Languages-beta: SL-3-Statements *

The PLanCompS Project

`SL-3-Statements.cbs` | PLAIN | PRETTY

Language "SL"

3 Statements

Syntax *Stmt* : stmt ::= expr ';'
| 'return' expr ';'
| 'return' ';'
| 'if' '(' expr ')' block
| 'if' '(' expr ')' block 'else' block
| 'while' '(' expr ')' block
| 'break' ';'
| 'continue' ';'
| block

Block : block ::= '{' stmt* '}'

Rule ⟦ 'if' '(' *Expr* ')' *Block* ⟧ : stmt =
⟦ 'if' '(' *Expr* ')' *Block* 'else' '{' '}' ⟧

Semantics exec⟦ *Stmt** : stmt* ⟧ : ⇒ null-type
Rule exec⟦ *Expr* ';' ⟧ = effect(eval⟦ *Expr* ⟧)
Rule exec⟦ 'return' *Expr* ';' ⟧ = return(eval⟦ *Expr* ⟧)
Rule exec⟦ 'return' ';' ⟧ = return(null-value)
Rule exec⟦ 'if' '(' *Expr* ')' $Block_1$ 'else' $Block_2$ ⟧ =
if-true-else(bool eval⟦ *Expr* ⟧, exec⟦ $Block_1$ ⟧, exec⟦ $Block_2$ ⟧)
Rule exec⟦ 'while' '(' *Expr* ')' *Block* ⟧ =
handle-break(
while-true(
bool eval⟦ *Expr* ⟧,
handle-continue(exec⟦ *Block* ⟧)))
Rule exec⟦ 'break' ';' ⟧ = break
Rule exec⟦ 'continue' ';' ⟧ = continue
Rule exec⟦ '{' *Stmt** '}' ⟧ = exec⟦ *Stmt** ⟧
Rule exec⟦ ⟧ = null-value
Rule exec⟦ *Stmt* $Stmt^+$ ⟧ =
sequential(exec⟦ *Stmt* ⟧, exec⟦ $Stmt^+$ ⟧)

---

*Suggestions for improvement: `plancomps@gmail.com`.
Reports of issues: `https://github.com/plancomps/CBS-beta/issues`.