AWA5.0

A Very Special Programming Language

For Very Special People

AWA5.0 Language Specifications

O Introduction O

AWA5.0 is a bubble-based esoteric language posturing itself as the first vtuber based programming language in the world (despite evidence to the contrary). AWA5.0 is designed to spring the world forward and disrupt markets everywhere as a leap straight over Web 5.0 and into Web 6.0.

O Structure O

AWA5.0 is written as a series of Awa commands and parameters (called 'Awa-tisms') that manipulate data stored in Bubbles contained in a large Bubble Abyss. All awatism commands are represented by 5-bit unsigned integer values, and awatism parameters are of variable length integers, both signed or unsigned depending on the command. Awatisms are written sequentially in Awatalk, a proprietary standard of distilling information down to its most pure and beautiful form.

The 5.0 in AWA5.0 represents the bit size of each awatism (5 bits) as well as the version of this AWA language (version 0). As of 5.0 there are 22 awatisms and 10 empty spaces for future awatisms.

Don't forget the initial awa!

Example code (AWA5.0):

awa awa awawa awawa awa awa awawawa awa awawa awa awawa awawa awa awa awa awa awawa awa awa awawawa awawa awa awa awawa awa ...

Example code (awatism shorthand):

```
blw 25   //blow value 5 into the abyss
blw 4    //blow value 3 into the abyss
mlt      //multiply together
prn      //print the result
...
```

O Awatalk O

All blocks of Awatalk are prepended with an initial "awa". This serves as both a simplified checksum as well as a means to prevent disgrammatical Awatalk (ex. Starting an "awa" as "wa"). An entire program only needs one of these "awa"s at the start.

Since awatisms are represented by varying-bit integers, Awatalk is a simple method of representing binary data. See the table below for full conversion details.

Binary	Awatalk
0	" awa"
1	"wa"

Example Awatism (submerge 5):
sbm 5 = 6 5 = 00110 00101 =
awa awawawa awa awa awawa awawa

Awatalk is case insensitive, and ignores any and all characters that aren't A, a, W, w or the space character.

While it is possible to write a converter from Awatism command shorthand to Awatalk, it's heavily recommended to write the Awatalk by hand to obtain the best experience.

O Error Handling O

AWA5.0 is perfect, and since only the truly savant could use this language, there is no specification for what happens when invalid or incompatible code is encountered as that will never happen.

O Bubble Abyss O

AWA5.0 revolves around its Bubble Abyss, an infinite FILO stack that hold bubbles. However due to current hardware limitations, Bubble Abysses are expected to cap out at some point. Bubbles are added to the top of the stack and push other values down, although the submerge awatism can push the top bubble further down the stack. All interactions with the Bubble Abyss must be handled through the top of the stack.

Bubbles can contain one of two things: either an individual integer value (standard 4 byte int, –2,147,483,648 to 2,147,483,647) or any number of other bubbles. These two types are called 'simple' bubbles and 'double' bubbles. Double bubbles can be blown to contain other bubbles, or popped to release bubbles contained inside them. Bubble order is maintained when transferring this way.

Double bubbles can invoke different behavior from awatisms, so please be careful.

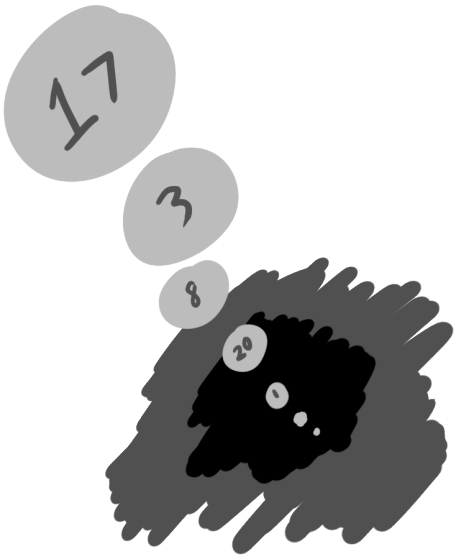
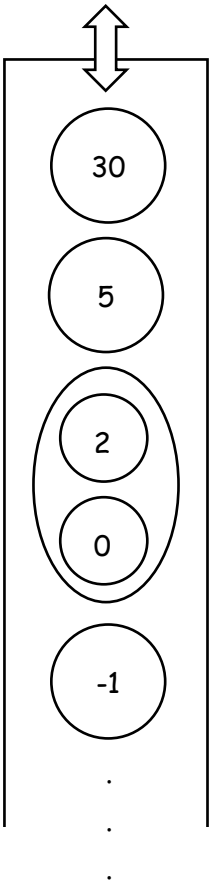


Fig. 1 Diagram of Bubble Abyss



O AwaSCII O

Awa Standard Code for Information Interchange (or AwaSCII) is a bold step forward in the re-imagining of the world's out-of-date standards. Normal ASCII takes 7 bits to represent 1 character, but AwaSCII refines and optimizes that to obtain 6 bits for that same character. See below for the table of AwaSCII values and associated characters. Do note that character positions have changed from the traditional alphabet to reflect how important they are.

Dec	Hex	Value	Dec	Hex	Value	Dec	Hex	Value
0	0x0	A	22	0x16	u	44	0x2C	2
1	0x1	W	23	0x17	m	45	0x2D	3
2	0x2	a	24	0x18	P	46	0x2E	4
3	0x3	w	25	0x19	C	47	0x2F	5
4	0x4	J	26	0x1A	N	48	0x30	6
5	0x5	E	27	0x1B	T	49	0x31	7
6	0x6	L	28	0x1C	p	50	0x32	8
7	0x7	Y	29	0x1D	c	51	0x33	9
8	0x8	H	30	0x1E	n	52	0x34	(space)
9	0x9	O	31	0x1F	t	53	0x35	.
10	0xA	S	32	0x20	B	54	0x36	,
11	0xB	I	33	0x21	D	55	0x37	!
12	0xC	U	34	0x22	F	56	0x38	'
13	0xD	M	35	0x23	G	57	0x39	(
14	0xE	j	36	0x24	R	58	0x3A	)
15	0xF	e	37	0x25	b	59	0x3B	~
16	0x10	l	38	0x26	d	60	0x3C	_
17	0x11	y	39	0x27	f	61	0x3D	/
18	0x12	h	40	0x28	g	62	0x3E	;
19	0x13	o	41	0x29	r	63	0x3F	\n
20	0x14	s	42	0x2A	0			
21	0x15	i	43	0x2B	1			

O Optimizations Over ASCII O

To fit the new 6 bit character size, some letters had to be obsoliticized. Please read the rationalizations below.

- k is basically a dressed up c
- q: see k above
- z is an s in a mirror
- v can be approximated by b (ex. 'very' -> 'bery')
- x is just silly

A few special letters had to be dropped as well to fit it all in.

- between ; and :, ; is bigger and would win in a fight
- ' can handle "'s job and more
- you can visit **https://shop.phase-connect.com** with / but not \
- special brackets [] and {} are excessive
- other symbols like +, - can be replaced with words (ex. 'plus', 'minus')
  - Due to clerical oversight, negative numbers should now be represented with ~ (ex. '~5')

O Awatisms O

See the table below for listed awatisms, as well as their command shorthand and associated parameter formats.

System	Pile Manipulation	Arithmetic	Program Flow
No Op 0x0 (nop)	Blow 0x5 (blo s8)	Add 0xB (4dd)	Label 0x10 (lbl u5)
Print 0x1 (prn)	Submerge 0x6 (sbm u5)	Subtract 0xC (sub)	Jump 0x11 (jmp u5)
Print Num 0x2 (pr1)	Pop 0x7 (pop)	Multiply 0xD (mul)	Equal To 0x12 (eql)
Read 0x3 (red)	Duplicate 0x8 (dpl)	Divide 0xE (div)	Less Than 0x13 (lss)
Read Num 0x4 (r3d)	Surround 0x9 (srn u5)	Count 0xF (cnt)	Greater Than 0x14 (gr8)
Terminate 0x1F (trm)	Merge 0xA (mrg)		

*parameters are given in the format ***xy***, where ***x*** is 's' for signed or 'u' for unsigned, and ***y*** is the number of bits in the parameter (typically 5 or 8)

Nop (nop) – 0x0

No operation, an empty command that has no effect.

nop – " awa awa awa awa awa"

Print (prn) – 0x01

Prints the top-most bubble's AwaSCII equivalent character and pops it. In the case of double bubbles, it prints all contained bubbles sequentially and pops them all.

prn – " awa awa awa awawa"

Print Num (pr1) – 0x02

Converts the top-most bubble's numerical value into an AwaSCII representation and prints it. In the case of double bubbles, it prints all contained bubbles sequentially, separated by spaces, and pops them all.

pr1 – " awa awa awawa awa"

Read (red) – 0x03

Reads input and pushes a double bubble containing a series of bubbles containing the AwaSCII values of the inputs. Non-AwaSCII characters are ignored.

red – " awa awa awawawa"

Read Num (r3d) – 0x04

Reads input and pulls a single number from it. This number is stored in a new bubble blown in the Abyss. Any input after the first consecutive set of numbers will be ignored. Please remember that '-' is NOT an AwaSCII character!

r3d – " awa awawa awa awa"

Blow (blo s8) – 0x05

Blows a new bubble containing the parameter value inside of it.

blw **1** – " awa awawa awawa **awa awa awa awa awa awa awawa**"

Submerge (sbm u5) – 0x06

Pushes the topmost bubble down X positions, where X is the parameter. If X is 0, it pushes the bubble all the way to the bottom.

sbm **6** – "awa awawawa awa **awa awawawa awa**"

Pop (pop) – 0x07

Pops the top bubble. If it is a double bubble, it releases all contained bubbles to the top of the Bubble Abyss.

pop – " awa awawawawa"

Duplicate (dpl) – 0x08

Makes a duplicate of the top-most bubble and pushes it into the Abyss.

dpl – "awawa awa awa awa"

Surround (srn u5) – 0x09

Surrounds the top X bubbles in a double bubble, where X is the parameter.

srn **3** – " awawa awa awawa **awa awa awawawa**"

Merge (mrg) – 0x0A

Merges the top two bubbles together. If both are simple bubbles, it acts like Add. If one or both are double bubbles, it will merge the contents of both bubbles into one bubble, maintaining bubble order. In this case, simple bubbles would be merged into the double bubble.

mrg – " awawa awawa awa"

Add (4dd) – 0x0B

Combines the top two bubbles into a new bubble with the value inside being the sum of the two. If one is a double bubble, it will add the simple bubble value to every bubble in the double bubble. If both are doubles, it will go through the bubbles one by one and combine them until one runs out of bubbles.

add – " awawa awawawa"

Subtract (sub) – 0x0C

Combines the top two bubbles into a new bubble with the value inside being the difference of the two. The bottom bubble gets subtracted from the top bubble. If one is a double bubble, it will apply the simple bubble value to every bubble in the double bubble. If both are doubles, it will go through the bubbles one by one and combine them until one runs out of bubbles.

sub – "awawawa awa awa"

Multiply (mul) – 0x0D

Combines the top two bubbles into a new bubble with the value inside being the product of the two. If one is a double bubble, it will apply the simple bubble value to every bubble in the double bubble. If both are doubles, it will go through the bubbles one by one and combine them until one runs out of bubbles.

mul – "awawawa awawa"

Divide (div) – 0x0E

Combines the top two bubbles into two new bubbles stored in a double bubble, with the top value being the original top bubble divided by the original bottom bubble (rounded down), and the bottom value being the remainder of that operation. If one is a double bubble, it will apply the simple bubble value to every bubble in the double bubble. If both are doubles, it will go through the bubbles one by one and combine them until one runs out of bubbles.

div – " awawawawa awa"

Count (cnt) – 0x0F

Blows a new bubble to the Abyss containing a value that reflects the number of bubbles contained in the current top bubble. If it's a simple bubble, this value is 0.

cnt – " awawawawawa"

Label (lbl u5) – 0x10

Marks a place in the code that can be jumped to with a Jump command. Can be tagged with a number 0 – 31.

lbl **17** – "~wa awa awa awa awa**wa awa awa awawa**"

Jump (jmp u5) – 0x11

Jumps to somewhere else in the code where the corresponding label command is. Can search for a label tagged 0 – 31.

jmp **17** – "~wa awa awa awawa**wa awa awa awawa**"

Equal To (eql) – 0x12

If the value of the top bubble is equal to the one below it, the next command is executed. Otherwise it skips that command.

eql – "~wa awa awawa awa"

Less Than (lss) – 0x13

If the value of the top bubble is less than the one below it, the next command is executed. Otherwise it skips that command.

lss – "~wa awa awawawa"

Greater Than (gr8) – 0x14

If the value of the top bubble is greater than the one below it, the next command is executed. Otherwise it skips that command.

gr8 – "~wa awawa awa awa"

Terminate (trm) – 0x1F

Marks the end of the program. A program doesn't need a terminate command to end, but any commands after the terminate command won't be read.

trm – "~wawawawawa"

O AWA5.0 Quick Reference Card O

Awatisms

System	Pile Manipulation	Arithmetic	Program Flow
No Op 0x0 (nop)	Blow 0x5 (blo s8)	Add 0x8 (4dd)	Label 0x10 (lbl u5)
Print 0x1 (prn)	Submerge 0x6 (sbm u5)	Subtract 0xC (sub)	Jump 0x11 (jmp u5)
Print Num 0x2 (pr1)	Pop 0x7 (pop)	Multiply 0xD (mul)	Equal To 0x12 (eql)
Read 0x3 (red)	Duplicate 0x8 (dpl)	Divide 0xE (div)	Less Than 0x13 (lss)
Read Num 0x4 (r3d)	Surround 0x9 (srn u5)	Count 0xF (cnt)	Greater Than 0x14 (gr8)
Terminate 0x1F (trm)	Merge 0xA (mrg)		

Awatalk

Binary	Awatalk
0	" awa"
1	"wa"

AwaSCII

Dec	Hex	Value	Dec	Hex	Value	Dec	Hex	Value
0	0x0	A	22	0x16	u	44	0x2C	2
1	0x1	W	23	0x17	m	45	0x2D	3
2	0x2	a	24	0x18	P	46	0x2E	4
3	0x3	w	25	0x19	C	47	0x2F	5
4	0x4	J	26	0x1A	N	48	0x30	6
5	0x5	E	27	0x1B	T	49	0x31	7
6	0x6	L	28	0x1C	p	50	0x32	8
7	0x7	Y	29	0x1D	c	51	0x33	9
8	0x8	H	30	0x1E	n	52	0x34	(space)
9	0x9	O	31	0x1F	t	53	0x35	.
10	0xA	S	32	0x20	B	54	0x36	,
11	0xB	I	33	0x21	D	55	0x37	!
12	0xC	U	34	0x22	F	56	0x38	'
13	0xD	M	35	0x23	G	57	0x39	(
14	0xE	j	36	0x24	R	58	0x3A	)
15	0xF	e	37	0x25	b	59	0x3B	~
16	0x10	l	38	0x26	d	60	0x3C	_
17	0x11	y	39	0x27	f	61	0x3D	/
18	0x12	h	40	0x28	g	62	0x3E	;
19	0x13	o	41	0x29	r	63	0x3F	\n
20	0x14	s	42	0x2A	0			
21	0x15	i	43	0x2B	1			

Thanks for printing me out! :^)

1: Cut here

2: Fold here

3: Staple here

O AWA5.0 Rights O

AWA5.0 is an open standard, meaning anyone can build on and produce their own version. The only request is that the AWA5.X name standard is preserved. For example, a minimalist reimagining might be called "AWA5.Min". Have fun, I know I did!

O Acknowledgements O

Temptonics LLLTD is a subsidiary of Kuro Co. and Temp FlavrWorks.

My personal special thanks to:

- Jelly Hoshiumi for being rad and getting me addicted to Shenzhen I/O.
- Sena Bonbon for the wonderful Awa5.0 logo/illustration
- Phase Connect for being my inspiration
- My FlaVRmates for pushing me to be my best everyday
- You, probably. Maybe. Eh.