Keyboard and mouse Commands

Instructions and functions to handle the keyboard and the mouse

Inkey$
Checks to see if a key has been pressed, and reports back its value in a string
Value returned:
string: The value of the last key pressed

Input$
This function loads a given number of characters into a string variable, waiting for the user to enter each character in turn.
Value returned:
string: The text entered by the user

Input VARIABLE
The INPUT command is used to enter information into one or more variables, separated byy commas
Parameters:
VARIABLE: The variable to store the return in, any type

Line Input VARIABLE
The LINE INPUT instruction is identical in usage to INPUT, except that is uses a press of the [Return] key to separate each value you enter via the keyboard instead of a comma.
Parameters:
VARIABLE: The variable to store the return in, any type

Key Speed TIME_LAG, DELAY_SPEED
Change key repeat speed. Not implented in HTML applications, maybe later in executable applications
Parameters:
TIME_LAG: The time-lag before repeat, in 1/50th of second
DELAY_SPEED: The delay before each repeated key, in 1/50th of second

Key State KEY_CODE
Test for a specific key press
Parameters:
KEY_CODE: The Javascript code of the key to test. The Amiga Scan Code in Amiga mode
Value returned:
boolean: true if the key is pressed, False if not

Key Shift
Test the status of control keys
Value returned:
integer: A mask of bits indicating which control key is down. Bit 0: left Shift, Bit 1: right Shift (note Javascript does not distinct left and right shift keys), Bit 2: Caps Lock, Bit 3: Ctrl, Bit 4: left Alt, Bit 5: right Alt, Bit 6: system key (Windows for PC, CMD for Mac, Amiga in Amiga emulation)

Put Key TEXT$
Load a string of characters directly into the keyboard buffer, enabling you to simulate user typing (in INPUT functions for example)

Parameters:
TEXT$: The texte to put in the buffer

ScanCode
Return the scancode of a key entered with INKEY$
Value returned:
integer: The code of the key that was entered by the user and detected by the last INKEY$. Will report Javascript key-codes for PC, and Amiga ScanCodes in Amiga emulation

ScanShift
Return shift status of key entered with INKEY$
Value returned:
integer: A mask of bits indicating the sate of the SHIFT key. =0 no Shift key pressed, <>0 one of the Shift key is pressed

Clear Key
Re-set the keyboard buffer. Might not have an effect on all platforms.

Wait Key
This simple command waits for a single key-press before acting on the next instruction

Key$ MACRO$
Reserved variable. Define a keyboard macro
Parameters:
MACRO$: The text of the macro

Scan$ SCANCODE, MASK
Return a scan-code for use with Key$
Parameters:
SCANCODE: The Javascript code of the key for PC, and the Amiga scan code for Amiga emulation
MASK: A mask of bits, wioth the same format as KEY SHIFT
Value returned:
string: The string to use with Key$

Hide On
Hide the mouse pointer (Deprecatred, use "Hide")

Hide
Hide the mouse pointer

Show On
Show the mouse pointer (Deprecatred, use "Hide")

Show
Show the mouse pointer

Show SHAPE
Change the shape of the pointer arrow
Parameters:
SHAPE: Number of the shape to use

X Mouse
Reserved Variable. Returns / Set the horizontal coordinate of the mouse. Note: setting the coordinate may not work on all platforms and in HTML applications.
Value returned:
integer: The horizontal coordinate of the mouse over the application in hardware coordinates

Y Mouse
Reserved Variable. Returns / Set the vertical coordinate of the mouse. Note: setting the coordinate may not work on all platforms and in HTML applications.
Value returned:
integer: The vertical coordinate of the mouse over the application in hardware coordinates

Mouse Key
Return the status of the mouse buttons
Value returned:
integer: A mask of bits. Bit 0: Left mouse button, Bit 1: Right mouse button, Bit 2: Middle mouse button if it exists

Mouse Click
Check for click of mouse button. This is similar to MOUSE KEY, but instead of checking to see whether or not a mouse button is held down, MOUSE CLICK is only interested in whether the user has just made a single click on a mouse button.
Value returned:
integer: A mask of bits. Bit 0: Left mouse button, Bit 1: Right mouse button, Bit 2: Middle mouse button if it exists

Limit Mouse ... To ... X1, Y1, X2, Y2
Change the shape of the pointer arrow
Parameters:
X1: Horizontal coordinate of the top-left corner of the bounding area
Y1: Vertical coordinate of the top-left corner of the bounding area
X2: Horizontal coordinate of the bottom-right corner of the bounding area
Y2: Vertical coordinate of the bottom-right corner of the bounding area

Limit Mouse X, Y, WIDTH, HEIGHT
Change the shape of the pointer arrow
Parameters:
X: Horizontal coordinate of the top-left corner of the bounding area
Y: Vertical coordinate of the top-left corner of the bounding area
WIDTH: Width of the bounding area
HEIGHT: Height of the bounding area

Limit Mouse
Without parameters, restore the displacement of the mouse to the whole screen

Mouse Wheel
Return the movement of the mouse wheel
Value returned:
integer: A displacement, positive or negative and dependant of the system