Languages-beta: OC-L-11-Module-Implementations *

The PLanCompS Project

`OC-L-11-Module-Implementations.cbs` | PLAIN | PRETTY

Language "OCaml Light"

11 Module implementations

Syntax
UI : unit-implementation ::= (semicolon-pair* module-items semicolon-pair*)?
MIS : module-items ::= definition
| expr
| module-items semicolon-pair* module-item
MI : module-item ::= definition
| semicolon-pair expr
D : definition ::= let-definition
| type-definition
| exception-definition
Lexis *SCP* : semicolon-pair ::= ';;'

Semantics interpret⟦ *UI* : unit-implementation ⟧ : ⇒ environments
Rule interpret⟦ ⟧ = map( )
Rule interpret⟦ SCP_1* *MIS* SCP_2* ⟧ =
handle-thrown(
scope(
ocaml-light-core-library,
accumulate(define-or-evaluate-items⟦ *MIS* ⟧)),
sequential(
print("Uncaught exception: ", ocaml-light-to-string given, "\n"),
map( )))

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

Semantics define-or-evaluate-items⟦ _ : module-items ⟧ : (⇒ envs)$^{+}$

Rule define-or-evaluate-items⟦ *LD* ⟧ =
ocaml-light-define-and-display define-values⟦ *LD* ⟧

Rule define-or-evaluate-items⟦ *TDS* ⟧ = define-types⟦ *TDS* ⟧

Rule define-or-evaluate-items⟦ *ED* ⟧ = define-exception⟦ *ED* ⟧

Rule define-or-evaluate-items⟦ *E* ⟧ =
ocaml-light-evaluate-and-display evaluate⟦ *E* ⟧

Rule define-or-evaluate-items⟦ *MIS SCP** *D* ⟧ =
(define-or-evaluate-items⟦ *MIS* ⟧, define-or-evaluate-items⟦ *D* ⟧)

Rule define-or-evaluate-items⟦ *MIS SCP** *SCP E* ⟧ =
(define-or-evaluate-items⟦ *MIS* ⟧, define-or-evaluate-items⟦ *E* ⟧)