Azalea Syntax Specification

The Azalea Team

2025-08-13

EBNF syntax

The Azalea syntax is defined formally using a variant of EBNF. The language is currently a work in progress, so think of this as a living document that will change as the language evolves.

- `ident` is a terminal, since it matches actual text in the source code (e.g a regular expression).
- `expr` is a non-terminal, since it is defined in terms of other terminals and non-terminals.

EBNF Grammar

```
-- Terminals
comment: "--" [^"\n"]*
reserved_type: "Int" | "Float" | "String" | "Bool" | "Unit"
ident:    "[_a-zA-Z][_a-zA-Z0-9]*"
number:   "[0-9]+"
float:    "[0-9]+\\.[0-9]+"
string:   "(\\[\\"]|[^"])*"
bool:     "true" | "false"
binop_op: "+" | "-" | "*" | "/" | "==" | "!=" | "<" | "<=" | ">" | ">="

type_param: ident | reserved_type
type_param_list: "[" type_param ("," type_param)* "]"

expr:
    literal
  | ident
  | member_access
  | binop
  | unop
  | record_expr
  | array_expr
  | array_index
  | call
  | lambda
  | if_expr

literal:
    number
  | float
  | string
  | bool

-- Expressions

member_access: expr "." ident
binop: expr binop_op expr
unop: binop_op expr

record_expr:
  "{" record_field_list "}"
record_field_list:
  ident ":" expr ("," ident ":" expr)*

array_expr:
  "[" expr ("," expr)* "]"
array_index:
  expr "." "[" expr "]"

call:
  expr "(" (expr ("," expr)*)? ")"
```

```
-- \(x) -> x, \(x, y) -> x + y
lambda:
  "\\" lambda_args "->" expr
lambda_args:
  ident ("," ident)*

if_expr:
    "if" expr "then" expr "else" expr      -- expression form, else required
  | "if" expr "then" block ("else" block)? -- statement form, else optional

-- Statements

block:
  "do" stmt* "end"

stmt:
    expr_stmt
  | let_stmt
  | mut_stmt
  | assign_stmt
  | for_stmt
  | while_stmt
  | toplevel_stmt

expr_stmt: expr
let_stmt:
  "let" ident ":" type? "=" expr
mut_stmt:
  "mut" ident ":" type? "=" expr
assign_stmt:
  ident "=" expr
for_stmt:
  "for" ident "in" expr block
while_stmt:
  "while" expr block

-- Top-level declarations
toplevel_stmt:
    extern_decl
  | record_toplevel_decl
  | enum_toplevel_decl
  | fn_toplevel_decl

-- External declarations for JS interop
extern_fn_param_list:
  ident ":" type ("," ident ":" type)*

extern_decl:
   "extern" "fn" string ident "(" extern_fn_param_list ")" ":" type

record_toplevel_decl:
```

```
  "record" ident "=" "{" record_decl_field_list "}"
record_decl_field_list:
  ident ":" type ("," ident ":" type)*

enum_toplevel_decl:
  "enum" ident "=" "{" enum_decl_variant_list "}"
enum_decl_variant_list:
  ident ("," ident)*

fn_toplevel_decl:
  | "fn" ident (type_param_list)? "(" fn_param_list ")" ":" type? block
  | "fn" ident (type_param_list)? "(" fn_param_list ")" ":" type? "=" expr
fn_param_list:
  ident ":" type ("," ident ":" type)*

-- Root is the root of the syntax tree.
-- It can contain either top-level statements or regular statements.
root:
    toplevel_stmt*
  | stmt*
```