XS Grammar

Syntax for production rules is:

$A \to B\alpha$

The following represent non terminal symbols: - uppercase letters

The following represent terminal symbols: - lowercase words/letters - greek symbols - keywords

1. Program

1.1. Literals

NUM $\to$ INT | FLT

VCT $\to$ `vector(NUM, NUM, NUM);`

1.2. Top Level Statements

X $\to$ I X | P

P $\to$ RL P | FN P | V_{top} P | ϵ

where

X := XS script

I := Include statement

P := Program

RL := Rule def

FN := Function def

V_{top} := Top level variable definition

1.3. Prelude

This consists of all the constants and functions described in

1. XS Constant Reference
2. XS Function Reference

2. Statement

S $\to$ V_{decl} | V_{def} | V_{asgn} | IE | W | F | SC | R | BR | CO | LBL | GT | DBG | BRPT | DP | DM

$\bar{\text{S}}$ $\to$ S $\bar{\text{S}}$ | ϵ

B $\to$ `{` $\bar{\text{S}}$ `}`

BS $\to$ B | S

where

S := Statement

$\bar{\text{S}}$:= Statements

B := Body

BS := Body or statement

V_{decl} := Variable declaration

V_{def} := Variable definition

V_{asgn} := Variable Assignment

IE := If (Else) statement

W := While loop

F := For loop

SC := Switch case

R := Return statement

BR := Break statement

CO := Continue statement

BRPT := Breakpoint

2.1. Top Level Var Def

$V_{top} \rightarrow$ `extern` V_{top} | `const` V_{top} | `static` V_{top}

$V_{top} \rightarrow$ DTYPE ID `=` LIT`;`

DTYPE $\rightarrow$ `int | float | bool | string | vect`

LIT $\rightarrow$ INT | FLT | STR | VCT | BOOL

Note: XS currently has bugs with defining top level strings and vectors

where

DTYPE := Datatype

ID := Identifier

LIT := Literal

2.2. Var Decl

$V_{decl} \rightarrow$ `static` V_{decl}

$V_{decl} \rightarrow$ DTYPE ID`;`

2.3. Var Def

$V_{def} \rightarrow$ `const` V_{def}

$V_{def} \rightarrow$ DTYPE ID `=` E`;`

2.4. Var Assign

$V_{asgn} \rightarrow$ ID `=` E`;`

where

E := Expression

2.5. If Else

IE → `if ( E )` BS ELSE

ELSE → `else` BS | ϵ

where

ELSE := Else branch

2.6. While

W → `while ( E )` BS

2.7. For

F → `for (` V_{asgn} OP_{rel} INT `)` BS

where

OP_{rel} := Relational operators

2.8. Switch

SC → `switch ( E ) {` CASES `}`

CASES → CASE CASES | DEFAULT $CASES_{no\ default}$ | ϵ

$CASES_{no\ default}$ → CASE $CASES_{no\ default}$ | ϵ

CASE → `case` LIT `:` B

DEFAULT → `default :` B

where

CASES := Optional case statements with one optional default case

$CASES_{no\ default}$:= Optional case statements only

CASE := case statement

DEFAULT := default statement

2.9. Functions

FN → `extern` FN | `mutable` FN

FN → RTYPE ID ($ARGS_{formal}$)B

RTYPE → `void` | DTYPE

$ARGS_{formal}$ → ARG | ARG, $ARGS_{formal}$

ARG → DTYPE ID = LIT | ϵ

where

RTYPE := Return type

$ARGS_{formal}$:= Formal Arguments

ARG := Argument

2.10. Return

R → `return ( E );`

R → `return;`

2.11. Rules

RL → `rule` ID RPS B

RPS → RP RPS | ϵ

RP → RP_{act} | RP_{grp} | RP_{sfq} | RP_{xfq} | RP_{rim} | RP_{pty}

RP_{act} → `active | inactive`

RP_{grp} → `group` ID

RP_{sfq} → `minInterval` INT | `highFrequency`

RP_{xfq} → `maxInterval` INT

RP_{rim} → `runImmediately`

RP_{pty} → `priority` INT

Note: only one of each parameter can be present in a rule def, this needs to be a linting time check, unfortunately its not possible to bake it into the grammar

RPS := Rule parameters

RP := Rule parameter

RP_{act} := Active state

RP_{grp} := Group

RP_{sfq} := Min frequency

RP_{xfq} := Max frequency

RP_{rim} := Run Immediately

RP_{pty} := Priority

2.12. Postfix

DP → ID`++`

DM → ID`--`

2.13. Include

I → `include` STR`;`

2.14. Break

BR → `break;`

2.15. Continue

CO → `continue;`

2.16. Label Def

LBL → `label` ID`;`

2.17. Goto

GT → `goto` ID`;`

2.18. Function Call (Statement)

FNCS → FNC`;`

2.19. Debug

DBG → `dbg` ID`;`

Note: I don't know what this does in XS, its valid syntax though.

2.20. Breakpoint

BRPT → `breakpoint;`

Note: This will pause XS execution. I don't know if its possible to resume execution/if this keyword is useful.

2.21. Class

CLS → `class` ID `{` MEM_VARS `};`

MEM_VARS → DTYPE ID `=` E`;` MEM_VARS | ϵ

Note: I don't know how to use classes in XS, its valid syntax though. The furthest I've gotten is declaring a class variable: `ClsName obj;`. Initialising it or accessing member values doesn't seem possible.

2.22. Docstring

Todo

3. Expression

E7 → LIT | ID | P | FNC

E6 → E7 | E6 `*` E7 | E6 `/` E7 | E6 `%` E7

E5 → E6 | E5 `+` E6 | E5 `-` E6

E4 → E5 | E4 `<` E5 | E4 `>` E5 | E4 `>=` E5 | E4 `<=` E5

E3 → E4 | E3 `==` E4 | E3 `!=` E4

E2 → E3 | E2 `&&` E3

E1 → E2 | E1 `||` E2

E → E1

3.1. Parenthesis

P → (E)

3.2. Function Call (Expression)

FNC → ID(ARGS$_{\text{actual}}$)

ARGS$_{\text{actual}}$ → ARGS$'$ | ϵ

ARGS$'$ → E | E, ARGS$'$

where

$\mathrm{ARGS}_{\mathrm{actual}} :=$ Actual Arguments

4. The Full Grammar