Syntax Semantics Idioms Libraries Tools

Colourless green ideas sleep furiously.

→ The big red ball.

→ The red big ball.

Semantics ⟶ "Static" semantics

→ Name resolution ("Lexical" scope)

→ Type checking / type inference

⟶ "Dynamic" semantics

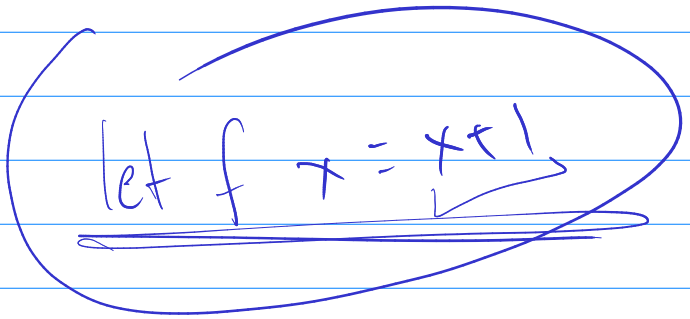
File "Compilation unit"

Sequence of variable definitions & expressions.

(top level lets)

let v = expression

variable name



let v = "Hello" ^ "World"

Integers | floating pt nums | Lists | Strings | Booleans |
Tuples | Arrays | Records | Unit

Expressions: Literals

| Variables | Conditionals | Boolean connectives
if then else (&& , ||)

| Function applications

f e1 e2 e3 e4 List.hd [2;8;9]

2+3 List.nth [2;8;9;6] 3

| Let expressions (let v = — in —)

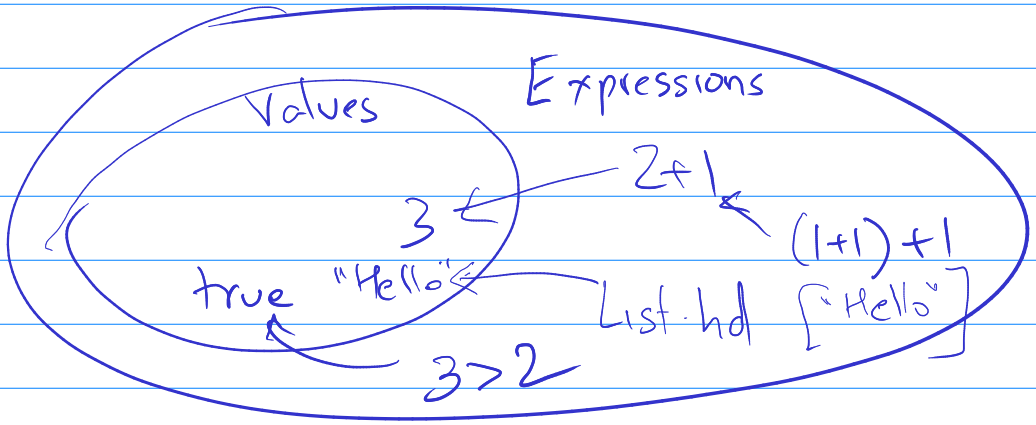
| Function definitions | Assertions

NP ::= Article Noun

Article ::= A | An | The

Sentence ::= NP VP

VP ::= Verb NP



Rule #1: When I see a fn application

$f\ e_1\ e_2\ \cdots\ e_n$

(1) Evaluate all inputs $e_1 \cdots e_n$, in some order

$e_1 \to v_1$, $e_n \to v_n$

(2) Substitute the results into the body of f.

(3) Continue evaluation until that turns into a value.

```
let f x = print "Hello"; x + 1

let g y = print "World"; y + 2

f (g 3)
```

Rule #2: To evaluate if e_1 then e_2 else e_3

(1) Evaluate e_1 first.

(2) If true, continue with e_2

(3) If false, continue with e_3