How to Set Up a React+Tauri Environment

Prerequisites

This tutorial will target Linux, primarily Ubuntu. It should be relatively easy to adapt the instructions to other distributions. You'll need Linux (WSL or otherwise), git, and a text editor.

Getting Cargo and Rust

Cargo is Rust's package manager, and it's how Tauri is distributed. If you've already installed Rust and Cargo, you can skip this section. If you've installed using a different method than the one described below, there may be some conflicts. 1. Install the required packages on Ubuntu and Debian by running the following commands:

```
sudo apt update
```

```
sudo apt install libwebkit2gtk-4.0-dev \
    build-essential \
    curl \
    wget \
    file \
    libssl-dev \
    libgtk-3-dev \
    libayatana-appindicator3-dev \
    librsvg2-dev \
    nsis \
    lld \
    llvm
```

2. Rust has a cringe distribution method that involves running a script you download. You can try using your system package manager but it may not work properly. The official method is to run this command: `curl --proto '=https' --tlsv1.2 https://sh.rustup.rs -sSf | sh`. If the TLSv1.2 flag causes issues, you can try removing it.
3. You now have Rust and Cargo, and Tauri's dependencies.

Setting Up PNPM

Because webdev is fun, there is no supported normal way to install Node on Ubuntu/Debian. You either need to use a Node manager or add the NodeSource repository to your package manager. I'll cover FNM, a Node manager, in detail. You can learn more about NodeSource here. PNPM is an alternative to the default Node package manager that is faster and more storage-efficient.

1. For some reason FNM also uses an install script. Use this command to run it: `curl -fsSL https://fnm.vercel.app/install | bash`

2. You now have FNM. Optionally, you can configure it to automatically switch to the specified Node version when you cd into a directory that contains a file specifying a certain Node version. Otherwise, you will need to switch manually whenever required. You can do that by editing your shell's config file and adding a line. If you understood that perfectly, instructions for specific shells and install methods are available here If you didn't understand any of that, do the following: run `sudo apt install nano`. Copy this string: `eval "$(fnm env --use-on-cd)"`. Run `nano ~/.bashrc`. Paste the line you copied at the end of the file.
3. Run `fnm install 20.11.1` and `fnm use 20.11.1`
4. You now have the latest node.js long-term support version at time of writing.
5. Run `corepack enable pnpm` to install pnpm.

Setting Up Tauri

See Tauri's documentation for additional resources.

1. Run the following command to install Tauri with Cargo: `cargo install tauri-cli --locked`

Getting Started

Getting the Codebase

1. You guys know this one: `git clone https://github.com/jaq-lagnirac/cs370-swe.git` or `git@github.com:jaq-lagnirac/cs370-swe.git`
2. Cd into the `org-manager` folder in the repo folder and run `pnpm install` to install all remaining dependencies.

Compiling the App

1. Compiling and running the app locally is as simple as running `cargo tauri dev`

Enabling Windows Compilation

1. Run `rustup target add x86_64-pc-windows-msvc` to enable targeting Windows with Rust.
2. Run `cargo install xwin` to download Windows-related libraries.
3. Run `xwin splat --output ~/.xwin` to make the libraries available. If this doesn't work, you need to add the directory cargo installs software in to your PATH. If you aren't sure how to do that, run `sudo apt install nano` then `nano ~/.bash_profile`. Paste `export PATH="${PATH}:/home/my_user/bin"` on a new line, but replace my_user with your username. Then run `source ~/.bash_profile` and retry

the command. In the org-manager/.cargo folder in the project, create config.toml with a text editor and add the following:

```
[target.x86\_64-pc-windows-msvc]
linker = "lld"
rustflags = [
  "-Lnative=/home/username/.xwin/crt/lib/x86\_64",
  "-Lnative=/home/username/.xwin/sdk/lib/um/x86\_64",
  "-Lnative=/home/username/.xwin/sdk/lib/ucrt/x86_64"
]
```

4. To compile targeting Windows, run `cargo tauri build --target x86_64-pc-windows-msvc`.

Dumping Ground for Stuff Only Required when Creating from Scratch

`corepack use pnpm@latest`, `pnpm add @tauri-apps/api`

I'm supposed to reaad this: https://nextjs.org/docs/app/building-your-application/deploying/static-exports