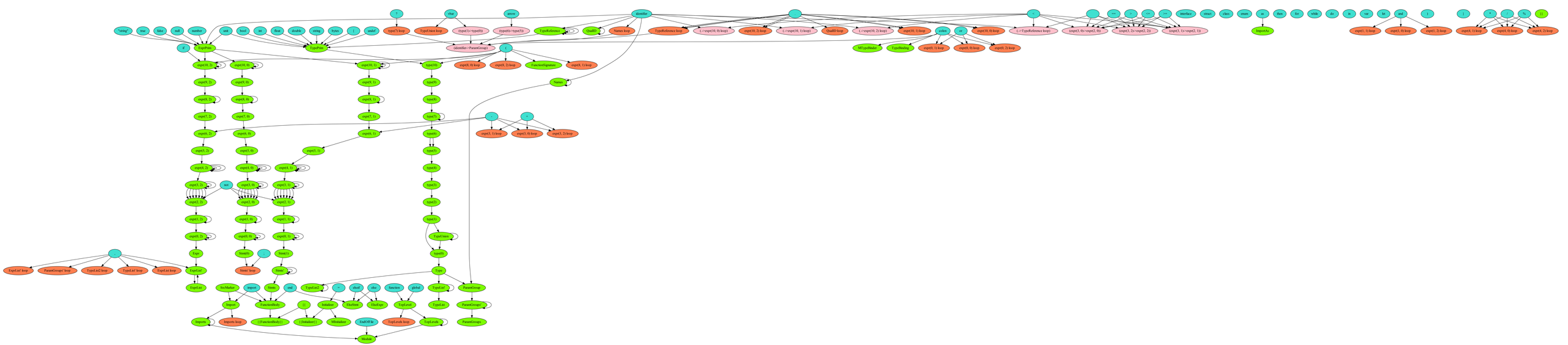
"string"
true
false
null
number
int
bool
int
float
double
string
bytes
{_}
undef
f
ExpPlus
typ(7) loop
TypeUnion loop
(typ(1)->typ(0))
(typ(6)->typ(5))
TypeReference*
Outdf
Names loop
TypeReference loop
identifier
(identifier->ParamGroup)
{_}
exp(8, 0) loop
exp(6, 2) loop
FunctionSignature
exp(0, 1) loop
Names*
exp(3, 1) loop
exp(1, 0) loop
exp(3, 2) loop
set
Eqp
ExpLit'
ExpLit
ExpLit' loop
ParamGroup' loop
TypeLit2 loop
TypeLit' loop
ExpLit loop
Stmt(0)
Stmt(1)
Stmt' loop
Stmt*
StrMarker
import
Stmt
end
TypeLit2
=
vkeif
else
function
global
TypeLit'
ParamGroup
import
FunctionBody
{{_}
InitMbr
ElseStmt
ElseExpr
TypeLit
ParamGroup'
Import*
Import loop
{{FunctionBody}}
{{InitMbr}}
MInitMbr
TypeLit.with loop
TypeLit.with
ParamGroups
EnOffRe
Module
Type
typ(1) loop
TypeLiteral
typ(0)
typ(5)
typ(4)
typ(3)
typ(2)
typ(7)
typ(6)
typ(8)
typ(9)
typ(10)
TypeBinding
MTypeBinding
exp(9, 1) loop
exp(9, 4) loop
exp(9, 2) loop
exp(10, 0) loop
exp(10, 1) loop
exp(10, 2) loop
(->exp(10, 0) loop)
(->exp(10, 1) loop)
(->exp(10, 2) loop)
OutdF loop
token
or
(->TypeReference loop)
(exp(3, 0)->exp(2, 0))
(exp(3, 2)->exp(2, 2))
(exp(3, 1)->exp(2, 1))
ImportAs
exp(1, 1) loop
exp(1, 0) loop
exp(1, 2) loop
exp(4, 1) loop
exp(4, 0) loop
exp(4, 2) loop
interface
struct
class
enum
as
then
for
while
do
in
var
let
and
}
]
%
(
%
))