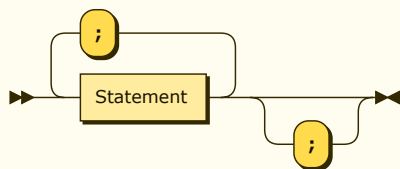
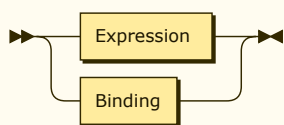


Statements:

```
Statements
  ::= Statement ( ';' Statement )* ';'?
```

referenced by:

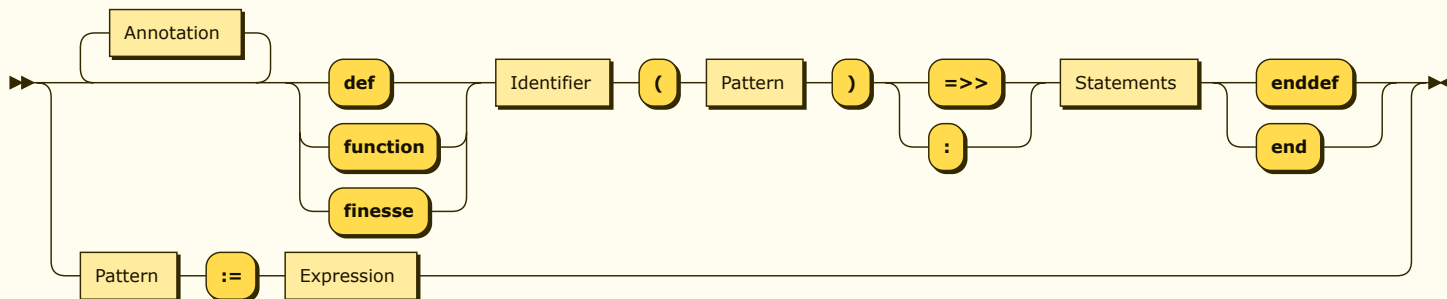
- [Binding](#)
- [IfExpression](#)
- [IfNotExpression](#)
- [LambdaExpression](#)
- [LetExpression](#)
- [LoopExpression](#)
- [SwitchExpression](#)

Statement:

```
Statement
  ::= Expression
  | Binding
```

referenced by:

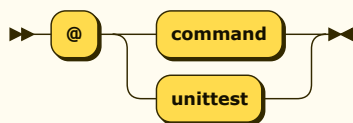
- [Statements](#)

Binding:

```
Binding ::= Pattern ':' Expression
  | Annotation* ( 'def' | 'function' | 'finesse' ) Identifier '(' Pattern ')' ( '=>' | ':' ) Statements ( 'enddef' | 'end' )
```

referenced by:

- [Query](#)
- [Statement](#)

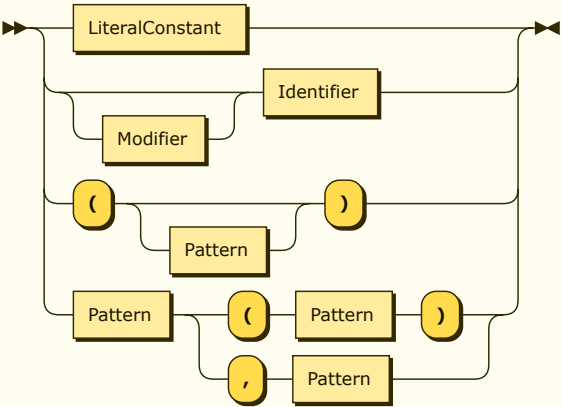
Annotation:

```
Annotation
  ::= '@' ( 'command' | 'unittest' )
```

referenced by:

- [Binding](#)

Pattern:

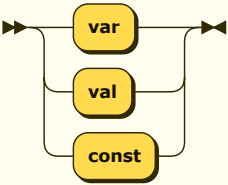


```
Pattern ::= LiteralConstant
        | Modifier? Identifier
        | '(' Pattern? ')'
        | Pattern ( '(' Pattern ')' | ',' Pattern )
```

referenced by:

- [Binding](#)
- [LambdaExpression](#)
- [Pattern](#)
- [Query](#)
- [SwitchExpression](#)

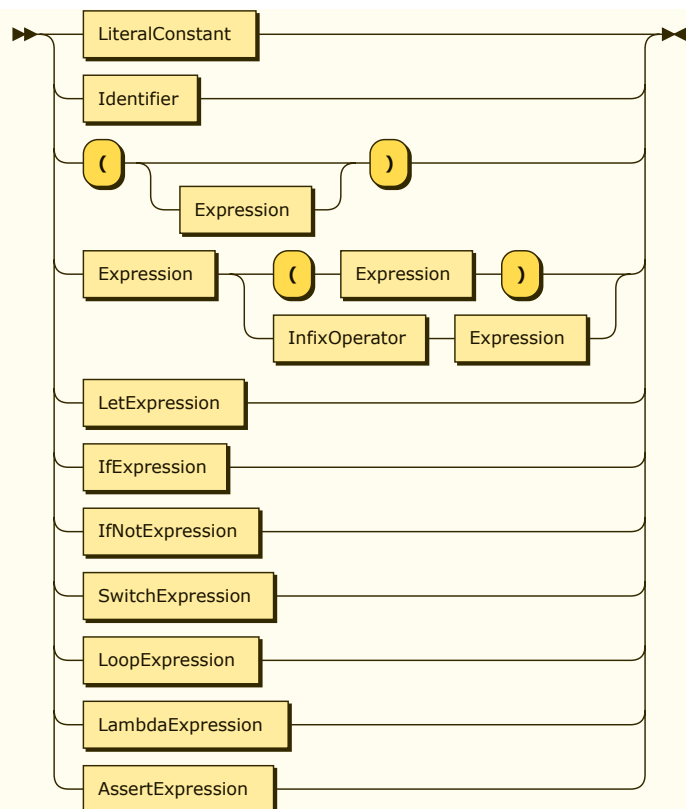
Modifer:



```
Modifer ::= 'var'
        | 'val'
        | 'const'
```

no references

Expression:



```

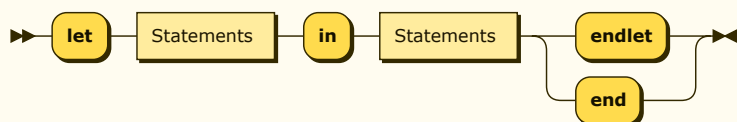
Expression
  ::= LiteralConstant
  | Identifier
  | '(' Expression? ')'
  | Expression ( '(' Expression ')' | InfixOperator Expression )
  | LetExpression
  | IfExpression
  | IfNotExpression
  | SwitchExpression
  | LoopExpression
  | LambdaExpression
  | AssertExpression

```

referenced by:

- [AssertExpression](#)
- [Binding](#)
- [Expression](#)
- [IfExpression](#)
- [IfNotExpression](#)
- [Query](#)
- [Statement](#)
- [SwitchExpression](#)

LetExpression:



```

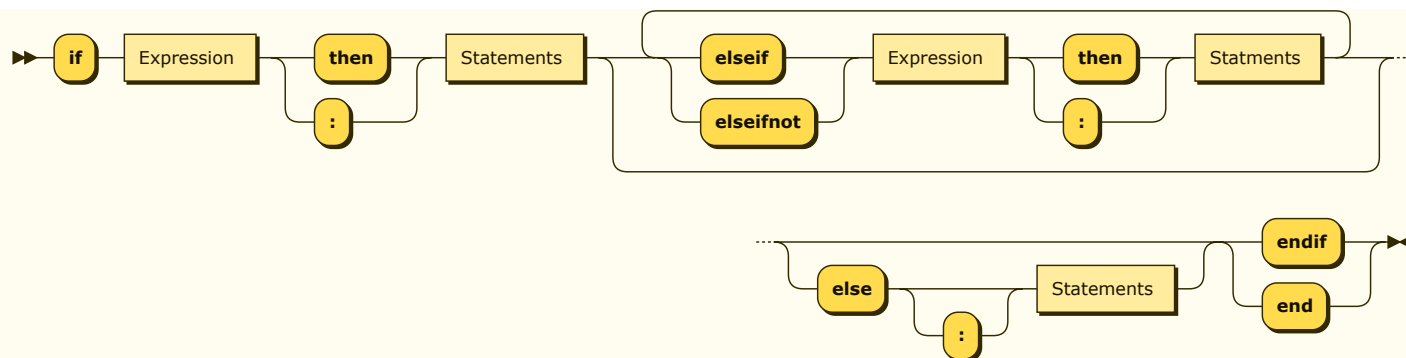
LetExpression
  ::= 'let' Statements 'in' Statements ( 'endlet' | 'end' )

```

referenced by:

- [Expression](#)

IfExpression:



```

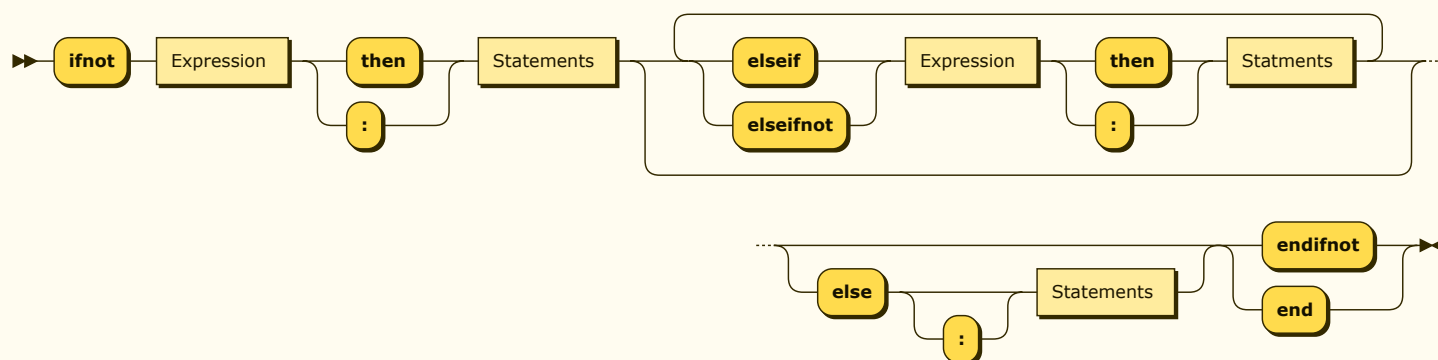
IfExpression
  ::= 'if' Expression ( 'then' | ':' ) Statements ( ( 'elseif' | 'elseifnot' ) Expression ( 'then' | ':' ) Statements )* ( 'else' ':'?
  Statements )? ( 'endif' | 'end' )

```

referenced by:

- [Expression](#)

IfNotExpression:



```

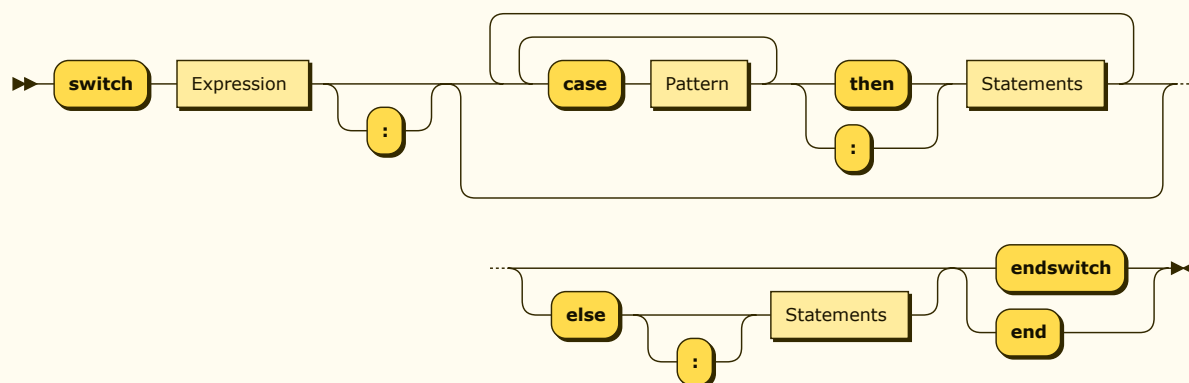
IfNotExpression
  ::= 'ifnot' Expression ( 'then' | ':' ) Statements ( ( 'elseif' | 'elseifnot' ) Expression ( 'then' | ':' ) Statements )* ( 'else' ':'?
  Statements )? ( 'endifnot' | 'end' )

```

referenced by:

- [Expression](#)

SwitchExpression:



```

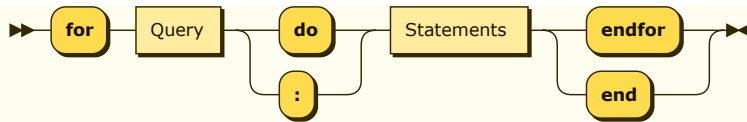
SwitchExpression
  ::= 'switch' Expression ':'? ( ( 'case' Pattern )+ ( 'then' | ':' ) Statements )* ( 'else' ':'? Statements )? ( 'endswitch' | 'end' )

```

referenced by:

- [Expression](#)

LoopExpression:

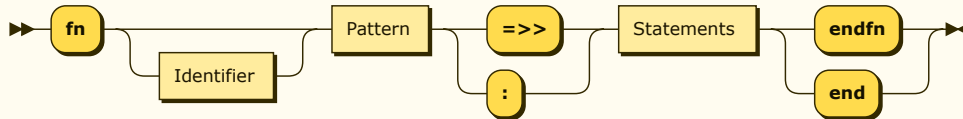


LoopExpression
 ::= 'for' Query ('do' | ':') Statements ('endfor' | 'end')

referenced by:

- [Expression](#)

LambdaExpression:

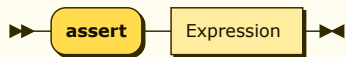


LambdaExpression
 ::= 'fn' Identifier? Pattern ('=>' | ':') Statements ('endfn' | 'end')

referenced by:

- [Expression](#)

AssertExpression:

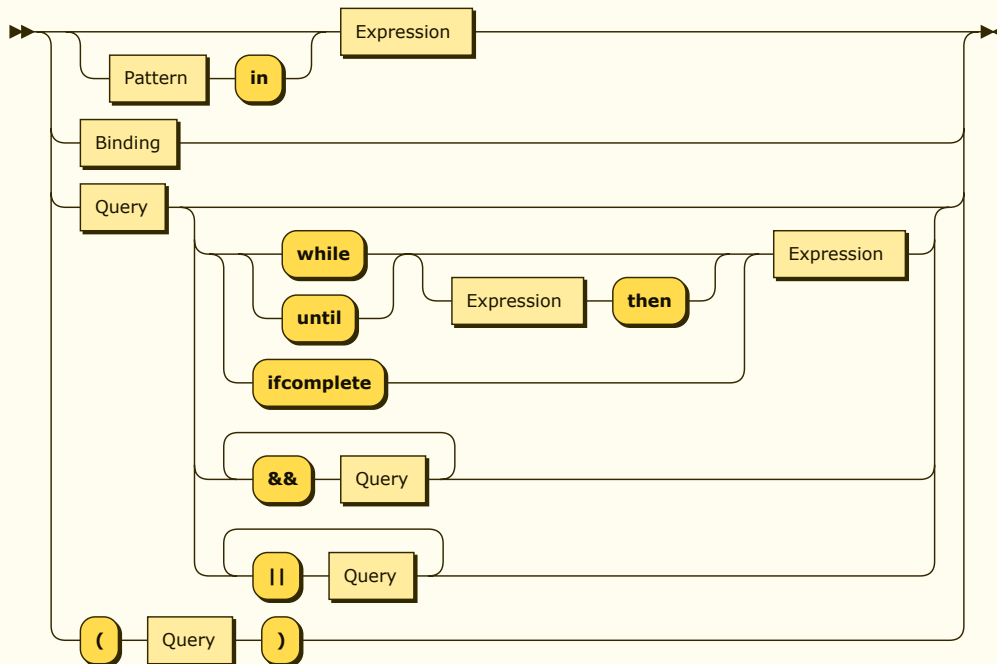


AssertExpression
 ::= 'assert' Expression

referenced by:

- [Expression](#)

Query:

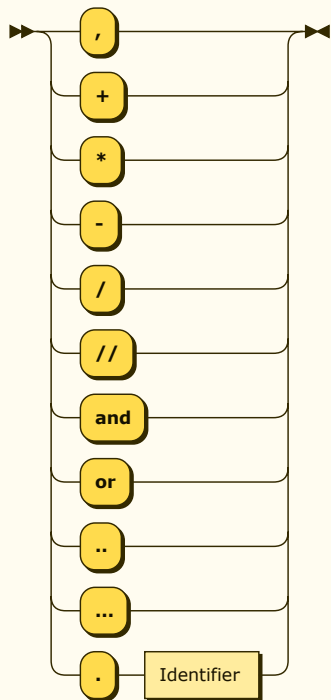


Query ::= (Pattern 'in')? Expression
 | Binding
 | Query (('while' | 'until') (Expression 'then')? | 'ifcomplete') Expression | ('&&' Query)* | ('||' Query)+)
 | '(' Query ')'

referenced by:

- [LoopExpression](#)
- [Query](#)

InfixOperator:



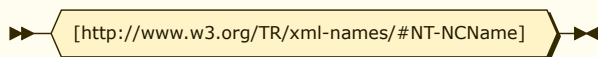
```

InfixOperator
    ::= '
        +
        *
        -
        /
        //'
        'and'
        'or'
        '..'
        '...'
        '.' Identifier
    
```

referenced by:

- [Expression](#)

Identifier:



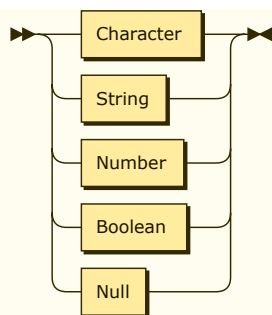
```

Identifier
    ::= [http://www.w3.org/TR/xml-names/#NT-NCName]
    
```

referenced by:

- [Binding](#)
- [Expression](#)
- [InfixOperator](#)
- [LambdaExpression](#)
- [Pattern](#)

LiteralConstant:



```
LiteralConstant
  ::= Character
  | String
  | Number
  | Boolean
  | Null
```

referenced by:

- [Expression](#)
- [Pattern](#)