```
#!/usr/bin/env bash

: ========================================
:   Introduction
: ========================================

# This script allows you to install the latest version of the
# "firebase" command by running:
#
:   curl -sL firebase.tools | bash
#
# If you do not want to use this script, you can manually
# download the latest "firebase" binary.
#
:   curl -Lo ./firebase_bin https://firebase.tools/bin/linux/latest
#
# Alternatively, you can download a specific version.
#
:   curl -Lo ./firebase_bin https://firebase.tools/bin/linux/v7.2.2
#
# Note: On Mac, replace "linux" with "macos" in the URL.
#
# For full details about installation options for the Firebase CLI
# please see our documentation.
#   https://firebase.google.com/docs/cli/
#
# Please report bugs / issues with this script on Github.
#   https://github.com/firebase/firebase-tools
#

: ========================================
:   Advanced Usage
: ========================================

# The behavior of this script can be modified at runtime by passing environmental
# variables to the `bash` process.
#
# For example, passing an argument called arg1 set to true and one called arg2 set
# to false would look like this.
#
: curl -sL firebase.tools | arg1=true arg2=false bash
#
# These arguments are optional, but be aware that explicitly setting them will help
# ensure consistent behavior if / when defaults are changed.
#

: ----------------------------------------
:  Upgrading - default: false
: ----------------------------------------

# By default, this script will not replace an existing "firebase" install.
# If you'd like to upgrade an existing install, set the "upgrade" variable to true.
#
```

```
: curl -sL firebase.tools | upgrade=true bash
#
# This operation could (potentially) break an existing install, so use it with caution.
#

: ---------------------------------------
:  Uninstalling - default false
: ---------------------------------------

# You can remove the binary by passing the "uninstall" flag.
#
: curl -sL firebase.tools | uninstall=true bash
#
# This will remove the binary file, the extracted cache, and any
# emulators which have been downloaded during use.
#

: ---------------------------------------
:  Analytics - default true
: ---------------------------------------

# This script reports anonymous success / failure analytics.
# You can disable this reporting by setting the "analytics" variable to false.
#
: curl -sL firebase.tools | analytics=false bash
#
# By default we report all data anonymously and do not collect any information
# except platform type (Darwin, Win, etc) in the case of an unsupported platform
# error.
#

: =======================================
:   Source Code
: =======================================

# This script contains a large amount of comments so you can understand
# how it interacts with your system. If you're not interested in the
# technical details, you can just run the command above.

# We begin by generating a unique ID for tracking the anonymous session.
CID=$(cat /dev/urandom | LC_CTYPE=c tr -dc 'a-zA-Z0-9' | fold -w 32 | head -n 1)
# Credit: https://gist.github.com/earthgecko/3089509

# We can use this CID in all calls to the Google Analytics endpoint via
# this reusable function.
send_analytics_event()
{
    # We only make the API call in the case of the "analytics" variable
    # not being set to "false". If it is set to false, this function
    # performs no operation.

    if [ ! "$analytics" = "false" ]; then
        curl -s https://www.google-analytics.com/collect \
            -d "tid=UA-154737150-1" \
            -d "t=event" \
```

```
            -d "ec=firebase.tools" \
            -d "ea=$1" \
            -d "v=1" \
            -d "cid=$CID" \
            -o /dev/null
    fi
}

# We send one event to count the number of times this script is ran. At the
# end we also report success / failure, but it's possible that the script
# will crash before we get to that point, so we manually count invocations here.
send_analytics_event start

# If the user asked for us to uninstall firebase, then do so.
if [ "$uninstall" = "true" ]; then
    if [[ ! "$(firebase --tool:setup-check)" == *bins* ]];
    then
        echo "Your \"firebase\" install was done via npm, not firebase.tools."
        echo "Run \"npm uninstall -g firebase-tools\" to uninstall."
        send_analytics_event uninstall_npm
        exit 0
    fi

    echo "-- Removing binary file..."
    sudo rm $(which firebase)
    echo "-- Removing emulator runtimes..."
    rm -rf ~/.cache/firebase/emulators
    echo "-- Removing npm cache..."
    rm -rf ~/.cache/firebase/tools
    echo "-- Removing firebase runtime..."
    rm -rf ~/.cache/firebase/runtime

    echo "-- firebase-tools has been uninstalled"
    echo "-- All Done!"

    send_analytics_event uninstall
    exit 0
fi

# For info about why we place the binary at this location, see
# https://unix.stackexchange.com/a/8658
INSTALL_DIR="/usr/local/bin"

# We need to ensure that the INSTALL_DIR exists.
# On some platforms like the Windows Subsystem for Linux it may not.
# We created it using a non-destructive mkdir command.
sudo mkdir -p "$INSTALL_DIR"

# We need to ensure that we don't mess up an existing "firebase"
# install, so before doing anything we check to see if this system
# has "firebase" installed and if so, we exit out.
echo "-- Checking for existing firebase-tools on PATH..."
HAS_FIREBASE_TOOLS=$(which firebase)

if [ ! -z "$HAS_FIREBASE_TOOLS" ]; then
```

```
    INSTALLED_FIREBASE_VERSION=$(firebase --version)

    # In the case of a corrupt firebase-tools install, we wont be able to
    # retrieve a version number, so to keep the logs correct, we refer to
    # your existing install as either the CLI version or as a "corrupt install"
    if [ ! -z "$INSTALLED_FIREBASE_VERSION" ]; then
        FIREBASE_TOOLS_NICKNAME="firebase-tools@$(firebase --version)"
    else
        FIREBASE_TOOLS_NICKNAME="a corrupted firebase-tools binary"
    fi

    # We are only capable of upgrading installs of the standalone binary version of firebase-tools
    # To detect if the version of firebase-tools installed is the binary version, we pass a hidden
    # flag which returns a JSON response from the binary or an error on a regular install.
    if [[ "$(firebase --tool:setup-check)" == error* ]]; then
        # If the install isn't a binary (i.e. it was installed via npm) then we can't help, so we
        echo "Your machine already has $FIREBASE_TOOLS_NICKNAME installed. Nothing to do."
        echo "-- All done!"

        send_analytics_event already_installed
        exit 0
    else
        # If the user didn't pass upgrade=true, then we print the command to do an upgrade and exi
        if [ ! "$upgrade" = "true" ]; then
            echo "Your machine has $FIREBASE_TOOLS_NICKNAME installed."
            echo "If you would like to upgrade your install run: curl -sL firebase.tools | upgrade=

            send_analytics_event already_installed
            exit 0
        else
            # If the user did pass upgrade=true, then we allow the script to continue and overwrit
            echo "-- Your machine has $FIREBASE_TOOLS_NICKNAME, attempting upgrade..."

            send_analytics_event upgrade
        fi

    fi
fi

echo "-- Checking your machine type..."

# Now we need to detect the platform we're running on (Linux / Mac / Other)
# so we can fetch the correct binary and place it in the correct location
# on the machine.

# We use "tr" to translate the uppercase "uname" output into lowercase
UNAME=$(uname -s | tr '[:upper:]' '[:lower:]')

# Then we map the output to the names used on the Github releases page
case "$UNAME" in
    linux*)     MACHINE=linux;;
    darwin*)    MACHINE=macos;;
esac

# If we never define the $MACHINE variable (because our platform is neither Mac
```

```
# or Linux), then we can't finish our job, so just log out a helpful message
# and close.
if [ -z "$MACHINE" ]
then
    echo "Your operating system is not supported, if you think it should be please file a bug."
    echo "https://github.com/firebase/firebase-tools/"
    echo "-- All done!"

    send_analytics_event "missing_platform_$UNAME"
    exit 0
fi

# We have enough information to generate the binary's download URL.
DOWNLOAD_URL="https://firebase.tools/bin/$MACHINE/latest"
echo "-- Downloading binary from $DOWNLOAD_URL"

# We use "curl" to download the binary with a flag set to follow redirects
# (Github download URLs redirect to CDNs) and a flag to show a progress bar.
sudo curl -o "$INSTALL_DIR/firebase" -L --progress-bar $DOWNLOAD_URL

# Once the download is complete, we mark the binary file as readable
# and executable (+rx).
echo "-- Setting permissions on binary..."
sudo chmod +rx "$INSTALL_DIR/firebase"

# If all went well, the "firebase" binary should be located on our PATH so
# we'll run it once, asking it to print out the version. This is helpful as
# standalone firebase binaries do a small amount of setup on the initial run
# so this not only allows us to make sure we got the right version, but it
# also does the setup so the first time the developer runs the binary, it'll
# be faster.
VERSION=$("$INSTALL_DIR/firebase" --version)

# If no version is detected then clearly the binary failed to install for
# some reason, so we'll log out an error message and report the failure
# to headquarters via an analytics event.
if [ -z "$VERSION" ]
then
    echo "Something went wrong, firebase has not been installed."
    echo "Please file a bug with your system information on Github."
    echo "https://github.com/firebase/firebase-tools/"
    echo "-- All done!"

    send_analytics_event failure
    exit 1
fi

# In order for the user to be able to actually run the "firebase" command
# without specifying the absolute location, the INSTALL_DIR path must
# be present inside of the PATH environment variable.

echo "-- Checking your PATH variable..."
if [[ ! ":$PATH:" == *":$INSTALL_DIR:"* ]]; then
    echo ""
    echo "It looks like $INSTALL_DIR isn't on your PATH."
```

```
    echo "Please add the following line to either your ~/.profile or ~/.bash_profile, then restart
    echo ""
    echo "PATH=\$PATH:$INSTALL_DIR"
    echo ""
    echo "For more information about modifying PATHs, see https://unix.stackexchange.com/a/26059"
    echo ""
    send_analytics_event missing_path
fi

# Since we've gotten this far we know everything succeeded. We'll just
# let the developer know everything is ready and take our leave.
echo "-- firebase-tools@$VERSION is now installed"
echo "-- All Done!"

send_analytics_event success
exit 0

# ----------------------------------------
#   Notes
# ----------------------------------------
#
# This script contains hidden JavaScript which is used to improve
# readability in the browser (via syntax highlighting, etc), right-click
# and "View source" of this page to see the entire bash script!
#
# You'll also notice that we use the ":" character in the Introduction
# which allows our copy/paste commands to be syntax highlighted, but not
# ran. In bash : is equal to `true` and true can take infinite arguments
# while still returning true. This turns these commands into no-ops so
# when ran as a script, they're totally ignored.
#
```