JAGL Documentation

Types

Jagl has four basic types: Numeric, Array, Block, and Function

Numeric

Numeric types can contain either floating point or integer numbers, and have a variety of syntaxes:

Decimal: *1, 1.0, -.4, 2.*

Scientific: *1e6, 1.8e-8*

Octal: *70o*

Hexadecimal: *8Fx*

Arrays

Arrays can contain any combination of types, and are created using normal brackets (with each item separated by a space, if needed). Example:

```
(1 2 (3 4 5)(6 7) 8 "String" {2+})
```

Strings are represented by an array of their ASCII values (Unicode is not yet supported). Strings have a syntactic sugar for definition, using quotes for normal strings, and ticks for strings containing escape characters. Example:

```
"string\n" -> (115 116 114 105 110 103 92 110) or (s t r i n g \\ n)
'string\n' -> (115 116 114 105 110 103 10)     or (s t r i n g \n)
```

Blocks

The Jagl blocks are analogous with Golfscript blocks, in that they contain a certain ordered list of tokens which can be evaluated or manipulated at a later point. Their definition is with braces:

```
{2 4+5*}
```

Functions

Functions are simply that, a set of operations to be done on the stack. In Jagl, almost every ASCII character is linked to a function which is why this documentation was created.

Function Reference

The following table contains all of the available functions in Jagl. You will see that there are headers for "Argument x", so here is a quick legend:

```
 BACK OF STACK (ARGUMENT1 ARGUMENT2 ARGUMENT3 FUNCTION) FRONT OF STACK
```

So, the higher the argument number, the closer to the front of the stack the value is.

Symbol	arg1	arg2	arg3	Description
+		Num	Num	Numeric addition
+		Array	Array	Concatenate arrays
+		Array	Any	Add item to END of array
+		Any	Array	Add item to BEGINNING of array
-		Num	Num	Numeric subtraction
-		Array	Array	Remove all items in arg3 from arg2, if it contains them
*		Num	Num	Numeric multiplication
*		Array	Num	Duplicate list arg3 times, and concatenate into a single array
*		Num	Array	Duplicate list arg3 times, and encase in an array without flattening
*		Block	Num	Execute block arg3 times
/		Num	Num	Numeric division
/		Array	Block	Map (Pop from array, execute block, pop from stack and add to new array, repeat)
=		Any	Any	Tests value equality, pushing 1 of equal and 0 otherwise
<		Any	Any	Compares values, pushing 1 if arg2 < arg3 and 0 otherwise
>		Any	Any	Compares values, pushing 1 if arg2 > arg3 and 0 otherwise
^		Num	Num	Push arg2 to the power of arg3
%		Num	Num	Numeric modulus
%		Array	Block	Filter (Pop from array, execute block, pop from stack if 1 and add to new array, discard otherwise, repeat)
%		Array	Array	Zip arrays, pushing a new array
&		Any	Any	Boolean AND. Push 1 if arg2 and arg3 evaluate to true, 0 otherwise
|		Any	Any	Boolean OR. Push 1 if arg2 or arg3 evaluate to true, 0 otherwise
@	Any	Any	Any	Rotate top 3 items on stack clockwise
~			Num	Bitwise not
~			Array	Convert to string and evaluate as Jagl code
~			Block	Run block once
:		Array	Num	Push the value at the index arg3 in arg2
:	Array	Num	Num	Push the values from index arg2 to arg3 in arg1
;		Array	Any	Push index of first occurrence of arg3 in arg2, -1 otherwise
[			Num	Increment number by 1
[			Array	Rotate array clockwise
]			Num	Decrement number by 1
]			Array	Rotate array counterclockwise
$			Any	Add arg3 between each value in array
a			Array	Pushes 1 if any value in arg3 evaluates to true, 0 otherwise
a			Num	Pushes the absolute value of arg3

Symbol	arg1	arg2	arg3	Description
A			Array	Pushes 1 if all values in arg3 evaluate to true, 0 otherwise
b			Array	Folds the array with + and pushes the resulting value
B			Array	Folds the array with * and pushes the resulting value
c				Cycle stack clockwise
C			Num	Cycle stack arg3 rotations (negative numbers rotate the stack counterclockwise)
C			Array	Reverse array
d			Any	Duplicate item
D			Any	Drop item
e		Array	Any	Pushes 1 if arg3 is in arg2, 0 otherwise
E			Array	Flatten array
E			Num	Push arg3 converted to a floating point number
f		Block	Any	If arg3 evaluates to true, then execute block
F	Block	Block	Any	If arg3 evaluates to true, then execute arg2, otherwise execute arg1
F	Any	Any	Any	If arg3 then push arg2, otherwise push arg1
g			Num	Push array of ASCII values corresponding to the string representing the number
h				Forcefully halt the program
i			Num	Convert character to number (48 -> 0, 49 -> 1, … , 58 -> 9)
i			Array	Convert array to string, and convert that to a number
I		Any	Any	If arg3, then keep arg2, otherwise drop
j			Array	Remove duplicates from array
j			Num	Convert to integer
J			Array	Sort array
l			Array	Push length of array
l			Block	Push length of block
l			Num	Push log e arg3
L		Num	Num	Push arg2 log arg3
m			Num	Pushes 1 if number is prime, 0 otherwise
M			Num	Converts to character, and pushes 1 if it is numeric, 0 otherwise
M			Array	Converts to string, and pushes 1 if all characters are numeric, 0 otherwise
n			Any	Pushes 0 if arg3 is true, 0 otherwise
N			Num	Convers to character, and pushes 1 if it is alphanumeric, 0 otherwise
N			Array	Convers to string, and pushes 1 if all characters are numeric, 0 otherwise
o		Array	Block	Performs a foldleft on arg2 with arg3 as the function
O			Num	Converts to character, and pushes 1 if it is whitespace, 0 otherwise

Symbol	arg1	arg2	arg3	Description
O			Array	Convers to string, and pushes 1 if all characters are numeric, 0 otherwise
p			Num	Print the character corresponding to the ASCII value arg3
p			Array	Print the string with each character corresponding to the values in the array
p			Any	Write rough string representation of value
P			Any	Write rough string representation of value
q			Array	Pushes the minimum value from arg3
Q			Array	Pushes the maximum value from arg3
r		Num	Num	Pushes an array containing arg2 through arg3 (exclusive)
r		Array	Block	Performs a foldLeft on arg2, using arg3 as the function
R			Array	Pushes a random element from arg3
R		Num	Num	Pushes a random number between arg2 and arg3
R		Array	Num	Pushes arg3 random items from arg2
s			Any	Pushes an array containing the string representation of arg3
S		Any	Any	Swap the top two items on the stack
t				Gets a character from stdin and pushes its ASCII value
T				Gets a line from stdin and pushes an array containing the characters' ASCII values
u			Block	Do block then pop value, if true then repeat
U			Array	Pushes all elements from arg3 to stack individually
v		Array	Array	Set intersection
V		Array	Array	Set difference
w			Block	Pop value, if true then execute block and repeat
W		Array	Array	Set symmetric difference
x				Pushes the length of the stack to the stack
X				Clears the stack
y		Array	Any	Split arg2 at occurrences of arg3
Y		Array	Any	Push an array containing all indexes at which arg3 appears in arg2
z				Push the space character (32)
Z				Push the linefeed character (10)
#				Print a textual representation of the stack for debugging purposes