Complete list of AppleScript key codes

Here it is. The complete list of AppleScript key codes. Use like so:

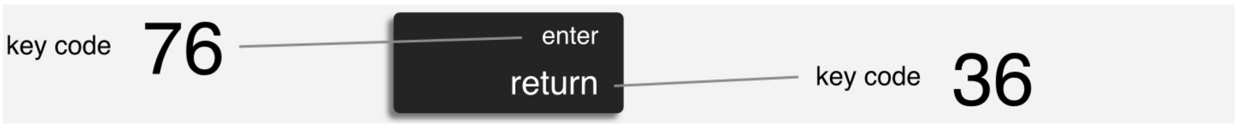
```
tell application "System Events"
        key code 49 --Key code 49 triggers the space bar
end tell
```

Just to be clear, everything after `--` in the example above is a comment. I believe that appropriate commenting is essential to effective programming, especially when there is any kind of collaboration involved.

If there are any other key codes that I missed, please let me know. They will be addd to the list.

Some keys cannot be reliably scripted using key codes. A good example of this is the play/pause function key. I have included instructions to perform these operations without the need for key codes. Right now the only function keys that don't work and don't have a good work-around are the increase/decrease keyboard brightness keys.

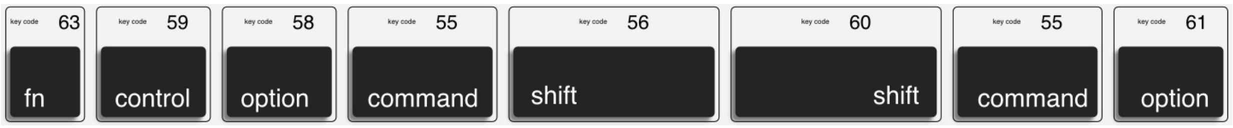
Return and enter



The enter key on most Macs is actually the return key (key code 36). The key code for enter is 76. The enter key is what you might see on a full size keyboard on the numpad side. On the more common, non-full-size Mac keyboards, enter can still be accomplished by hitting fn + enter . This is why the enter key says return *and* enter on it.

Most applications don't really care about the difference between return and enter. In pretty much every text editor both enter and return will result in the same new line. One example of where these two keys do different things is with iTunes. Interestingly, in iTunes, enter (76) renames the currently selected song while return (36) plays the song from the beginning. This is still true as of iTunes 12 running on OS X 10.10 Yosemite. Off topic side note: option + enter adds the selected track to the up next list.

Modifier keys



Most of the modifiers have two different key codes. One for the left and one for the right. So instead of just triggering, say, `option`, you can trigger (right) `option` specifically. This applies to `option`, `shift`, and `control`. It appears that `command` has only one key code which applies to either key. I think it is worth noting that the keyboard on the MacBook Air I'm using right now has only one `control` key on its physical keyboard.

Right control	62
Right option	61
Right shift	60

While this is all great and dandy, I imagine you will probably want to use these keys to modify other keys. That *is* what they're for, after all. To do this, we actually don't need to use key codes. For example, you could do `command` + `A` like this:

```
tell application "System Events" to keystroke a using co
```

Use multiple modifier keys with `{}`. In this example, we do a "Paste and Match Style" with `option` + `shift` + `command` + `V` like this:

```
tell application "System Events" to key code 9 using {op
```

Key code 9, in the example above, is the `V` key. Please note that we could just as easily use keystroke, like this:

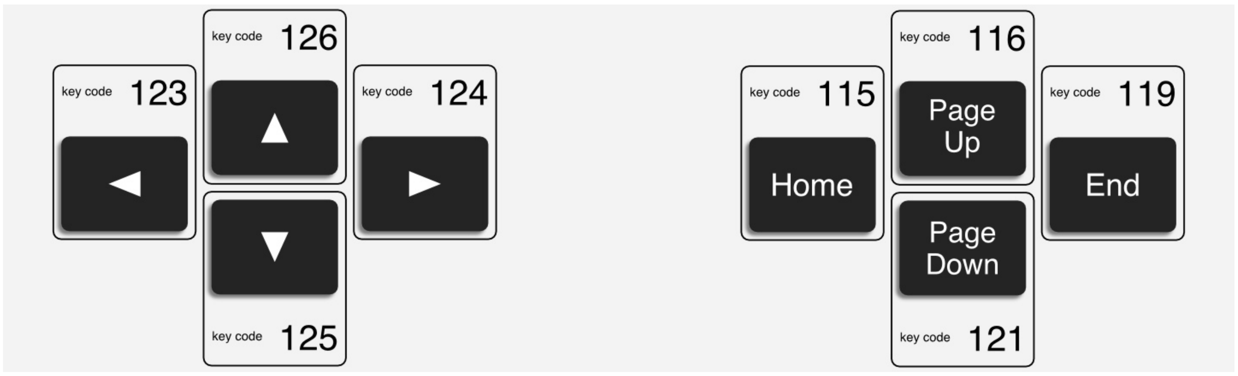
```
tell application "System Events" to keystroke "V" using
```

For numbers, letters and symbols, using keystroke is typically preferable to using key codes. Keystroke is easier to use, I think. However, all of the letter, number and symbol keys do have associated key codes standing by.

Remember, when using keystroke, place the corresponding characters in quotes.

```
keystroke "Hello world"
```

Arrow keys, page up, page down, home and end



Arrow keys can be incredibly useful. So can the page up/down, home and end keys.

Here are some interesting uses of the arrow keys. Try these out in a text editor.

Skip ahead one word:

```
tell application "System Events" to key code 124 using o
```

Go back one word:

```
tell application "System Events" to key code 123 using o
```

Skip to the end of the paragraph:

```
tell application "System Events" to key code 125 using o
```

Go back to the beginning of the paragraph:

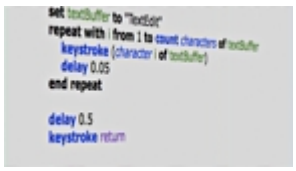
```
tell application "System Events" to key code 126 using o
```

Skip to the end of the line:

```
tell application "System Events" to key code 124 using c
```

Go back to the beginning of the line:

```
tell application "System Events" to key code 123 using c
```



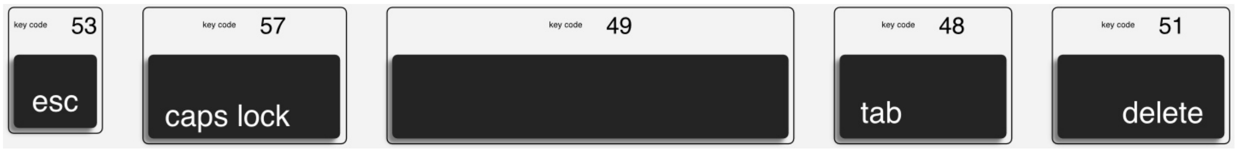
```
set textBuffer to "TextEdit"
repeat with i from 1 to count characters of textBuffer
    keystroke (character i of textBuffer)
    delay 0.05
end repeat

delay 0.5
keystroke return
```

How to automate your keyboard in Mac OS X with AppleScript

Scripting keyboard events with AppleScript is a quick and easy way to automate those common, repetitive tasks in...

Read more

Esc, space bar, tab, delete, caps lock



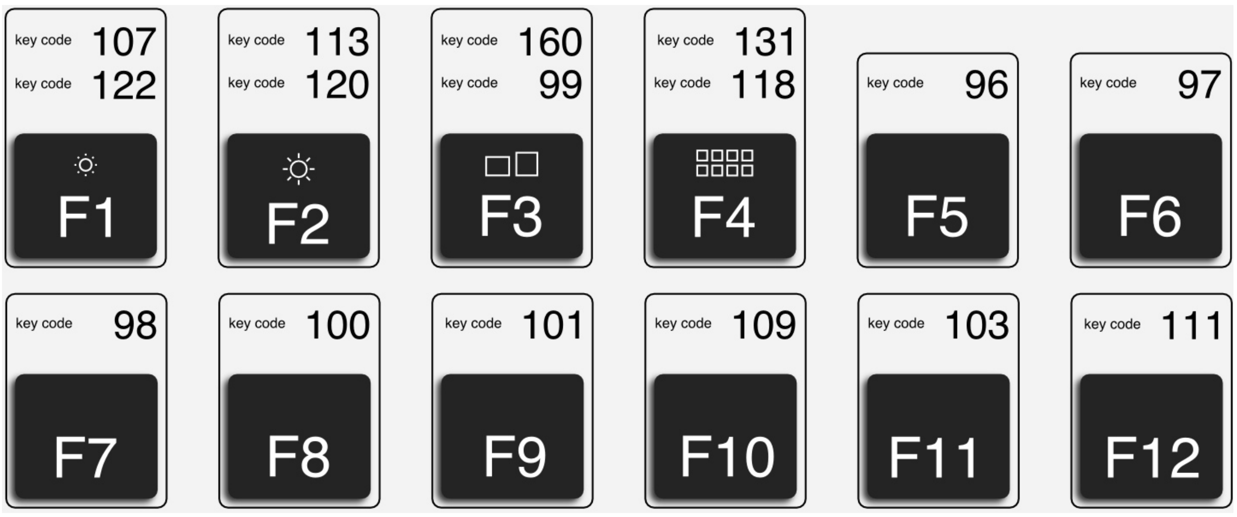
All of these work as expected, except for caps lock. Key code 57 does not appear to enable or disable the caps lock.

You should be aware that return, space, and tab can be used with keystroke.

```
keystroke return
keystroke space
keystroke tab
```

In this case we don't use quotation marks. If you did, AppleScript would write out the name of the key instead of actually invoking that key.

F keys and some of the function keys



Not all of the function keys can be reliably scripted using key codes. See further down in this guide for work arounds.

In the diagram above, some of the keys have two key codes associated with them. The top number corresponds to the function and the bottom number corresponds to the F key. For example, this would increase your screen brightness by one increment, the same as if you had pressed that key.

```
 tell application "System Events" to key code 113
```

What about F13-F20? We can script those, too!

F13	key code 105
F14	key code 107
F15	key code 113
F16	key code 106
F17	key code 64
F18	key code 79

F19	key code 80
F20	key code 90

Play, pause, fast forward, rewind and volume function keys

I am not aware of a good way to do this using key codes. However, these basic functions are still very easy to accomplish without key codes.

Play: `tell application "iTunes" to play`

Pause: `tell application "iTunes" to pause`

Rewind (previous track): `tell application "iTunes" to previous track`

Fast forward (next track): `tell application "iTunes" to next track`

If you’re using something other than iTunes, you can still try the above. Just substitute “iTunes” with the name of the app. Depending on how scriptable the app is, this may or may not work.

Fortunately, setting the volume is a little less prone to error than play/pause/fast forward/rewind. While the play/pause function is dependent on app scriptability, setting the volume changes the whole system’s volume. Mute, unmute, set and increment volume like so:

Mute: `set volume with output muted`

Unmute: `set volume without output muted`

Set volume to 100%: `set volume output volume 100`

Set volume to 50%: `set volume output volume 50`

Set volume to 1%: `set volume 1`

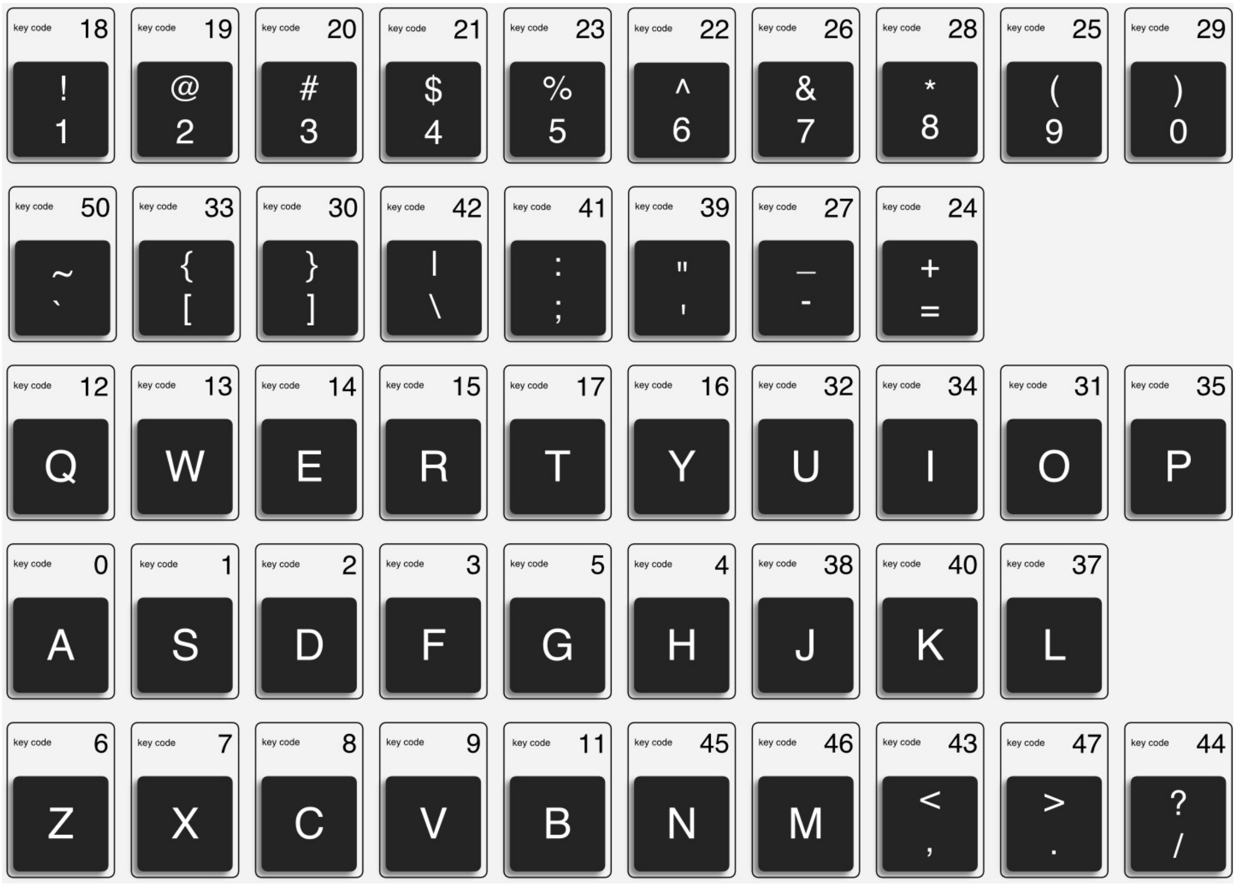
Make your system volume slowly fade out with this nifty little script. Adjust the “delay 0.2” bit in the middle of the loop to speed up or slow down the fade.

```
set a to output volume of (get volume settings) --set a

repeat while a is not 0 --repeat until volume is zero
        set a to (a - 1) --set volume decrement to curre
        delay 0.2 --delay 0.2 seconds between each decre
        set volume output volume a --decrement volume by
end repeat
```

Controlling keyboard brightness

As far as I am aware, there is no easy way to do this with AppleScript… If you must control keyboard brightness with AppleScript, probably your best bet would be to find an app to do this. Then you could control that app from AppleScript.

Letters, numbers and symbols



All of these can be used with keystroke, like this:

```
tell application "System Events" to keystroke "abcd"
```

That said, all of these keys still have corresponding key codes.

Numpad key codes

Numpad 1	83
Numpad 2	84
Numpad 3	85
Numpad 4	86
Numpad 5	87
Numpad 6	88

Numpad 7	89
Numpad 8	91
Numpad 9	92
Numpad 0	82
Numpad *	67
Numpad /	75
Numpad +	69
Numpad -	78
Numpad =	81
Numpad .	65
Numpad clear	71

Last updated June 28, 2015

By CK

email@eastmanreference.com

subscribe via RSS