



webtalk - a Javascript and CSS based HyperTalk interpreter.

Table of Contents

Release Notes.....	2
License agreement.....	45
Other footnotes:.....	45

Release Notes

Version 1

now I have the basic structure in place, add simple sum parsing (with get and put):

```
get 10 + 5  
(stores the result in it )
```

Version 2

added the 'put' command

Version 3

Now you can try these commands:

```
put word 1 of the date  
put char 1 of the time  
put the date  
put the time
```

This should output:

```
The day of the week (e.g., "Wednesday")  
The first character of the time (e.g., "1")  
The full date  
The full time
```

Version 4

```
put char 1 of "hello"  
put word 2 of "hello world"
```

now understands how to pick out certain parts (char x and word x)

Version 5

is now also aware of the length:

```
put the length of "hello"  
put the length of the date  
put char 3 of "test"  
put word 2 of "hello world"  
put the time
```

Version 6

Added word/char/character/words/chars/characters n to m of functionality, which allows getting ranges of text. This is a powerful HyperTalk feature that lets you extract ranges like:

```
put words 2 to 4 of "this is a test message"  
put chars 2 to 5 of "hello world"
```

Version 7

Added string concatenation with & and the is in operator for string containment checks. These were both commonly used in HyperTalk. Here's the implementation:

```
put "hello" & " " & "world"
put "test" is in "this is a test"
put "xyz" is in "abc"
put word 1 of "hello" & " there"
put chars 1 to 3 of "hello" & chars 4 to 5 of "world"
put the time
put the number of chars of "test" & "ing"
```

Version 8

added string transformations that were common in HyperTalk: the first/last/middle word/char/character of and item n of (with comma as default delimiter). These are powerful text manipulation features:

example:

```
put the first word of "hello world there"
put the last word of "hello world there"
put the middle word of "hello world there"
put the first char of "testing"
put the last character of "testing"
put the middle character of "test"
put item 2 of "apple,banana,cherry"
put item 1 of "1,2,3" & item 3 of "a,b,c"
put "test" is in the last word of "hello world test"
put chars 1 to 3 of the first word of "hello there"
```

outputs:

```
hello
there
world
t
g
e
banana
1c
true
hel
```

Version 9

Not my strong point at all this, so bear with me.

The dreaded 'math' functions. (or maths to me, because there's more than one)... anyway.

Took me ages to work this out.

Testing with:

```
put 5 > 3
put 10 < 5
put 6 is 6
put "hello" is "hello"
put 7 <> 7
put 8 >= 8
put 4 <= 3
put 10 + 5
put 20 - 8
put 6 * 7
put 15 / 2
```

```
put 15 div 2
put 17 mod 5
put the length of "test" > 2
put item 2 of "1,2,3" is "2"
```

returns the following output:

```
true
false
true
true
false
true
false
15
12
42
7.5
7
2
true
true
```

Version 10

string transformations: convert to uppercase/lowercase and the number of items of. This will let users do case conversions and count comma-separated items:

```
put convert "Hello World" to uppercase
put convert "Hello World" to lowercase
put convert the first word of "Hello World" to uppercase
put the number of items of "apple,banana,cherry"
put the number of items of ""
put convert item 2 of "hello,WORLD,test" to lowercase
put convert "test" & "ing" to uppercase
put the number of items of "1,2,3" > 2
put 5 + the number of items of "a,b,c,d"
```

Oops! -- the last line throws an error:

Error: Invalid expression: 5 + the number of items of "a,b,c,d"

Version 11

I knew I'd have a problem with the maths functions, and was dreading this.

I fixed the previous error, so running:

```
put 10 - the number of items of "x,y"
```

now returns:

9

(which is **STILL** wrong. Should be 8)

I also created another error in the parser:

```
put the number of items of "1,2,3" + the number of items of "a,b"
```

returns error:

Error: Math operations require numeric values

Version 12

Sorted out my terrible error (caused by my terrible understanding of maths!)

```
put 10 - the number of items of "x,y"  
put "test" is in "testing"
```

now returns (as you'd expect):

```
8  
true
```

Version 13

Added toLower and toUpper conversion:

```
put toUpper(char 1 of "hello")  
put toLower(char 3 of "HELLO")  
put toUpper(word 2 of "hello world")  
put char 1 of "hello"  
put toUpper("hello")
```

returns:

```
H  
l  
WORLD  
h  
HELLO
```

Version 14

Added string replacement:

```
put replace "l" with "L" in "hello"  
put replace "world" with "everyone" in "hello world"  
put replace "," with ";" in "one,two,three"  
put replace char 1 of "hello" with "H" in "hello"  
put replace toUpper("l") with "*" in "hello"
```

returns:

```
hello  
hello everyone  
one;two;three  
Hello  
hello
```

Version 15

Added the offset function:

```
put offset("world" in "hello world")  
put offset("l" in "hello")  
put offset("x" in "hello")  
put offset(char 1 of "hello" in "hello")  
put offset(toUpper("l") in "hello")
```

returns (as you'd expect):

```
7  
3  
0  
1  
0
```

Version 16

Added repeat loops.

Testing:

```
repeat 3 times
  put "Hello"
end repeat
```

returns:

```
Hello
Hello
Hello
```

```
repeat 2 times
  put toUpper("test")
  put offset("e" in "hello")
end repeat
```

returns:

```
TEST
2
TEST
2
```

and...

```
repeat 4 times
  put replace "l" with "*" in "hello"
end repeat
```

returns:

```
he**o
he**o
he**o
he**o
```

Version 17

adding if/then/else:

```
if offset("world" in "hello world") > 0 then
  put "Found world!"
else
  put "No world found"
end if
```

returns:

```
Found world!
```

```
if toUpper(char 1 of "hello") is "H" then
  put "Starts with H"
  put "Good!"
else
  put "Doesn't start with H"
end if
```

returns:

```
Starts with H
```

Good!

and...

```
if the number of items of "a,b,c" > 2 then
  put "More than 2 items"
else
  put "2 or fewer items"
end if
```

returns:

More than 2 items

Version 18

adding repeat while:

```
put "0" into counter
repeat while counter < "3"
  put counter
  put counter + 1 into counter
end repeat
```

```
put "hello" into msg
repeat 1 times
  put toUpper(msg) into msg
end repeat
put msg
```

I also added the sort function:

```
put "banana,apple,cherry" into fruits
put sort items of fruits

put "5,3,1,4,2" into numbers
put sort items of numbers in ascending order
put sort items of numbers in descending order
```

```
put "Z,a,B,y,C,x" into mixed
put sort items of mixed
```

returns:

```
apple, banana, cherry
1, 2, 3, 4, 5
5, 4, 3, 2, 1
a, B, C, x, y, Z
```

Version 19

added date conversion and dateitems:

```
put the date
put the long date
put the abbreviated date
put the time
put the dateitems
```

```
put convert the date to english date
put convert "2024,2,6,3,14,30,0" to long date
put convert the long date to short date
```

returns:
2/6/2025
Thursday, February 6, 2025
Thu, Feb 6, 2025
2:18 PM
2025,2,6,5,14,18,48
Thursday, February 6, 2025
Tuesday, February 6, 2024
2/6/2025

Version 20

added the random function:

```
put the random of 6
put random(6)
put random(1,100)

repeat 5 times
  put the random of 10
end repeat
```

Output:

```
3
4
94
3
1
2
3
7
```

Version 21

added string chunks/chunking:

```
put "Hello, World! How are you?" into msg
put chars 1 to 5 of msg
put words 1 to 3 of msg
put word -1 of msg
put first word of msg
put last word of msg
put middle word of msg

put "apple,banana,cherry,date" into fruits
put items 2 to 3 of fruits
put first item of fruits
put last item of fruits
```

Output:

```
Hello
Hello, World! How
you?
Hello,
you?
How
banana,cherry
apple
date
```


Version 22

Adding combine and split.

```
put "apple,banana,cherry" into fruits
put combine fruits with " and "
put combine fruits with "|"

put "hello world how are you" into msg
put split msg by " "
put split msg by "o"
put "2024-02-06" into date
put split date by "-"
put combine date with "/"
```

Output

```
hello, world, how, are, you
hell, w, rld h, w are y, u
2024, 02, 06
2024-02-06
```

I also added the wait command (which is harder than it sounds in a web implementation!)

```
put "Starting..."
wait 2 seconds
put "After 2 seconds"
wait 500 milliseconds
put "After 500ms"
```

(now works as you'd expect)

Version 23

repeat with function:

```
repeat with i = 1 to 5
  put "Counter is " && i
end repeat

repeat with x = 10 to 6
  put "Counting down: " && x
end repeat
```

Output:

```
Counter is 01
Counter is 02
Counter is 03
Counter is 04
Counter is 05
Counting down: 10
Counting down: 09
Counting down: 08
Counting down: 07
Counting down: 06
```

Version 24

testing my repeat logic:

```
put 10 into tcount
repeat tcount
  put "output"
end repeat

put 5 into n
repeat n times
  put "another output"
end repeat
```

Result is:

```
output
output
output
output
output
output
output
output
output
output
another output
another output
another output
another output
another output
```

Version 25

added the 'starts with' and 'ends with' condition:

```
put "abc123" into test
if test starts with "abc" and test ends with "123" then
  put "Matches both patterns"
  wait 1 second
  put "Still matching!"
end if
```

Output:

```
Matches both patterns
Still matching!
```

I can also verify it further with:

```
put "abc1234" into test
if test starts with "abc" and test ends with "123" then
  put "Matches both patterns"
else
  put "no match"
end if
```

output:

```
no match
```

(as I'd expect as I changed the string to end with 1234 deliberately)

Version 26

Added the ability to +1 and -1 to a variable.

Also interchangeability using add 1 to variable and subtract 1 from variable

```
put 1 into myvar
repeat 11
  put myvar + 1 into myvar
  put myvar
  put myvar -2 into myvar
  put myvar
end repeat
```

```
put 1 into myvar
repeat 11
  add 1 to myvar
  put myvar
  subtract 2 from myvar
  put myvar
end repeat
```

Version 27

Added a testing.html file so that I can easily find out if I break a feature by adding something else.

This is just to help me debug stuff while I put this together!

Version 28

going through, debugging - fixing errors.

Just resolved the

"Put word x of string" function.

The quoted string handler was running first and stripping the quotes, which was causing the rest of the string to be treated incorrectly.

Version 29

Backtracking, fixing bugs:

we didn't have a specific handler for "the length of" expressions. Now fixed, so the interpreter will do the following:

Match expressions like "the length of X"

Evaluate X (which could be a variable, string, or property like "the time")

Convert the result to a string

Return it's length

Version 30

Backtracking, fixing bugs.

Annoyingly these are creeping in when adding extra features, because one might be inter-related to the other due to how they are handled in the interpreter.

Length of variable test

put the length of the time

Length of string test

put the length of "a sample string"

Number of words test

put the number of words of "this is another test"

Version 31

fixed my previous mistakes:

Number of chars in variable

put the number of chars of the long date

Number of chars in string

success

put the number of chars of "abcdefghijklmnop"

Version 32

fixed "is in" function:

"is in" variable test

success

if "1" is in the date then put "1 is in the date"

"is in" string test

success

if "0" is in "hell0" then put "0 is in our string"

Version 33

bug fixing:

First char of string

success

put the first char of "test"

Last char of string

success

put the last char of "testing"

First char of variable

success

put the first char of the time

Last char of variable

success

put the last char of the time

Version 34

bug fixing:

To uppercase

success

put toUpper("this should be uppercase")

To lowercase

success

put toLower("ThIs ShoULD bE lOweRcaSe")

Version 35

bug fixing:

Count items

success

put the number of items in tItems

Version 36

Bug fixing:

Delete last char

success

put "strings" into tstring

delete last char of tstring

put tstring

Version 37

Bug fixing mode:

Offset test

success

put offset("needle" in "this is the needle somewhere in a haystack")

Version 38

bug fixing (because I'm an idiot)

The repeat until command itself is working correctly.

The rest of the repeat until structure was correct

I should have used:

put true into tVar

instead of

set tVar to true

This is now fixed :)

Version 39

fixed sort items:

Sort items

success

put "f, b, d, c, e, a" into tItems

sort items of tItems ascending

Version 40

bug fix:

DateItems test

success

put the dateItems into tDateBits

put tDateBits

Version 41

bug fixing:
corrected conversion of the dateitems back to the long date.
(note to self: I nearly broke EVERYTHING doing this!!)

```
put the dateItems into tDateBits
convert tDateBits to the long date
put tDateBits
```

Version 42

fixed the "starts with", "begins with" and "ends with" function:

```
put "abc" into tItems
if tItems starts with "a" then
  put "starts with a"
end if
```

```
put "xyz" into tItems
if tItems begins with "x" then
  put "begins with x"
end if
```

```
put "test" into tItems
if tItems ends with "st" then
  put "ends with st"
end if
```

Version 43

This is where it could all go wrong, as I'm really out of my depth now!
(yes, it's maths-related!) not my strong point.

It took me ages to work this out, and a lot of googling, but I think I've got it:

```
put sin(90) into tResult
put tResult
put 30 into tAngle
put sin(tAngle) into tResult
put tResult
```

returns:

```
1
0.49999999999999994
```

I think that's right!

Version 44

This is pretty much over my head, but google tells me that since I added SIN, then there's COS too.

After a lot of head scratching, and a bit of ChatGPT thrown in, I've managed to do:

Cos function test
success

```
put cos(0) into tResult
put tResult
put 60 into tAngle
```

```
put cos(tAngle) into tResult
put tResult
```

returns output:

```
1
0.5000000000000001
```

Version 45

I had an issue with tan lines. (haha)
Not that kind. Although that kind makes more sense.

If I run:

```
put tan(45) into tResult
put tResult
put tan(0) into tResult
put tResult
put 60 into tAngle
put tan(tAngle) into tResult
put tResult
```

I now get:

```
0.9999999999999999
0
1.7320508075688767
```

which I think (no, which I *HOPE*) is correct!

Version 46

Added square root calculation (I think!)

```
put sqrt(16) into tResult
put tResult
put 25 into tNumber
put sqrt(tNumber) into tResult
put tResult
put sqrt(0) into tResult
put tResult
```

output is:

```
4
5
0
```

Version 47

Added the round function.

I do actually use this, so it's one I have a better understanding of:

```
put round(3.7) into tResult
put tResult
put round(3.2) into tResult
put tResult
put 4.6 into tNumber
put round(tNumber) into tResult
put tResult
```

the output is:

```
4
```

3
5

4 (3.7 rounded up)
3 (3.2 rounded down)
5 (4.6 rounded up)

Version 48

The Absolute value function test:

```
put abs(-5) into tResult
put tResult
put abs(3.2) into tResult
put tResult
put -7.5 into tNumber
put abs(tNumber) into tResult
put tResult
```

The output is:

5
3.2
7.5

Version 49

Added the "contains" feature, as in:

```
put "Hello World" contains "World" into tResult
put tResult
put "Hello" contains "xyz" into tResult
put tResult
put "Hello" into tString
put tString contains "el" into tResult
put tResult
```

Output:

Hello World" contains "World
Hello" contains "xyz
true

Version 50

Added the reverse funtion:

```
put reverse("Hello") into tResult
put tResult
put "World" into tString
put reverse(tString) into tResult
put tResult
put reverse("A man a plan") into tResult
put tResult
```

the result of all this is:

olleH
dlroW
nalp a nam A

Seems to work.

Version 51

Added an answer dialog!

```
answer "Do you want to continue?" with "Yes" or "No"
put the result
answer "Pick a number" with "1" or "2" or "3"
put the result
```

(now correctly returns '*the result*').

Version 52

added an ask dialog, since I just added an answer one!

```
ask "What are you thinking?"
put it

put "What kind of mood are you in?" into tPrompt
ask tPrompt
put it
```

Version 53

I thought I added the replace function already, but I must be getting tired as it doesn't seem to be there.
So...

```
put replace("hello world", "world", "there") into tResult
put tResult

put "hello hello hello" into tString
put replace(tString, "hello", "hi") into tResult
put tResult

put "no matches here" into tString
put replace(tString, "xyz", "abc") into tResult
put tResult
```

Now returns:

```
hello there
hi hi hi
no matches here
```

Version 54

sort command test:

```
put "banana,apple,cherry" into tFruits
sort items of tFruits
put tFruits

put "30,10,20" into tNumbers
sort items of tNumbers ascending
put tNumbers

put "cat,dog,bird" into tAnimals
sort items of tAnimals descending
put tAnimals
```

Output:

```
apple,banana,cherry
10,20,30
dog,cat,bird
```

Version 55

Added conversion of date formats:

```
put 2025,2,7,12,30,0 into tDate -- store the date
convert tDate to longdate -- convert to long format
put tDate
```

```
put 2025-02-07 into tDate
convert tDate from date to long date
put tDate
```

```
put 2025-02-07 into tDate
convert tDate to dateitems
put tDate
```

Oh, and you'll notice you can now comment with "--" too, as I thought that would be handy!

Version 56

Added "the power" of number.

Thanks go to my 11 year old daughter for helping with this!

```
put power(2, 3) into tResult -- 2^3 = 8
put tResult
```

```
put 5 into tBase
put 2 into tExponent
put power(tBase, tExponent) into tResult -- 5^2 = 25
put tResult
```

```
put power(3, 0) into tResult -- 3^0 = 1
put tResult
```

```
put power(2, -2) into tResult -- 2^-2 = 0.25
put tResult
```

the output is (which between us, we think is correct):

```
8
25
1
0.25
```

Version 57

The "Mod" function.

Again - thanks to my 11 year old for this one.

```
put mod(7, 3) into tResult -- 7 mod 3 = 1
put tResult
```

```
put 17 into tNumber
put 5 into tDivisor
put mod(tNumber, tDivisor) into tResult -- 17 mod 5 = 2
put tResult
```

```
put mod(10, 2) into tResult -- 10 mod 2 = 0  
put tResult
```

```
put mod(-7, 3) into tResult -- -7 mod 3 = 2  
put tResult
```

the resultant output is:

```
1  
2  
0  
2
```

Version 58

Round and Trunc functions.

(For those times where you need a round trunk... I'm joking):

```
put round(3.7) into tResult -- rounds to 4  
put tResult
```

```
put round(3.2) into tResult -- rounds to 3  
put tResult
```

```
put round(-3.7) into tResult -- rounds to -4  
put tResult
```

```
put 3.7 into tNumber  
put round(tNumber) into tResult -- rounds to 4  
put tResult
```

```
put trunc(3.7) into tResult -- truncates to 3  
put tResult
```

```
put trunc(-3.7) into tResult -- truncates to -3  
put tResult
```

```
put 3.7 into tNumber  
put trunc(tNumber) into tResult -- truncates to 3  
put tResult
```

the result is:

```
4  
3  
-4  
4  
3  
-3  
3
```

Version 59

Added the ability to play ogg sounds:

```
play "beep.ogg"  
put "beep.ogg" into tSound  
play tSound  
play "beep.ogg" until done
```

Version 60

Added musical note playing:

```
play "c d e f g" tempo 120
wait 1 second
play "c4 d4 e4" tempo 60
wait 1 second
play "c e g" tempo 240 until done
```

Version 61

Added the ability to play sounds at different tempos (ogg files)

```
play "boing.ogg" tempo 60
wait 1 second
play "boing.ogg" tempo 240
wait 1 second
play "boing.ogg" tempo 120 until done
```

Version 62

added the factors:

put the factors of 1000

outputs:

1,2,4,5,8,10,20,25,40,50,100,125,200,250,500,1000

Version 63

made checkboxes on the testing page, so that we don't have to sit through sounds and dialogs every time we (I) test a new feature.

Version 64

added the mousseloc feature:

put the mousseloc
returns the position of the mouse x and y

tested with:

```
repeat 10
  put the mousseloc
  wait 1 second
end repeat
```

Version 65

Added the mouseH and the mouseV

put the mouseH
put the mouseV

Version 66

added "the platform", as in:

get the platform
or
put the platform

returns output:

"MacOS"
"Windows"
"Linux"
"Android"
"iOS"
"unknown"

Version 67

added the screenrect:

put the screenrect

returns:

0,24,1920,1056

(the area minus any panels that might be at the top of the screen) - this way you know the height of the menubar too.

Version 68

added browser identification, so we know what browser is running the implementation:

put the browsername
put the browserversion

returns (for example):

Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/20100101 Firefox/135.0
135.0

Version 69

added a check for screenorientation:

put the screenorientation

returns either:

landscape or portrait

Version 70

Added touchscreen check:

put the hasTouchscreen

returns either true or false, depending on if the device you are running the browser on has a touchscreen or not.

Version 71

The interpreter now records the last clicked position the user made. We can reference that in script using:

put the lastclickloc
put the lastclickh
put the lastclickv

it would return (for example):

225,577
225
577

Version 72

another handy feature I thought to add, was the detection of the latitude and longitude.

put the latlon

if the user hasn't consented to allowing the browser to get their location, this will return:

unavailable

if it has access though, it would return:

25.0, -71.0

(in this example, I use the middle of the Bermuda Triangle, rather than show my location).

Version 73

added get the title, and, set the title.

This is so we can change the name of the page shown in the titlebar of the browser.

The syntax is:

get the title

put "Test Title" into testTitle
set the title to testTitle

Version 74

added the itemdelimiter function, as in:

set the itemdelimiter to ","
put item 1 of the date
set the itemdelimiter to "/"
put item 1 of the date

the first line would output the entire date string because my itemdelimiter is set to "," which is the default.

The second time I do it, only the first item is output because I set the itemdelimiter to a "/" which is what the date is containing. (if that makes sense).

Version 75

added support for "the seconds" as in:

put the seconds
if the seconds > 1739184668 then put "seconds are more than 10:51 on the date 10/2/2025"

Version 76

add support for the ticks:

put the ticks
put the ticks > 0
put "Start"
wait 60 ticks
put "One second later"
wait 1 tick
put "Plus one tick"

you can also use
wait 1 jiffy
which is interchangeable with the word tick

Version 77

Oops! forgot to add the exit repeat function!

now works. Tested with:

```
put "Start"
put 1 into x
repeat until x > 5
  put x
  exit repeat
  put x + 1 into x
end repeat
put "Done"

put "Start"
repeat 5 times
  put "In loop"
  exit repeat
  put "Should not see this"
end repeat
put "Done"
```

Version 78

added the isnumber function:

```
put isNumber("20") -- returns true
put isNumber("a") -- returns false
```

Version 79

Added uppercase and lowercase detection of a character:

```
put "Wed" into tString
get isUpper(char 2 of tString)
put it

put isUpper(char 1 of the long date)
```

(returns true if a character is a capital, false if not)

Version 80

Added the ability to set the clipboard (text) using script:

```
put "Hello World" into tString
set the clipboardText to tString
```

or:

```
set the clipboardText to "Test clipboard"
```

Version 81

added in the return command:

```
put "this is " & return & "my text" into tVar
```

```
put tVar
put "A$B" into tVar2
replace "$" with return in tVar2
put tVar2
put "line1" & return & "line2" & return & "line3" into tVar3
put tVar3
```

Output

```
this is
my text
A
B
line1
line2
line3
```

Version 82

Added the paragraph count feature:

```
put "this is a piece of text" & return & "another piece of text" into tParagraph
put the number of paragraphs in tParagraph
```

```
put "single paragraph" into tSingle
put the number of paragraphs in tSingle
```

```
put "p1" & return & "p2" & return & "p3" into tThree
put the number of paragraphs in tThree
```

would output:

```
2
1
3
```

Version 83

Changed the itemdelimiter a bit.

This was so I could type:

```
set the itemdelimiter to slash
put "a/b/c" into tVar
put item 1 of tVar
```

If I wanted to set the itemdelimiter to a slash (between quotes), I also ran this test:

```
set the itemdelimiter to "/"
put "x/y/z" into tVar
put item 1 of tVar
```

this outputs (as expected):

```
a -- in the first example
x -- in the last example
```

Version 84

Added the ability to set the itemdelimiter to comma (both ways)

```
set the itemdelimiter to comma
put "a,b,c" into tVar
put item 3 of tVar
```


output: c

```
set the itemdelimiter to ","
put "a,b,c" into tVar
put item b of tVar
```

output: b

Version 85

You can now use "ask password" with the following example:

```
ask password "Complete registration:" with "username,email,password,confirm password"
put it
```

returns a comma separated list with the results of each field the user typed into.

Version 86

added the "ask list" option:

```
put "item 1" & return & "item 2" & return & "item 3" into treturndelimitedList
ask list "Please pick an item:" with treturndelimitedList
put it
```

The would return whatever the user chose from that list into the it variable

Version 87

Added the average function, and tried to make this really robust:

Testing:

```
--Average of two numbers: success
get average(12,23)
put it
```

```
--Average direct output: success
put average(34,23)
```

```
--Average into variable: success
put average(32,43,23) into tAverageValue
put tAverageValue
```

```
--Average with multiple numbers: success
put average(4,36,45,50,75)
```

```
--Average with variables: success
put 10 into tNum1
put 20 into tNum2
put average(tNum1,tNum2)
```

```
--Average with expressions: success
put average(5+5, 10*2)
put average(2*3, 4+5, 10/2)
```

The output of running the above would be:

18

29
33
42
15
15
7

Version 88

Added the charToNum and numToChar feature:

```
put charToNum("A")  
put numToChar("65")
```

Output:

65
A

Version 89

added the exp function:

```
put exp(1)
```

Returns the natural exponential of a number. (Yeah, I copied and pasted that explanation!)

Version 90

added sum function:

```
put sum(1,2,3,4) into tResult  
put tResult  
put tResult = 10
```

Output:

10
true

Version 91

Added stripList function.

What this does:

Suppose I had the script:

```
put "/home/tom/Desktop/sort/PPC-logo.png" & return &  
"/home/tom/Desktop/sort/screenshot.png" into tFilesList
```

So now, the contents of the variable tFilesList is:

```
/home/tom/Desktop/sort/PPC-logo.png  
/home/tom/Desktop/sort/screenshot.png
```

Rather than having to use repeat loops on the variable where we have to do something like:

```
put 0 into tCount  
repeat the number of lines in tFilesList  
  add 1 to tCount  
  delete char 7 to 18 of line tCount of tFilesList  
end repeat  
put tFilesList
```

This takes a long time if you have a variable with several hundred lines (or more). That's where the stripList function comes in:

```
put stripList(tFilesList,7,18) into tStrippedList
put tStrippedList
```

Version 92

added 'the value' as in:

```
put the value of 45 + 12
put the value of (5 * 3) + 2
put the value of (5 * (3 + 2))
```

output:

```
57
17
25
```

Version 93

added the beginnings of GUI objects.

I'm starting with a simple button.

You can now use:

```
create button "test" -- creates a button named "test" in the middle of the card area
delete button "test" -- deletes a button named "test" if it exists
```

Version 94

added the howmany function, as in:

```
put "a" into tSearch
put "A string containing a few a characters." into tString
put howmany(tSearch,tString)
```

output:

```
5
```

Version 95

The testing page now reports the result of each test properly underneath each test.

Version 96

added handler support (on and end):

```
on handler1
  put "handler1 called"
handler2
end handler1
```

```
on handler2
  put "handler2 called"
end handler2
```

handler1

outputs:

```
handler1 called
handler2 called
```

however, there's an exception to this:

```
repeat -- we don't need an 'on repeat' here.  
  put "do something"  
  exit repeat  
end repeat
```

repeat statements don't need an "on repeat" to begin, so we ignore the need for an on repeat.

Version 97

I've added the Logarithm function and tests for this in testing2.html

As in:

```
put log(2.718) into tResult -- natural log of e ≈ 1  
put tResult  
put log(1) into tResult -- natural log of 1 = 0  
put tResult  
put 10 into tNumber  
put log(tNumber) into tResult -- natural log of 10  
put tResult  
put log(100) into tResult -- natural log of 100  
put tResult
```

returns:

0.999896315728952 0 2.302585092994046 4.605170185988092

Logarithm with base function test (Testing logarithm with base function with literal numbers and variables)passed

```
put log(100, 10) into tResult -- log base 10 of 100 = 2  
put tResult  
put log(8, 2) into tResult -- log base 2 of 8 = 3  
put tResult  
put 16 into tNumber  
put 2 into tBase  
put log(tNumber, tBase) into tResult -- log base 2 of 16 = 4  
put tResult  
put log(10, 10) into tResult -- log base 10 of 10 = 1  
put tResult  
put log(1, 10) into tResult -- log base 10 of 1 = 0  
put tResult
```

returns:

2 3 4 1 0

Then I thought I'd try and trip it up on purpose:

```
put log(0) into tResult
```

result returned is:

Correctly caught error: log: argument must be a positive number

What about if I use a negative number?

```
put log(-1) into tResult
```

result returned is:

Correctly caught error: log: argument must be a positive number

Now, what happens if I try and trip it up with multiple items?

```
put log(10, 0) into tResult
```

returned result is:

Correctly caught error: log: base must be a positive number not equal to 1

I also fixed the display of the results in the "testing 2" page, because it wasn't showing the syntax. (I'd forgotten to implement the <pre> html element. Oops!

Version 98

Added a checkbox "☒ clear after run" on the main index page.

This is so you can optionally have the message box clear it's text when you run a command.

I've fixed an issue with duplicate output lines.

For example, if I ran:

```
put the time
```

I might get:

```
7:37 AM
```

```
7:37 AM
```

The problem was that when executing a "put" command, the interpreter was sending the output through the outputHandler function - but also returning the same value back to main.js, which was then displaying it again (oops!)

Version 99

Added the ability to send a handler in <time> (pendingmessages)

As in:

```
on testA
  put "A"
end testA
```

```
on testB
  put "B"
end testB
```

I could then run:

```
send testA to me in 1 second
send testB to me in 2 seconds
put "This appears immediately"
```

Output is:

This appears immediately -- *self explanatory, appears immediately as you'd expect*
A -- *appears in message box after 1 second*
B -- *appears in message box after 2 seconds*

Reversing this, I could also use:

```
send testA to me in 1 second
send testB to me in 2 milliseconds
put "This appears immediately"
```

The output is:

This appears immediately -- *self explanatory, appears immediately as you'd expect*
B -- *appears in message box after 2 milliseconds (so, first)*
A -- *appears in message box after 1 second*

Version 100

Thank you to Paul McClernan (OpenXTalk Paul) for this.

<https://github.com/PaulMcClernan>

He's very kindly added the appearance function, as in:

put the appearance

returns:

dark / light

(depending on what system appearance the browser is set to use)

Version 101

Added a script editor!

If you create a button object with:

create button "test"

You can now right-click on a button object to edit it's script.

Or, you can also use:

edit the script of button "test"

Both will show the script editor now.

I've also allowed this to be styled in css with the file "script-editor.css"

Version 102

I noticed the mouseup message wasn't being handled properly. Neither did buttons know about other things (mousedown, mouseenter, mouseleave, mousewithin) so I've just added those:

```
on mouseDown
  put "Mouse button pressed down"
end mouseDown
```

```
on mouseUp
  put "Mouse button released"
end mouseUp
```

```
on mouseEnter
  put "Mouse entered the button"
end mouseEnter
```

```
on mouseLeave
  put "Mouse left the button"
end mouseLeave
```

```
on mouseWithin
  put "Mouse is within the button"
```

```
end mouseWithin
```

These would now put the corresponding messages in the "message box" as you'd expect.

Version 103

Paul McClernan (OpenXTalk Paul), has very kindly added the backdrop property.
As in, you can get and set the backdrop colour to a different one.

I also expanded upon this slightly: so now the interpreter also looks at "col-references.css".
You can now get the backdrop colour and store this in a variable:

```
put the backdrop into tOriginalBackdrop
```

You can also set the backdrop in a variety of ways:

```
set the backdrop to darkGreen
set the backdrop to #FF5500
set the backdrop to 200,100,50
set the backdrop to tOriginalBackdrop
```

Version 104

I added speech support via the WebSpeech API, and have devised this test for it:

```
put the speechVoices into tVoices
if tVoices is empty or tVoices contains "No speech" then
  put "No voices available"
else
  put "Available voices:" & return & tVoices
  put the speechVoice into tOriginalVoice
  put "Original voice: " & tOriginalVoice
  set the speechVoice to "Default"
  speak "This is a test of the speech synthesis feature"
  set the speechVoice to tOriginalVoice
end if
```

I also added this to the "testing2.html" page.

However, this shows that speech support (depending on distro) is lagging behind compared to Windows or MacOS. Many distros show:

```
No voices available
```

Version 105

Implemented the "loc" property (location for objects).

So, I tested this with:

```
create button "test"
put the loc of button "test" -- Returns the x,y position of the button's center
set the loc of button "test" to "200,200" -- Positions button with its center at 200,200
```

I also began work on tool modes, as it seemed a necessary thing when messing about with objects. We can now run:

```
choose browse tool -- the default, where you click and interact with objects
choose pointer tool -- akin to edit mode, where you drag objects around the card
```

As well as "choose [x] tool", you can set the mode instead (which runs the same command):

```
set the mode to browse -- same as "choose browse tool"  
set the mode to edit -- same as "choose pointer tool"
```

I did some more work to object logic. For example, setting the script of our newly created button above to:

```
on mouseEnter  
  set the top of me to 20  
  set the left of me to 20  
end mouseEnter  
  
on mousedown  
  set the top of me to 100  
  set the left of me to 400  
end mousedown
```

As you'd expect, that now moves the button around the card upon the mouse entering it, and when it's clicked. (It understands the context of what "me" is) - which was actually quite hard.

Added a checkbox in the bottom left of the script editor that allows you to toggle autocomplete on/off. The checkbox is unchecked by default, so autocomplete suggestions won't appear until this is turned on. When enabled, eventually the autocomplete will suggest keywords and functions as you type. You'll be able to navigate the suggestions using arrow keys and select with Tab or Enter.

I also fixed a visual issue with the script editor, as it had an overprinting text layer.

Version 106

As mentioned by Paul McClernan, I corrected the platform function to also accept platform()

I added tests for this in testing2.html

```
put the platform into tPlatformProperty  
put platform() into tPlatformFunction  
put "Platform property: " & tPlatformProperty  
put "Platform function: " & tPlatformFunction  
put tPlatformProperty = tPlatformFunction into tAreEqual  
put "Are equal: " & tAreEqual -- returns "true" if all goes to plan
```

Version 107

I wanted the interpreter to allow me to be a bit more "fuzzy" when I reference lines of a string:

```
put "apple" & return & "banana orange" & return & "cherry grape kiwi" into tString  
put word 1 of line 3 of tString into tResult -- two line conditions at the same time  
put tResult
```

outputs:
cherry

```
put word 2 of line 2 of tString into tResult -- two line conditions at the same time  
put tResult
```

outputs:
orange

```
put word 3 of line 3 of tString into tResult
```



```
put tResult
```

outputs:

kiwi

So, then I remembered that HyperTalk allows you to use first, second, third etc - so added this:

```
put "dog" & return & "cat mouse" & return & "fish bird snake" into tString
put first word of third line of tString into tResult -- phonetically referencing
put tResult
put second word of third line of tString into tResult
put tResult
put first word of second line of tString into tResult
put tResult
```

this would output:

fish
bird
cat

Then I implemented "last" keyword for lines in strings:

```
put "cat" & return & "snake dog chicken" & return & "elderly people" into tString
put first word of first line of tString into tResult
put tResult
put second word of second line of tString into tResult
put tResult
put last line of tString into tResult -- now knows about "last line" of...
put tResult
```

outputs:

cat
dog
elderly people

Version 108

Added the backgroundColor (and synonym backgroundColour):

```
create button "test"
set the backgroundColor of button "test" to "255,0,0" -- set background RGB values
set the backgroundColour of button "test" to "Green" -- background colour, named ref
put the backgroundColour of button "test"
```

So it also made sense to add the foregroundColor (and synonym foregroundColour) at the same time, as in:

```
create button "test"
set the foregroundColor of button "test" to "255,0,0" -- set foreground RGB values
set the foregroundColor of button "test" to "Red" -- Named color
set the foregroundColour of button "test" to "Blue" -- UK-English colour, named ref
get the foregroundColor of button "test" -- we can get using US spelling of property
get the foregroundColour of button "test" -- we can get using UK spelling of property
```

Version 109

Added a new handler: mouseupoutside

What this does, imagine we create a new button using the following script:

```
create button "test"
```

Then we set the script of that button to:

```
on mousedown
    set the backgroundcolour of me to red
    set the foregroundcolour of me to white
end mousedown

on mouseup
    set the backgroundcolour of me to white
    set the foregroundcolour of me to red
end mouseup
```

When the user clicks their mouse **down** on the button - it turns red with white text.

When the user **releases** their mouse on the button - it turns white with red text.

All good so far. But, what if the user clicks their mouse down on the button, moves their mouse off the button then releases?

That's where mouseupoutside comes in:

```
on mouseupoutside
    send mouseup to me -- loops back to the standard mouseup in this example
    -- answer "you released the mouse outside the button!" -- commented out
    -- uncomment the above line to warn the user
end mouseupoutside
```

Version 110

Added the getting/setting/putting of custom properties of objects.

As in:

```
create button "test"
set the tProp of button "test" to "something"
```

Then in the script of button "test":

```
on mousedown
    answer the tProp of me
end mousedown
```

output: displays answer dialog with the string "something" because it retrieved the custom property of the button object.

Version 111

Added the field object!

```
create field "test"
put "testing" into line 1 of field "test"
```

So far, I've implemented the following field properties (some are in preparation for an inspector)

```
put the textalign of field "test"
put the scrollable of field "test"
put the locktext of field "test"
put the multiline of field "test"
put the autotab of field "test"
```

put the widemargins of field "test"
put the sharedtext of field "test"
put the textcolour of field "test"
put the backgroundcolour of field "test"
put line 1 of field "test"
set the rotation of field "test" to 10
set the angle of field "test" to 15
set the textsize of field "test" to 45

Just to go into some of these properties in greater detail:

textalign -- *Controls the text alignment in the field*
We can use 'left', 'center', 'right', 'justify'. The default is: 'left'

scrollable -- *Determines if the field can be scrolled*
We can use: true, false - the default: true

locktext -- *Controls whether the text in the field can be edited*
We toggle with: true, false - default is: false

multiline -- *Determines if the field can have multiple lines of text*
Values are: true, false - the default is: true

autotab -- *Controls automatic tabbing behaviour*
Values are: true, false - The default: false (although may be overruled by browser)

widemargins -- *Controls the padding inside the field*
Values: true, false. Default setting is: false
(When true, padding is 10px; when false, padding is 5px)

sharedtext -- *Determines if the text is shared across cards*
Values are: true, false - Default is not shared: false

left -- *The left position of the field in pixels*
top -- *The top position of the field in pixels*
width -- *The width of the field in pixels*
height -- *The height of the field in pixels*
loc or location -- *The center position of the field as "x,y"*

textcolor / textcolour / foregroundcolor / foregroundcolour -- *The colour of the text*
(default is empty / null) -- *uses browser inherited colour*

backgroundcolor / backgroundcolour -- *The background colour of the field*
(default is empty / null)

text / content - The text content of the field, As in:
put "testing" into line 1 of field "test"
set the text of fld "test" to "another"
set the content of fld "test" to "example"

edit the script of field "test" -- *The script associated with the field*

set the rotation of field "test" to 10 -- *the rotation / angle of the field*
set the angle of field "test" to 15 -- *you can use angle as a synonym of rotation*

set the textsize of field "test" to 45 -- *change the size of text in the field.*

Version 112

Added "the name of me" for objects. As in:
put the name of me

On my to-do list next: I've got to also add the setting of the name of me, or the setting of the name for <object> too.

(Object renaming, which also needs to update the DOM and the WebObjects when a name change occurs, as this is used to track what's on the card)

I also cleaned up the script editor. It now gets the indentation correct for else if conditions.

I added a "format" button, in case you paste text in from somewhere else, you can get it to tidy up the formatting with this button.

I do want to get the formatting of what's between quotes correct too, but I'm getting there.

Version 113

Added the acos math functions to the interpreter (Thanks **TerryL**).

I've also reorganised the entire DOM and WebObjects map so they work with IDs as well as object names (in tandem).

Also added the renaming function, as in:

```
set the name of button "test" to "renamed"  
set the name of btn id 2 to "renamed again"
```

Or in the script of an object:

```
on mousedown  
  set the name of me to "blah"  
end mousedown
```

You can now also get by ID in the script of the object itself too:

```
on mouseup  
  put the id of me into tID  
  put tID  
end mouseup
```

...and rename by ID too:

```
on mouseup  
  put the id of me into tID  
  set the name of button id tID to "new"  
end mouseup
```

Boring bit:

Worth mentioning; when a button is renamed, the event handlers remained attached to the original DOM element, but the script lookup used the new name. This was the: **executeObjectScript** method in **interpreter.js**. Now, when the user renames a button via script, it updates the DOM element's dataset and the objects map, but the event handlers still referenced the original name that was used when the button was created. I then modified the event handlers to use the updated reference, to ensure mouse events keep working on the renamed object.

I did extensive testing as I was trying to confuse it, but seems to pass all these tests:

```
create button "test"
put the id of button "test" into myID
set the name of btn id myID to "renamed"
set the width of btn id myID to 400
set the name of btn id myID to "original"
put the name of btn id myID
```

Version 114

Added the ceiling (and ceil synonym) function, as in this example:

```
put ceiling(33.25) into tResult -- outputs 34
put ceil(12.9) into tResult -- outputs 13
```

If I try and trip it up:

```
put ceiling("abc")
```

returns:

Error: ceiling: argument must be a number

Version 115

More tweaks to the script editor (I needed a rest from adding math(s) functions).

Now, if I put this in a button:

```
on mouseup
  put "this is a test" into tstring
  put tstring
  -- yes, this is now looking a bit more promising!
end mouseup
```

It's actually formatted correctly with the correct colours, it knows how to handle strings and comments properly too.

Version 116

Added the beep command. You can now use:

```
beep -- play an audible beep once
beep 4 -- beeps 4 times
```

Added floor math function.

```
put floor(33.25) -- outputs 33
put floor(12.9) -- outputs 12
put floor(-3.7) -- outputs -4
```

Added geometric mean function.

```
put geometricMean(10,20,25) -- outputs 17.1
put geometricMean(4,9) -- outputs 6
```

Added harmonic mean function.

```
put harmonicMean(10,20,25) -- outputs 15.79
put harmonicMean(4,9) -- outputs 5.54
```

Added median function

```
put median(10,20,30) -- outputs 20
put median(10,20,30,40) -- outputs 25
put median(5,1,3,2,4) -- outputs 3
```

Added max function

```
put max(10,20,30) -- outputs 30
put max(42,17,29) -- outputs 42
put max(-5,-10,-3) -- outputs -3
```

added min function

```
put min(10,20,30) -- outputs 10
put min(42,17,29) -- outputs 17
put min(-5,-10,-3) -- outputs -10
```

Added round and statround functions

```
put round(12.5) -- outputs 13
put round(12.4) -- outputs 12
put round(-3.7) -- outputs -4
put statRound(10.5) -- outputs 10
put statRound(11.5) -- outputs 12
put statRound(10.1) -- outputs 10
```

Version 117

Added a new section to the top of each testing page.

This now tracks the number of tests that are failed, passed, and skipped.

If any tests fail, it'll put these at the top of each testing page (sorted alphabetically).

This is useful when adding/testing new functions.

Test Summary			
Total Tests:	Passed:	Failed:	Skipped:
224	212	0	12

I also added a "testing3.html" page, so that can be used as a blank template in future if we want more test pages.

Version 118

(lots of changes here)

Firstly, these can all be found in "testing3.html"

I split them up to make it easier to manage.

Because the interpreter was getting a bit large, I added an extra "extended-functions.js" which is called as an extension of the interpreter now. Just means that as the file grows in size, we can add extras coming off it. (I was getting too confused by the size of the "interpreter.js" and it was getting difficult to make changes).

Mentioning that, I've been busy adding the following functions:

format	split
baseConvert	join
pi	base64Encode
variance	base64Decode
standardDeviation	md5Digest
union	sha1Digest
intersection	

Examples:

Format function

```
put format(123.456, "#.#") into tResult
put tResult -- outputs 123.5
```

BaseConvert function -- (Convert decimal 255 to hexadecimal)

```
put baseConvert(255, 10, 16) into tResult
put tResult -- would output ff
```

Pi function (Get the value of pi)

```
put pi() into tResult
put tResult -- outputs 3.141592653589793
```

Variance function - (Calculate variance of 1,2,3,4,5)

```
put variance("1,2,3,4,5") into tResult
put tResult -- outputs 2
```

Variance function - (Calculate variance using variable)

```
put "1,2,3,4,5" into tNumbers
put variance(tNumbers) into tResult
put tResult -- outputs 2
```

StandardDeviation function - (Calculate standard deviation of 1,2,3,4,5)

```
put standardDeviation("1,2,3,4,5") into tResult
put tResult -- outputs 1.4142135623730951
```

Union function example - using variables as input

```
put "a,b,c" into tList1
put "c,d,e" into tList2
put union(tList1, tList2) into tResult
put tResult -- outputs a,b,c,d,e
```

Intersection function - (Intersection example)

```
put "a,b,c" into tList1
put "c,d,e" into tList2
put intersection(tList1, tList2) into tResult
put tResult -- outputs c
```

Split function - (Split string by delimiter)

```
put split("a-b-c", "-") into tResult
put tResult -- outputs a,b,c
```

Join function - (Join list with delimiter)

```
put "a,b,c" into tList1
put "-" into tDelimiter -- new delimiter to put between items of tList1
put join(tList1, tDelimiter) into tResult
put tResult -- outputs a-b-c
```

Base64Encode function - (Encode string to base64)
put base64Encode("Hello, World!") into tResult
put tResult -- *returns SGVsbG8sIFdvcmxkIQ==*

Base64Decode function - (Decode base64 to string)
put base64Decode("SGVsbG8sIFdvcmxkIQ==") into tResult
put tResult -- *returns Hello, World!*

MD5Digest function - (Calculate MD5 hash)
put md5Digest("Hello, World!") into tResult
put tResult
-- *outputs: 06050a080e02070d080807090208030803010b0606040b0d080b070f000a0d04*

SHA1Digest function - (Calculate SHA1 hash)
put sha1Digest("Hello, World!") into tResult
put tResult -- *returns 0a0a9f2a6772942557ab5355-28950bbe-709a1ff*

Just a further note about the additional "extended-functions.js" file:

I modified the prototype of the InterpreterClass. This makes sure that wherever I include the interpreter, including the one in in main.js, it'll have access to the extended functions.
I've also added a transparent retry check that will attempt to extend the InterpreterClass multiple times if it's not available. (this silently prevents errors, and I thought it was better than adding any wait commands)
This support situations where the script loading order might not be right for example.
This means the interpreter now uses the original evaluateExpression as a *function reference* instead of a *class inheritance*. This should be more reliable for edits and changes when you want to see it in realtime.

Version 119

Just a small change. See "extended-functions.js", and my comment:

// Try to extend immediately

I've modified the timing of the extension mechanism (10 milliseconds), so it tries a lot harder to load the extended functions, and does not give up as readily (whereas it was 10% determined to load this, it's now 80% determined to do so).

Version 120

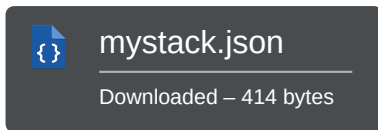
Added the userAgent (thanks **Paul McClellan**), so you can use:

put the userAgent -- *returns the browser's user agent string*
put the browserName -- *does the same as above*

Version 121

Added the ability for the user to save their stacks with:
save stack as "mystack"

This writes the current stack to a JSON file, in this example:
mystack.json



```
{
  "version": "1.0",
  "timestamp": "2025-03-22T10:37:51.336Z",
  "objects": [
    {
      "name": "testing",
      "id": "1",
      "type": "button",
      "position": {
        "left": "50px",
        "top": "50px",
        "width": "100px",
        "height": "30px"
      },
      "script": "on mouseup\n  answer \"I want to test if this gets stored.\"\n  \nend mouseup",
      "properties": {}
    }
  ]
}
```

Version 121

To go along with our saving command above, I've also added the load stack function.

The filename can be specified with or without quotes. If the interpreter can't find it (usually due to browser restrictions), it'll prompt the user to choose a file instead.

Before loading, the user is prompted with a confirmation dialog:

"Are you sure you want to load the stack from file [filename]? This will replace any stack content you currently have."

There are two buttons shown: "Yes, load" and "No, Cancel" -- I tried to make this as clear as possible to avoid any confusion (rather than just a simple yes or no).

If "No, Cancel" is selected, the load process is cancelled and nothing should be cleared from the current stack.

Assuming we chose "Yes, load", when a JSON file is selected (hopefully one we made earlier), the interpreter reads and tries to make sense of the JSON data and validates its structure.

All existing objects are cleared from the card, all event listeners are properly removed to prevent memory leaks.

Each object from the saved stack is recreated with the same:

Type (button, field, etc.), name, position and size, script content, any custom properties, any field content (for field objects if applicable)

I've also added some error handling when parsing the JSON structure. It should result in detailed error messages being displayed to the user.

Version 123

I've added the `ByteOffset` function, and added a few test cases to support it:

ByteOffset function - (Find position of 'c' in 'abcde')

```
put byteOffset("c", "abcde") into tResult
put tResult -- returns 3
```

ByteOffset function - with skip parameter (Find position of 'b' in 'abacadabra' starting after 2 characters)

```
put byteOffset("b", "abacadabra", 2) into tResult
put tResult -- returns 6
```

ByteOffset function - substring in string (Find position of 'bark' in 'embarking')

```
put byteOffset("bark", "embarking") into tResult
put tResult -- returns 3
```

ByteOffset function - not found (Character not found in string)

```
put byteOffset("z", "abcde") into tResult
put tResult -- returns 0 as character not found in string
```

ByteOffset function - negative skip count

```
put byteOffset("c", "abcde", -1) into tResult
put tResult -- returns expected error: byteOffset: skip count must be non-negative
```

Version 124

`create fld "test"` -- now understands 'fld' as a shorthand for "field"

Corrected this function:

`put "potatoes" into me` -- previously didn't work if used in the script of an object.

This is now fixed and would put "potatoes" into a field for example.

`put "test" into line 2 of field "test"` -- this now works properly (multi-line)

Changes to rotation or angle function:

```
on mousedown
  put the angle of fld a +1 into tAngle
  set the angle of fld "a" to tAngle at 20,20 -- set hinge at 20,20 px of the object
  -- this is the exact point from the topleft, where the object will rotate around
end mousedown
```

alternatively:

```
on mousedown
  put the angle of fld a +1 into tAngle
  set the angle of fld "a" to tAngle -- default, hinge point now middle of the object
end mousedown
```

`put the topleft of fld "test"` -- previously didn't work (neither did topright, bottomleft, bottomright).

These are now reported correctly. You should be able to use any of these such as:

```
set the topleft of fld "test" to 40,20
set the topright of fld "test" to 50,10
set the bottomright of fld "test" to 400,200
set the bottomleft of fld "test" to 10,100
```

Added intersect for objects:

```
put intersect(button "a", button "b") -- returns true or false, depending if they touch
```

Added 'special character' by name reference. For example, if I type:

```
put slash -- should return "/" without quotes
put space -- should return " " without quotes
put QUOTE -- should return a quote character
put comma -- should return a "," without quotes
```

Added some card scripting:

```
on mousedown
  set the backgroundcolor of this card to "yellow" -- in button script
end mousedown
```

```
on opencard
  -- implemented the 'on opencard' message handler
  answer "Thank you for trying out my stack!"
end opencard
```

Version 125

Added support for:

```
create btn "btnName" -- we could only use "create button..." previously.
```

Added the start of graphic objects:

```
create graphic "lineA"
set the points of graphic "lineA" to 40,40,160,160
set the lineSize of graphic "lineA" to 5
```

In case anyone is wondering, I'm sticking to SVG drawing conventions here, so the JSON of this in the saved stack looks like:

```
"attributes": {
  "x1": 40,
  "y1": 40,
  "x2": 160,
  "y2": 160,
  "stroke": "black",
  "stroke-width": 5
}
```

Version 126

```
create grc "test" -- could only use "create graphic..." previously
set the showborder of grc "test" to false -- turns off border around graphics (default)
put the points of graphic "test" -- supports returning the points of a graphic
```

Now into some real 'fun' curve calculations:

Simple straight line from (10,10) to (200,100)

```
set the points of graphic "test" to "10,10,200,100"
```

Quadratic Bezier curve

```
-- M = starting point (x,y)
-- Q = control point (x,y), end point (x,y)
set the points of graphic "test" to "M20,20 Q100,10 180,100"
```

```
-- Another example with different control point
set the points of graphic "test" to "M50,50 Q150,10 200,150"
```

Cubic Bezier curve

```
-- M = starting point (x,y)
-- C = control point 1 (x,y), control point 2 (x,y), end point (x,y)
set the points of graphic "test" to "M20,20 C50,10 150,10 180,100"
```

S-curve example

```
set the points of graphic "test" to "M20,100 C60,20 140,180 180,100"
```

Arc curve

```
-- M = starting point (x,y)
-- A = rx,ry rotation large-arc-flag,sweep-flag end-point-x,end-point-y
-- Parameters: rx,ry = radius x,y; rotation = angle in degrees;
-- large-arc-flag = 0 for small arc, 1 for large arc;
-- sweep-flag = 0 for counter-clockwise, 1 for clockwise
set the points of graphic "test" to "M20,100 A80,50 0 1,1 180,100"
```

Half circle example

```
set the points of graphic "test" to "M20,100 A80,80 0 1,0 180,100"
```

Path with multiple segments (line + curve)

```
-- L = line to x,y
set the points of graphic "test" to "M20,20 L100,20 C120,20 180,50 180,100"
```

Closed path (adds Z at the end to close the path)

```
set the points of graphic "test" to "M20,20 L100,20 C120,20 180,50 180,100 Z"
```

Path with smooth curve transition

```
-- S = smooth cubic bezier (uses previous control point reflection)
set the points of graphic "test" to "M20,100 C60,20 60,180 100,100 S140,20 180,100"
```

T = smooth quadratic bezier

```
set the points of graphic "test" to "M20,100 Q60,20 100,100 T180,100"
```

-- one last thing!

```
set the points of graphic "test" to "M20,100 L100,20 L180,100 Z" -- We use a closed path
here and draw a triangle. Because the shape is closed, how do we fill the inside?
```

Set a red fill (inside closed shapes with Z set)

```
set the fill of grc "test" to "red"
```

License agreement:

This license agreement MUST accompany the "webtalk" software.

Rather than put you through a long and boring license agreement, here's an easier to read version of the unique license agreement:

What you can do:

- Use it without registering
- Use it without paying anything each month.
- Use it in any free projects you create.
- Improve it
- Feel free to redistribute it as long as this agreement is also included.
- Feel free to include with Linux live distros.
- Check out openxtalk.org for more.

What you can't do:

- Do not sell, market, or re-sell this software
- Do not fork this software to subvert the agreement
- If you are in any way affiliated with "Livecode":
 1. Delete all copies of this software you obtained.
 2. Do not use any part of this software.
 3. Do not copy any of the code for your own use.
 4. Do not pass [in full / in part] to Livecode testers.
 5. Do not use internally within your company[ies].
 6. Persons working on behalf of Livecode's interests are prohibited from using, decompiling or reverse-engineering this project in any way.
- Do not modify this documentation
- Do not falsify the origin of this software

If you are in any doubt over any particular licensing issues, feel free to visit openxtalk.org/forum for further clarification. By using the "webtalk" software, you agree to the points raised above. These terms may/may not change in future releases.

Other footnotes:

Webtalk is loosely based on the HyperTalk scripting language, and the goal is to be compatible in most major respects. There are additions to it and perhaps things missing from it. The boring bit is I'm not implying it's fit for any particular purpose, and there's no guarantees implied or otherwise about how it might run on your system. (Being based on Javascript and CSS, it heavily relies on what your browser is capable of).

It is being created over at the openxtalk.org website, mainly by myself (Tom Perry [*tperry2x*]), but also with help and suggestions from a small selection of interested users.