-transfer-example/perl-gtk3-starter-basic

- Use system package manager for native packages
- “It just works.” — me (ca. just now)
- Docker: some tools don’t like being run as root

Use own Perl during development.

Docker, no root (Anki — Qt Web Engine, CEF ; both Chromium based)

Gtk3 on macOS

- macOS has several package managers: tested with Homebrew
- Do not use system Perl (good advice for any platform)
- Architecture: `x86_64`, will need testing on `arm64`: https://doesitarm.com/

System Perl would list multiple architectures in `ccflags` which would break compilation of `Glib.pm` (see `ARCHFLAGS` environment variable and how it relates to universal binaries).

See discussion at https://mail.gnome.org/archives/gtk-perl-list/2016-October/msg00004.html and https://docs.brew.sh/Gems,-Eggs-and-Perl-Modules#avoiding-sudo-altogether-for-perl.

Gtk3 on MSWin32

- Use MSYS2.
- `ExtUtils::MakeMaker` hacks
- `#define MAX_PATH 260`
- Disable layered windows

```
BEGIN {
  if( $^O eq 'MSWin32' ) {
    $ENV{GDK_WIN32_LAYERED} = 0;
  }
}

use Gtk3 -init;
```


MSYS2 is a rolling release.

There are workarounds for the max path issue (using prefix or registry setting).

Layered windows https://stackoverflow.com/questions/38375102/unable-to-embed-gstreamer-video-in-a-gtk-window, gone in GTK4 https://gitlab.gnome.org/GNOME/gtk/-/merge_requests/2782.

Building on GitHub Actions

Debug build using https://github.com/mxschmitt/action-tmate

Using Vagrant for local debugging

Provision a VM locally

```
vagrant up buster64 # Debian
vagrant up win10    # Windows 10
#vagrant up macOS   # macOS
```

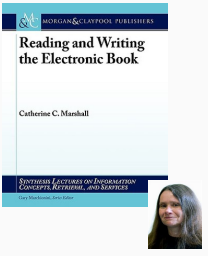
Creating distributable packages

- Windows: `pacman -Ql`, `PAR::Packer`, WiX Toolset
- macOS: Homebrew tap, create-dmg (*TODO*)
- Linux: .deb/.rpm, Flatpak (*TODO*)

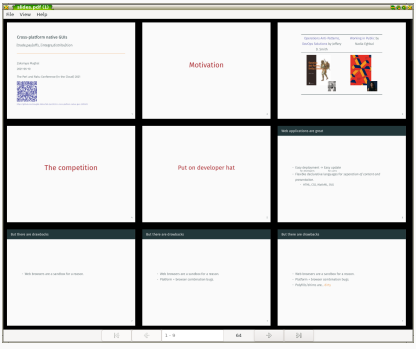
Example Perl Homebrew tap https://github.com/sqitchers/homebrew-sqitch

Why I wrote a ++(document reader)

Reading and Writing the Electronic Book by Catherine C. Marshall



Curie




```
cpanm -n Renard::Curie
# https://github.com/project-renard/curie
```

Curie stack

- Perl
- Gtk3
- Cairo
- MuPDF (Alien::MuPDF)
- Custom scene graph
- Graphene (Alien::Graphene)
- Kiwisolver (Alien::Kiwisolver)
- Festival
- IO::Async

Future

- AndroWish
- Port to e-ink device
- Binary Perl dist packaging on CI?

Compare with binary wheels in Python packaging https://github.com/pypa/cibuildwheel, https://github.com/pypa/manylinux

Acknowledgements

- Chirag Ghanshani, Stanislav Yotov, Jesus Hernandez
- CPAN Testers, Slaven Rezić, Thibault Duponchelle
- PerlAlien, #native, GTK-Perl mailing list

Questions & Contact

- on IRC: `sivoais` on irc://irc.perl.org/#native (Alien and FFI!) or irc://irc.perl.org/#pdl (scientific and numerical computing!),
- on Twitter: @zmughal,
- on GitHub: https://github.com/zmughal.