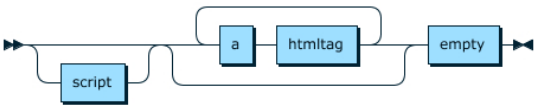
program:

program ::= 'PROGRAM' 'ID' program1

no references

program1:



program1 ::= script? (a htmltag)* empty

referenced by:

- program

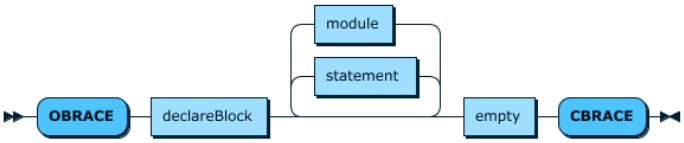
script:

script ::= 'SCRIPT' snp_script_start block

referenced by:

- program1

block:

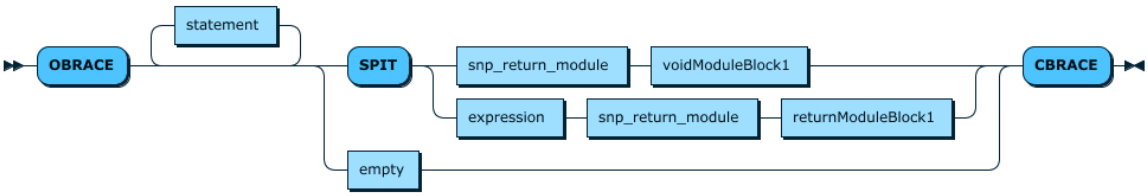


block ::= 'OBRACE' declareBlock (statement | module)* empty 'CBRACE'

referenced by:

- script

simpleBlock:

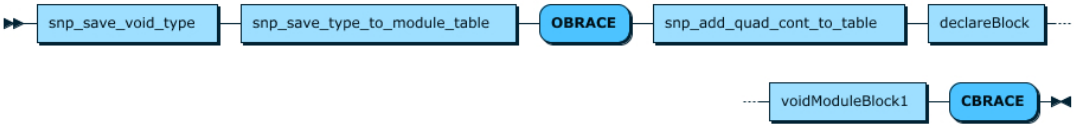


simpleBlock
::= 'OBRACE' statement* ('SPIT' (snp_return_module voidModuleBlock1 | expression snp_return_module returnModuleBlock1) | empty) 'CBRACE'

referenced by:

- condition
- condition1
- cycle
- doCycle

voidModuleBlock:

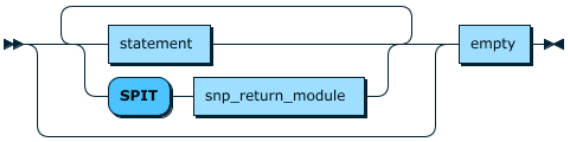


voidModuleBlock
::= snp_save_void_type snp_save_type_to_module_table 'OBRACE' snp_add_quad_cont_to_table declareBlock voidModuleBlock1 'CBRACE'

referenced by:

- module1

voidModuleBlock1:

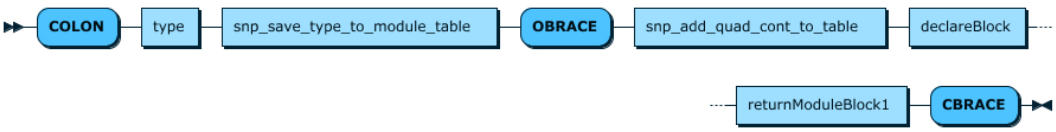


voidModuleBlock1
        ::= (statement | 'SPIT' snp_return_module)* empty

referenced by:

- simpleBlock
- voidModuleBlock

returnModuleBlock:

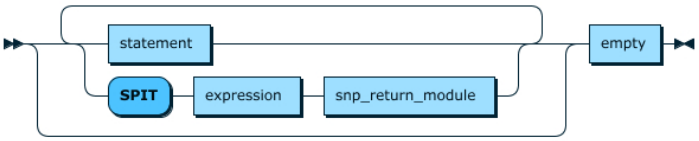


returnModuleBlock
        ::= 'COLON' type snp_save_type_to_module_table 'OBRACE' snp_add_quad_cont_to_table declareBlock returnModuleBlock1 'CBRACE'

referenced by:

- module1

returnModuleBlock1:

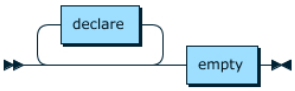


returnModuleBlock1
        ::= (statement | 'SPIT' expression snp_return_module)* empty

referenced by:

- returnModuleBlock
- simpleBlock

declareBlock:

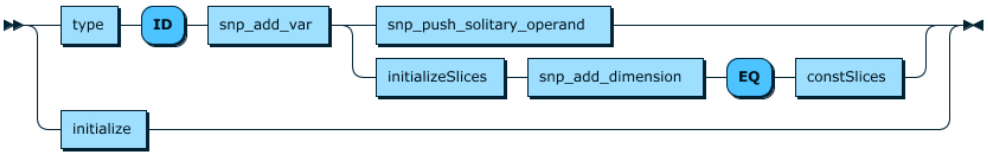


declareBlock
        ::= declare* empty

referenced by:

- block
- returnModuleBlock
- voidModuleBlock

declare:

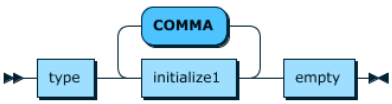


declare ::= type 'ID' snp_add_var (snp_push_solitary_operand | initializeSlices snp_add_dimension 'EQ' constSlices)
           | initialize

referenced by:

- declareBlock

initialize:



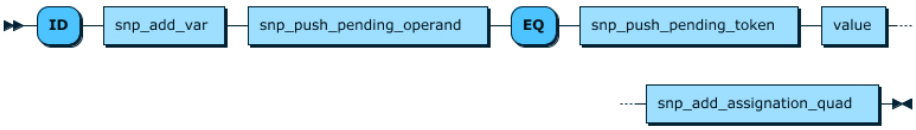
initialize

```
        ::= type initialize1 ( 'COMMA' initialize1 )* empty
```

referenced by:

- declare

initialize1:




```
initialize1
        ::= 'ID' snp_add_var snp_push_pending_operand 'EQ' snp_push_pending_token value snp_add_assignation_quad
```

referenced by:

- initialize

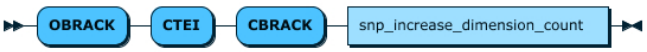
initializeSlices:


```
initializeSlices
        ::= initializeSlices1D
          | initializeSlices2D
```

referenced by:

- declare

initializeSlices1D:




```
initializeSlices1D
        ::= 'OBRACK' 'CTEI' 'CBRACK' snp_increase_dimension_count
```

referenced by:

- initializeSlices

constSlices:


```
constSlices
        ::= constSlice1D
```

referenced by:

- declare

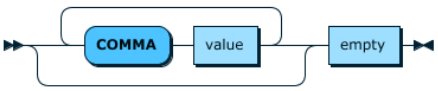
constSlice1D:


```
constSlice1D
        ::= 'OBRACK' 'CBRACK' snp_init_slice_1d
```

referenced by:

- constSlices

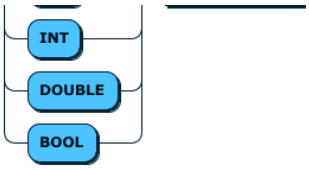
constSlice1D1:




```
constSlice1D1
        ::= ( 'COMMA' value )* empty
```

no references

type:

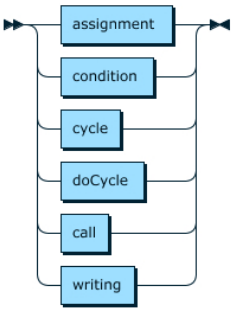


type ::= ('STR' | 'INT' | 'DOUBLE' | 'BOOL') snp_save_type

referenced by:

- arguments
- declare
- initialize
- returnModuleBlock

statement:

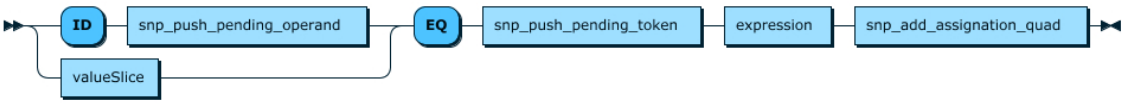


statement
::= assignment
| condition
| cycle
| doCycle
| call
| writing

referenced by:

- block
- returnModuleBlock1
- simpleBlock
- voidModuleBlock1

assignment:



assignment
::= ('ID' snp_push_pending_operand | valueSlice) 'EQ' snp_push_pending_token expression snp_add_assignation_quad

referenced by:

- statement

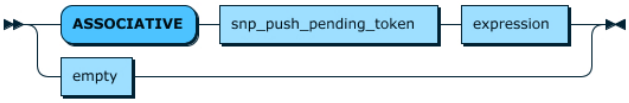
expression:

expression
::= rel_expression expression1 snp_check_precedence_and_create_quadruple_for_logic

referenced by:

- assignment
- condition
- cycle
- doCycle
- expression1
- factor
- html_assignment
- html_condition
- html_cycle
- html_end_do_cycle
- htmlscript
- params
- returnModuleBlock1
- simpleBlock
- writing1

expression1:

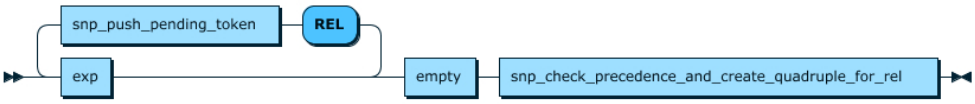


expression1
        ::= 'ASSOCIATIVE' snp_push_pending_token expression
          | empty

referenced by:

- expression

rel_expression:

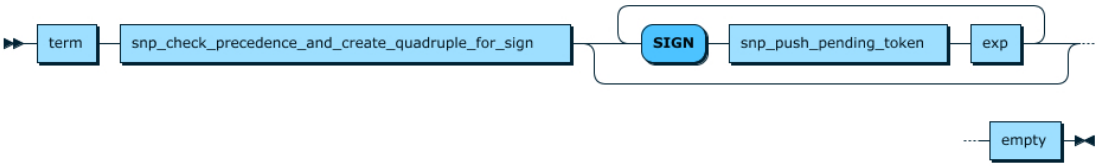


rel_expression
        ::= exp ('REL' snp_push_pending_token exp)* empty snp_check_precedence_and_create_quadruple_for_rel

referenced by:

- expression

exp:

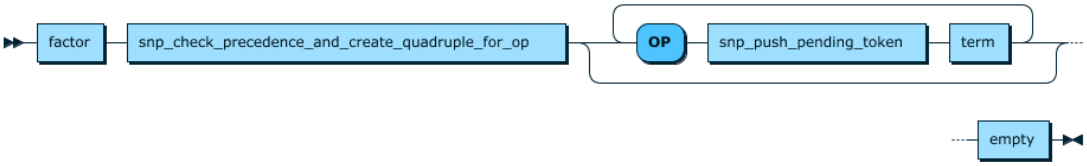


exp   ::= term snp_check_precedence_and_create_quadruple_for_sign ('SIGN' snp_push_pending_token exp)* empty

referenced by:

- exp
- rel_expression

term:

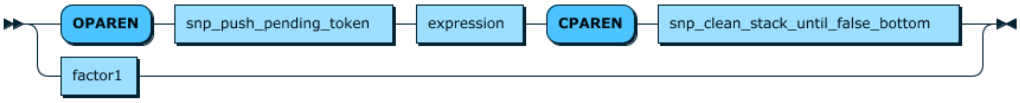


term   ::= factor snp_check_precedence_and_create_quadruple_for_op ('OP' snp_push_pending_token term)* empty

referenced by:

- exp
- term

factor:

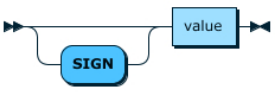


factor   ::= 'OPAREN' snp_push_pending_token expression 'CPAREN' snp_clean_stack_until_false_bottom
          | factor1

referenced by:

- term

factor1:

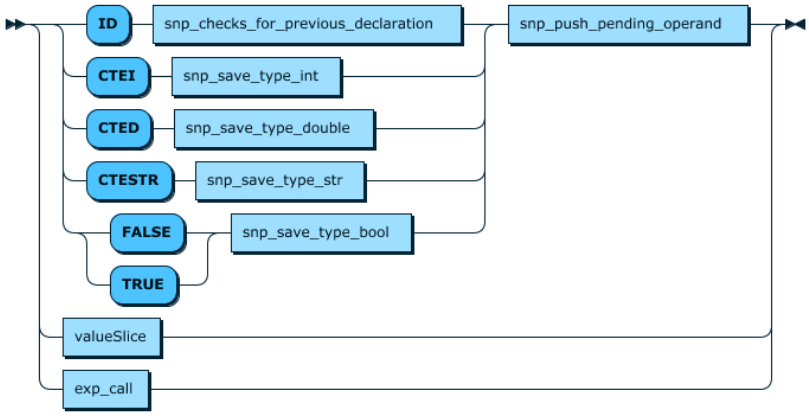


factor1 ::= 'SIGN'? value

referenced by:

- factor

value:




```
value    ::= ( 'ID' snp_checks_for_previous_declaration | 'CTEI' snp_save_type_int | 'CTED' snp_save_type_double | 'CTESTR' snp_save_type_str | ( 'FALSE' | 'TRUE' )
snp_save_type_bool ) snp_push_pending_operand
           | valueSlice
           | exp_call
```

referenced by:

- constSlice1D1
- factor1
- initialize1

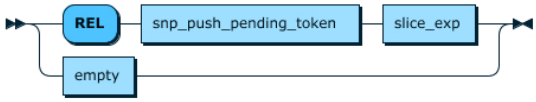
slice_expression:


```
slice_expression
         ::= slice_exp slice_expression1 snp_check_precedence_and_create_quadruple_for_rel
```

referenced by:

- slice_factor
- valueSlice1D

slice_expression1:

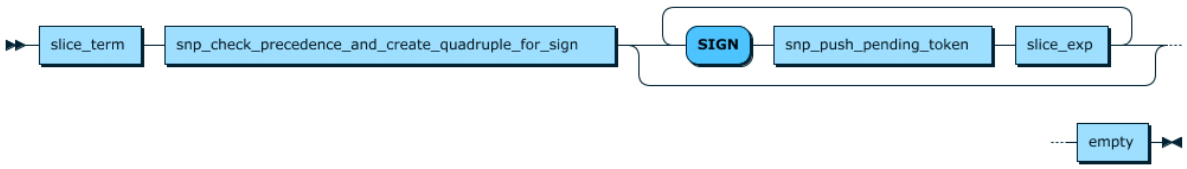



```
slice_expression1
         ::= 'REL' snp_push_pending_token slice_exp
           | empty
```

referenced by:

- slice_expression

slice_exp:

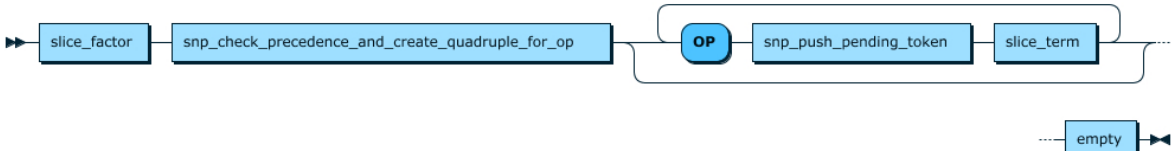



```
slice_exp
         ::= slice_term snp_check_precedence_and_create_quadruple_for_sign ( 'SIGN' snp_push_pending_token slice_exp )* empty
```

referenced by:

- slice_exp
- slice_expression
- slice_expression1

slice_term:



slice_term
        ::= slice_factor snp_check_precedence_and_create_quadruple_for_op ('OP' snp_push_pending_token slice_term)* empty

referenced by:

- slice_exp
- slice_term

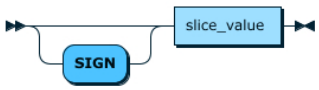
slice_factor:

slice_factor
        ::= 'OPAREN' snp_push_pending_token slice_expression 'CPAREN' snp_clean_stack_until_false_bottom
          | slice_factor1

referenced by:

- slice_term

slice_factor1:

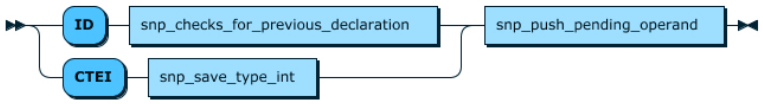


slice_factor1
        ::= 'SIGN'? slice_value

referenced by:

- slice_factor

slice_value:



slice_value
        ::= ('ID' snp_checks_for_previous_declaration | 'CTEI' snp_save_type_int) snp_push_pending_operand

referenced by:

- slice_factor1

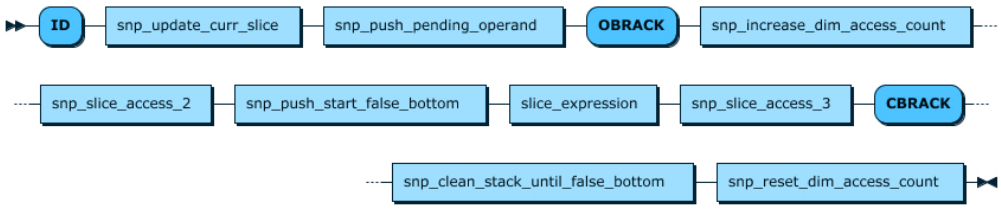
valueSlice:

valueSlice
        ::= valueSlice1D

referenced by:

- assignment
- value

valueSlice1D:

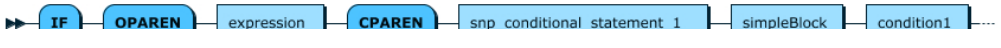


valueSlice1D
        ::= 'ID' snp_update_curr_slice snp_push_pending_operand 'OBRACK' snp_increase_dim_access_count snp_slice_access_2 snp_push_start_false_bottom
slice_expression snp_slice_access_3 'CBRACK' snp_clean_stack_until_false_bottom snp_reset_dim_access_count

referenced by:

- valueSlice

condition:

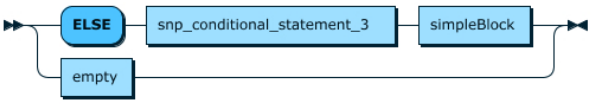


condition
        ::= 'IF' 'OPAREN' expression 'CPAREN' snp_conditional_statement_1 simpleBlock condition1 snp_conditional_statement_2

referenced by:

- statement

condition1:

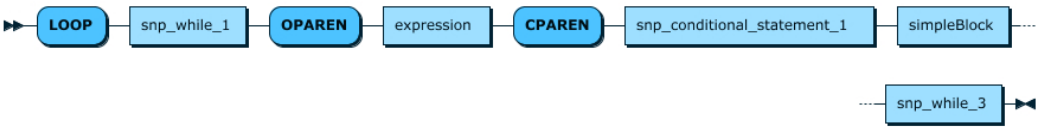


condition1
        ::= 'ELSE' snp_conditional_statement_3 simpleBlock
          | empty

referenced by:

- condition

cycle:



cycle   ::= 'LOOP' snp_while_1 'OPAREN' expression 'CPAREN' snp_conditional_statement_1 simpleBlock snp_while_3

referenced by:

- statement

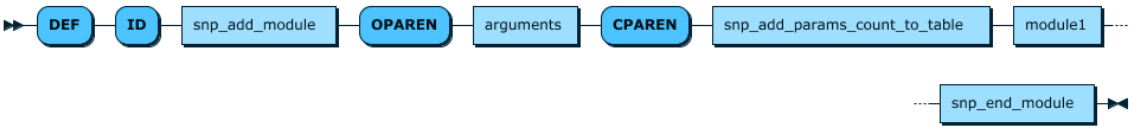
doCycle:

doCycle ::= 'DO' snp_while_1 simpleBlock 'LOOP' 'OPAREN' expression 'CPAREN' snp_do_while_gotot

referenced by:

- statement

module:

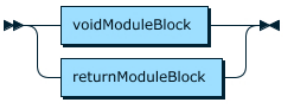


module   ::= 'DEF' 'ID' snp_add_module 'OPAREN' arguments 'CPAREN' snp_add_params_count_to_table module1 snp_end_module

referenced by:

- block

module1:



module1 ::= voidModuleBlock
           | returnModuleBlock

referenced by:

- module

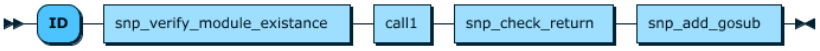
call:

call   ::= 'ID' snp_verify_module_existance call1 snp_add_gosub

referenced by:

- statement

exp_call:

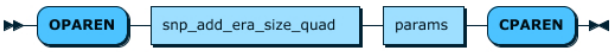



```
exp_call ::= 'ID' snp_verify_module_existance call1 snp_check_return snp_add_gosub
```

referenced by:

- value

call1:

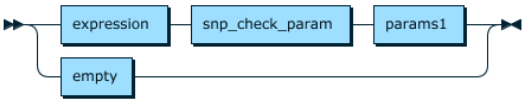



```
call1    ::= 'OPAREN' snp_add_era_size_quad params 'CPAREN'
```

referenced by:

- call
- exp_call

params:

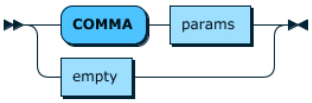



```
params   ::= expression snp_check_param params1
           | empty
```

referenced by:

- call1
- html_call
- params1

params1:




```
params1  ::= 'COMMA' params
           | empty
```

referenced by:

- params

writing:


```
writing  ::= 'EVAL' 'OPAREN' writing1 'CPAREN'
```

referenced by:

- html_statement
- statement

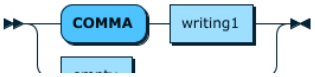
writing1:


```
writing1 ::= snp_push_pending_eval_token expression snp_add_eval_quad writing2
```

referenced by:

- writing
- writing2

writing2:

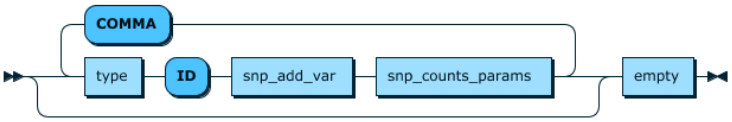


writing2 ::= 'COMMA' writing1
| empty

referenced by:

- writing1

arguments:



arguments
::= (type 'ID' snp_add_var snp_counts_params ('COMMA' type 'ID' snp_add_var snp_counts_params)*)? empty

referenced by:

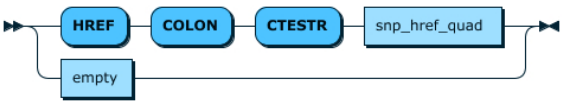
- module

html_class:

html_class
::= 'CLASS' 'COLON' 'CTESTR' snp_class_quad
| empty

no references

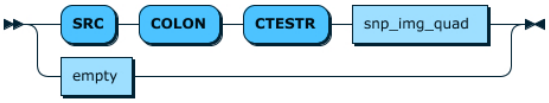
html_href:



html_href
::= 'HREF' 'COLON' 'CTESTR' snp_href_quad
| empty

no references

html_img:



html_img ::= 'SRC' 'COLON' 'CTESTR' snp_img_quad
| empty

no references

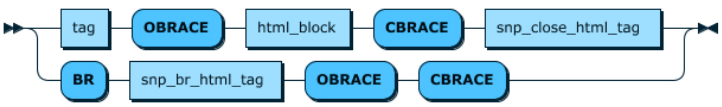
htmlscript:

htmlscript
::= 'OEVALSCRIPT' snp_push_eval_pending_token expression 'CEVALSCRIPT' snp_add_eval_quad

referenced by:

- html_statement

htmltag:

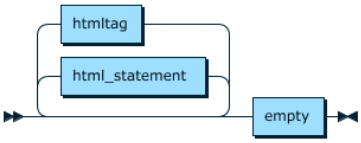


htmltag ::= tag 'OBRACE' html_block 'CBRACE' snp_close_html_tag
| 'BR' snp_br_html_tag 'OBRACE' 'CBRACE'

referenced by:

- html_block
- program1

html_block:

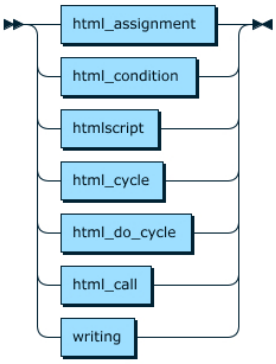



```
html_block
        ::= ( html_statement | htmltag )* empty
```

referenced by:

- html_condition
- html_condition1_else
- html_cycle
- html_do_cycle
- htmltag

html_statement:

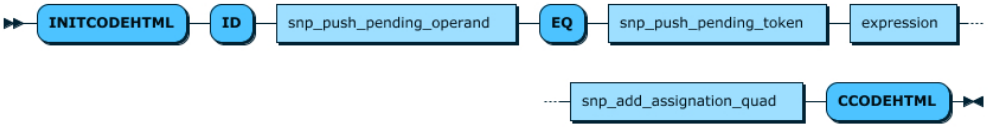



```
html_statement
        ::= html_assignment
          | html_condition
          | htmlscript
          | html_cycle
          | html_do_cycle
          | html_call
          | writing
```

referenced by:

- html_block

html_assignment:

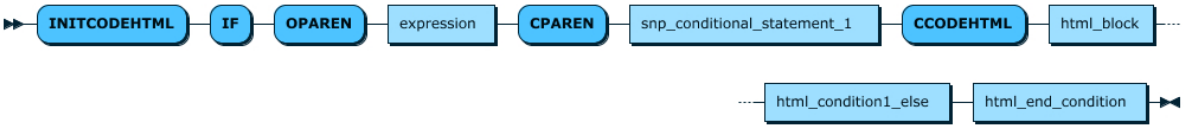



```
html_assignment
        ::= 'INITCODEHTML' 'ID' snp_push_pending_operand 'EQ' snp_push_pending_token expression snp_add_assignation_quad 'CCODEHTML'
```

referenced by:

- html_statement

html_condition:

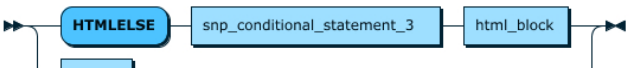



```
html_condition
        ::= 'INITCODEHTML' 'IF' 'OPAREN' expression 'CPAREN' snp_conditional_statement_1 'CCODEHTML' html_block html_condition1_else html_end_condition
```

referenced by:

- html_statement

html_condition1_else:



html_condition1_else
        ::= 'HTMLELSE' snp_conditional_statement_3 html_block
          | empty

referenced by:

- html_condition

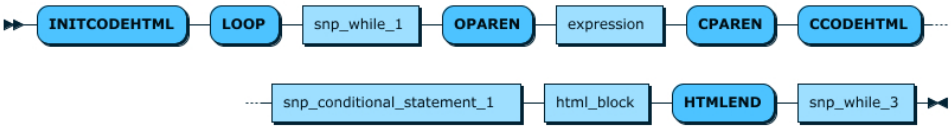
html_end_condition:

html_end_condition
        ::= 'HTMLEND' snp_conditional_statement_2

referenced by:

- html_condition

html_cycle:



html_cycle
        ::= 'INITCODEHTML' 'LOOP' snp_while_1 'OPAREN' expression 'CPAREN' 'CCODEHTML' snp_conditional_statement_1 html_block 'HTMLEND' snp_while_3

referenced by:

- html_statement

html_do_cycle:

html_do_cycle
        ::= 'INITCODEHTML' 'DO' 'CCODEHTML' snp_while_1 html_block html_end_do_cycle

referenced by:

- html_statement

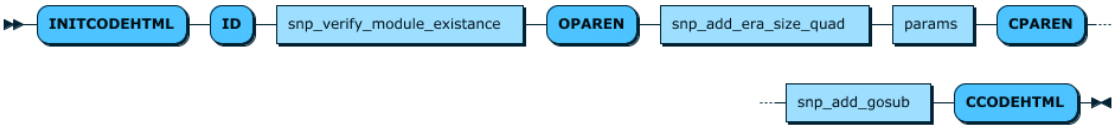
html_end_do_cycle:

html_end_do_cycle
        ::= 'HTMLENDDO' 'OPAREN' expression 'CPAREN' 'CCODEHTML' snp_do_while_gotot

referenced by:

- html_do_cycle

html_call:



html_call
        ::= 'INITCODEHTML' 'ID' snp_verify_module_existance 'OPAREN' snp_add_era_size_quad params 'CPAREN' snp_add_gosub 'CCODEHTML'

referenced by:

- html_statement
