A production may be parameterized by a subscripted suffix of the form "$_{[parameters]}$", which may appear after the nonterminal symbol defined by the production. "$_{parameters}$" may be either a single name or a comma separated list of names. A parameterized production is a short hand for a set of productions defining all combinations of the parameter names appended to the parameterized nonterminal symbol. This means that:

StatementList$_{[Return]}$ **:**
    ReturnStatement
    ExpressionStatement

is a convenient abbreviation for:

StatementList **:**
    ReturnStatement
    ExpressionStatement

StatementListReturn **:**
    ReturnStatement
    ExpressionStatement

and that:

StatementList$_{[Return, In]}$ **:**
    ReturnStatement
    ExpressionStatement

is a convenient abbreviation for:

StatementList **:**
    ReturnStatement
    ExpressionStatement

StatementListReturn **:**
    ReturnStatement
    ExpressionStatement

StatementListIn **:**
    ReturnStatement
    ExpressionStatement

StatementListReturnIn **:**
    ReturnStatement
    ExpressionStatement

References to nonterminals on the right hand side of a production can also be parameterized. For example:

StatementList **:**
    ReturnStatement
    ExpressionStatement$_{[In]}$

is equivalent to saying:

StatementList **:**
    ReturnStatement
    ExpressionStatementIn

A nonterminal reference may have both a parameter list and an "$_{opt}$" suffix. For example:

VariableDeclaration **:**
BindingIdentifier Initialiser$_{[In]opt}$

is a convenient abbreviation for:

VariableDeclaration **:**
BindingIdentifier
BindingIdentifier InitialiserIn

The prefixing a parameter name with "$_{?}$"on a right hand side nonterminal reference make that parameter value dependent upon the occurrence of the parameter name on the reference to the current productions symbol. For example:

VariableDeclaration$_{[In]}$ **:**
BindingIdentifier Initialiser$_{[?In]}$

is a convenient abbreviation for:

VariableDeclaration **:**
BindingIdentifier Initialiser

VariableDeclarationIn **:**
BindingIdentifier InitialiserIn

If a right hand side alternative is prefixed with "[+parameter]" that alternative is only available is the named parameter was used in referencing the production's nonterminal symbol. If a right hand side alternative is prefixed with "[~parameter]" that alternative is only available is the named parameter was *not* used in referencing the production's nonterminal symbol. This means that:

StatementList$_{[Return]}$ **:**
[+Return] *ReturnStatement*
ExpressionStatement

is an abbreviation for:

StatementList **:**
ExpressionStatement

StatementListReturn **:**
ReturnStatement
ExpressionStatement

and that

StatementList$_{[Return]}$ **:**
[~Return] *ReturnStatement*
ExpressionStatement

is an abbreviation for:

StatementList **:**
ReturnStatement
ExpressionStatement

StatementListReturn **:**
ExpressionStatement

And here is a sample of the real ES grammar

PrimaryExpression[yield] **:**
this
IdentifierReference[?yield]
Literal
ArrayInitialiser[?yield]
ObjectLiteral[?yield]
FunctionExpression
ClassExpression
GeneratorExpression
GeneratorComprehension[?yield]
RegularExpressionLiteral
TemplateLiteral[?yield]
CoverParenthesisedExpressionAndArrowParameterList[?yield]

CoverParenthesisedExpressionAndArrowParameterList[yield] **:**
(*Expression*[in, ?yield] **)**
( )
(... *BindingIdentifier*[?yield] **)**
(*Expression*[in, ?yield] **, ...** *BindingIdentifier*[?yield] **)**

IdentifierReference[yield] **:**
Identifier
[~yield] **yield**

LexicalDeclaration[in, yield] **:**
LetOrConst BindingList[?in, ?yield] **;**

LetOrConst **:**
let
const

BindingList[in, yield] **:**
LexicalBinding[?in, ?yield]
BindingList[?in, ?yield] **,** *LexicalBinding*[?in, ?yield]

LexicalBinding[in, yield] **:**
BindingIdentifier[?yield] *Initialiser*[?in, ?yield]opt
BindingPattern[?yield] *Initialiser*[?in, ?yield]

BindingIdentifier[default, yield] **:**
[~default] **default**
[~yield] **yield**
Identifier