Shell Scripting for Font Builds

The basics of making font builds that are approachable, scalable, and repeatable

@ArrowType

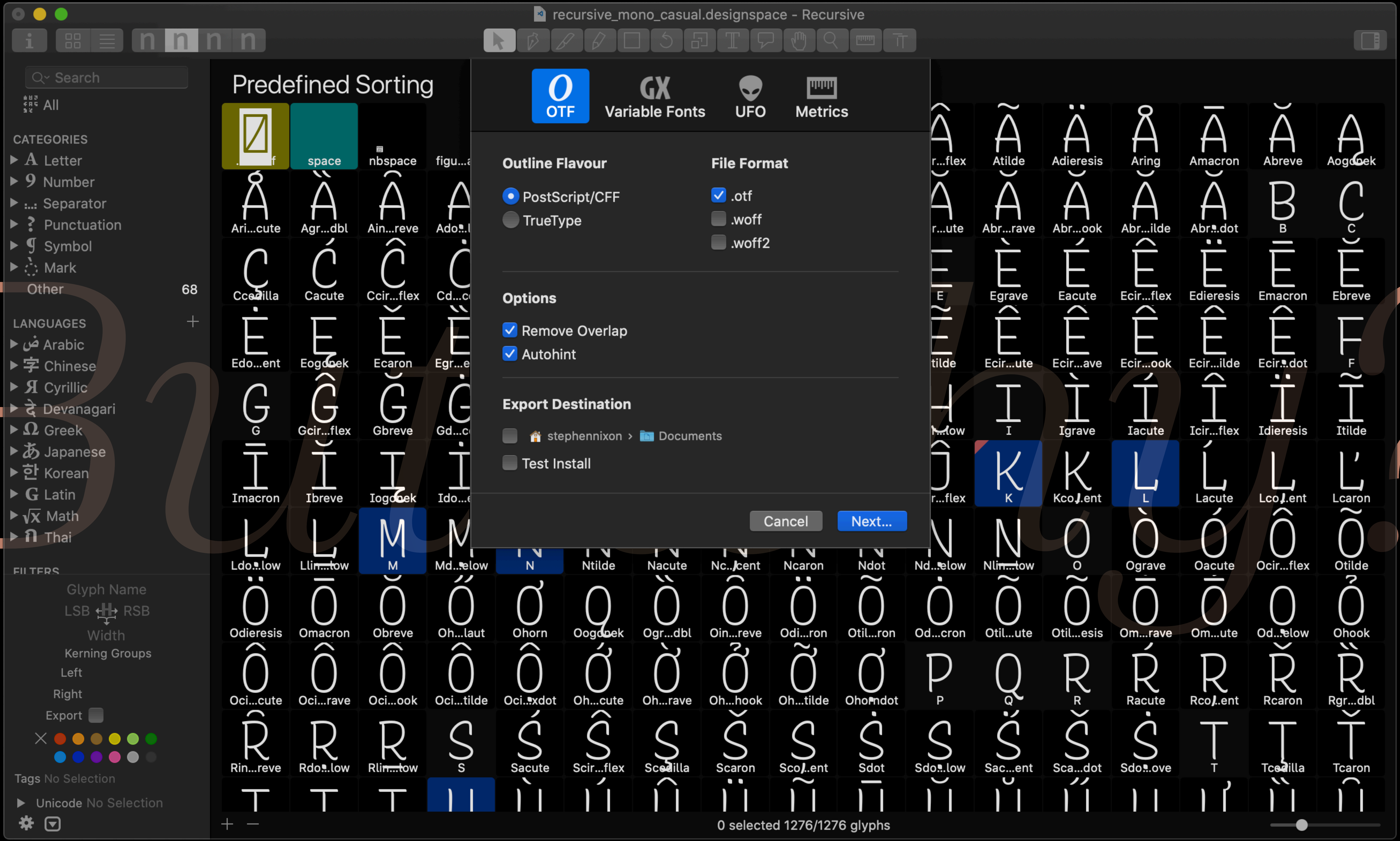
TypeLab 2021

github.com/arrowtype/typelab-2021

DISCLAIMERS

1. This talk is Mac-specific
2. This is just one approach, mostly for .glyphs / .ufo
3. I’m still learning!

DISCLAIMERS



recursive_mono_casual.designspace - Recursive

OTF | Variable Fonts | UFO | Metrics

Outline Flavour

- [✓ PostScript/CFF]
- TrueType

File Format

- [✓ .otf]
- .woff
- .woff2

Options

- [✓ Remove Overlap]
- [✓ Autohint]

Export Destination

- stephennixon › Documents
- Test Install

Cancel | Next...

0 selected 1276/1276 glyphs

Glyphs, RoboFont, FontLab, and other editors are awesome. Why not just use their export tools?

Why build fonts with code?

Building fonts with code is…

- → **Customizable:** fuller control over what you make
- → **Repeatable:** fewer steps to remember, less clicking & dragging
- → **Durable:** open-source build tools are future-proof (mostly)
- → **Debuggable:** you can dig into underlying code to solve problems

fonts used today

Recursive

https://recursive.design

Name Sans

v0.6 on Future Fonts!

Lang Syne

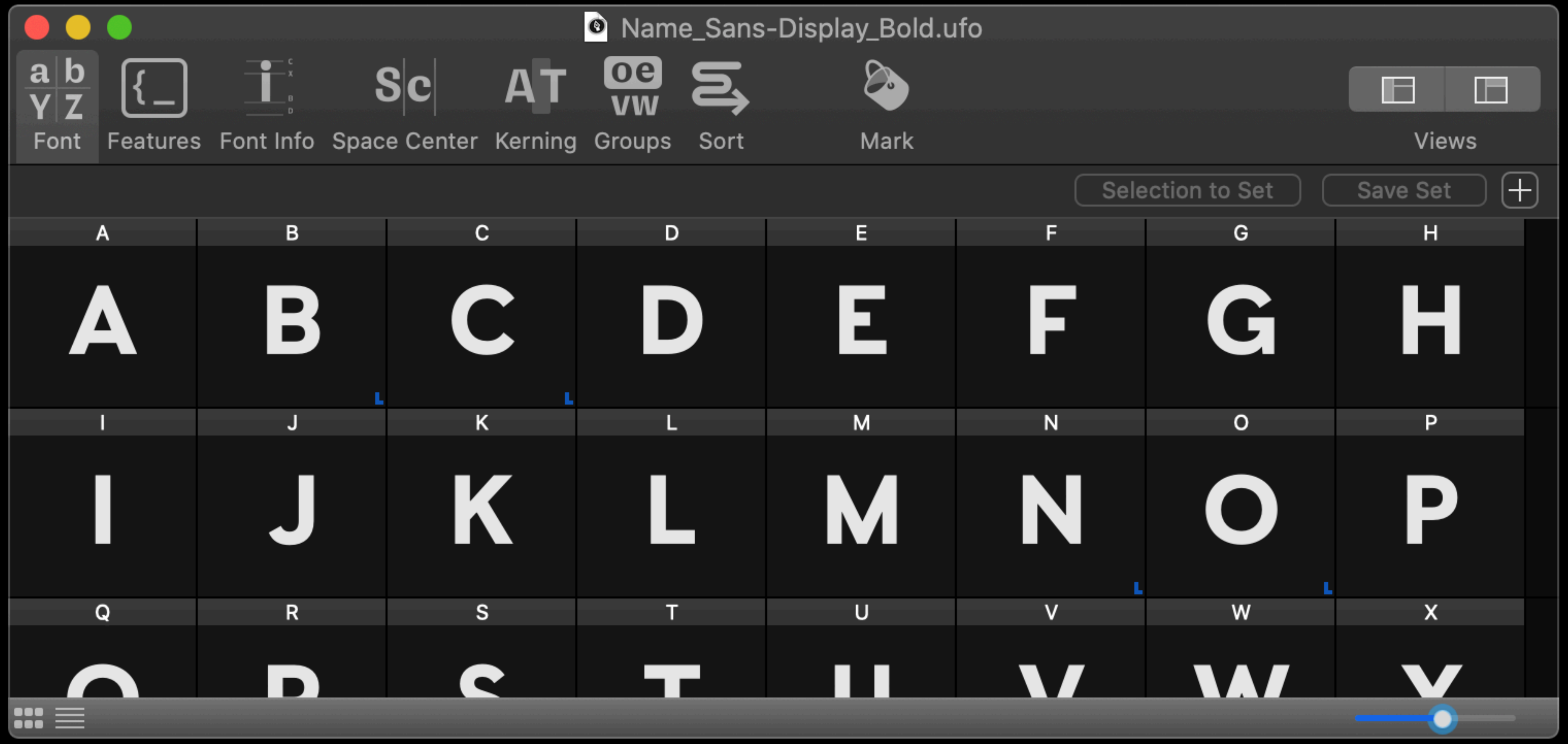
v0.1 coming soon!

fonts used today

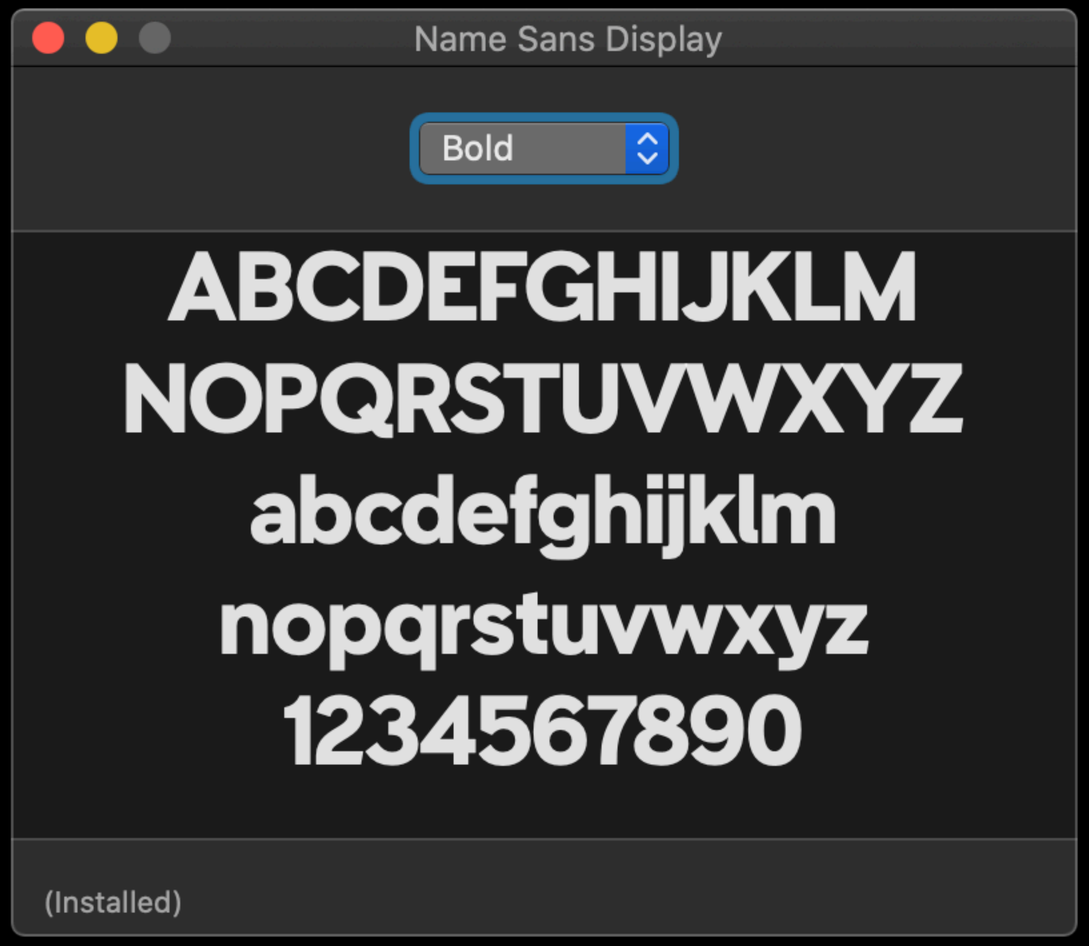
Some useful *definitions*

Font Building

→ The process of creating working font files (.ttf, .otf, .woff2, etc) from the type sources you draw (.ufo, .glyphs, .vfb, etc).

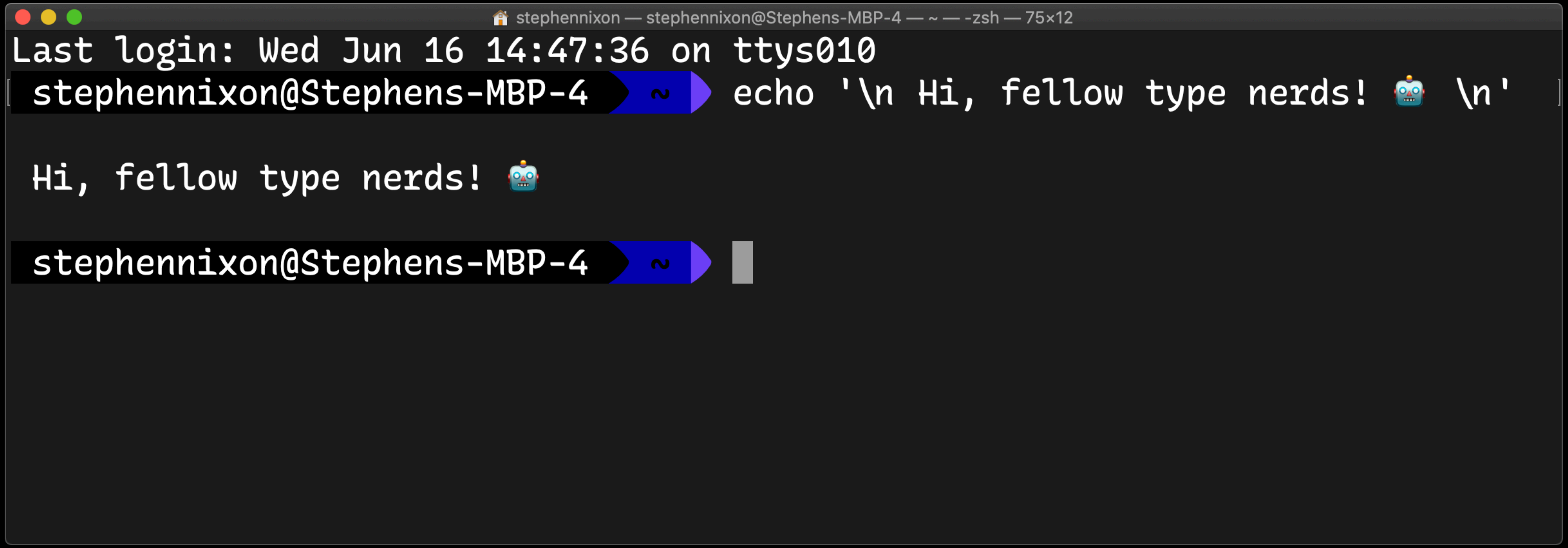


→



Terminal / Shell / Command Line

→ A tool that lets you control a computer with code


stephennixon — stephennixon@Stephens-MBP-4 — ~ — -zsh — 75×12

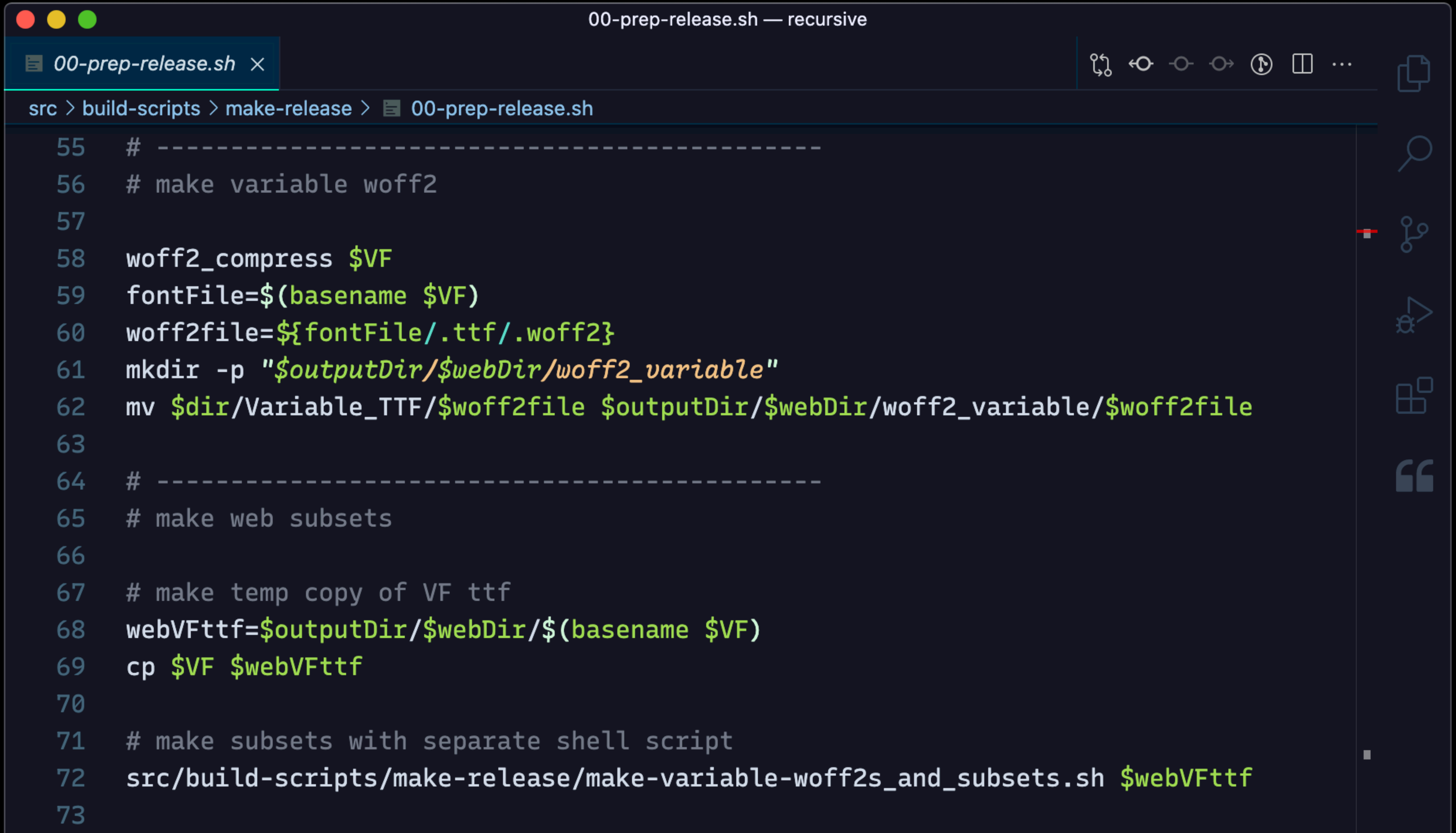
```
Last login: Wed Jun 16 14:47:36 on ttys010
stephennixon@Stephens-MBP-4  ~  echo '\n Hi, fellow type nerds! 🤖 \n'

 Hi, fellow type nerds! 🤖

stephennixon@Stephens-MBP-4  ~  
```

Shell Scripts

→ Scripts that allow you to program a series of shell commands


00-prep-release.sh — recursive
00-prep-release.sh
src > build-scripts > make-release > 00-prep-release.sh

```
# ---------------------------------------------
# make variable woff2

woff2_compress $VF
fontFile=$(basename $VF)
woff2file=${fontFile/.ttf/.woff2}
mkdir -p "$outputDir/$webDir/woff2_variable"
mv $dir/Variable_TTF/$woff2file $outputDir/$webDir/woff2_variable/$woff2file

# ---------------------------------------------
# make web subsets

# make temp copy of VF ttf
webVFttf=$outputDir/$webDir/$(basename $VF)
cp $VF $webVFttf

# make subsets with separate shell script
src/build-scripts/make-release/make-variable-woff2s_and_subsets.sh $webVFttf
```

Why use shell scripting?

Shell scripting is…

→ **Supported:** many font dev tools have Command-Line Interfaces (CLIs)

→ **Helpful:** you **could** remember CLI commands, but you don't have to

→ **Powerful:** you can sequence many tools & steps in a font build, easily

→ **Concise:** a shell script can coordinate CLIs, Python, and other code

A few *details*

A typical build workflow might include...

- → **Prep:** take working source UFOs and set info, remove draft glyphs, etc
- → **Build:** build sources into static/variable fonts, fix font data in post
- → **Organize:** sort outputs into a custom folder structure, copy in docs
- → **Test:** run FontBakery to check for errors in font data
- → **Proof:** make PDF specimens with DrawBot, web tests with Python, etc

The anatomy of a terminal command

```
cd <dest>
```

Program

e.g. “Change Directory”

Argument(s)

Angle brackets mean “your argument goes here”

Basic Terminal commands

`cd <dest>` – change directory (move location)

`mv <path> <dest>` – move a file to another path

`cp <path> <dest>` – copy a file to another path

`echo <text>` – print text to output

`say <text>` – speak text aloud in a robotic computer voice

Flags

```
fontmake --help
```

Program

e.g. FontMake, a program that builds fonts

Flag(s)

- Most programs have a “**`--help`**” flag
- Flags specify optional arguments
- Many flags have abbreviations, like “**`-h`**”

A full command

Variable

```
designspace=“sources/Example.designspace"
```

```
fontmake -m $designspace -o variable --output-path "fonts/Example.ttf"
```

Program

Flags with **Arguments**

I could show syntax all day,

but...

DEMO TIME

Drawbacks

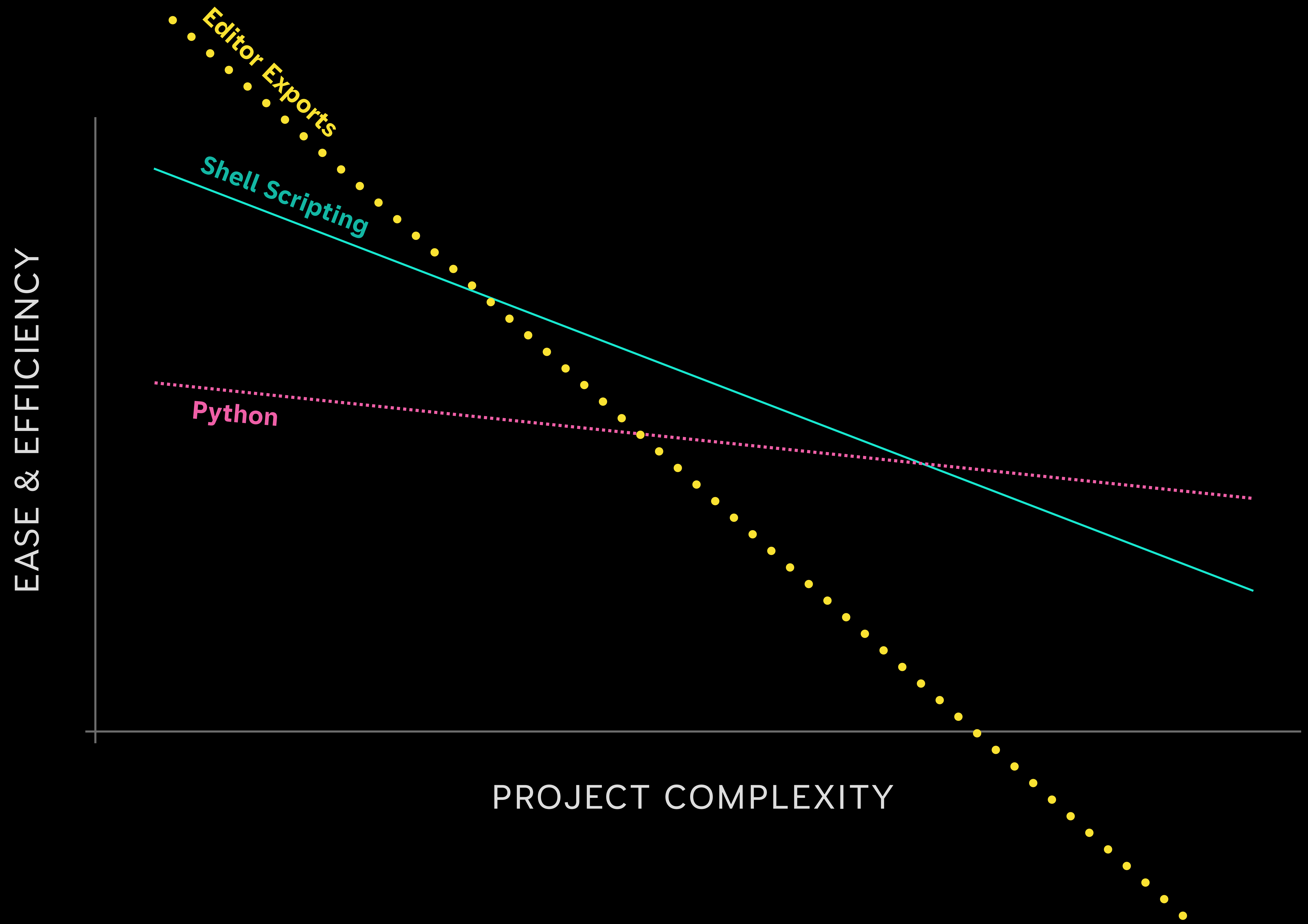
Compared to Python, shell scripting can be…

→ **Annoying:** syntax can be picky, and some things require Googling

→ **Semi-repetitive:** Python packages are better for repeat-use code

→ **Inflexible:** shell scripts are best when kept concise & high-level

an unscientific chart



At the end of the day…

Shell scripting is so useful & approachable, it’s worth learning.

Where to learn more

How to Create and Use Bash Scripts – By Tania Rascia

A Guide to Python’s Virtual Environments – Matthew Sarmiento

Open-source font projects like Recursive

Git repos for FontMake, FontBakery, woff2, GF Tools, and FontTools github.com/arrowtype/typelab-2021

@ArrowType