Writing native Linux desktop apps with JavaScript

Philip Chimento • 🧸 🍂 ptomato • 🐱 @therealptomato

Linux Application Summit, May 13, 2021
Introduction

- I maintain GJS (GNOME JavaScript)
- This talk is a bit of an experiment for me
- Can web JS programmers ramp up quickly on writing a desktop app?
What this talk is

- For JavaScript developers and enthusiasts
  - who are curious about writing a desktop app
- A walk through creating and publishing a desktop app in JS
  - Technologies: GJS, GTK, Flatpak, Flathub
- A slide deck that you can read later
  - [https://ptomato.name/talks/las2021/](https://ptomato.name/talks/las2021/)
What this talk is not

- A step-by-step tutorial on how to write an app
  - There's already a good one on gjs.guide
- Presented by an experienced web developer
Let's get started!
App: "Bloatpad"
the unnecessary note-taking app
Have something to start with

• Can also use gtk-js-app
- a Meson build system
- a placeholder icon
- resource bundles
- a `.desktop` file
- a settings schema
- an AppStream meta info file
- infrastructure for i18n
- skeleton code
- a Flatpak manifest
Build systems

- Meson is probably a good one to stick with
- You will need it if your app ever includes any C code
- Coming from JS development you still might want something more familiar

$ yarn init

"scripts": {
   "prebuild": "test -d _build || meson _build",
   "build": "ninja -C _build",
   "start": "meson compile -C _build devel",
   "test": "meson test -C _build"
}
Yarn

```bash
$ yarn build
$ yarn start
```

Hello, World!
**Lint**

- May as well install prettier and never again worry about code style
- **eslint** for usage

```bash
$ yarn add --dev prettier eslint eslint-config-prettier
```

```
"lint": "eslint . --fix && prettier --write ."
```
**TypeScript**

- You can write in TypeScript, it *mostly* works
- Or write JS with type annotations in comments and use TypeScript to typecheck
- Thanks to the **hard work** of Evan Welsh
Other build tools

- Bundlers are probably not needed
  - Tree shaking can be useful
  - use e.g. `find-unused-exports`
- Minifiers are probably not needed
- Babel probably works
Assembling the UI

Photo by Anna Shvets from Pexels
XML UI files or no?

- XML-CSS-JS is like the trinity of HTML-CSS-JS
- Alternative is to build your UI in code
XML UI files or no?

```xml
<object class="GtkListView" id="notesList">
    <property name="show-separators">True</property>
    <signal name="activate" handler="_onNotesListActivate"/>
</object>
```

vs.

```javascript
this._notesList = new Gtk.ListView({ showSeparators: true });
this._notesList.connect("activate", this._onNotesListActivate.bind(this));
```
XML UI files

- Tedious to write by hand
- **Glade UI Designer**
  - GTK 3 only
  - GTK 4 alternative **underway**
Result
CSS

.large-icon {
  color: #888a85;
  -gtk-icon-shadow: #d3d7cf 1px 1px;
  padding-right: 8px;
}
CSS
Time to write code
See Gio.File.load_bytes for more information.

**load_contents(cancellable)**

Parameters:

- cancellable (Gio.Cancellable) — optional Gio.Cancellable object, null to ignore

Returns:

- ok (Boolean) — true if the this's contents were successfully loaded. false if there were errors.
- contents (ByteArray) — a location to place the contents of the file
- etag_out (String) — a location to place the current entity tag for the file, or null if the entity tag is not needed

Throws exception:

Yes

Loads the content of the file into memory. The data is always zero-terminated, but this is not included in the resultant _length_.
About the API

- Every UI element is based on `Gtk.Widget`
- Roughly equivalent to a HTML DOM element
  - Methods
  - Properties
  - Signals (events)
  - CSS element name and classes
- Things that are not UI elements are based on `GObject.Object`
ES modules

```javascript
import Gdk from "gi://Gtk";
import Gio from "gi://Gio";
import GObject from "gi://GObject";
import Gtk from "gi://Gtk";

import { NotesListItem } from "./item.js";
```
Async operations

- GNOME platform has asynchronous, cancellable I/O
- Experimental opt-in support for JS `await`

```javascript
Gio._promisify(Gio.OutputStream.prototype, 'write_bytes_async', 'write_bytes_finish');

// ...

let bytesWritten = 0;
while (bytesWritten < bytes.length) {
  bytesWritten = await stream.write_bytes_async(bytes, priority, cancellable);
  bytes = bytes.slice(bytesWritten);
}
```
Popular runtime libraries

- These may or may not work
- Check if you actually need the dependency
- Use ES module directly if it doesn't have other deps
- Some modules ship a browser bundle, this might work
- Else, build a UMD bundle with Browserify and vendor it
Build a UMD bundle with browserify

```
yarn add my-library
mkdir -p src/vendor
npx browserify -r my-library -s myLibrary -o src/vendor/my-library.js

import './vendor/my-library.js';
// myLibrary is now a global object
```
Top 5 most used NPM libraries

1. lodash
2. chalk
3. request
4. commander
5. react
Lodash

- In some cases not necessary
- Use lodash-es if you need lodash

```javascript
import _ from './vendor/lodash-es/lodash.js';
_.defaults({ 'a': 1 }, { 'a': 3, 'b': 2 });
```
Chalk

- No bundle, so make a Browserified one
- Color support detection code is Node-only
  - Edit bundle, change `stdout: false` and `stderr: false` to `true`

```javascript
import './vendor/chalk.js';
print(chalk.blue('Hello') + ' World' + chalk.red('!'));
```
Request

- Deprecated
- Use Soup instead

```javascript
const request = require('request');
request('https://ptomato.name', function (error, response, body) {
  console.error('error:', error);
  console.log('statusCode:', response && response.statusCode);
  console.log('body:', body);
});

import Soup from 'gi://Soup';
const session = new Soup.Session();
session.queue_message(msg, (_, {statusCode, responseBody}) => {
  log(`statusCode: ${statusCode}`);
  log(`body: ${responseBody.data}`);
});
```
Commander

- No bundle, so make a Browserified one

```javascript
import System from 'system';
import './vendor/commander.js';
const { Command } = commander;

const options = new Command()
  .option('-p, --pizza-type <type>', 'flavour of pizza')
  .parse(System.programArgs, { from: 'user' })
  .opts(); // ^^^^^^^^^^^^^^^^^

if (options.pizzaType) print(`pizza flavour: ${options.pizzaType}`);
```
React

- Not applicable

P.S. Although it would be cool if React Native worked with GTK
Fast-forward to the **written code**

(Live demo, but in case that doesn't work out, screenshots follow)
To Do List, May 13

* Present talk at LAS
* Attend other talks
* Relax
To Do List, May 13
* Present talk at LAS

Random Ideas
Blobatpad for cats?
Distributing your app to users
How?

- Flathub
- Requirements
  - Luckily, the generated project skeleton meets all of these
  - Only need to fill in a few things
AppStream meta info

- This file is used to provide the description that users see on Flathub
- And in their software updater application
- Description of file format

Polari

An Internet Relay Chat Client for GNOME

Install

Make sure to follow the setup guide before installing

A simple Internet Relay Chat (IRC) client that is designed to integrate seamlessly with GNOME; it features a simple and beautiful interface which allows you to focus on your conversations.

You can use Polari to publicly chat with people in a channel, and to have private one-to-one conversations. Notifications make sure that you never miss an important message — for private conversations, they even allow you to reply instantly without switching back to the application!
AppStream meta info

- **Generator** to get you started
- Asks you a few questions
- Asks for URLs of screenshots
- Flathub guidelines
- **OARS** rating
  - OARS **Generator**
Desktop file

- Tells how to display your app in the desktop
- Description of file format
- List of categories

```plaintext
[Desktop Entry]
Name=Bloatpad
Comment=Unnecessary note-taking application
Exec=name.ptomato.Bloatpad
Icon=name.ptomato.Bloatpad
Terminal=false
Type=Application
Categories=Utility;GTK;
StartupNotify=true
```
Application icon

- Tobias Bernard on Designing an Icon for your App
Submit it to Flathub

- Instructions here
Translate your UI

- Gettext is built-in to the platform
  - Venerable framework for UI translations
- Use a website like Transifex
- Recruit volunteer translators
- Or translate the UI yourself in whatever languages you speak
Conclusion

- Some things might seem familiar to JS developers, others might not
- We should reduce the friction for these developers
- But not everything from the web or Node.js applies well to the desktop
Questions
Thanks

- Andy Holmes, Evan Welsh, Sri Ramkrishna for discussions and their work on improving the GJS developer experience

License

- Presentation licensed under Creative Commons BY-NC-ND 4.0
- Bloatpad code, permissive MIT license