```
--property jarvisname : (do shell script "defaults read /Library/Preferences/Jarvis
        TwitterName")
--property jarvispwd : (do shell script "defaults read /Library/Preferences/Jarvis
        TwitterPwd")
--property commandLocal : (do shell script "defaults read /Library/Preferences/
        Jarvis commandLocal")
--property myMobile : (do shell script "defaults read /Library/Preferences/Jarvis
        myMobile")
--property myEmail : (do shell script "defaults read /Library/Preferences/Jarvis
        myEmail")
--property myGoogleLatID : (do shell script "defaults read /Library/Preferences/
        Jarvis googleLatID")

--SEND AN IM TO A BUDDY ON ICHAT
on sendIM(themessage, theBuddy)
	tell application "Messages" to send themessage to theBuddy
end sendIM

--SEND A TEXT(SMS) TO PHONE NUMBER
on sendText(themessage, theNum)
	--if phone number isn't supplied, your mobile number is assumed
	if theNum is "" then set theNum to myMobile
	--+1 is required, this checks for it and adds if needed (assumed for a US
			phone number)
	if (count of characters of theNum) is 10 then set theNum to "+1" &
			theNum
	if theNum does not start with "+" then set theNum to "+" & theNum
	if theNum does not start with "AIM:" then set theNum to "AIM:" &
			theNum
	tell application "Messages" to send themessage to buddy theNum
end sendText

--SPEECH -the variable input "messageArray" is actually an array
on speak(messageArray)
	--determines whether to use the "private mode" text to speak, or "public
			mode".
	set override to first item of messageArray
	if (count of messageArray) > 2 then
		try
			if (do shell script "defaults read /Library/Preferences/Jarvis
					privateMode") = "1" then
				set themessage to item 2 of messageArray
			else
				set themessage to item 3 of messageArray
			end if
		on error
```

```
			set themessage to item 2 of messageArray
			do shell script "defaults write /Library/Preferences/Jarvis
					privateMode -bool FALSE"
		end try
	else
		set themessage to second item of messageArray
	end if
	--enable speech
	if themessage is "true" or themessage is true or themessage is "on" or
			themessage is "enable" then
		do shell script "defaults write /Library/Preferences/Jarvis speak TRUE"
		if (do shell script "defaults read /Library/Preferences/Jarvis speak") as
				boolean = true then
			set volume output volume 75
			beep
			delay 0.1
			beep
			return "Speech is enabled"
		else
			return "Error enabling speech"
		end if
		--disable speech
	else if themessage is "false" or themessage is false or themessage is "off"
			or themessage is "disable" then
		do shell script "defaults write /Library/Preferences/Jarvis speak FALSE"
		if (do shell script "defaults read /Library/Preferences/Jarvis speak") as
				boolean = false then
			--say "Speech is disabled"
			--set volume output volume 0
			beep
			delay 0.1
			beep
			delay 0.1
			beep
			return "Speech is disabled"
		else
			return "Error disabling speech"
		end if
		--disable speech quietly
	else if themessage is "false silent" then
		do shell script "defaults write /Library/Preferences/Jarvis speak FALSE"
		if (do shell script "defaults read /Library/Preferences/Jarvis speak") as
				boolean = false then
			return "Speech is disabled"
		else
			return "Error disabling speech"
```

```
        end if
        --set to private speech mode
    else if themessage is "private" then
        do shell script "defaults write /Library/Preferences/Jarvis privateMode -
                    bool TRUE"
        return "Private speech enabled"
        --set to public speech mode
    else if themessage is "public" then
        do shell script "defaults write /Library/Preferences/Jarvis privateMode -
                    bool FALSE"
        return "Public speech enabled"
        --return a status of speech
    else if themessage is "status" then
        set temp1 to ((do shell script "defaults read /Library/Preferences/
                    Jarvis speak"))
        if temp1 is "TRUE" then
            set temp1 to "Speech is enabled"
        else
            set temp1 to "Speech is disabled"
        end if
        if (do shell script "defaults read /Library/Preferences/Jarvis
                    privateMode") is "0" then
            set temp2 to "Public mode"
        else
            set temp2 to "Private mode"
        end if
        return (temp1 & return & temp2)
        --repeat the last thing that was said
    else if themessage is "repeat" then
        try
            set lastTalk to (do shell script "cat /Library/Preferences/
                        com.Jarvis.lastTalk.txt")
            return lastTalk
        on error themessage
            return themessage
        end try
    else
        --remove URLs from being spoken aloud
        if themessage contains "http" then
            set myChar to the offset of "http" in themessage
            repeat with x from myChar to count of (every character of
                        themessage)
                if character x of themessage = space or character x of
                            themessage = return then
                    set themessage to (characters 1 thru (myChar - 1) of
                                themessage) & (characters x thru -1 of
                                themessage) as string
                        exit repeat
                    end if
                    if x = (count of (every character of themessage)) then
                        set themessage to characters 1 thru (myChar - 1) of
                                themessage as string
                    end if
                end repeat
            end if
            set themessage to replaceText("\"", "", themessage)
            set speakable to true
            try
                set speakable to (do shell script "defaults read /Library/
                        Preferences/Jarvis speak") as boolean
            end try
            if speakable is true or override is true then
                try
                    try
                        --speaker warm up audio, to make the wireless speakers
                                turn on before text is spoken. Only used if
                                speakers need "warm up" before usage.
                        if ((current date) - (date (do shell script "defaults
                                read /Library/Preferences/Jarvis
                                lastTalkTime"))) > 240 then
                            say "A new notification. " --without waiting until
                                    completion
                            delay 0.5
                        end if
                        --speak text
                        say (themessage) --without waiting until completion
                        --save last spoken message for "repeat" command
                        if themessage is not "" then
                            try
                                do shell script "echo " & quoted form of
                                        themessage & " > /Library/
                                        Preferences/
                                        com.Jarvis.lastTalk.txt"
                            on error errorMessage
                                logger(((current date) as string) & ": Error - "
                                        & errorMessage)
                            end try
                        end if
                    end try
                    do shell script "defaults write /Library/Preferences/Jarvis
                            lastTalkTime " & quoted form of ((current date)
                            as string)
                        return true
                on error
                        return false
                end try
        else
                return false
        end if
    end if
end speak

--CONVERTS TEXT TO A ENCODING BETTER SUITED FOR TWITTER
on text_to_tweet(theTweet)
    set theTweet to theTweet as «class utf8»
    if (count of every character of theTweet) > 140 then
        set theTweet to characters 1 thru 140 of theTweet as string
    end if
    --set theTweet to encode_text(theTweet, false, false)
    return theTweet
end text_to_tweet

--UPDATE JARVIS' TWITTER STATUS
on update_status(tweet_message)
    set tweet_message to text_to_tweet(tweet_message)
    try
        set ret to do shell script "curl --basic --user " & jarvisname & ":" &
                    jarvispwd & "  --data status=" & quoted form of
                    tweet_message & " http://twitter.com/statuses/update.xml"
    on error themessage
        logger(themessage)
        return themessage
    end try
end update_status

--SEND @REPLY TO A USER ON TWITTER
on reply(sn, tweet_message)
    set tweet_message to "@" & sn & space & tweet_message
    set tweet_message to text_to_tweet(tweet_message)
    do shell script "curl --basic --user " & jarvisname & ":" & jarvispwd & " --
            data status=" & quoted form of tweet_message & " http://
            twitter.com/statuses/update.xml"
end reply

--SEND DIRECT MESSAGE ON TWITTER TO USER
on sendDM(tweet_message, theUser)
    logger(tweet_message)
    set tweet_message to text_to_tweet(tweet_message)
```

```
        logger(tweet_message)
        do shell script "curl --basic --user " & jarvisname & ":" & jarvispwd & "  --
                 data text=" & quoted form of tweet_message & " --data user=" &
                 theUser & " http://twitter.com/direct_messages/new.xml"
end sendDM

--SEND EMAIL. USES APPLE MAIL, SO APPLE MAIL NEEDS TO BE SETUP WITH AN
        ACCOUNT.
on send_email(theAddress, theSubject, thebody)
        tell application "Mail"
                set mymessage to make new outgoing message with properties
                        {visible:true, subject:theSubject, content:thebody}
                tell mymessage to make new bcc recipient at end of to recipients with
                        properties {address:theAddress}
                send mymessage
        end tell
end send_email

--XML READER. RETURNS ARRAY OF TAGS REQUESTED.
on xml_processor(rawdata, theItem, tags)
        set rawtext to rawdata
        set myoutput to {}
        set myItems to xml_parser(rawtext, theItem)
        repeat with x from 1 to count of myItems
                set temp to {}
                repeat with z from 1 to count of tags
                        set temp to temp & xml_parser(item x of myItems, item z of tags)
                end repeat
                set myoutput to myoutput & {temp}
        end repeat
        return myoutput
end xml_processor
on xml_parser(rawtext, theID)
        set myoutput to {}
        repeat
                try
                        if rawtext contains ("<" & theID & ">") then
                                set myoutput to myoutput & {text ((offset of ("<" & theID &
                                                ">") in rawtext) + 2 + (count of characters of
                                                theID)) thru ((offset of ("</" & theID & ">") in
                                                rawtext) - 1) of rawtext}
                                set rawtext to text ((offset of ("</" & first word of theID &
                                                ">") in rawtext) + 2 + (count of characters in
                                                theID)) thru -1 of rawtext
                        else
                                exit repeat
```

```
				end if
			on error
				exit repeat
			end try
		end repeat
		return myoutput
	end xml_parser

	--SHORT URL GENERATOR. USES HTTP://TR.IM API
	on trim(myURL)
		set myURL to encode_text(myURL, true, true)
		return (do shell script "curl http://api.tr.im/api/trim_simple?url=" &
				myURL)
	end trim

	--LOGS A MESSAGES TO CONSOLE. TAGS IS WITH KEYWORD "JARVIS"
	on logger(themessage)
		--do shell script "logger -t Jarvis " & quoted form of themessage
		do shell script "echo " & quoted form of themessage & " >> ~/Library/
				Logs/Jarvis.log"
	end logger

	--GET TIME
	on getTimeInHoursAndMinutes()
		-- Get the "hour"
		set timeStr to time string of (current date)
		set Pos to offset of ":" in timeStr
		set thehour to characters 1 thru (Pos - 1) of timeStr as string
		set timeStr to characters (Pos + 1) through end of timeStr as string

		-- Get the "minute"
		set Pos to offset of ":" in timeStr
		set theMin to characters 1 thru (Pos - 1) of timeStr as string
		set timeStr to characters (Pos + 1) through end of timeStr as string

		--Get "AM or PM"
		set Pos to offset of " " in timeStr
		set theSfx to characters (Pos + 1) through end of timeStr as string

		return (thehour & ":" & theMin & " " & theSfx) as string
	end getTimeInHoursAndMinutes

	--RETURN SUFFIX OF THE DAY
	on daySuffix(num)
		if num is 1 or num is 21 or num is 31 then
			set ending to "st"
	else if num is 2 or num is 22 then
		set ending to "nd"
	else if num is 3 or num is 23 then
		set ending to "rd"
	else
		set ending to "th"
	end if
	return ending --(("Today is " & the weekday of currentdate as string) & ",
			the " & temp as string) & ending & ". "
end daySuffix

--RETURN DAYTIME
on daytime()
	set thehour to do shell script "date +%H"
	set daytime to "morning"
	if thehour ≥ 12 then
		set daytime to "afternoon"
		if thehour ≥ 17 then
			set daytime to "evening"
		end if
	end if
	return daytime
end daytime

--COUNTS THE NUMBER OF SECONDS INTO WEEKS, DAYS, HOURS, MINUTES
on interpret_Time(myTime)
	set theWeeks to 0
	set theDays to 0
	set theHours to 0
	set theMinutes to 0
	set theSeconds to 0
	set myResult to ""

	if myTime < 60 then
		if myTime > 1 then set myResult to myResult & myTime & " seconds"
		if myTime = 1 then set myResult to myResult & myTime & " second"
	else
		if myTime ≥ 604800 then
			set theWeeks to myTime div 604800
			set myTime to myTime mod 604800
			if theWeeks > 1 then set myResult to myResult & theWeeks & "
					weeks"
			if theWeeks = 1 then set myResult to myResult & theWeeks & "
					week"
		end if
		if myTime ≥ 86400 then
```

```
            set theDays to myTime div 86400
            set myTime to myTime mod 86400
            if theDays > 1 then set myResult to myResult & theDays & " days"
            if theDays = 1 then set myResult to myResult & theDays & " day"
        end if
        if myTime ≥ 3600 then
            set theHours to myTime div 3600
            set myTime to myTime mod 3600
            if myResult is not equal to "" then set myResult to myResult & ",
                        "
            if theHours > 1 then set myResult to myResult & theHours & "
                        hours"
            if theHours = 1 then set myResult to myResult & theHours & "
                        hour"
        end if
        if myTime ≥ 60 then
            set theMinutes to myTime div 60
            set myTime to myTime mod 60
            if myResult is not equal to "" then set myResult to myResult & ",
                        and "
            if theMinutes > 1 then set myResult to myResult & theMinutes & "
                        minutes"
            if theMinutes = 1 then set myResult to myResult & theMinutes & "
                        minute"
        end if
    end if
    (*set theSeconds to myTime
    if theSeconds > 0 then
        if myResult is not equal to "" then set myResult to myResult & ", and "
        if theSeconds = 1 then
            set myResult to myResult & theSeconds & " seconds"
        else
            set myResult to myResult & theSeconds & " seconds."
        end if
    end if*)
    return myResult
end interpret_Time

--GETS GPS LOCATION USING GOOGLE LATITUDE. RETURNS RECORD WITH
     STREET, CITY, STATE, ZIPCODE, COUNTRY, LONGITUDE, LATITUDE
on getGPSLocation()
    try
        set rawdata to do shell script "curl 'http://www.google.com/latitude/
                apps/badge/api?user=" & myGoogleLatID & "&type=atom'"
        set mylocation to replaceText(space, ",", (first item of
                xml_parser(rawdata, "georss:point")))
        set myLongitude to first word of mylocation
        set myLatitude to last word of mylocation
        set rawdata to do shell script "curl 'http://maps.google.com/maps/
                geo?output=xml&key=abcdefg&q=" & mylocation & "'"
        set myZipcode to first item of xml_parser(rawdata,
                "PostalCodeNumber")
        set myCountry to first item of xml_parser(rawdata, "CountryName")
        set myState to first item of xml_parser(rawdata,
                "AdministrativeAreaName")
        set myTown to first item of xml_parser(rawdata, "LocalityName")
        set myStreet to first item of xml_parser(rawdata,
                "ThoroughfareName")
        set mylocation to {street:myStreet, city:myTown, state:myState,
                zipcode:myZipcode, country:myCountry,
                longitude:myLongitude, latitude:myLatitude}
    on error themessage
        set mylocation to {street:"n/a", city:"n/a", state:"n/a", zipcode:"n/a",
                country:"n/a", longitude:"n/a", latitude:"n/a"}
        return themessage
    end try
    return mylocation
end getGPSLocation
--GETS GPS LOCATION USING GOOGLE LATITUDE. RETURNS JUST ZIPCODE
on getZipcode()
    try
        set rawdata to do shell script "curl 'http://www.google.com/latitude/
                apps/badge/api?user=" & myGoogleLatID & "&type=atom'"
        set mylocation to replaceText(space, ",", (first item of
                xml_parser(rawdata, "georss:point")))
        set updated to first item of xml_parser(rawdata, "updated")
        set rawdata to do shell script "curl 'http://maps.google.com/maps/
                geo?output=xml&key=abcdefg&q=" & mylocation & "'"
        set myZipcode to first item of xml_parser(rawdata,
                "PostalCodeNumber")
    on error
        set myZipcode to "n/a"
    end try
    return myZipcode
end getZipcode

--SUBROUTINE TO SPLIT A STRING
to split(aString, sep)
    local aList, delims
    tell AppleScript
        set delims to text item delimiters
        set text item delimiters to sep
```

```applescript
        set aList to text items of aString
        set text item delimiters to delims
    end tell
    return aList
end split

--UPPERCASES TEXT
on upperCase(s)
    set uc to "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    set lc to "abcdefghijklmnopqrstuvwxyz"
    repeat with i from 1 to 26
        set AppleScript's text item delimiters to character i of lc
        set s to text items of s
        set AppleScript's text item delimiters to character i of uc
        set s to s as text
    end repeat
    set AppleScript's text item delimiters to ""
    return s
end upperCase

--LOWERCASES TEXT
on lowerCase(s)
    set uc to "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
    set lc to "abcdefghijklmnopqrstuvwxyz"
    repeat with i from 1 to 26
        set AppleScript's text item delimiters to character i of uc
        set s to text items of s
        set AppleScript's text item delimiters to character i of lc
        set s to s as text
    end repeat
    set AppleScript's text item delimiters to ""
    return s
end lowerCase

--PERCENT ENCODING
on encode_char(this_char)
    set the ASCII_num to (the ASCII number this_char)
    set the hex_list to {"0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "A", "B",
        "C", "D", "E", "F"}
    set x to item ((ASCII_num div 16) + 1) of the hex_list
    set y to item ((ASCII_num mod 16) + 1) of the hex_list
    return ("%" & x & y) as string
end encode_char
on encode_text(this_text, encode_URL_A, encode_URL_B)
    set the standard_characters to "abcdefghijklmnopqrstuvwxyz0123456789"
    set the URL_A_chars to "$+!'/?;&@=#%><{}[]\"~`^\\|*"
```

```
	set the URL_B_chars to ".-_:"
	set the acceptable_characters to the standard_characters
	if encode_URL_A is false then set the acceptable_characters to the
			acceptable_characters & the URL_A_chars
	if encode_URL_B is false then set the acceptable_characters to the
			acceptable_characters & the URL_B_chars
	set the encoded_text to ""
	repeat with this_char in this_text
		if this_char is in the acceptable_characters then
			set the encoded_text to (the encoded_text & this_char)
		else
			set the encoded_text to (the encoded_text &
					encode_char(this_char)) as string
		end if
	end repeat
	return the encoded_text
end encode_text

--REPLACES TEXT IN A STRING
on replaceText(find, replace, someText)
	set prevTIDs to text item delimiters of AppleScript
	set text item delimiters of AppleScript to find
	set someText to text items of someText
	set text item delimiters of AppleScript to replace
	set someText to "" & someText
	set text item delimiters of AppleScript to prevTIDs
	return someText
end replaceText
```