HyperTalk™
Quick Reference Card

The information in this Quick Reference Card pertains to HyperCard version 1.2.2.

Script editor command summary

Key combination	Action
Command-A	Select entire script
Command-C	Copy selection to Clipboard
Command-F	Find text (same as Find button)
Command-G	Find next occurrence of same text
Command-H	Find current selection
Command-P	Print selection or (if no selection) entire script (same as Print button)
Command-period	Close script without saving changes (same as Cancel button)
Command-V	Paste Clipboard contents at insertion point
Command-X	Cut selection to Clipboard
Enter	Close script and save changes (same as OK button)
Option-Return	Wrap line without return character ("soft" return—symbolized by ¬ in scripts. Don't use a "soft" return inside quotation marks.)
Return	Return character—indicates end of HyperTalk statement
Tab	Format script

Shortcuts for seeing scripts

Key combination	Effect
Command-Option	Display buttons; click a button with keys down to edit its script
Shift-Command-Option	Display fields and buttons; click a field (or button) with keys down to edit its script
Command-Option-C	Edit script of current card
Command-Option-B	Edit script of current background
Command-Option-S	Edit script of current stack

Commands

In the statements listed below, square brackets [ ] enclose optional elements. (Don't type the square brackets.) Words in *italic* are placeholders describing general elements, not specific names; you must replace them in an actual command. It doesn't matter whether you use uppercase or lowercase letters in HyperTalk; names formed from two words are shown with an embedded capital letter (`likeThis`) merely to make them more readable. The HyperTalk prepositions `of` and `in` are interchangeable.

`add` *expression* `to` *destination*

`answer "`*question*`" [with "`*reply*`" [or "`*reply2*`" [or "`*reply3*`"]]]`

`arrowKey` *keyName*

`ask [password]` *question* `[with` *defaultAnswer*`]`

`beep` *number*

`choose` *toolName* `tool`

`click at` *location* `[with` *key*`[,` *key2*`[,` *key3*`]]]`

`close file` *fileName*

`close printing`

`convert` *container* `to` *format* `[and` *format*`]`

`controlKey` *asciiNumber*

`delete` *chunk* `[of` *container*`]`

`dial` *expression* `[with modem [`*modemCommands*`]]`

`divide` *destination* `by` *expression*

`doMenu` *menuItem*

`drag from` *start* `to` *finish* `[with` *key*`[,` *key2*`[,` *key3*`]]]`

`edit script of` *object*

`enterKey`

`find [chars]` *expression* `[in field` *fieldDesignator*`]`

`find [word]` *expression* `[in field` *fieldDesignator*`]`

`find string` *expression* `[in field` *fieldDesignator*`]`

`find whole` *expression* `[in field` *fieldDesignator*`]`

`functionKey` *keyNumber*

`get` *expression*

`go [to] [stack] "`*stackName*`"`

`go [to]` *bkgndDescriptor* `[of [stack] "`*stackName*`"]`

`go [to]` *cardDescriptor* `[of` *bkgndDescriptor*`] [of [stack] "`*stackName*`"]`

`help`

`hide menuBar`

`hide` *windowName*

`hide` *object*

`hide` *picture*

```
lock screen
multiply destination by expression
open [document with] application
open file fileName
open printing [with dialog]
play "voice" [tempo tempoValue] ["notes"]
play stop
pop card [preposition destination]
print card
print expression cards
print cardDescriptor
print document with application
push cardDescriptor
put expression [preposition destination]
read from file fileName until character
read from file fileName for numberOfCharacters
reset paint
returnKey
select object
select [preposition] expression of field
select [preposition] expression of msg
select [preposition] text of field
select empty
set [the] property [of object] to value
show [all] cards
show number cards
show menuBar
show windowName [at h, v]
show object [at h, v]
show picture
sort [direction] [style] by expression
subtract expression from destination
tabKey
type expression [with key[, key2[, key3]]]
unlock screen [with effectName]
visual [effect] effectName [speed] [to image]
wait [for] number [seconds]
wait until condition
wait while condition
write source to file fileName
```

Functions

In the statements listed below, square brackets `[ ]` enclose optional elements. (Don't type the square brackets.) Words in italic are placeholders describing general elements, not specific names; you must replace them in an actual command. It doesn't matter whether you use uppercase or lowercase letters in HyperTalk; names formed from two words are shown with an embedded capital letter (`likeThis`) merely to make them more readable. The HyperTalk prepositions `of` and `in` are interchangeable.

When using functions in HyperTalk statements you must either use the word `the` before the function name or add parentheses after it. Both forms are shown in the list that follows. *Factor* is a single value, such as the number 5 or a container holding a value; *expression* can be a single factor or a combination of several factors and operators that results in a value, such as `(2+3)` or `(2+(field 1))`. Parameters in a list must be separated by commas.

`the abs of` *factor* `abs (`*expression*`)`	Absolute value
`annuity (`*rate, periods*`)`	Calculates an annuity
`the atan of` *factor* `atan (`*expression*`)`	Arc tangent—radians
`average (`*list*`)`	Calculates an average
`the charToNum of` *factor* `charToNum (`*expression*`)`	Returns the ASCII value of a character
`the clickH`	Gives horizontal coordinate of where the user last clicked
`the clickLoc` `clickLoc()`	Gives horizontal and vertical coordinates of where the user last clicked
`the clickV`	Gives vertical coordinate of where the user last clicked
`the commandKey` `commandKey()`	Condition of the Command key: `up` or `down`
`compound (`*rate, periods*`)`	Calculates compound interest
`the cos of` *factor* `cos (`*expression*`)`	Cosine—radians
`the [`*modifier*`] date`	Current date set in the Macintosh: long or short
`the diskSpace` `diskSpace()`	Amount of free space on the current disk
`the exp of` *factor* `exp (`*expression*`)`	Mathematical exponential
`the exp1 of` *factor* `exp1 (`*expression*`)`	1 less than mathematical exponential: `exp()-1`
`the exp2 of` *factor* `exp2 (`*expression*`)`	The value of 2 raised to the power of *factor*

`the foundText`	Returns characters found by the `find` command
`the foundChunk`	Returns a description of where the text is found
`the foundLine`	Tells which line the found text is in
`the foundField`	Tells which field the found text is in
`the length of` *factor* `length(`*expression*`)`	Number of characters in a text string
`the ln of` *factor* `ln(`*expression*`)`	Natural logarithm—base-*e*
`the ln1 of` *factor* `ln1(`*expression*`)`	1 plus the natural logarithm: `ln(1+`*factor*`)`
`the log2 of` *factor* `log2(`*expression*`)`	Base-2 logarithm
`max(`*list*`)`	Returns the highest number value of a list
`min(`*list*`)`	Returns the lowest number value of a list
`the mouse` `mouse()`	Condition of the mouse button: `up` or `down`
`the mouseClick` `mouseClick()`	Returns `true` if the mouse button is clicked
`the mouseH` `mouseH()`	Horizontal position of the pointer on the screen
`the mouseLoc` `mouseLoc()`	Horizontal and vertical coordinates of the pointer
`the mouseV` `mouseV()`	Vertical position of the pointer
`[the] number of` *objects*	Number of buttons/fields on current card or bg
`[the] number of` *chunks* `in` *factor*	Number of characters, words, lines, and so on in text string
`[the] number of cards of` *background*	Number of cards in specified background
`the numToChar of` *factor* `numToChar(`*expression*`)`	Returns the character corresponding to an ASCII value
`offset(`*string1*`,` *string2*`)`	Gives number of characters between the beginnings of two strings
`the optionKey` `optionKey()`	Condition of the Option key: `up` or `down`
`the param of` *factor* `param(`*expression*`)`	Returns the value of a parameter in a list
`the paramCount` `paramCount()`	The total number of parameters
`the params` `params()`	The entire list of parameters

`the random of` *factor* `random(`*expression*`)`	Gives a random integer from 1 to the value of *factor*
`the result` `result()`	Returns a text string if `find` or `go` is unsuccessful
`the round of` *factor* `round(`*expression*`)`	Rounds to nearest integer: an odd integer plus 0.5 rounds up; an even integer, down
`the screenRect` `screenRect()`	The rectangle of the screen in which the menu bar is displayed: left, top, right, bottom coordinates
`the seconds` `seconds()`	Number of seconds between midnight January 1, 1904, and the current time in your Macintosh
`the selectedText`	Returns the text currently selected
`the selectedChunk`	Describes the location of the selected text
`the selectedLine`	Tells which line the selected text is in
`the selectedField`	Tells which field the selected text is in
`the shiftKey` `shiftKey()`	Condition of the Shift key: `up` or `down`
`the sin of` *factor* `sin(`*expression*`)`	Sine—radians
`the sound` `sound()`	Name of sound resource currently playing, or `"done"` if none is playing
`the sqrt of` *factor* `sqrt(`*expression*`)`	Square root of a positive number—a negative number gives the result `NAN(001)` meaning "not a number"
`the tan of` *factor* `tan(`*expression*`)`	Tangent—radians
`the target` `target()`	Identifies the original recipient of a message
`the ticks` `ticks()`	Number of ticks (⅙₀ second) since the Macintosh was turned on or restarted
`the [`*modifier*`] time` `time()`	Gives time as a text string: long, short, abbreviated
`the tool` `tool()`	Name of currently chosen tool
`the trunc of` *factor* `trunc(`*expression*`)`	The integer part of a number in *function*
`the value of` *factor* `value(`*expression*`)`	Gives the value of a string as an expression
`the [long] version [of HyperCard]` `version()`	Returns the version number of HyperCard
`the version of` *stackDescriptor*	Tells version of HyperCard used to create, compact, change since compacted, and make latest changes, plus the date modified in seconds since January 1, 1904

Operator precedence

Order	Operators	Type of operator
1	`( )`	Grouping
2	`-`	Minus sign for numbers
	`not`	Logical negation for Boolean values
3	`^`	Exponentiation for numbers
4	`* / div mod`	Multiplication and division for numbers
5	`+ -`	Addition and subtraction for numbers
6	`& &&`	Concatenation of text
7	`> < <= >= ≤ ≥`	Comparison for numbers or text
	`is in contains`	Comparison for text
	`is not in`	Comparison for text
8	`= is is not <> ≠`	Comparison for numbers or text
9	`and`	Logical for Boolean values
10	`or`	Logical for Boolean values