

Setanta grammar specification

Below is the full specification for the *Setanta* syntax, as used by *tsPEG* to generate it's input parser

```

1  ———
2  import { Environment } from "./env";
3  import { callFunc , idxList , Value } from "./values";
4  import { unescapeChars } from "./litreacha";
5  import * as Asserts from "./asserts";
6  import * as Checks from "./checks";
7  import { orBinOp , orQuickBinOp , andBinOp , andQuickBinOp ,
8          binOpEvalFn , binOpQuickEvalFn } from "./binops";
9  import { objLookupsEval , postfixArgsEval , csArgsEval , prefEval , EvalFn } from "./
    evals";
10 import { qEvalToEval } from "./evals";
11 import * as Quick from "./quickevals";
12 ———
13 Program      := stmts=AsgnStmt* _
14 AsgnStmt     := IfStmt
15                | BlockStmt
16                | NuairStmt
17                | LeStmt
18                | CCSmt
19                | BrisStmt
20                | CtlchStmt
21                | GniomhStmt
22                | ToradhStmt
23                | AssgnStmt
24                | DefnStmt
25                | Expr
26 NonAsgnStmt  := IfStmt
27                | NuairStmt
28                | LeStmt
29                | CCSmt
30                | BrisStmt
31                | ToradhStmt
32                | BlockStmt
33                | AssgnStmt
34                | Expr
35 IfStmt       := _ 'm[áa]' \&gap expr=Expr \&gap stmt=NonAsgnStmt elsebranch={_ 'n[oó]
    ' \&gap stmt=NonAsgnStmt}?
36 BlockStmt   := _ '{' blk=AsgnStmt* _ '}'
37 NuairStmt   := _ 'nuair-a' expr=Expr \&gap stmt=NonAsgnStmt
38 LeStmt      := _ 'le' \&gap id=ID _ 'idir' _ '\(' strt=Expr _ ',' end=Expr step={_ ',
    ' step=Expr}? _ '\)' stmt=NonAsgnStmt
39 DefnStmt    := _ id=ID _ ':=' _ expr=Expr
40 AssgnStmt   := _ lhs=Postfix _ op=AsgnOp _ expr=Expr
41 GniomhStmt  := _ 'gn[ii]omh' \&gap id=ID _ '\(' args=CSIDs? _ '\)' _ '{'
42             stmts=AsgnStmt*
43             _ '}'
44 CtlchStmt   := _ 'creatlach' \&gap id=ID tuis={_ 'ó' id=ID}? _ '{'
45             gniomhs=GniomhStmt*
46             _ '}'
47 BrisStmt    := _ 'bris'

```

```

48 CCStmt      := _ 'chun-cinn'
49 ToradhStmt  := _ 'toradh' \&gap exp=Expr?
50 Expr        := And
51 And         := head=Or tail={_ '\\&' trm=Or}*
52             .evalfn = EvalFn { return andBinOp(this); }
53             .qeval = Quick.MaybeEv { return andQuickBinOp(this); }
54 Or          := head=Eq tail={_ '\\|' trm=Eq}*
55             .evalfn = EvalFn { return orBinOp(this) }
56             .qeval = Quick.MaybeEv { return orQuickBinOp(this); }
57 Eq          := head=Comp tail={_ op='[!]=' trm=Comp}*
58             .evalfn = EvalFn { return binOpEvalFn(this) }
59             .qeval = Quick.MaybeEv { return binOpQuickEvalFn(this); }
60 Comp        := head=Sum tail={_ op=Compare trm=Sum}*
61             .evalfn = EvalFn { return binOpEvalFn(this) }
62             .qeval = Quick.MaybeEv { return binOpQuickEvalFn(this); }
63 Sum         := head=Product tail={_ op=PlusMinus trm=Product}*
64             .evalfn = EvalFn { return binOpEvalFn(this) }
65             .qeval = Quick.MaybeEv { return binOpQuickEvalFn(this); }
66 Product     := head=Prefix tail={_ op=MulDiv trm=Prefix}*
67             .evalfn = EvalFn { return binOpEvalFn(this); }
68             .qeval = Quick.MaybeEv { return binOpQuickEvalFn(this); }
69 Prefix      := _ op='-|!?' pf=Postfix
70             .evalfn = EvalFn { return prefEval(this); }
71             .qeval = Quick.MaybeEv { return Quick.qPrefEval(this); }
72 Postfix     := at=ObjLookups ops=PostOp*
73             .evalfn = EvalFn { return postfixArgsEval(this); }
74             .qeval = Quick.MaybeEv { return Quick.qPostfixArgsEval(this); }
75 ObjLookups  := _ attrs={id=ID '@' !wspace}* root=Atom
76             .evalfn = EvalFn { return objLookupsEval(this); }
77             .qeval = Quick.MaybeEv { return Quick.qObjLookupsEval(this); }
78 PostOp      := '\\(' args=CSArgs? _ '\\)' | '\\[' expr=Expr '\\]'
79 Atom        := _ '\\(' trm=Expr '\\)'
80             .evalfn = EvalFn { return (env: Environment) => this.trm.evalfn(env);
81             }
82             .qeval = Quick.MaybeEv {
83             const childF = this.trm.qeval;
84             return childF === null ? null : childF.bind(this.trm);
85             }
86             | ID
87             | Litreacha
88             | Int
89             | Bool
90             | Neamhni
91             | ListLit
92 LSpec        := id=ID ops=PostOp*
93 ListLit      := _ '\\[' els=CSArgs? _ '\\]'
94             .evalfn = EvalFn {
95             return (env: Environment) => this.els ? this.els.evalfn(env) :
96             Promise.resolve([]);
97             }
98             .qeval = Quick.MaybeEv { return Quick.qListLitEval(this); }
99 CSArgs       := head=Expr tail={_ ',' exp=Expr}*
100             .evalfn = (env: Environment) => Promise<Value[]> { return csArgsEval(
101             this); }
102             .qeval = ((env: Environment) => Value[]) | null { return Quick.qCSArgsEval

```

```

    (this); }
100 CSIDs      := head=ID tail={_ ',' id=ID}*
101 ID         := _ !{Keyword gap} id='[a-zA-Z_áéíóúÁÉÍÓÚ]+'
102           .evalfn = EvalFn { return qEvalToEval(Quick.qIdEval(this.id)); }
103           .qeval = Quick.EvalFn { return Quick.qIdEval(this.id); }
104 Bool       := _ bool='f[ií]or|br[eé]ag'
105           .evalfn = EvalFn { return qEvalToEval(Quick.qBoolEval(this.bool)); }
106           .qeval = Quick.EvalFn { return Quick.qBoolEval(this.bool); }
107 Neamhni    := _ 'neamhn[ií]'
108           .evalfn = EvalFn { return () => Promise.resolve(null); }
109           .qeval = Quick.EvalFn { return () => null; }
110 Int        := _ int='-[0-9]+(?:\.[0-9]+)?'
111           .evalfn = EvalFn { return qEvalToEval(Quick.qIntEval(this.int)); }
112           .qeval = Quick.EvalFn { return Quick.qIntEval(this.int); }
113 Litreacha  := _ '\'' val='([\^\'\\]|\\.)*' '\''
114           .evalfn = EvalFn { return qEvalToEval(Quick.qLitreachaEval(this.val))
115                               ; }
116           .qeval = Quick.EvalFn { return Quick.qLitreachaEval(this.val); }
116 _         := wspace*
117 wspace     := '(?:\s|>--(?:?!--<).)*(--<|\n|$))'
118 gap       := { wspace | '[^a-zA-Z0-9áéíóúÁÉÍÓÚ]' }+ | '$'
119 PlusMinus := '\+|- '
120 AsgnOp    := '=|\+=|\*=|-|=|%=|\/='
121 MulDiv    := '\*|\/|\/|\/|\/|\/'
122 Compare   := '<|=|>|<|>'
123 Keyword   := 'm[áa]' | 'n[oó]' | 'nuair-a' | 'f[ií]or|br[eé]ag'
124           | 'gn[ií]omh' | 'chun-cinn' | 'neamhn[ií]' | 'toradh' | 'creatlach'

```