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

fonts used today

Recursive

Name Sans

Work in progress!

Lang Syne

Coming soon!

fonts used today

disclaimers

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

disclaimers

Why build fonts with code?

Building fonts with code is…

→ **Repeatable:** fewer steps to remember, less clicking & dragging

→ **Durable:** open-source build tools will still work in 10+ years

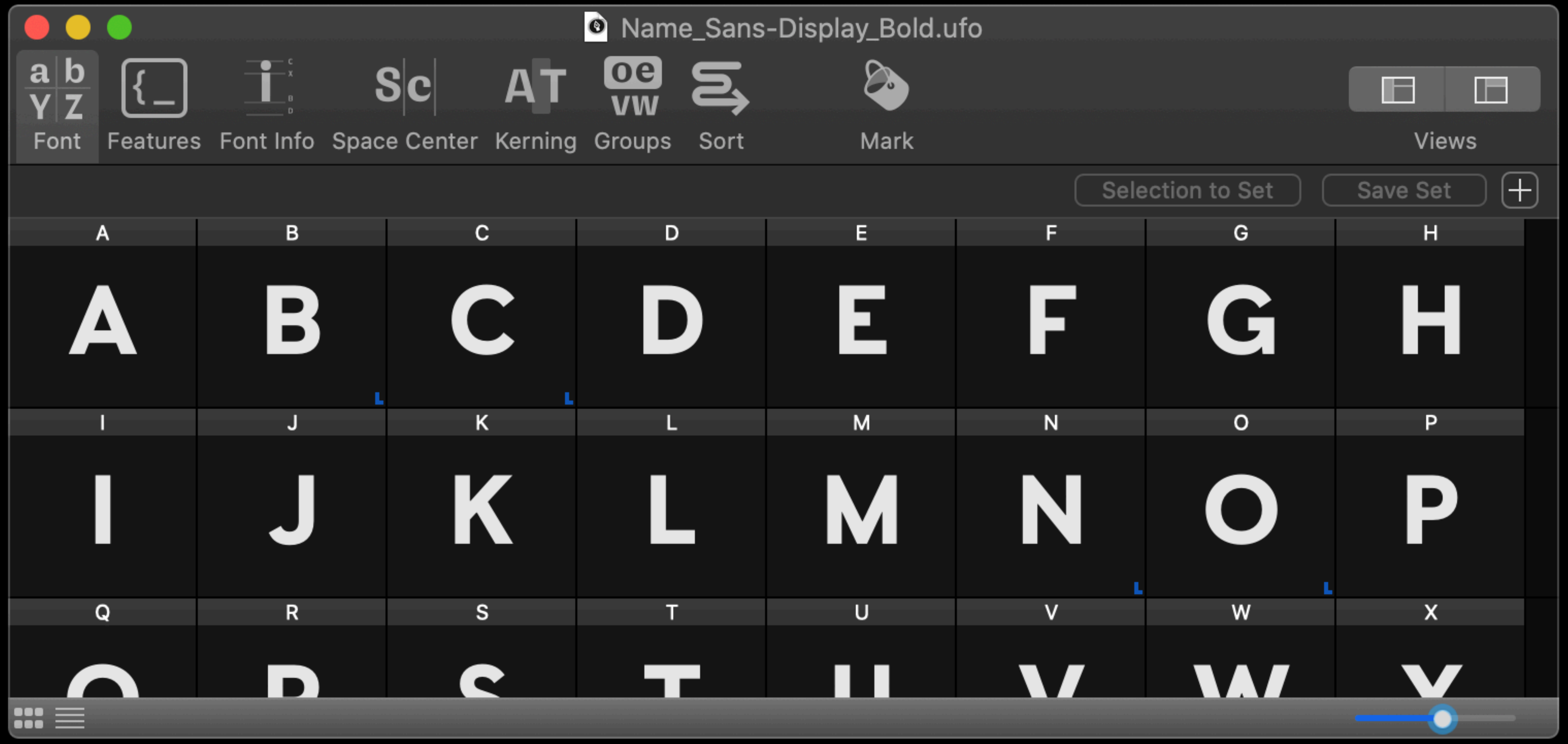
→ **Customizable:** full control over outputs, sorting, and more

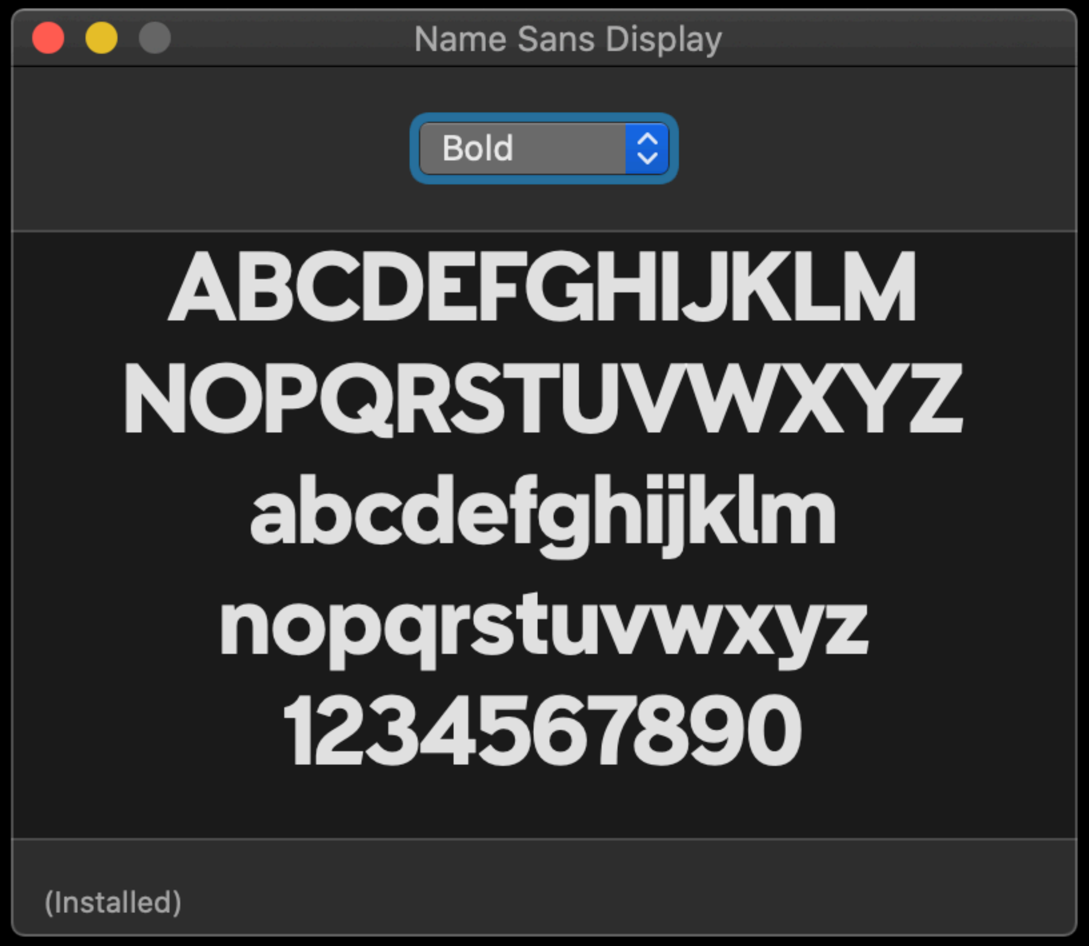
→ **Debuggable:** you can dig into build tools to find & solve problems

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).


Name_Sans-Display_Bold.ufo


Name Sans Display

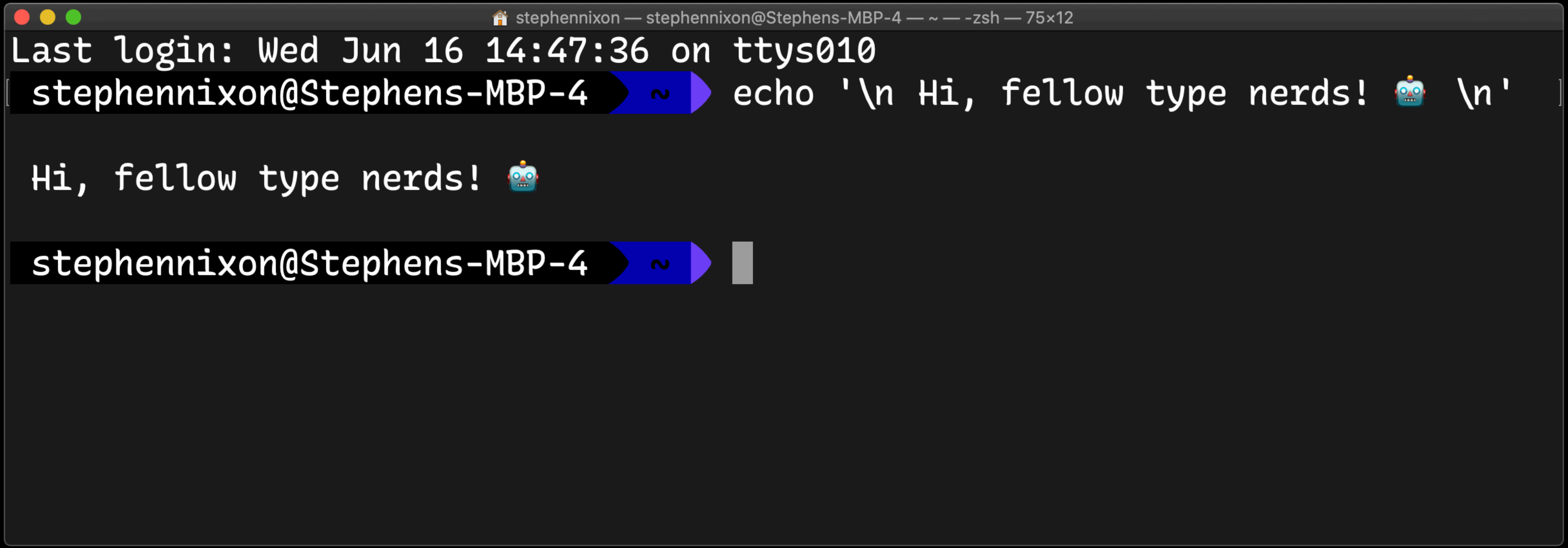
Bold

ABCDEFGHIJKLM
NOPQRSTUVWXYZ
abcdefghijklm
nopqrstuvwxyz
1234567890

(Installed)

Terminal / Shell / Command Line

→ A tool that lets you control a computer with text



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

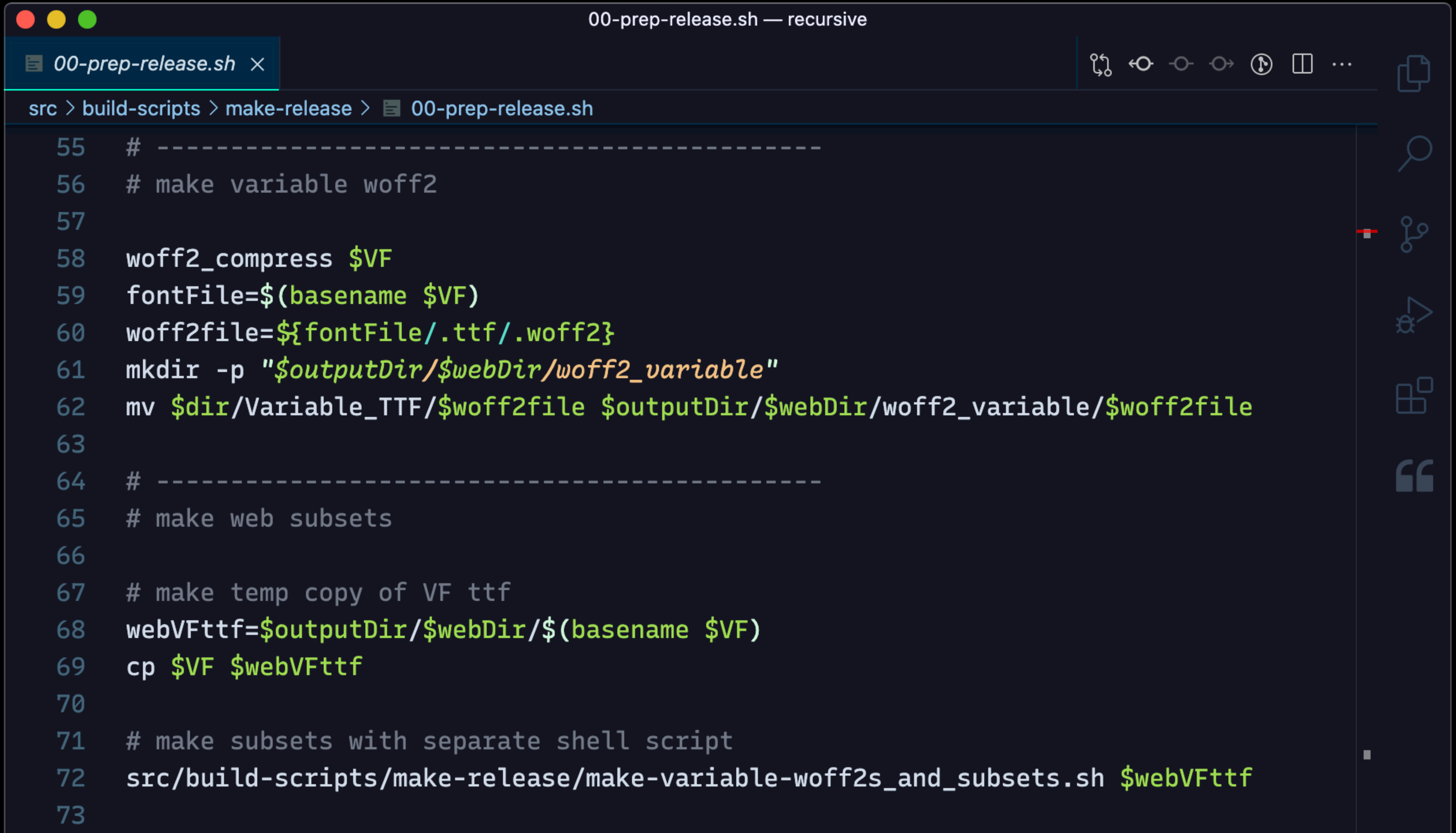
```
Last login: Wed Jun 16 14:47:36 on ttys010
 stephennixon@Stephens-MBP-4  ~  echo '\n Hi, fellow type nerds! 🤖 \n'

 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 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

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

We could discuss syntax all day,

but...

DEMO TIME

Drawbacks vs Python

Shell scripting can be...

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

→ **Inefficient:** a Python package is probably better for repeat-use code

→ **Not as flexible:** shell scripts are best when kept concise & high-level

Nevertheless

Shell scripting is so approachable and useful, it’s a nice tool to have in your mental toolbox.

Where to learn more

How to Create and Use Bash Scripts – By Tania Rascia

A Guide to Python's Virtual Environments – Matthew Sarmiento

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

@ArrowType