MacOS setup

Created: January 13 2021 UTC, 23:18:46 Modified: January 13 2021 UTC, 23:19:54
Note URL:
https://www.evernote.com/Home.action#n=260d4ff1-777e-c92b-0c60-9ea76f2ae6fb&b=24c8b1ee-0e30-4e09-bb62-633842194afc&ses=4&sh=1&sds=5

```
# Move to your download directory and use your terminal to run the script and install selected apps and packages by running 'sh setup.sh'

# Run the script: `$ ./macbook-setup.sh`

#!/bin/bash

### Must do first ###

# Ask for the administrator password upfront.
sudo -v

# Install Xcode
xcode-select --install

# Keep-alive: update existing `sudo` time stamp until `.macos` has finished
while true; do sudo -n true; sleep 60; kill -0 "$$" || exit; done 2>/dev/null &

# Install   Homebrew"
/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

# Allow third party software
sudo spctl --master-disable


# # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # # #

echo "  Starting setup"

# Install Oh-My-Zsh
sh -c "$(curl -fsSL https://raw.github.com/robbyrussell/oh-my-zsh/master/tools/install.sh)"

# Change login shell to zsh
chsh -s /bin/zsh

# Add /usr/local/bin/sbin to Default Path
install_paths () {
  if ! grep -Fq "/usr/local/sbin" /etc/paths; then
    sudo sed -i "" -e "/\/usr\/sbin/{x;s/$/\/usr\/local\/sbin/;G;}" /etc/paths
  fi
}

# Look for developer tools (needed for Homebrew)
xcode-select -p
if [ $? -eq 0 ]; then
    echo "Found XCode Tools"
else
```

```
    echo "Installing XCode Tools"

    xcode-select --install
fi

# Flags
set -e # Global exit on error flag
set -x # Higher verbosity for easier debug
set -o pipefail # Exit on pipe error

# Add Homebrew Packages to Brewfile
install_brewfile_brew_pkgs () {
  printf "%s\n" "${_pkgs}" | \
  while IFS="$(printf '\t')" read pkg; do
    # printf 'brew "%s", args: [ "force-bottle" ]\n' "${pkg}" >> "${BREWFILE}"
    printf 'brew "%s"\n' "${pkg}" >> "${BREWFILE}"
  done
  printf "\n" >> "${BREWFILE}"
}

# Install Git
brew install git&
which git
# Download the .gitconfig file to your home directory:
cd ~
curl -O https://raw.githubusercontent.com/nicolashery/mac-dev-setup/master/.gitconfig

# Install Node.js with nodenv
install_node_sw () {
  if which nodenv > /dev/null; then
    NODENV_ROOT="/usr/local/node" && export NODENV_ROOT

    sudo mkdir -p "$NODENV_ROOT"
    sudo chown -R "$(whoami):admin" "$NODENV_ROOT"

    p "Installing Node.js with nodenv"
    git clone https://github.com/nodenv/node-build-update-defs.git \
      "$(nodenv root)"/plugins/node-build-update-defs
    nodenv update-version-defs > /dev/null

    nodenv install --skip-existing 8.7.0
    nodenv global 8.7.0

    grep -q "${NODENV_ROOT}" "/etc/paths" || \
    sudo sed -i "" -e "1i\\
${NODENV_ROOT}/shims
" "/etc/paths"

    init_paths
    rehash
  fi

  T=$(printf '\t')
```

```
  printf "%s\n" "$_npm" | \
  while IFS="$T" read pkg; do
    npm install --global "$pkg"
  done

  rehash
}

git config --global user.name "Liselot3";
git config --global user.email "sweetthunder33@gmail.com"

git config --global credential.helper osxkeychain

# Install Python with pyenv
install_python_sw () {
  if which pyenv > /dev/null; then
    CFLAGS="-I$(brew --prefix openssl)/include" && export CFLAGS
    LDFLAGS="-L$(brew --prefix openssl)/lib" && export LDFLAGS
    PYENV_ROOT="/usr/local/python" && export PYENV_ROOT

    sudo mkdir -p "$PYENV_ROOT"
    sudo chown -R "$(whoami):admin" "$PYENV_ROOT"

    p "Installing Python 2 with pyenv"
    pyenv install --skip-existing 2.7.13
    p "Installing Python 3 with pyenv"
    pyenv install --skip-existing 3.6.2
    pyenv global 2.7.13

    grep -q "${PYENV_ROOT}" "/etc/paths" || \
    sudo sed -i "" -e "1i\\
${PYENV_ROOT}/shims
" "/etc/paths"

    init_paths
    rehash

    pip install --upgrade "pip" "setuptools"

    # Reference: https://github.com/mdhiggins/sickbeard_mp4_automator
    pip install --upgrade "babelfish" "guessit<2" "qtfaststart" "requests" "stevedore==1.19.1" "subliminal<2"
    pip install --upgrade "requests-cache" "requests[security]"

    # Reference: https://github.com/pixelb/crudini
    pip install --upgrade "crudini"
  fi
}

# Install Ruby with rbenv
install_ruby_sw () {
  if which rbenv > /dev/null; then
    RBENV_ROOT="/usr/local/ruby" && export RBENV_ROOT
```

```
    sudo mkdir -p "$RBENV_ROOT"
    sudo chown -R "$(whoami):admin" "$RBENV_ROOT"

    p "Installing Ruby with rbenv"
    rbenv install --skip-existing 2.4.2
    rbenv global 2.4.2

    grep -q "${RBENV_ROOT}" "/etc/paths" || \
    sudo sed -i "" -e "1i\\
${RBENV_ROOT}/shims
" "/etc/paths"

    init_paths
    rehash

    printf "%s\n" \
      "gem: --no-document" | \
    tee "${HOME}/.gemrc" > /dev/null

    gem update --system > /dev/null

    trash "$(which rdoc)"
    trash "$(which ri)"
    gem update

    gem install bundler
  fi
}

# Add Homebrew Taps to Brewfile
install_brewfile_taps () {
  printf "%s\n" "${_taps}" | \
  while IFS="$(printf '\t')" read tap; do
    printf 'tap "%s"\n' "${tap}" >> "${BREWFILE}"
  done
  printf "\n" >> "${BREWFILE}"
}

# Add Caskroom Options to Brewfile
install_brewfile_cask_args () {
  printf 'cask_args \' >> "${BREWFILE}"
  printf "%s\n" "${_args}" | \
  while IFS="$(printf '\t')" read arg dir; do
    printf '\n  %s: "%s",' "${arg}" "${dir}" >> "${BREWFILE}"
  done
  sed -i "" -e '$ s/,/\
/' "${BREWFILE}"
}

# Add Homebrew Casks to Brewfile
install_brewfile_cask_pkgs () {
  printf "%s\n" "${_casks}" | \
```

```
  while IFS="$(printf '\t')" read cask; do
    printf 'cask "%s"\n' "${cask}" >> "${BREWFILE}"
  done
  printf "\n" >> "${BREWFILE}"
}

# Add App Store Packages to Brewfile
install_brewfile_mas_apps () {
  open "/Applications/App Store.app"
  run "Sign in to the App Store with your Apple ID" "Cancel" "OK"

  MASDIR="$(getconf DARWIN_USER_CACHE_DIR)com.apple.appstore"
  sudo chown -R "$(whoami)" "${MASDIR}"
  rsync -a --delay-updates \
    "${CACHES}/mas/" "${MASDIR}/"

  printf "%s\n" "${_mas}" | \
  while IFS="$(printf '\t')" read app id; do
    printf 'mas "%s", id: %s\n' "${app}" "${id}" >> "${BREWFILE}"
  done
}

### Install packages
brew install npm
brew install xcodegen
brew install git
brew install node
brew install ruby
brew install rsync
brew install openssl
brew install trash
brew install terminal-notifier
brew install zsh
brew install zsh-syntax-highlighting
brew install environments
brew install python3 # Python version 3.7, preinstalled is 2.7
brew install pandoc # Markup to Word/Open office converter needed by Typora
brew install bash # newest bash version. System's default is 3
brew install coreutils # GNU File, Shell, and Text utilities
brew install dmg2img # Utilities for converting macOS DMG images
brew install wget # Internet file retriever
brew install mint # Dependency mng, installs & runs Swift command-line tool packages

#Unused
#brew install vapor/tap/vapor # backend framework

echo "  Updating homebrew..."
brew update

#Install Ruby gems
GEMS=(
    xcpretty
    bundler
```

```
)

echo "  Installing Ruby gems"
sudo gem install ${GEMS[@]} -N


echo "  Cleaning up..."
brew cleanup -s


### Install non-AppStore apps
nonAppStoreApps=(
#Browsers
   google-chrome
#Developer
   visual-studio-code # Modern code editor with community-driven plugins
   iterm2 # Alternative terminal
   sublime-text # Cross-platform code editor with own high performance rendering engine
   pycharm
   visual-studio-code
   visual-studio
   appcode
   haskell-for-mac
   atom
   brackets
   sublime-text
   macdown
   little-snitch
   github
   gitfinder
   clipy
   iterm3
   google-cloud-sdk
#HDD
   drivedx # SMART status and HDD health tool
   paragon-ntfs # Support for NTFS file system
#Video & Audio
   vlc # Most popular cross-platform video watching app
   handbrake # Video Transcoder
   spotify
#Notes/Education
   evernote
   microsoft-teams
#Graphics
   gimp # Graphics editor
#Cloud Storage
   dropbox
   google-backup-and-sync
#Social
   whatsapp
#Utilities
   wavebox
   Dashlane # Password Manager
```

```
    find-empty-folders
    bettertouchtool # Custom gestures for touchpad and touchbar & reverse scrolling setting
#Other
    grammarly
    evernote
    transmission # Torrents client
    calibre # Mobi/epub e-book converter
    anki # App for learning with flashcard
#Unused
    # microsoft-office
    # skype # Communicator
    # virtualbox # Virtual Machine
    # virtualbox-extension-pack # Extensions for virtualbox such as display resolution and USB
    # docker # App to make containers for environments
    # zoomus # Video conference App
)

echo "Install non-AppStore apps"
brew cask install ${nonAppStoreApps[@]}

# Install AppStore CLI installer
brew install mas

### Install AppStore apps
appStoreApps=(
    497799835 # Xcode (Apple IDE)
    462054704 # Microsoft Word (Documents editor)
    462058435 # Microsoft Excel (Spreadsheets editor)
    462062816 # Microsoft PowerPoint (Presentations editor)
    985367838 # Microsoft Outlook (Email client)
    1388020431 # DevCleaner for Xcode (deletes old Xcode files in ~/Library/Developer folder)
#Unused
    # 425424353 # The Unarchiver (Archives extractor)
    # 405399194 # Kindle (Mobi file format reader)
    # 985367838 # Microsoft Outlook (Email client)
    # 1278508951 # Trello (Project management tool)
    # 430798174 # HazeOver (App that dims unfocused windows)
    # 425955336 # Skitch (App for marking pictures)
    # 411643860 # Daisy Disk (App for recovering disk space)
)

echo "Install AppStore apps"
mas install ${appStoreApps[@]}

# python
sudo easy_install pip;
pip install --upgrade;

#Link System Utilities to Applications
install_links () {
  printf "%s\n" "${_links}" | \
  while IFS="$(printf '\t')" read link; do
    find "${link}" -maxdepth 1 -name "*.app" -type d -print0 2> /dev/null | \
```

```
    xargs -0 -I {} -L 1 ln -s "{}" "/Applications" 2> /dev/null
  done
}

# Install terminal colors for Bash (choose between light and dark theme)
echo -e "export CLICOLOR=1\n#light theme\nexport LSCOLORS=ExFxBxDxCxegedabagacad\n#dark theme\n#export
LSCOLORS=GxFxCxDxBxegedabagaced" >> ~/.bash_profile

# Fix Catalina bug with "Insecure completion-dependent directories detected"
compaudit | xargs chmod g-w,o-w

# Install terminal colors for zsh (light theme)
# Use this generator to translate BSD colors and Linux colors: https://geoff.greer.fm/lscolors/
echo -e 'export LSCOLORS="ExFxBxDxCxegedabagacad"' >> ~/.zshrc
echo -e 'export LS_COLORS="di=1;34:ln=1;35:so=1;31:pi=1;33:ex=1;32:bd=34;46:cd=34;43:su=30;41:sg=30;46:tw=30;42:ow=30;43"'
>> ~/.zshrc
echo -e "zstyle ':completion:*:default' list-colors \${(s.:.)LS_COLORS}" >> ~/.zshrc

# Uncomment en_US.UTF-8 locale for console, making them undependable from macOS locale settings
sed -i -e 's/# export LANG=en_US.UTF-8/export LANG=en_US.UTF-8/g' ~/.zshrc

# Copy SF Mono font (available only in Xcode and Terminal.app) to the system
cp -R /System/Applications/Utilities/Terminal.app/Contents/Resources/Fonts/. /Library/Fonts/

# Remove ALL icons (except Finder) from dock
echo "Removing all icons (except Finder) from the dock…"
defaults write com.apple.dock persistent-apps -array

# Install developer friendly quick look plugins; see https://github.com/sindresorhus/quick-look-plugins
brew cask install qlcolorcode qlstephen qlmarkdown quicklook-json qlprettypatch quicklook-csv betterzip qlimagesize webpquicklook
suspicious-package quicklookase qlvideo

# List of dock icons
dockIcons=(
    /System/Applications/Utilities/Terminal.app
    /System/Applications/Mail.app
    /System/Applications/Messages.app
    /Applications/Safari.ap
    "/Applications/Visual Studio Code.app"
    /Applications/Xcode.app
)

# Adding icons
for icon in "${dockIcons[@]}"
do
    echo "Adding $icon to the dock…"
    defaults write com.apple.dock persistent-apps -array-add
    "<dict><key>tile-data</key><dict><key>file-data</key><dict><key>_CFURLString</key><string>$icon</string>
<key>_CFURLStringType</key><integer>0</integer></dict></dict></dict>"
done

echo "Setting up dock size…"
defaults write com.apple.dock tilesize -int 40
```

```
echo "Restarting dock…"
killall Dock

# As we installed homebrew before xcode, we need to switch to Xcode Command Line Tools
sudo xcode-select -s /Applications/Xcode.app/Contents/Developer

# List of mint packages
mintPackages=(
   MakeAWishFoundation/SwiftyMocky # Library to autogenerate mocks
)

mint install ${mintPackages[@]}

# install CocoaPods dependency manager for iOS apps
sudo gem install cocoapods

# install CocoaPods Keys plugin
sudo gem install cocoapods-keys

### Set Preferences ###

# Xcode won't ask for password with every build
sudo DevToolsSecurity -enable

# Create symlinks between home folder & iCloud folders
#WARNING This command uses "~" sign, so don't just copy paste it using ZSH. Use bash for this one!
ln -s ~/ ~/Library/Mobile\ Documents/com~apple~CloudDocs/$(whoami)\ home\ symlink
ln -s ~/Library/Mobile\ Documents/com~apple~CloudDocs/Documents ~/Documents\ symlink
ln -s ~/Desktop ~/Desktop\ symlink

# Make a Developer folder synchronise
mkdir ~/Library/Mobile\ Documents/com~apple~CloudDocs/Developer
ln -s ~/Library/Mobile\ Documents/com~apple~CloudDocs/Developer ~/Developer\ symlink

# Show debug menu on AppStore
sudo defaults write com.apple.appstore ShowDebugMenu -bool true

# Show debug menu on contacts
sudo defaults write com.apple.addressbook ABShowDebugMenu -bool true

# Remove Apple Remote Desktop Settings
sudo rm -rf /var/db/RemoteManagement ; \
sudo defaults delete /Library/Preferences/com.apple.RemoteDesktop.plist ; \
defaults delete ~/Library/Preferences/com.apple.RemoteDesktop.plist ; \
sudo rm -r /Library/Application\ Support/Apple/Remote\ Desktop/ ; \
rm -r ~/Library/Application\ Support/Remote\ Desktop/ ; \
rm -r ~/Library/Containers/com.apple.RemoteDesktop

# Use Plain Text Mode as Default
defaults write com.apple.TextEdit RichText -int 0

# Remove All Unavailable Simulators on Xcode
```

```
xcrun simctl delete unavailable

#brew
brew install emacs --with-cocoa;
brew linkapps emacs

# Link all applications
mkdir $HOME/Applications /
brew linkapps

#DOCK
#Auto Rearrange Spaces Based on Most Recent Use
defaults write com.apple.dock mru-spaces -bool true && \
killall Dock
# Lock the Dock Size
defaults write com.apple.Dock size-immutable -bool yes && \
killall Dock

# Convert File to HTML
# Supported formats are plain text, rich text (rtf) and Microsoft Word (doc/docx).
textutil -convert html file.ext

# Activate And Restart the ARD Agent and Helper
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -activate -restart -agent -
console

# Deactivate and Stop the Remote Management Service
sudo /System/Library/CoreServices/RemoteManagement/ARDAgent.app/Contents/Resources/kickstart -deactivate -stop

# Remove Apple Remote Desktop Settings
sudo rm -rf /var/db/RemoteManagement ; \
sudo defaults delete /Library/Preferences/com.apple.RemoteDesktop.plist ; \
defaults delete ~/Library/Preferences/com.apple.RemoteDesktop.plist ; \
sudo rm -r /Library/Application\ Support/Apple/Remote\ Desktop/ ; \
rm -r ~/Library/Application\ Support/Remote\ Desktop/ ; \
rm -r ~/Library/Containers/com.apple.RemoteDesktop


##Visual Studio Code##
# Enable VSCode key repeat
defaults write com.microsoft.VSCode ApplePressAndHoldEnabled -bool false

# Remove All Unavailable Simulators
xcrun simctl delete unavailable;
sudo diskutil repairPermissions /
sudo systemsetup -setstartupdisk /System/Library/CoreServices

#clear all ACLs
sudo chmod -RN /

# Disable guest user
sudo dscl . -delete /Users/Guest;
sudo security delete-generic-password -a Guest -s com.apple.loginwindow.guest-account -D "application password"
```

```
/Library/Keychains/System.keychain;
sudo defaults write /Library/Preferences/com.apple.loginwindow GuestEnabled -bool FALSE

# Disable compatibility check
sudo nvram boot-args="-no_compat_check"

###############################################################################
# Finder                                                                      #
###############################################################################

# Show All File Extensions
sudo defaults write -g AppleShowAllExtensions -bool true
# Show Full Path in Finder Title
sudo defaults write com.apple.finder _FXShowPosixPathInTitle -bool true
# Show "Quit Finder" Menu Item
sudo defaults write com.apple.finder QuitMenuItem -bool true && \
killall Finder
# Expand Save Panel by Default
sudo defaults write -g NSNavPanelExpandedStateForSaveMode -bool true && \
sudo defaults write -g NSNavPanelExpandedStateForSaveMode2 -bool true
# Show path bar
sudo defaults write com.apple.finder ShowPathbar -bool true
# Show status bar
sudo defaults write com.apple.finder ShowStatusBar -bool true
# Save to Disk by Default, not iCloud
sudo defaults write -g NSDocumentSaveNewDocumentsToCloud -bool false
# Set Sidebar icon size to medium
sudo defaults write -g NSTableViewDefaultSizeMode -int 2
# Disable Creation of Metadata Files on USB Volumes ( .DS_Store )
defaults write com.apple.desktopservices DSDontWriteUSBStores -bool true
# Change Working Directory to Finder Path
cd "$(osascript -e 'tell app "Finder" to POSIX path of (insertion location as alias)')
# Keep folders on top when sorting by name
sudo defaults write com.apple.finder _FXSortFoldersFirst -bool true"
# Allow quitting via ⌘ + Q; doing so will also hide desktop icons
sudo defaults write com.apple.finder QuitMenuItem -bool true
# When performing a search, search the current folder by default
sudo defaults write com.apple.finder FXDefaultSearchScope -string "SCcf"
# Show the /Volumes folder
sudo chflags nohidden /Volumes
# Expand the following File Info panes:
# “General”, “Open with”, and “Sharing & Permissions”
sudo defaults write com.apple.finder FXInfoPanesExpanded -dict \
   General -bool true \
   OpenWith -bool true \
   Privileges -bool true

# Automatic Restart on System Freeze
sudo systemsetup -setrestartfreeze on

# Menu bar: show remaining battery  percentage
defaults write com.apple.menuextra.battery ShowPercent -string "YES"
```

```
# Configure Guest Users
  sudo sysadminctl -guestAccount off

# Configure   system
defaults write -g InitialKeyRepeat -int 15
defaults write com.apple.AppleMultitouchTrackpad Clicking -bool true
defaults write -g KeyRepeat -int 2

# Eliminate prompts for passwords
printf "%s\n" "%wheel ALL=(ALL) NOPASSWD: ALL" | \
sudo tee "/etc/sudoers.d/wheel" > /dev/null && \
sudo dscl /Local/Default append /Groups/wheel GroupMembership "$(whoami)"

# Set Defaults for Sleep
sudo pmset -a sleep 0
sudo pmset -a disksleep 0

# App settings
set -e         # Exit on error
set -o pipefail # Exit on pipe error
set -x         # Enable verbosity

# Disable recently played videos
sudo defaults write org.videolan.vlc NSRecentDocumentsLimit 0;
sudo defaults write org.videolan.vlc.LSSharedFileList RecentDocuments -dict-add MaxAmount 0

# Set Software Update Check Interval (check daily instead of weekly)
sudo defaults write com.apple.SoftwareUpdate ScheduleFrequency -int 1

# Set the dark style
sudo defaults write NSGlobalDomain AppleInterfaceStyle -string 'Dark'

# Menu bar: show remaining battery  percentage
sudo defaults write com.apple.menuextra.battery ShowPercent -string "YES"

# Disable the “Are you sure you want to open this application?” dialog
sudo defaults write com.apple.LaunchServices LSQuarantine -bool false

# Disable the crash reporter
sudo defaults write com.apple.CrashReporter DialogType -string "none"

# Disable Notification Center and remove the menu bar icon
launchctl unload -w /System/Library/LaunchAgents/com.apple.notificationcenterui.plist 2> /dev/null

# Enable full keyboard access for all controls (e.g. enable Tab in modal dialogs)
defaults write NSGlobalDomain AppleKeyboardUIMode -int 3

# Disable machine sleep while charging
sudo pmset -c sleep 0

# Save screenshots to the desktop
defaults write com.apple.screencapture location -string "${HOME}/Desktop"
```

```
# Save screenshots in PNG format (other options: BMP, GIF, JPG, PDF, TIFF)
defaults write com.apple.screencapture type -string "png"

# Remove Gatekeeper Exception
spctl --remove /path/to/Application.app

# Manage Gatekeeper
# Especially helpful with the annoying macOS 10.15 (Catalina) system popup blocking execution of non-signed apps.
# Status
spctl --status
# Enable (Default)
sudo spctl --master-enable
# Disable
sudo spctl --master-disable

# Don’t display the annoying prompt when quitting iTerm
sudo defaults write com.googlecode.iterm2 PromptOnQuit -bool false

###############################################################################
# TextEdit                                                    #
###############################################################################

# Use plain text mode for new TextEdit documents
sudo defaults write com.apple.TextEdit RichText -int 0

# Open and save files as UTF-8 in TextEdit
sudo defaults write com.apple.TextEdit PlainTextEncoding -int 4;
sudo defaults write com.apple.TextEdit PlainTextEncodingForWrite -int 4



###############################################################################
# Mac App Store                                               #
###############################################################################

# Enable the automatic update check
defaults write com.apple.SoftwareUpdate AutomaticCheckEnabled -bool true

# Check for software updates daily, not just once per week
defaults write com.apple.SoftwareUpdate ScheduleFrequency -int 1

# Download newly available updates in background
defaults write com.apple.SoftwareUpdate AutomaticDownload -int 1

# Install System data files & security updates
defaults write com.apple.SoftwareUpdate CriticalUpdateInstall -int 1

# Turn on app auto-update
defaults write com.apple.commerce AutoUpdate -bool true

# Allow the App Store to reboot machine on macOS updates
defaults write com.apple.commerce AutoUpdateRestartRequired -bool true
```

```
### CONFIGURE ###

# Define Function config
config () {
  config_admin_req
  config_bbedit
  config_desktop
  config_emacs
  config_environment
  config_istatmenus
  config_openssl
  config_sysprefs
  config_zsh
  config_guest

  which custom
}

# Define Function config_defaults
config_defaults () {
  printf "%s\n" "${1}" | \
  while IFS="$(printf '\t')" read domain key type value host; do
    ${2} defaults ${host} write ${domain} "${key}" ${type} "${value}"
  done
}

# Define Function config_plist
T="$(printf '\t')"

config_plist () {
  printf "%s\n" "$1" | \
  while IFS="$T" read command entry type value; do
    case "$value" in
      (\$*)
        $4 /usr/libexec/PlistBuddy "$2" \
          -c "$command '${3}${entry}' $type '$(eval echo \"$value\")'" 2> /dev/null ;;
      (*)
        $4 /usr/libexec/PlistBuddy "$2" \
          -c "$command '${3}${entry}' $type '$value'" 2> /dev/null ;;
    esac
  done
}

# Define Function config_launchd
config_launchd () {
  test -d "$(dirname $1)" || \
    $3 mkdir -p "$(dirname $1)"

  test -f "$1" && \
    $3 launchctl unload "$1" && \
    $3 rm -f "$1"

  config_plist "$2" "$1" "$4" "$3" && \
```

```
    $3 plutil -convert xml1 "$1" && \
    $3 launchctl load "$1"
}

# Configure Desktop Picture
config_desktop () {
  sudo rm -f "/Library/Caches/com.apple.desktop.admin.png"

  base64 -D << EOF > "/Library/Desktop Pictures/Solid Colors/Solid Black.png"
<<black.png.b64>>
EOF
}

# Configure Environment Variables
config_environment () {
  sudo tee "/etc/environment.sh" << 'EOF' > /dev/null
<<environment.sh>>
EOF
  sudo chmod a+x "/etc/environment.sh"
  rehash

  la="/Library/LaunchAgents/environment.user"
  ld="/Library/LaunchDaemons/environment"

  sudo mkdir -p "$(dirname $la)" "$(dirname $ld)"
  sudo launchctl unload "${la}.plist" "${ld}.plist" 2> /dev/null
  sudo rm -f "${la}.plist" "${ld}.plist"

  config_defaults "$_environment_defaults" "sudo"
  sudo plutil -convert xml1 "${la}.plist" "${ld}.plist"
  sudo launchctl load "${la}.plist" "${ld}.plist" 2> /dev/null
}
/etc/environment.sh

#!/bin/sh

set -e

if test -x /usr/libexec/path_helper; then
  export PATH=""
  eval `/usr/libexec/path_helper -s`
  launchctl setenv PATH $PATH
fi

osascript -e 'tell app "Dock" to quit'

# Configure OpenSSL
# Create intentionally invalid certificate for use w/DNS-based ad blocker, e.g. https://pi-hole.net

config_openssl () {
  _default="/etc/letsencrypt/live/default"
  test -d "$_default" || mkdir -p "$_default"
```

```
  cat << EOF > "${_default}/default.cnf"
<<openssl.cnf>>
EOF

  openssl req -days 1 -new -newkey rsa -x509 \
    -config "${_default}/default.cnf" \
    -out "${_default}/default.crt"

  cat << EOF > "/usr/local/etc/nginx/servers/default.conf"
<<default.conf>>
EOF
}
/etc/letsencrypt/live/default/default.cnf
[ req ]
default_bits = 4096
default_keyfile = ${_default}/default.key
default_md = sha256
distinguished_name = dn
encrypt_key = no
prompt = no
utf8 = yes
x509_extensions = v3_ca

[ dn ]
CN = *

[ v3_ca ]
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer
basicConstraints = CA:true
/usr/local/etc/nginx/servers/default.conf
server {
  server_name .$(hostname -f | cut -d. -f2-);

  listen 80;
  listen [::]:80;

  return 301 https://\$host\$request_uri;
}

server {
  listen 80 default_server;
  listen [::]:80 default_server;

  listen 443 default_server ssl http2;
  listen [::]:443 default_server ssl http2;

  ssl_certificate ${_default}/default.crt;
  ssl_certificate_key ${_default}/default.key;

  ssl_ciphers NULL;

  return 204;
```

```
}

# Configure energy saver
config_energy () {
  printf "%s\n" "${_energy}" | \
  while IFS="$(printf '\t')" read flag setting value; do
    sudo pmset $flag ${setting} ${value}
  done
}

# Configure git
# On a specific repository
git config user.email "sweetthunder33@gmail.com"
git config user.name "Liselot3"
git config user.signingkey <KeyIdVALUE>
git config commit.gpgsign true

# or globally
git config --global commit.gpgsign true
git config --global user.signingkey <KeyIdVALUE>

# Configure Z-Shell
config_zsh () {
  grep -q "$(which zsh)" /etc/shells ||
  print "$(which zsh)\n" | \
  sudo tee -a /etc/shells > /dev/null

  case "$SHELL" in
    ($(which zsh)) ;;
    (*)
      chsh -s "$(which zsh)"
      sudo chsh -s $(which zsh) ;;
  esac

  sudo tee -a /etc/zshenv << 'EOF' > /dev/null
<<etc-zshenv>>
EOF
  sudo chmod +x "/etc/zshenv"
  . "/etc/zshenv"

  sudo tee /etc/zshrc << 'EOF' > /dev/null
<<etc-zshrc>>
EOF
  sudo chmod +x "/etc/zshrc"
  . "/etc/zshrc"
}

#  Set Permissions on Install Destinations
init_perms () {
  printf "%s\n" "${_dest}" | \
  while IFS="$(printf '\t')" read d; do
    test -d "${d}" || sudo mkdir -p "${d}"
    sudo chgrp -R admin "${d}"
```

```
    sudo chmod -R g+w "${d}"
  done
}

# Mark Applications Requiring Administrator Account
config_admin_req () {
  printf "%s\n" "${_admin_req}" | \
  while IFS="$(printf '\t')" read app; do
    sudo tag -a "Red, admin" "/Applications/${app}"
  done
}

# Reinstate sudo Password
config_rm_sudoers () {
  sudo -- sh -c \
    "rm -f /etc/sudoers.d/wheel; dscl /Local/Default -delete /Groups/wheel GroupMembership $(whoami)"

  /usr/bin/read -n 1 -p "Press any key to continue.
" -s
  if run "Log Out Then Log Back In?" "Cancel" "Log Out"; then
    osascript -e 'tell app "loginwindow" to «event aevtrlgo»'
  fi
}



# Customize Home
custom_githome () {
  git -C "${HOME}" init

  test -f "${CACHES}/dbx/.zshenv" && \
    mkdir -p "${ZDOTDIR:-$HOME}" && \
    cp "${CACHES}/dbx/.zshenv" "${ZDOTDIR:-$HOME}" && \
    . "${ZDOTDIR:-$HOME}/.zshenv"

  a=$(ask "Existing Git Home Repository Path or URL" "Add Remote" "")
  if test -n "${a}"; then
    git -C "${HOME}" remote add origin "${a}"
    git -C "${HOME}" fetch origin master
  fi

  if run "Encrypt and commit changes to Git and push to GitHub, automatically?" "No" "Add AutoKeep"; then
    curl --location --silent \
      "https://github.com/ptb/autokeep/raw/master/autokeep.command" | \
      . /dev/stdin 0

    autokeep_remote
    autokeep_push
    autokeep_gitignore
    autokeep_post_commit
    autokeep_launchagent
    autokeep_crypt

    git reset --hard
```

```
    git checkout -f -b master FETCH_HEAD
  fi

  chmod -R go= "${HOME}" > /dev/null 2>&1
}

# Customize Dropbox
test -d "/Applications/Dropbox.app" && \
    open "/Applications/Dropbox.app"

# Customize Default UTIs
custom_duti () {
  if test -x "/usr/local/bin/duti"; then
    test -f "${HOME}/Library/Preferences/org.duti.plist" && \
      rm "${HOME}/Library/Preferences/org.duti.plist"

    printf "%s\n" "${_duti}" | \
    while IFS="$(printf '\t')" read id uti role; do
      defaults write org.duti DUTISettings -array-add \
        "{
          DUTIBundleIdentifier = '$a';
          DUTIUniformTypeIdentifier = '$b';
          DUTIRole = '$c';
        }"
    done

    duti "${HOME}/Library/Preferences/org.duti.plist" 2> /dev/null
  fi
}

##Finder##
custom_finder () {
  config_defaults "${_finder}"
  defaults write "com.apple.finder" "NSToolbar Configuration Browser" \
    '{
      "TB Display Mode" = 2;
      "TB Item Identifiers" = (
        "com.apple.finder.BACK",
        "com.apple.finder.PATH",
        "com.apple.finder.SWCH",
        "com.apple.finder.ARNG",
        "NSToolbarFlexibleSpaceItem",
        "com.apple.finder.SRCH",
        "com.apple.finder.ACTN"
      );
    }'
}

# Customize SSH
custom_ssh () {
  if ! test -d "${HOME}/.ssh"; then
    mkdir -m go= "${HOME}/.ssh"
    e="$(ask 'New SSH Key: Email Address?' 'OK' '')"
```

```
    ssh-keygen -t ed25519 -a 100 -C "$e"
    cat << EOF > "${HOME}/.ssh/config"
Host *
  AddKeysToAgent yes
  IdentityFile ~/.ssh/id_ed25519
EOF
    pbcopy < "${HOME}/.ssh/id_ed25519.pub"
    open "https://github.com/settings/keys"
  fi
}

# Customize Clock
defaults -currentHost write com.apple.systemuiserver dontAutoLoad \
    -array-add "/System/Library/CoreServices/Menu Extras/Clock.menu"
defaults write com.apple.menuextra.clock DateFormat \
    -string "EEE MMM d  h:mm:ss a"

#SSH
pbcopy < ~/.ssh/id_ed25519.pub
open https://github.com/settings/keys
terminal-notifier -title SSH key -message  "Copied SSH key to clipboard, add it to GitHub"


### Fonts ###

brew tap homebrew/fonts

### programming fonts
brew cask install font-fira-mono-for-powerline
brew cask install font-fira-code

### SourceCodePro + Powerline + Awesome Regular (for powerlevel 9k terminal icons)
cd ~/Library/Fonts && { curl -O 'https://github.com/Falkor/dotfiles/blob/master/fonts/SourceCodePro+Powerline+Awesome+Regular.ttf?
raw=true' ; cd -; }

# Close any open System Preferences panes, to prevent them from overriding
# settings we're about to change
osascript -e 'tell application "System Preferences" to quit'



### Quicklook plugins ###

brew cask install qlcolorcode;
# syntax highlighting in preview
brew cask install qlstephen ;
# preview plaintext files without extension
brew cask install qlmarkdown;
# preview markdown files;
brew cask install quicklook-json;
# preview json files
brew cask install epubquicklook ;
# preview epubs, make nice icons
```

```
brew cask install quicklook-cs;

### End ###

# Restore Sane Shell
stty sane

# Clear DNS Cache
sudo dscacheutil -flushcache && \
sudo killall -HUP mDNSResponder

# Clear Font Cache for Current User
atsutil databases -removeUser && \
atsutil server -shutdown && \
atsutil server -ping

# Recursively Delete .DS_Store Files
find . -type f -name '*.DS_Store' -ls -delete

# Build Locate Database
sudo launchctl load -w /System/Library/LaunchDaemons/com.apple.locate.plist

#Install macOS Updates
  sudo softwareupdate --install --all

# Update installed apps & clear caches
brew analytics off;
brew outdated;
brew update;
brew upgrade;
brew cask upgrade;
brew cleanup;
brew tap "homebrew/bundle";
rm -rf ~/Library/Caches/Homebrew

sudo purge

echo "⚠ Some changes aren't applied until you log out and back in."
echo "  Setup complete!"




### Extras ###

# Root User
# Enable
#dsenableroot

# Disable (Default)
#dsenableroot -d
```

Safe Mode Boot##
Status
#nvram boot-args

Enable
#sudo nvram boot-args="-x"

Disable (Default)
#sudo nvram boot-args=""

Figure out which apps you've installed via the App Store
#find /Applications \
-path '*Contents/_MASReceipt/receipt' \
-maxdepth 4 -print |\
sed 's#.app/Contents/_MASReceipt/receipt#.app#g; s#/Applications/##'

https://www.dropbox.com/s/29ou62smphr9i4b/macbook-setup_v2.sh?dl=0