Text Comparison

Documents Compared

grammar.pdf

grammar.pdf

Summary

873 word(s) added

746 word(s) deleted

5999 word(s) matched

51 block(s) matched

To see where the changes are, scroll down.

grammar.pdf

~~ECMAScript 4th Edition Grammar~~

~~##~~ ~~ED SURFACE SYNTAX~~ ~~(## - production number; ED - edition introduced)~~

TEXT STRUCTURE

Line terminator normalization

1 The character sequence CRLF, and the single characters CR, LS, and PS, are all converted to a single LF character, in all source contexts, before tokenization takes place.

Cf stripping (Compatibility Note)

2 Format Control characters (category Cf in the Unicode database) will no longer be stripped from the source text of a program [see Ecma-262 section 7.1]

Byte order mark (BOM) handling

3 The character sequences for BOM shall be replaced with a single white space character ~~before tokenization takes place.~~

Unicode escapes

4 The escape sequence of the form **\u{n..n}** will be replace by the unicode character whose code point is the value of the hexidecimal number between **{** and **}**

LEXICAL STRUCTURE

ReservedIdentifier [one of]

1 **break case cast catch class const continue debugger default delete do dynamic else false final finally for function if in instanceof interface is let like namespace native new null override return static super switch this throw true try type typeof use var void while with yield __proto__**

ContextuallyReservedIdentifier [one of]

2 **each extends generator get implements set standard strict undefined**

Punctuator [one of]

3 **. .< … ! != !== % %= & &= && &&= * *= + += ++ - -= -- / /= < <= << <<= = == === > ~~>= >==~~ >> >>= >>> >>>= ^ ^= | |= || ||= : :: ( ) [ ] { } ~ , ; ?**

VirtualSemicolon

4 [If the first through the *n*th tokens of an ECMAScript program form are grammatically valid but the first through the n+1st tokens are not and there is a line break between the nth tokens and the n+1st tokens, then the parser tries to parse the program again after inserting a VirtualSemicolon token between the nth and the n+1st tokens]

Identifier

5 [see Ecma-262 section 7.6]

StringLiteral

6 [see Ecma-262 section 7.8.4]

7 [see Line continuations spec: http://wiki.ecmascript.org/doku.php?id=features_specs:line_continuation_in_strings]

DoubleLiteral

8 [see Ecma-262 section 7.8.3]

DecimalLiteral

9 [Literals that denote decimal objects can be expressed as numeric literals (see E262 sec 7.8.3) with a suffix “m”: 10m; 12.48m; 1.5e-7m. When these literals are evaluated they yield new instances of decimal objects]

ECMAScript 4th Edition Grammar

ID ED SURFACE SYNTAX

TEXT STRUCTURE

Line terminator normalization

1 The character sequence CRLF, and the single characters CR, LS, and PS, are all converted to a single LF character, in all source contexts, before tokenization takes place.

Cf stripping (Compatibility Note)

2 Format Control characters (category Cf in the Unicode database) will no longer be stripped from the source text of a program [see Ecma-262 section 7.1]

Byte order mark (BOM) handling

3 The character sequences for BOM shall be replaced with a single white space character (0x20) before tokenization takes place if the BOM occurs outside of a string or regular expression literal.

Unicode escapes

4 The escape sequence of the form **\u{n..n}** will be replace by the unicode character whose code point is the value of the hexidecimal number between **{** and **}**

LEXICAL STRUCTURE

ReservedIdentifier [one of]

1 **break case cast catch class const continue debugger default delete do dynamic else false final finally for function if in instanceof interface is let like namespace native new null override return static super switch this throw true try type typeof use var void while with yield __proto__**

ContextuallyReservedIdentifier [one of]

2 **each extends generator get implements set standard strict undefined**

Punctuator [one of]

3 **. .< ... ! != !== % %= & &= && &&= * *= + += ++ - -= -- / /= < <= << <<= = == === > >= >= >> >>= >>> >>>= ^ ^= | |= || ||= : :: ( ) [ ] { } ~ , ; ?**

VirtualSemicolon

4 [If the first through the *n*th tokens of an ECMAScript program form are grammatically valid but the first through the n+1st tokens are not and there is a line break between the nth tokens and the n+1st tokens, then the parser tries to parse the program again after inserting a VirtualSemicolon token between the nth and the n+1st tokens]

Identifier

5 [see Ecma-262 section 7.6]

StringLiteral

6 [see Ecma-262 section 7.8.4]

7 [see Line continuations spec: http://wiki.ecmascript.org/doku.php?id=features_specs:line_continuation_in_strings]

DoubleLiteral

8 [see Ecma-262 section 7.8.3]

DecimalLiteral

9 [Literals that denote decimal objects can be expressed as numeric literals (see E262 sec 7.8.3) with a suffix “m”: 10m; 12.48m; 1.5e-7m. When these literals are evaluated they yield new instances of decimal objects]

RegExpInitialiser

10 [see Ecma-262 section 7.8.5]

11 [see Extend RegExp: http://developer.mozilla.org/es4/proposals/extend_regexps.html]

12 [see Line continuations spec: http://wiki.ecmascript.org/doku.php?id=features_specs:line_continuation_in_strings]

PROGRAM STRUCTURE

EXPRESSIONS

α = { allowColon, noColon }
β = { allowIn, noIn }

Identifier

1 3 **Identifier**
2 3 **ContextuallyReservedIdentifier**

PropertyIdentifier

3 3 Identifier
4 4 **ReservedIdentifier**

NameExpression

5 3 Identifier
6 4 NamespaceExpression **::** PropertyIdentifier

NamespaceExpression

7 4 NameExpression
8 4 ***StringLiteral***

ParenExpression

9 3 **(** CommaExpression$^{allowColon, allowIn}$ **)**

FunctionExpression$^{\alpha, \beta}$

10 3 **function** PropertyIdentifier FunctionSignature FunctionExpressionBody$^{\alpha, \beta}$
11 3 **function** FunctionSignature FunctionExpressionBody$^{\alpha, \beta}$

FunctionExpressionBody$^{\alpha, \beta}$

12 3 **{** Directives^{local} **}**
13 4 CommaExpression$^{\alpha, \beta}$

ObjectInitialiser^{noColon}

14 3 InitialiserAttribute **{** FieldList **}**

ObjectInitialiser^{allowColon}

15 3 InitialiserAttribute **{** FieldList **}**
16 4 InitialiserAttribute **{** FieldList **}** **:** TypeExpression

FieldList

17 3 «empty»
18 3 Field
19 3 Field **,** FieldList

Field

~~20 3 InitialiserAttribute FieldName : AssignmentExpression$^{allowColon, allowIn}$~~

RegExpInitialiser

10 [see Ecma-262 section 7.8.5]

11 [see Extend RegExp: http://developer.mozilla.org/es4/proposals/extend_regexps.html]

12 [see Line continuations spec: http://wiki.ecmascript.org/doku.php?id=features_specs:line_continuation_in_strings]

PROGRAM STRUCTURE

EXPRESSIONS

α = { allowColon, noColon }
β = { allowIn, noIn }

Identifier

1 3 **Identifier**
2 3 **ContextuallyReservedIdentifier**

PropertyIdentifier

3 3 Identifier
4 4 **ReservedIdentifier**

NameExpression

5 3 Identifier
6 4 NamespaceExpression **::** PropertyIdentifier

NamespaceExpression

7 4 NameExpression
8 4 **StringLiteral**

ParenExpression

9 3 **(** CommaExpression$^{\text{allowColon, allowIn}}$ **)**

FunctionExpression$^{\alpha,\beta}$

10 3 **function** PropertyIdentifier FunctionSignature FunctionExpressionBody$^{\alpha,\beta}$
11 3 **function** FunctionSignature FunctionExpressionBody$^{\alpha,\beta}$

FunctionExpressionBody$^{\alpha,\beta}$

12 3 **{** Directives$^{\text{local}}$ **}**
13 4 [lookahead !{ **{** }] CommaExpression$^{\alpha,\beta}$

ObjectInitialiser$^{\text{noColon}}$

14 3 InitialiserAttribute **{** FieldList **}**

ObjectInitialiser$^{\text{allowColon}}$

15 3 InitialiserAttribute **{** FieldList **}**
16 4 InitialiserAttribute **{** FieldList **}** **:** TypeExpression

FieldList

17 3 «empty»
18 3 Field
19 3 Field **,** FieldList

Field

21 4 InitialiserAttribute **get** FieldName GetterSignature FunctionExpressionBody$^{\text{allowColon, allowIn}}$
22 4 InitialiserAttribute **set** FieldName SetterSignature FunctionExpressionBody$^{\text{allowColon, allowIn}}$
23 ~~3~~ **__proto__** **:** AssignmentExpression$^{\text{allowColon, allowIn}}$

InitialiserAttribute
24 3 «empty»
25 4 **const**
26 4 **var**

FieldName
27 3 NameExpression
28 3 **StringLiteral**
29 3 **NumberLiteral**
30 4 [lookahead !{**__proto__**}] **ReservedIdentifier**

ArrayInitialiser$^{\text{noColon}}$
31 3 InitialiserAttribute **[** ArrayElements **]**

ArrayInitialiser$^{\text{allowColon}}$
32 3 InitialiserAttribute **[** ArrayElements **]**
33 4 InitialiserAttribute **[** ArrayElements **]** **:** TypeExpression

ArrayElements
34 3 ArrayElementList
35 4 ArrayComprehension

ArrayElementList
36 3 «empty»
37 3 AssignmentExpression$^{\text{allowColon, allowIn}}$
38 4 SpreadExpression
39 3 **,** ArrayElementList
40 3 AssignmentExpression$^{\text{allowColon, allowIn}}$ **,** ArrayElementList

SpreadExpression
41 4 **...** AssignmentExpression$^{\text{allowColon, allowIn}}$

ArrayComprehension
42 4 AssignmentExpression$^{\text{allowColon, allowIn}}$ ComprehensionExpression

ComprehensionExpression
43 4 **for** **(** TypedPattern$^{\text{noIn}}$ **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** ComprehensionClause
44 4 **for** **each** **(** TypedPattern$^{\text{noIn}}$ **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** ComprehensionClause
45 4 **let** **(** LetBindingList **)** ComprehensionClause
46 4 **if** ParenExpression ComprehensionClause

ComprehensionClause
47 4 «empty»
48 4 ComprehensionExpression

PrimaryExpression$^{\alpha, \beta}$
49 3 **null**
50 3 **true**
~~51~~ ~~3~~ ~~**false**~~

20 3 InitialiserAttribute FieldName **:** AssignmentExpression$^{\text{allowColon, allowIn}}$
21 4 InitialiserAttribute **get** FieldName GetterSignature FunctionExpressionBody$^{\text{allowColon, allowIn}}$
22 4 InitialiserAttribute **set** FieldName SetterSignature FunctionExpressionBody$^{\text{allowColon, allowIn}}$
23 4 **__proto__** **:** AssignmentExpression$^{\text{allowColon, allowIn}}$

InitialiserAttribute
24 3 «empty»
25 4 **const**
26 4 **var**

FieldName
27 3 NameExpression
28 3 **StringLiteral**
29 3 **NumberLiteral**
30 4 [lookahead !{**__proto__**}] **ReservedIdentifier**

ArrayInitialiser$^{\text{noColon}}$
31 3 InitialiserAttribute **[** ArrayElements **]**

ArrayInitialiser$^{\text{allowColon}}$
32 3 InitialiserAttribute **[** ArrayElements **]**
33 4 InitialiserAttribute **[** ArrayElements **]** **:** TypeExpression

ArrayElements
34 3 ArrayElementList
35 4 ArrayComprehension

ArrayElementList
36 3 «empty»
37 3 AssignmentExpression$^{\text{allowColon, allowIn}}$
38 4 SpreadExpression
39 3 **,** ArrayElementList
40 3 AssignmentExpression$^{\text{allowColon, allowIn}}$ **,** ArrayElementList

SpreadExpression
41 4 **...** AssignmentExpression$^{\text{allowColon, allowIn}}$

ArrayComprehension
42 4 AssignmentExpression$^{\text{allowColon, allowIn}}$ ComprehensionExpression

ComprehensionExpression
43 4 **for** **(** TypedPattern$^{\text{noIn}}$ **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** ComprehensionClause
44 4 **for** **each** **(** TypedPattern$^{\text{noIn}}$ **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** ComprehensionClause
45 4 **let** **(** LetBindingList **)** ComprehensionClause
46 4 **if** ParenExpression ComprehensionClause

ComprehensionClause
47 4 «empty»
48 4 ComprehensionExpression

PrimaryExpression$^{\alpha, \beta}$
49 3 **null**
50 3 **true**

52	3	**DoubleLiteral**
53	4	**DecimalLiteral**
54	3	**StringLiteral**
55	3	**RegExpInitialiser**
56	3	ArrayInitialiser$^{\alpha}$
57	3	ObjectInitialiser$^{\alpha}$
58	3	FunctionExpression$^{\alpha, \beta}$
59	3	ThisExpression
60	4	LetExpression$^{\alpha, \beta}$
61	3	ParenExpression
62	3	NameExpression
		ThisExpression
63	3	**this**
64	4	**this** [no line break] **function**
65	4	**this** [no line break] **generator**
		LetExpression$^{\alpha, \beta}$
66	4	**let** **(** LetBindingList **)** CommaExpression$^{\alpha, \beta}$
		LetBindingList
67	4	«empty»
68	4	VariableBindingList$^{\text{allowIn}}$
		Arguments
69	3	**( )**
70	3	**(** SpreadExpression **)**
71	3	**(** ArgumentList **)**
72	3	**(** ArgumentList **,** SpreadExpression **)**
		ArgumentList
73	3	AssignmentExpression$^{\text{allowColon, allowIn}}$
74	3	ArgumentList **,** AssignmentExpression$^{\text{allowColon, allowIn}}$
		PropertyOperator
75	4	**.** **ReservedIdentifier**
76	3	**.** NameExpression
77	3	Brackets
78	4	TypeApplication
		Brackets
79	3	**[** CommaExpression$^{\text{noColon, allowIn}}$ **]**
80	4	**[** SliceExpression **]**
		SliceExpression
81	4	OptionalExpression$^{\text{noColon}}$ **:** OptionalExpression$^{\text{noColon}}$
82	4	OptionalExpression$^{\text{noColon}}$ **:** OptionalExpression$^{\text{noColon}}$ **:** OptionalExpression$^{\text{allowColon}}$
83	4	**::** OptionalExpression$^{\text{allowColon}}$
84	4	OptionalExpression$^{\text{noColon}}$ **::**
		OptionalExpression$^{\alpha}$
85	4	«empty»
~~86~~	~~4~~	~~CommaExpression$^{\alpha, \text{allowIn}}$~~

51 3 **false**
52 3 **DoubleLiteral**
53 4 **DecimalLiteral**
54 3 **StringLiteral**
55 3 **RegExpInitialiser**
56 3 ArrayInitialiser$^{\alpha}$
57 3 ObjectInitialiser$^{\alpha}$
58 3 FunctionExpression$^{\alpha, \beta}$
59 3 ThisExpression
60 4 LetExpression$^{\alpha, \beta}$
61 3 ParenExpression
62 3 NameExpression

ThisExpression
63 3 **this**
64 4 **this** [no line break] **function**
65 4 **this** [no line break] **generator**

LetExpression$^{\alpha, \beta}$
66 4 **let** **(** LetBindingList **)** CommaExpression$^{\alpha, \beta}$

LetBindingList
67 4 «empty»
68 4 VariableBindingList^{allowIn}

Arguments
69 3 **( )**
70 3 **(** SpreadExpression **)**
71 3 **(** ArgumentList **)**
72 3 **(** ArgumentList **,** SpreadExpression **)**

ArgumentList
73 3 AssignmentExpression$^{allowColon, allowIn}$
74 3 ArgumentList **,** AssignmentExpression$^{allowColon, allowIn}$

PropertyOperator
75 4 **. ReservedIdentifier**
76 3 **.** NameExpression
77 3 Brackets
78 4 TypeApplication

Brackets
79 3 **[** CommaExpression$^{noColon, allowIn}$ **]**
80 4 **[** SliceExpression **]**

SliceExpression
81 4 OptionalExpression^{noColon} **:** OptionalExpression^{noColon}
82 4 OptionalExpression^{noColon} **:** OptionalExpression^{noColon} **:** OptionalExpression^{allowColon}
83 4 **::** OptionalExpression^{allowColon}
84 4 OptionalExpression^{noColon} **::**

OptionalExpression$^{\alpha}$
85 4 «empty»

TypeApplication

87 4 **.<** TypeExpressionList **>**

MemberExpression$^{\alpha,\beta}$

88 3 PrimaryExpression$^{\alpha,\beta}$

89 3 **new** MemberExpression$^{\alpha,\beta}$ Arguments

90 ~~4~~ ~~SuperExpression PropertyOperator~~

~~91~~ ~~3~~ ~~MemberExpression$^{\alpha,\beta}$ PropertyOperator~~

SuperExpression

92 4 **super**

93 4 **super** ParenExpression

CallExpression$^{\alpha,\beta}$

94 3 MemberExpression$^{\alpha,\beta}$ Arguments

95 3 CallExpression$^{\alpha,\beta}$ Arguments

96 3 CallExpression$^{\alpha,\beta}$ PropertyOperator

NewExpression$^{\alpha,\beta}$

97 3 MemberExpression$^{\alpha,\beta}$

98 3 **new** NewExpression$^{\alpha,\beta}$

LeftHandSideExpression$^{\alpha,\beta}$

99 3 NewExpression$^{\alpha,\beta}$

100 3 CallExpression$^{\alpha,\beta}$

PostfixExpression$^{\alpha,\beta}$

101 3 LeftHandSideExpression$^{\alpha,\beta}$

102 3 LeftHandSideExpression$^{\alpha,\beta}$ [no line break] **++**

103 3 LeftHandSideExpression$^{\alpha,\beta}$ [no line break] **--**

UnaryExpression$^{\alpha,\beta}$

104 3 PostfixExpression$^{\alpha,\beta}$

105 3 **delete** ~~PostfixExpression$^{\alpha}$~~$^{\beta}$

106 3 **void** UnaryExpression$^{\alpha,\beta}$

107 3 **typeof** UnaryExpression$^{\alpha,\beta}$

108 3 **++** ~~PostfixExpression$^{\alpha,\beta}$~~

~~109~~ ~~3~~ ~~**--** PostfixExpression$^{\alpha}$~~$^{\beta}$

110 3 **+** UnaryExpression$^{\alpha,\beta}$

111 3 **-** UnaryExpression$^{\alpha,\beta}$

112 3 **~** UnaryExpression$^{\alpha,\beta}$

113 3 **!** UnaryExpression$^{\alpha,\beta}$

114 4 **type** TypeExpression

MultiplicativeExpression$^{\alpha,\beta}$

115 3 UnaryExpression$^{\alpha,\beta}$

116 3 MultiplicativeExpression$^{\alpha,\beta}$ ***** UnaryExpression$^{\alpha,\beta}$

117 3 MultiplicativeExpression$^{\alpha,\beta}$ **/** UnaryExpression$^{\alpha,\beta}$

118 3 MultiplicativeExpression$^{\alpha,\beta}$ **%** UnaryExpression$^{\alpha,\beta}$

AdditiveExpression$^{\alpha,\beta}$

~~119~~ ~~3~~ ~~MultiplicativeExpression$^{\alpha,\beta}$~~

86 4 CommaExpression$^{\alpha, \text{allowIn}}$

TypeApplication

87 4 **.<** TypeExpressionList **>**

MemberExpression$^{\alpha, \beta}$

88 3 PrimaryExpression$^{\alpha, \beta}$

89 3 **new** MemberExpression$^{\alpha, \beta}$ Arguments

90 3 MemberExpression$^{\alpha, \beta}$ PropertyOperator

91 4 SuperExpression PropertyOperator

SuperExpression

92 4 **super**

93 4 **super** ParenExpression

CallExpression$^{\alpha, \beta}$

94 3 MemberExpression$^{\alpha, \beta}$ Arguments

95 3 CallExpression$^{\alpha, \beta}$ Arguments

96 3 CallExpression$^{\alpha, \beta}$ PropertyOperator

NewExpression$^{\alpha, \beta}$

97 3 MemberExpression$^{\alpha, \beta}$

98 3 **new** NewExpression$^{\alpha, \beta}$

LeftHandSideExpression$^{\alpha, \beta}$

99 3 NewExpression$^{\alpha, \beta}$

100 3 CallExpression$^{\alpha, \beta}$

PostfixExpression$^{\alpha, \beta}$

101 3 LeftHandSideExpression$^{\alpha, \beta}$

102 3 LeftHandSideExpression$^{\alpha, \beta}$ [no line break] **++**

103 3 LeftHandSideExpression$^{\alpha, \beta}$ [no line break] **--**

UnaryExpression$^{\alpha, \beta}$

104 3 PostfixExpression$^{\alpha, \beta}$

105 3 **delete** UnaryExpression$^{\alpha, \beta}$

106 3 **void** UnaryExpression$^{\alpha, \beta}$

107 3 **typeof** UnaryExpression$^{\alpha, \beta}$

108 3 **++** UnaryExpression$^{\alpha, \beta}$

109 3 **--** UnaryExpression$^{\alpha, \beta}$

110 3 **+** UnaryExpression$^{\alpha, \beta}$

111 3 **-** UnaryExpression$^{\alpha, \beta}$

112 3 **~** UnaryExpression$^{\alpha, \beta}$

113 3 **!** UnaryExpression$^{\alpha, \beta}$

114 4 **type** TypeExpression

MultiplicativeExpression$^{\alpha, \beta}$

115 3 UnaryExpression$^{\alpha, \beta}$

116 3 MultiplicativeExpression$^{\alpha, \beta}$ ***** UnaryExpression$^{\alpha, \beta}$

117 3 MultiplicativeExpression$^{\alpha, \beta}$ **/** UnaryExpression$^{\alpha, \beta}$

118 3 MultiplicativeExpression$^{\alpha, \beta}$ **%** UnaryExpression$^{\alpha, \beta}$

AdditiveExpression$^{\alpha, \beta}$

120	3	AdditiveExpression$^{\alpha, \beta}$ **+** MultiplicativeExpression$^{\alpha, \beta}$
121	3	AdditiveExpression$^{\alpha, \beta}$ **-** MultiplicativeExpression$^{\alpha, \beta}$
		ShiftExpression$^{\alpha, \beta}$
122	3	AdditiveExpression$^{\alpha, \beta}$
123	3	ShiftExpression$^{\alpha, \beta}$ **<<** AdditiveExpression$^{\alpha, \beta}$
124	3	ShiftExpression$^{\alpha, \beta}$ **>>** AdditiveExpression$^{\alpha, \beta}$
125	3	ShiftExpression$^{\alpha, \beta}$ **>>>** AdditiveExpression$^{\alpha, \beta}$
		RelationalExpression$^{\alpha, \beta}$
126	3	ShiftExpression$^{\alpha, \beta}$
127	3	RelationalExpression$^{\alpha, \beta}$ **<** ShiftExpression$^{\alpha, \beta}$
128	3	RelationalExpression$^{\alpha, \beta}$ **>** ShiftExpression$^{\alpha, \beta}$
129	3	RelationalExpression$^{\alpha, \beta}$ **<=** ShiftExpression$^{\alpha, \beta}$
130	3	RelationalExpression$^{\alpha, \beta}$ **>=** ShiftExpression$^{\alpha, \beta}$
131	3	RelationalExpression$^{\alpha, \beta}$ [β == allowIn] **in** ShiftExpression$^{\alpha, \beta}$
132	3	RelationalExpression$^{\alpha, \beta}$ **instanceof** ShiftExpression$^{\alpha, \beta}$
133	4	RelationalExpression$^{\alpha, \beta}$ **cast** TypeExpression
134	4	RelationalExpression$^{\alpha, \beta}$ **is** TypeExpression
135	4	RelationalExpression$^{\alpha, \beta}$ **like** TypeExpression
		EqualityExpression$^{\alpha, \beta}$
136	3	RelationalExpression$^{\alpha, \beta}$
137	3	EqualityExpression$^{\alpha, \beta}$ **==** RelationalExpression$^{\alpha, \beta}$
138	3	EqualityExpression$^{\alpha, \beta}$ **!=** RelationalExpression$^{\alpha, \beta}$
139	3	EqualityExpression$^{\alpha, \beta}$ **===** RelationalExpression$^{\alpha, \beta}$
140	3	EqualityExpression$^{\alpha, \beta}$ **!==** RelationalExpression$^{\alpha, \beta}$
		BitwiseAndExpression$^{\alpha, \beta}$
141	3	EqualityExpression$^{\alpha, \beta}$
142	3	BitwiseAndExpression$^{\alpha, \beta}$ **&** EqualityExpression$^{\alpha, \beta}$
		BitwiseXorExpression$^{\alpha, \beta}$
143	3	BitwiseAndExpression$^{\alpha, \beta}$
144	3	BitwiseXorExpression$^{\alpha, \beta}$ **^** BitwiseAndExpression$^{\alpha, \beta}$
		BitwiseOrExpression$^{\alpha, \beta}$
145	3	BitwiseXorExpression$^{\alpha, \beta}$
146	3	BitwiseOrExpression$^{\alpha, \beta}$ **|** BitwiseXorExpression$^{\alpha, \beta}$
		LogicalAndExpression$^{\alpha, \beta}$
147	3	BitwiseOrExpression$^{\alpha, \beta}$
148	3	LogicalAndExpression$^{\alpha, \beta}$ **&&** BitwiseOrExpression$^{\alpha, \beta}$
		LogicalOrExpression$^{\alpha, \beta}$
149	3	LogicalAndExpression$^{\alpha, \beta}$
150	3	LogicalOrExpression$^{\alpha, \beta}$ **||** LogicalAndExpression$^{\alpha, \beta}$
		ConditionalExpression$^{\alpha, \beta}$
151	4	YieldExpression$^{\alpha, \beta}$
152	3	LogicalOrExpression$^{\alpha, \beta}$
153	3	LogicalOrExpression$^{\alpha, \beta}$ **?** AssignmentExpression$^{\text{noColon}, \beta}$
~~154~~		~~**:** AssignmentExpression$^{\alpha, \beta}$~~

119 3 MultiplicativeExpression$^{\alpha, \beta}$
120 3 AdditiveExpression$^{\alpha, \beta}$ **+** MultiplicativeExpression$^{\alpha, \beta}$
121 3 AdditiveExpression$^{\alpha, \beta}$ **-** MultiplicativeExpression$^{\alpha, \beta}$

ShiftExpression$^{\alpha, \beta}$
122 3 AdditiveExpression$^{\alpha, \beta}$
123 3 ShiftExpression$^{\alpha, \beta}$ **<<** AdditiveExpression$^{\alpha, \beta}$
124 3 ShiftExpression$^{\alpha, \beta}$ **>>** AdditiveExpression$^{\alpha, \beta}$
125 3 ShiftExpression$^{\alpha, \beta}$ **>>>** AdditiveExpression$^{\alpha, \beta}$

RelationalExpression$^{\alpha, \beta}$
126 3 ShiftExpression$^{\alpha, \beta}$
127 3 RelationalExpression$^{\alpha, \beta}$ **<** ShiftExpression$^{\alpha, \beta}$
128 3 RelationalExpression$^{\alpha, \beta}$ **>** ShiftExpression$^{\alpha, \beta}$
129 3 RelationalExpression$^{\alpha, \beta}$ **<=** ShiftExpression$^{\alpha, \beta}$
130 3 RelationalExpression$^{\alpha, \beta}$ **>=** ShiftExpression$^{\alpha, \beta}$
131 3 RelationalExpression$^{\alpha, \beta}$ [β == allowIn] **in** ShiftExpression$^{\alpha, \beta}$
132 3 RelationalExpression$^{\alpha, \beta}$ **instanceof** ShiftExpression$^{\alpha, \beta}$
133 4 RelationalExpression$^{\alpha, \beta}$ **cast** TypeExpression
134 4 RelationalExpression$^{\alpha, \beta}$ **is** TypeExpression
135 4 RelationalExpression$^{\alpha, \beta}$ **like** TypeExpression

EqualityExpression$^{\alpha, \beta}$
136 3 RelationalExpression$^{\alpha, \beta}$
137 3 EqualityExpression$^{\alpha, \beta}$ **==** RelationalExpression$^{\alpha, \beta}$
138 3 EqualityExpression$^{\alpha, \beta}$ **!=** RelationalExpression$^{\alpha, \beta}$
139 3 EqualityExpression$^{\alpha, \beta}$ **===** RelationalExpression$^{\alpha, \beta}$
140 3 EqualityExpression$^{\alpha, \beta}$ **!==** RelationalExpression$^{\alpha, \beta}$

BitwiseAndExpression$^{\alpha, \beta}$
141 3 EqualityExpression$^{\alpha, \beta}$
142 3 BitwiseAndExpression$^{\alpha, \beta}$ & EqualityExpression$^{\alpha, \beta}$

BitwiseXorExpression$^{\alpha, \beta}$
143 3 BitwiseAndExpression$^{\alpha, \beta}$
144 3 BitwiseXorExpression$^{\alpha, \beta}$ **^** BitwiseAndExpression$^{\alpha, \beta}$

BitwiseOrExpression$^{\alpha, \beta}$
145 3 BitwiseXorExpression$^{\alpha, \beta}$
146 3 BitwiseOrExpression$^{\alpha, \beta}$ **|** BitwiseXorExpression$^{\alpha, \beta}$

LogicalAndExpression$^{\alpha, \beta}$
147 3 BitwiseOrExpression$^{\alpha, \beta}$
148 3 LogicalAndExpression$^{\alpha, \beta}$ **&&** BitwiseOrExpression$^{\alpha, \beta}$

LogicalOrExpression$^{\alpha, \beta}$
149 3 LogicalAndExpression$^{\alpha, \beta}$
150 3 LogicalOrExpression$^{\alpha, \beta}$ **||** LogicalAndExpression$^{\alpha, \beta}$

ConditionalExpression$^{\alpha, \beta}$
151 4 YieldExpression$^{\alpha, \beta}$
152 3 LogicalOrExpression$^{\alpha, \beta}$
153 3 LogicalOrExpression$^{\alpha, \beta}$ **?** AssignmentExpression$^{noColon, \beta}$

		NonAssignmentExpression$^{\alpha, \beta}$
155	4	YieldExpression$^{\alpha, \beta}$
156	3	LogicalOrExpression$^{\alpha, \beta}$
157	3	LogicalOrExpression$^{\alpha, \beta}$ **?** NonAssignmentExpression$^{noColon, \beta}$
158	3	**:** NonAssignmentExpression$^{\alpha, \beta}$
		YieldExpression$^{\alpha, \beta}$
159	4	**yield**
160	4	**yield** [no line break] AssignmentExpression$^{\alpha, \beta}$
		AssignmentExpression$^{\alpha, \beta}$
161	3	ConditionalExpression$^{\alpha, \beta}$
162	3	Pattern$^{\alpha, \beta, allowExpr}$ **=** AssignmentExpression$^{\alpha, \beta}$
163	3	SimplePattern$^{\alpha, \beta, allowExpr}$ CompoundAssignmentOperator AssignmentExpression$^{\alpha, \beta}$
		CompoundAssignmentOperator
164	3	***=**
165	3	**/=**
166	3	**%=**
167	3	**+=**
168	3	**-=**
169	3	**<<=**
170	3	**>>=**
171	3	**>>>=**
172	3	**&=**
173	3	**^=**
174	3	**|=**
175	~~3~~	~~**&&=**~~
~~176~~	~~3~~	**||=**
		CommaExpression$^{\alpha, \beta}$
177	3	AssignmentExpression$^{\alpha, \beta}$
178	3	CommaExpression$^{\alpha, \beta}$ **,** AssignmentExpression$^{\alpha, \beta}$

PATTERNS

γ = { allowExpr, noExpr }

		Pattern$^{\alpha, \beta, \gamma}$
179	3	SimplePattern$^{\alpha, \beta, \gamma}$
180	4	ObjectPattern$^{\alpha, \beta, \gamma}$
181	4	ArrayPattern$^{\gamma}$
		SimplePattern$^{\alpha, \beta, noExpr}$
182	3	Identifier
		SimplePattern$^{\alpha, \beta, allowExpr}$
183	3	LeftHandSideExpression$^{\alpha, \beta}$
		ObjectPattern$^{\gamma}$
184	4	**{** FieldListPattern$^{\gamma}$ **}**

154		**:** AssignmentExpression$^{\alpha, \beta}$
		NonAssignmentExpression$^{\alpha, \beta}$
155	4	YieldExpression$^{\alpha, \beta}$
156	3	LogicalOrExpression$^{\alpha, \beta}$
157	3	LogicalOrExpression$^{\alpha, \beta}$ **?** NonAssignmentExpression$^{\text{noColon}, \beta}$
158	3	**:** NonAssignmentExpression$^{\alpha, \beta}$
		YieldExpression$^{\alpha, \beta}$
159	4	**yield**
160	4	**yield** [no line break] AssignmentExpression$^{\alpha, \beta}$
		AssignmentExpression$^{\alpha, \beta}$
161	3	ConditionalExpression$^{\alpha, \beta}$
162	3	Pattern$^{\alpha, \beta, \text{allowExpr}}$ **=** AssignmentExpression$^{\alpha, \beta}$
163	3	SimplePattern$^{\alpha, \beta, \text{allowExpr}}$ CompoundAssignmentOperator AssignmentExpression$^{\alpha, \beta}$
		CompoundAssignmentOperator
164	3	***=**
165	3	**/=**
166	3	**%=**
167	3	**+=**
168	3	**-=**
169	3	**<<=**
170	3	**>>=**
171	3	**>>>=**
172	3	**&=**
173	3	**^=**
174	3	**|=**
175	4	**&&=**
176	4	**||=**
		CommaExpression$^{\alpha, \beta}$
177	3	AssignmentExpression$^{\alpha, \beta}$
178	3	CommaExpression$^{\alpha, \beta}$ **,** AssignmentExpression$^{\alpha, \beta}$

PATTERNS

γ = { allowExpr, noExpr }

		Pattern$^{\alpha, \beta, \gamma}$
179	3	SimplePattern$^{\alpha, \beta, \gamma}$
180	4	ObjectPattern$^{\alpha, \beta, \gamma}$
181	4	ArrayPattern$^{\gamma}$
		SimplePattern$^{\alpha, \beta, \text{noExpr}}$
182	3	Identifier
		SimplePattern$^{\alpha, \beta, \text{allowExpr}}$
183	3	LeftHandSideExpression$^{\alpha, \beta}$
		ObjectPattern$^{\gamma}$
184	4	**{** FieldListPattern$^{\gamma}$ **}**

FieldListPattern$^{\gamma}$

185 4 «empty»

186 4 FieldPattern$^{\gamma}$

187 4 FieldListPattern$^{\gamma}$ **,**

188 4 FieldListPattern$^{\gamma}$ **,** FieldPattern$^{\gamma}$

FieldPattern$^{\gamma}$

189 4 FieldName

190 4 FieldName **:** Pattern$^{\text{allowColon, allowIn, }\gamma}$

ArrayPattern$^{\gamma}$

191 4 **[** ElementListPattern$^{\gamma}$ **]**

ElementListPattern$^{\gamma}$

192 4 «empty»

193 4 ElementPattern$^{\gamma}$

194 4 **...** SimplePattern$^{\text{allowColon, allowIn, }\gamma}$

195 4 **,** ElementListPattern$^{\gamma}$

196 4 ElementPattern$^{\gamma}$ **,** ElementListPattern$^{\gamma}$

ElementPattern$^{\gamma}$

197 4 Pattern$^{\text{allowColon, allowIn, }\gamma}$

TypedIdentifier

198 3 ~~Identifier~~

~~199~~ ~~4~~ ~~Identifier~~ **:** TypeExpression

TypedPattern$^{\beta}$

200 3 Pattern$^{\text{noColon, }\beta\text{, noExpr}}$

201 4 Pattern$^{\text{noColon, }\beta\text{, noExpr}}$ **:** TypeExpression

LikenedPattern$^{\beta}$

202 4 Pattern$^{\text{noColon, }\beta\text{, noExpr}}$ **like** TypeExpression

TYPE EXPRESSIONS

TypeExpression

203 4 BasicTypeExpression

204 4 **?** BasicTypeExpression

205 4 **!** BasicTypeExpression

BasicTypeExpression

206 4 *****

207 4 **null**

208 4 **undefined**

209 4 TypeName

210 4 FunctionType

211 4 UnionType

212 4 RecordType

213 4 ArrayType

TypeName

~~214~~ ~~4~~ ~~NameExpression~~

FieldListPattern$^{\gamma}$

185 4 «empty»
186 4 FieldPattern$^{\gamma}$
187 4 FieldListPattern$^{\gamma}$ **,**
188 4 FieldListPattern$^{\gamma}$ **,** FieldPattern$^{\gamma}$

FieldPattern$^{\gamma}$

189 4 FieldName
190 4 FieldName **:** Pattern$^{\text{allowColon, allowIn, }\gamma}$

ArrayPattern$^{\gamma}$

191 4 **[** ElementListPattern$^{\gamma}$ **]**

ElementListPattern$^{\gamma}$

192 4 «empty»
193 4 ElementPattern$^{\gamma}$
194 4 **...** SimplePattern$^{\text{allowColon, allowIn, }\gamma}$
195 4 **,** ElementListPattern$^{\gamma}$
196 4 ElementPattern$^{\gamma}$ **,** ElementListPattern$^{\gamma}$

ElementPattern$^{\gamma}$

197 4 Pattern$^{\text{allowColon, allowIn, }\gamma}$

TypedIdentifier

198 3 PropertyIdentifier
199 4 PropertyIdentifier **:** TypeExpression

TypedPattern$^{\beta}$

200 3 Pattern$^{\text{noColon, }\beta\text{, noExpr}}$
201 4 Pattern$^{\text{noColon, }\beta\text{, noExpr}}$ **:** TypeExpression

LikenedPattern$^{\beta}$

202 4 Pattern$^{\text{noColon, }\beta\text{, noExpr}}$ **like** TypeExpression

TYPE EXPRESSIONS

TypeExpression

203 4 BasicTypeExpression
204 4 **?** BasicTypeExpression
205 4 **!** BasicTypeExpression

BasicTypeExpression

206 4 *
207 4 **null**
208 4 **undefined**
209 4 TypeName
210 4 FunctionType
211 4 UnionType
212 4 RecordType
213 4 ArrayType

TypeName

215 4 NameExpression TypeApplication

FunctionType

216 4 **function** FunctionSignatureType

FunctionSignatureType

217 4 TypeParameters **( )** ResultType

218 4 TypeParameters **(** ParametersType **)** ResultType

219 4 TypeParameters **(** **this** **:** TypeName **)** ResultType

220 4 TypeParameters **(** **this** **:** TypeName **,** ParametersType **)** ResultType

ParametersType

221 4 RestParameterType

222 4 NonRestParametersType

223 4 NonRestParametersType **,** RestParameterType

NonRestParametersType

224 4 ParameterType **,** NonRestParametersType

225 4 ParameterType

226 4 OptionalParametersType

OptionalParametersType

227 4 OptionalParameterType

228 4 OptionalParameterType **,** OptionalParametersType

OptionalParameterType

229 4 ParameterType **=**

ParameterType

230 4 TypeExpression

231 4 Identifier **:** TypeExpression

RestParameterType

232 4 **...**

233 4 **...** Identifier

UnionType

234 4 **(** TypeUnionList **)**

TypeUnionList

235 4 «empty»

236 4 NonemptyTypeUnionList

NonemptyTypeUnionList

237 4 TypeExpression

238 4 TypeExpression **|** NonemptyTypeUnionList

RecordType

239 4 **{** FieldTypeList **}**

FieldTypeList

240 4 «empty»

~~241~~ ~~4~~ ~~FieldType~~

214 4 NameExpression
215 4 NameExpression TypeApplication

FunctionType
216 4 **function** FunctionSignatureType

FunctionSignatureType
217 4 TypeParameters **( )** ResultType
218 4 TypeParameters **(** ParametersType **)** ResultType
219 4 TypeParameters **(** **this** **:** TypeName **)** ResultType
220 4 TypeParameters **(** **this** **:** TypeName **,** ParametersType **)** ResultType

ParametersType
221 4 RestParameterType
222 4 NonRestParametersType
223 4 NonRestParametersType **,** RestParameterType

NonRestParametersType
224 4 ParameterType **,** NonRestParametersType
225 4 ParameterType
226 4 OptionalParametersType

OptionalParametersType
227 4 OptionalParameterType
228 4 OptionalParameterType **,** OptionalParametersType

OptionalParameterType
229 4 ParameterType **=**

ParameterType
230 4 TypeExpression
231 4 Identifier **:** TypeExpression

RestParameterType
232 4 **...**
233 4 **...** Identifier

UnionType
234 4 **(** TypeUnionList **)**

TypeUnionList
235 4 «empty»
236 4 NonemptyTypeUnionList

NonemptyTypeUnionList
237 4 TypeExpression
238 4 TypeExpression **|** NonemptyTypeUnionList

RecordType
239 4 **{** FieldTypeList **}**

FieldTypeList
240 4 «empty»

242 4 FieldType **,** FieldTypeList

FieldType

243 4 FieldName

244 4 FieldName **:** TypeExpression

ArrayType

245 4 **[** ElementTypeList **]**

ElementTypeList

246 4 «empty»

247 4 TypeExpression

248 4 **...** TypeExpression

249 4 **,** ElementTypeList

250 4 TypeExpression **,** ElementTypeList

TypeExpressionList

251 4 TypeExpression

252 4 TypeExpressionList **,** TypeExpression

STATEMENTS

τ = { constructor, class, global, interface, local, statement }

ω = { abbrev, noShortIf, full }

Statement$^{\omega}$

253 3 BlockStatement

254 3 BreakStatement Semicolon$^{\omega}$

255 3 ContinueStatement Semicolon$^{\omega}$

256 3 DoWhileStatement ~~Semicolon$^{\omega}$~~

257 3 EmptyStatement

258 3 ExpressionStatement Semicolon$^{\omega}$

259 3 ForStatement$^{\omega}$

260 3 IfStatement$^{\omega}$

261 3 LabeledStatement$^{\omega}$

262 4 LetBlockStatement

263 3 ReturnStatement Semicolon$^{\omega}$

264 3 SwitchStatement

265 4 SwitchTypeStatement

266 3 ThrowStatement Semicolon$^{\omega}$

267 3 TryStatement

268 3 WhileStatement$^{\omega}$

269 3 WithStatement$^{\omega}$

Substatement$^{\omega}$

270 3 Statement$^{\omega}$

271 3 VariableDefinition$^{\text{noIn, statement}}$

Semicolon$^{\text{abbrev}}$

272 3 **;**

273 3 **VirtualSemicolon**

274 3 «empty»

241 4 FieldType
242 4 FieldType **,** FieldTypeList

FieldType
243 4 FieldName
244 4 FieldName **:** TypeExpression

ArrayType
245 4 **[** ElementTypeList **]**

ElementTypeList
246 4 «empty»
247 4 TypeExpression
248 4 **...** TypeExpression
249 4 **,** ElementTypeList
250 4 TypeExpression **,** ElementTypeList

TypeExpressionList
251 4 TypeExpression
252 4 TypeExpressionList **,** TypeExpression

STATEMENTS

τ = { constructor, class, global, interface, local, statement }
ω = { abbrev, noShortIf, full }

Statement$^{\omega}$
253 3 BlockStatement
254 3 BreakStatement Semicolon$^{\omega}$
255 3 ContinueStatement Semicolon$^{\omega}$
256 3 DoWhileStatement Semicolon^{abbrev}
257 3 EmptyStatement
258 3 ExpressionStatement Semicolon$^{\omega}$
259 3 ForStatement$^{\omega}$
260 3 IfStatement$^{\omega}$
261 3 LabeledStatement$^{\omega}$
262 4 LetBlockStatement
263 3 ReturnStatement Semicolon$^{\omega}$
264 3 SwitchStatement
265 4 SwitchTypeStatement
266 3 ThrowStatement Semicolon$^{\omega}$
267 3 TryStatement
268 3 WhileStatement$^{\omega}$
269 3 WithStatement$^{\omega}$

Substatement$^{\omega}$
270 3 Statement$^{\omega}$
271 3 VariableDefinition$^{noIn,\ statement}$

Semicolon^{abbrev}
272 3 **;**
273 3 **VirtualSemicolon**
274 3 «empty»

Semicolon$^{\text{noShortIf}}$

275 3 **;**
276 3 **VirtualSemicolon**
277 3 «empty»

Semicolon$^{\text{full}}$

278 3 **;**
279 3 **VirtualSemicolon**

EmptyStatement

280 3 **;**

ExpressionStatement

281 3 [lookahead !{ **{**, **const**, **function**, **let**, **var** }] CommaExpression$^{\text{allowColon, allowIn}}$

BlockStatement

282 3 **{** Directives$^{\text{local}}$ **}**

LabeledStatement$^{\omega}$

283 3 Identifier **:** Substatement$^{\omega}$

LetBlockStatement

284 4 **let** **(** LetBindingList **)** **{** Directives$^{\text{local}}$ **}**

IfStatement$^{\text{abbrev}}$

285 3 **if** ParenExpression Substatement$^{\text{abbrev}}$
286 3 **if** ParenExpression Substatement$^{\text{noShortIf}}$ **else** Substatement$^{\text{abbrev}}$

IfStatement$^{\text{full}}$

287 3 **if** ParenExpression Substatement$^{\text{full}}$
288 3 **if** ParenExpression Substatement$^{\text{noShortIf}}$ **else** Substatement$^{\text{full}}$

IfStatement$^{\text{noShortIf}}$

289 3 **if** ParenExpression Substatement$^{\text{noShortIf}}$ **else** Substatement$^{\text{noShortIf}}$

WithStatement$^{\omega}$

290 3 **with** ParenExpression Substatement$^{\omega}$

SwitchStatement

291 3 **switch** ParenExpression **{** CaseElements **}**

CaseElements

292 3 CaseClauses$^{\text{full}}$ DefaultClause$^{\text{full}}$ CaseClauses$^{\text{abbrev}}$
293 3 CaseClauses$^{\text{full}}$ DefaultClause$^{\text{abbrev}}$
294 3 CaseClauses$^{\text{abbrev}}$

CaseClauses$^{\omega}$

295 3 «empty»
296 3 CaseClauses$^{\text{full}}$ CaseClause$^{\omega}$

CaseClause$^{\omega}$

297 3 **case** CommaExpression$^{\text{allowColon, allowIn}}$ **:** Directives$^{\text{local, }\omega}$

Semicolon$^{\text{noShortIf}}$

275 3 **;**

276 3 **VirtualSemicolon**

277 3 «empty»

Semicolon$^{\text{full}}$

278 3 **;**

279 3 **VirtualSemicolon**

EmptyStatement

280 3 **;**

ExpressionStatement

281 3 [lookahead !{ **{**, **const**, **function**, **let**, **type**, **var** }] CommaExpression$^{\text{allowColon, allowIn}}$

BlockStatement

282 3 **{** Directives$^{\text{local}}$ **}**

LabeledStatement$^{\omega}$

283 3 Identifier **:** Substatement$^{\omega}$

LetBlockStatement

284 4 **let** **(** LetBindingList **)** **{** Directives$^{\text{local}}$ **}**

IfStatement$^{\text{abbrev}}$

285 3 **if** ParenExpression Substatement$^{\text{abbrev}}$

286 3 **if** ParenExpression Substatement$^{\text{noShortIf}}$ **else** Substatement$^{\text{abbrev}}$

IfStatement$^{\text{full}}$

287 3 **if** ParenExpression Substatement$^{\text{full}}$

288 3 **if** ParenExpression Substatement$^{\text{noShortIf}}$ **else** Substatement$^{\text{full}}$

IfStatement$^{\text{noShortIf}}$

289 3 **if** ParenExpression Substatement$^{\text{noShortIf}}$ **else** Substatement$^{\text{noShortIf}}$

WithStatement$^{\omega}$

290 3 **with** ParenExpression Substatement$^{\omega}$

SwitchStatement

291 3 **switch** ParenExpression **{** CaseElements **}**

CaseElements

292 3 CaseClauses$^{\text{full}}$ DefaultClause$^{\text{full}}$ CaseClauses$^{\text{abbrev}}$

293 3 CaseClauses$^{\text{full}}$ DefaultClause$^{\text{abbrev}}$

294 3 CaseClauses$^{\text{abbrev}}$

CaseClauses$^{\omega}$

295 3 «empty»

296 3 CaseClauses$^{\text{full}}$ CaseClause$^{\omega}$

CaseClause$^{\omega}$

297 3 **case** CommaExpression$^{\text{allowColon, allowIn}}$ **:** Directives$^{\text{local, }\omega}$

DefaultClause$^{\omega}$

298 3 **default** **:** Directives$^{\text{local}, \omega}$

SwitchTypeStatement

299 4 **switch** **type** ParenExpression **{** TypeCaseElements **}**

TypeCaseElements

300 4 TypeCaseElement

301 4 TypeCaseElements TypeCaseElement

TypeCaseElement

302 4 **case** **(** TypedPattern$^{\text{allowColon, allowIn}}$ **)** **{** Directives$^{\text{local}}$ **}**

DoWhileStatement

303 3 **do** ~~Substatement$^{\text{abbrev}}$~~ **while** ParenExpression

WhileStatement$^{\omega}$

304 3 **while** ParenExpression Substatement$^{\omega}$

ForStatement$^{\omega}$

305 3 **for** **(** ForInitialiser **;** OptionalExpression$^{\text{allowColon}}$ **;** OptionalExpression$^{\text{allowColon}}$ **)** Substatement$^{\omega}$

306 3 **for** **(** ForInBinding **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** Substatement$^{\omega}$

307 4 **for** **each** **(** ForInBinding **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** Substatement$^{\omega}$

ForInitialiser

308 3 «empty»

309 3 CommaExpression$^{\text{allowColon, noIn}}$

310 3 VariableDefinition$^{\text{noIn}, \tau}$

ForInBinding

311 3 Pattern$^{\text{allowColon, noIn, allowExpr}}$

312 3 VariableDefinitionKind$^{\text{local}}$ VariableBinding$^{\text{noIn}}$

ContinueStatement

313 3 **continue**

314 3 **continue** [no line break] Identifier

BreakStatement

315 3 **break**

316 3 **break** [no line break] Identifier

ReturnStatement

317 3 **return**

318 3 **return** [no line break] CommaExpression$^{\text{allowColon, allowIn}}$

ThrowStatement

319 3 **throw** CommaExpression$^{\text{allowColon, allowIn}}$

TryStatement

320 3 **try** **{** Directives$^{\text{local}}$ **}** CatchClauses

321 3 **try** **{** Directives$^{\text{local}}$ **}** CatchClauses **finally** **{** Directives$^{\text{local}}$ **}**

322 3 **try** **{** Directives$^{\text{local}}$ **}** **finally** **{** Directives$^{\text{local}}$ **}**

DefaultClause$^{\omega}$
298 3 **default** **:** Directives$^{\text{local}, \omega}$

SwitchTypeStatement
299 4 **switch** **type** ParenExpression **{** TypeCaseElements **}**

TypeCaseElements
300 4 TypeCaseElement
301 4 TypeCaseElements TypeCaseElement

TypeCaseElement
302 4 **case** **(** TypedPattern$^{\text{allowColon, allowIn}}$ **)** **{** Directives$^{\text{local}}$ **}**

DoWhileStatement
303 3 **do** Substatement$^{\text{full}}$ **while** ParenExpression

WhileStatement$^{\omega}$
304 3 **while** ParenExpression Substatement$^{\omega}$

ForStatement$^{\omega}$
305 3 **for** **(** ForInitialiser **;** OptionalExpression$^{\text{allowColon}}$ **;** OptionalExpression$^{\text{allowColon}}$ **)** Substatement$^{\omega}$
306 3 **for** **(** ForInBinding **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** Substatement$^{\omega}$
307 4 **for** **each** **(** ForInBinding **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** Substatement$^{\omega}$

ForInitialiser
308 3 «empty»
309 3 CommaExpression$^{\text{allowColon, noIn}}$
310 3 VariableDefinition$^{\text{noIn}, \tau}$

ForInBinding
311 3 Pattern$^{\text{allowColon, noIn, allowExpr}}$
312 3 VariableDefinitionKind$^{\text{local}}$ VariableBinding$^{\text{noIn}}$

ContinueStatement
313 3 **continue**
314 3 **continue** [no line break] Identifier

BreakStatement
315 3 **break**
316 3 **break** [no line break] Identifier

ReturnStatement
317 3 **return**
318 3 **return** [no line break] CommaExpression$^{\text{allowColon, allowIn}}$

ThrowStatement
319 3 **throw** CommaExpression$^{\text{allowColon, allowIn}}$

TryStatement
320 3 **try** **{** Directives$^{\text{local}}$ **}** CatchClauses
321 3 **try** **{** Directives$^{\text{local}}$ **}** CatchClauses **finally** **{** Directives$^{\text{local}}$ **}**
322 3 **try** **{** Directives$^{\text{local}}$ **}** **finally** **{** Directives$^{\text{local}}$ **}**

CatchClauses

323 3 CatchClause

324 3 CatchClauses CatchClause

CatchClause

325 3 **catch** **(** Parameter **)** **{** Directives$^{\text{local}}$ **}**

SuperStatement

326 4 **super** ~~**(** Arguments **)**~~

DIRECTIVES

Directives$^{\tau}$

327 3 «empty»

328 3 DirectivesPrefix$^{\tau}$ Directive$^{\tau,\text{ abbrev}}$

DirectivesPrefix$^{\tau}$

329 3 «empty»

330 3 DirectivesPrefix$^{\tau}$ Directive$^{\tau,\text{ full}}$

Directive$^{\text{class},\,\omega}$

331 4 Pragma$^{\text{class}}$

332 4 **static** [no line break] **{** Directives$^{\text{local}}$ **}**

333 4 ~~AnnotatableDirective$^{\text{class},\,\omega}$~~

Directive$^{\text{interface},\,\omega}$

334 4 Pragma$^{\text{interface}}$

335 4 ~~AnnotatableDirective$^{\text{interface},\,\omega}$~~

Directive$^{\text{constructor},\,\omega}$

336 4 Pragma$^{\text{local}}$

337 4 SuperStatement Semicolon$^{\omega}$

338 4 Statement$^{\omega}$

339 4 ~~AnnotatableDirective$^{\text{local},\,\omega}$~~

Directive$^{\tau,\,\omega}$

340 4 Pragma$^{\tau}$

341 3 Statement$^{\omega}$

342 3 ~~AnnotatableDirective$^{\tau,\,\omega}$~~

~~AnnotatableDirective$^{\text{global},\,\omega}$~~

343 4 Attribute [no line break] ~~AnnotatableDirective$^{\text{global},\,\omega}$~~

344 3 VariableDefinition$^{\text{allowIn, global}}$ Semicolon$^{\omega}$

345 3 FunctionDefinition$^{\text{global},\,\omega}$

346 4 NamespaceDefinition Semicolon$^{\omega}$

347 4 ClassDeclaration Semicolon$^{\omega}$

348 4 ClassDefinition

349 4 InterfaceDeclaration Semicolon$^{\omega}$

350 4 InterfaceDefinition

351 4 TypeDeclaration Semicolon$^{\omega}$

352 4 TypeDefinition Semicolon$^{\omega}$

~~AnnotatableDirective$^{\text{class},\,\omega}$~~

CatchClauses

323 3 CatchClause

324 3 CatchClauses CatchClause

CatchClause

325 3 **catch** **(** Parameter **)** **{** Directives$^{\text{local}}$ **}**

SuperStatement

326 4 **super** Arguments

DIRECTIVES

Directives$^{\tau}$

327 3 «empty»

328 3 DirectivesPrefix$^{\tau}$ Directive$^{\tau,\text{ abbrev}}$

DirectivesPrefix$^{\tau}$

329 3 «empty»

330 3 DirectivesPrefix$^{\tau}$ Directive$^{\tau,\text{ full}}$

Directive$^{\text{class},\ \omega}$

331 4 Pragma$^{\text{class}}$

332 4 **static** [no line break] **{** Directives$^{\text{local}}$ **}**

333 4 AttributedDirective$^{\text{class},\ \omega}$

Directive$^{\text{interface},\ \omega}$

334 4 Pragma$^{\text{interface}}$

335 4 AttributedDirective$^{\text{interface},\ \omega}$

Directive$^{\text{constructor},\ \omega}$

336 4 Pragma$^{\text{local}}$

337 4 SuperStatement Semicolon$^{\omega}$

338 4 Statement$^{\omega}$

339 4 AttributedDirective$^{\text{local},\ \omega}$

Directive$^{\tau,\ \omega}$

340 4 Pragma$^{\tau}$

341 3 Statement$^{\omega}$

342 3 AttributedDirective$^{\tau,\ \omega}$

AttributedDirective$^{\text{global},\ \omega}$

343 4 Attribute [no line break] AttributedDirective$^{\text{global},\ \omega}$

344 3 VariableDefinition$^{\text{allowIn, global}}$ Semicolon$^{\omega}$

345 3 FunctionDefinition$^{\text{global},\ \omega}$

346 4 NamespaceDefinition Semicolon$^{\omega}$

347 4 ClassDeclaration Semicolon$^{\omega}$

348 4 ClassDefinition

349 4 InterfaceDeclaration Semicolon$^{\omega}$

350 4 InterfaceDefinition

351 4 TypeDeclaration Semicolon$^{\omega}$

352 4 TypeDefinition Semicolon$^{\omega}$

grammar.pdf

~~353~~ ~~4~~ ~~Attribute [no line break] AnnotatableDirective$^{class, \omega}$~~
354 4 VariableDefinition$^{allowIn, class}$ Semicolon$^{\omega}$
355 4 FunctionDefinition$^{class, \omega}$
356 4 NamespaceDefinition Semicolon$^{\omega}$
357 4 TypeDefinition Semicolon$^{\omega}$

~~AnnotatableDirective$^{interface, \omega}$~~
358 4 Attribute [no line break] ~~AnnotatableDirective^{interface}~~
~~359~~ ~~4~~ ~~FunctionDeclaration Semicolon$^{\omega}$~~

~~AnnotatableDirective$^{local, \omega}$~~
360 3 VariableDefinition$^{allowIn, local}$ Semicolon$^{\omega}$
361 3 FunctionDefinition$^{local, \omega}$

Attribute
362 4 NamespaceExpression
363 4 **dynamic**
364 4 **final**
365 4 **override**
366 4 **__proto__**
367 4 **static**

DEFINITIONS

VariableDefinition$^{\beta, \tau}$
368 3 VariableDefinitionKind$^{\tau}$ VariableBindingList$^{\beta}$

VariableDefinitionKind^{statement}
369 3 **var**

VariableDefinitionKind$^{\tau}$
370 4 **const**
371 4 **let**
372 3 **var**

VariableBindingList$^{\beta}$
373 3 VariableBinding$^{\beta}$
374 3 VariableBindingList$^{\beta}$ **,** VariableBinding$^{\beta}$

VariableBinding$^{\beta}$
375 3 TypedIdentifier
376 3 TypedPattern$^{\beta}$ VariableInitialisation$^{\beta}$

VariableInitialisation$^{\beta}$
377 3 **=** AssignmentExpression$^{allowColon, \beta}$

~~FunctionDeclaration~~
~~378~~ ~~4~~ ~~**function** PropertyIdentifier FunctionSignatureType~~
~~379~~ ~~4~~ ~~**function get** PropertyIdentifier GetterSignature~~
~~380~~ ~~4~~ ~~**function set** PropertyIdentifier SetterSignature~~

~~FunctionDefinition^{class}~~
~~381~~ 4 **function** Identifier [Identifier == outer classname] ConstructorSignature **{** Directives^{constructor} **}**

AttributedDirective$^{\text{class}, \omega}$
353 4 Attribute [no line break] AttributedDirective$^{\text{class}, \omega}$
354 4 VariableDefinition$^{\text{allowIn, class}}$ Semicolon$^{\omega}$
355 4 FunctionDefinition$^{\text{class}, \omega}$
356 4 NamespaceDefinition Semicolon$^{\omega}$
357 4 TypeDefinition Semicolon$^{\omega}$

AttributedDirective$^{\text{interface}, \omega}$
358 4 Attribute [no line break] AttributedDirective$^{\text{interface}, \omega}$
359 4 FunctionDeclaration$^{\text{interface}}$ Semicolon$^{\omega}$

AttributedDirective$^{\text{local}, \omega}$
360 3 VariableDefinition$^{\text{allowIn, local}}$ Semicolon$^{\omega}$
361 3 FunctionDefinition$^{\text{local}, \omega}$

Attribute
362 4 NamespaceExpression
363 4 **dynamic**
364 4 **final**
365 4 **override**
366 4 **__proto__**
367 4 **static**

DEFINITIONS

VariableDefinition$^{\beta, \tau}$
368 3 VariableDefinitionKind$^{\tau}$ VariableBindingList$^{\beta}$

VariableDefinitionKind$^{\text{statement}}$
369 3 **var**

VariableDefinitionKind$^{\tau}$
370 4 **const**
371 4 **let**
372 3 **var**

VariableBindingList$^{\beta}$
373 3 VariableBinding$^{\beta}$
374 3 VariableBindingList$^{\beta}$ **,** VariableBinding$^{\beta}$

VariableBinding$^{\beta}$
375 3 TypedIdentifier
376 3 TypedPattern$^{\beta}$ VariableInitialisation$^{\beta}$

VariableInitialisation$^{\beta}$
377 3 **=** AssignmentExpression$^{\text{allowColon}, \beta}$

FunctionDeclaration$^{\text{interface}}$
378 4 **function** PropertyIdentifier FunctionSignatureType

FunctionDeclaration$^{\tau}$
379 4 **function** PropertyIdentifier FunctionSignatureType
380 4 **function get** PropertyIdentifier GetterSignature

~~382~~ ~~4~~ ~~**function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}}$~~

~~383~~ ~~4~~ ~~**function get** PropertyIdentifier GetterSignature FunctionBody$^{\text{allowIn}}$~~

~~384~~ ~~4~~ ~~**function set** PropertyIdentifier SetterSignature FunctionBody$^{\text{allowIn}}$~~

~~385~~ ~~4~~ ~~**native** FunctionDeclaration~~

~~FunctionDefinition$^{\text{local}}$~~

~~386~~ ~~4~~ ~~**const function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}}$~~

~~387~~ 3 **function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}, \omega}$

FunctionDefinition$^{\tau, \omega}$

~~388~~ ~~4~~ ~~**const function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}}$~~

~~389~~ ~~3~~ ~~**function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}}$~~

~~390~~ ~~4~~ ~~**function get** PropertyIdentifier GetterSignature FunctionBody$^{\text{allowIn}}$~~

~~391~~ ~~4~~ ~~**function set** PropertyIdentifier SetterSignature FunctionBody$^{\text{allowIn}}$~~

~~392~~ ~~4~~ ~~**native** FunctionDeclaration~~

~~FunctionSignature~~

~~393~~ ~~3~~ ~~TypeParameters **( )** ResultTypeOrLike~~

~~394~~ ~~3~~ ~~TypeParameters **(** Parameters **)** ResultTypeOrLike~~

~~395~~ 4 TypeParameters **(this :** TypeName **)** ResultTypeOrLike

~~396~~ 4 TypeParameters **(this :** TypeName **,** Parameters **)** ResultTypeOrLike

GetterSignature

~~397~~ ~~4~~ ~~**( )** ResultTypeOrLike~~

~~SetterSignature~~

~~398~~ 4 **(** Parameter **)** ResultTypeVoid

FunctionBody$^{\alpha, \beta, \omega}$

~~399~~ ~~3~~ ~~**{** Directives$^{\text{local}}$ **}**~~

~~400~~ ~~4~~ ~~CommaExpression$^{\alpha, \beta}$ Semicolon$^{\omega}$~~

~~TypeParameters~~

~~401~~ ~~3~~ ~~«empty»~~

~~402~~ ~~4~~ ~~**.<** TypeParameterList **>**~~

~~TypeParameterList~~

~~403~~ ~~4~~ ~~Identifier~~

~~404~~ ~~4~~ ~~Identifier **,** TypeParameterList~~

~~Parameters~~

~~405~~ ~~4~~ ~~RestParameter~~

~~406~~ ~~3~~ ~~NonRestParameters~~

~~407~~ ~~4~~ ~~NonRestParameters **,** RestParameter~~

~~NonRestParameters~~

~~408~~ ~~3~~ ~~Parameter **,** NonRestParameters~~

~~409~~ ~~3~~ ~~Parameter~~

~~410~~ ~~3~~ ~~OptionalParameters~~

~~OptionalParameters~~

~~411~~ ~~4~~ ~~OptionalParameter~~

~~412~~ ~~4~~ ~~OptionalParameter **,** OptionalParameters~~

381 4 **function set** PropertyIdentifier SetterSignature

FunctionDefinition$^{\text{class}, \omega}$

382 4 **function** Identifier [Identifier == outer classname] ConstructorSignature **{** Directives$^{\text{constructor}}$ **}**

383 4 **function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}, \omega}$

384 4 **function get** PropertyIdentifier GetterSignature FunctionBody$^{\text{allowIn}, \omega}$

385 4 **function set** PropertyIdentifier SetterSignature FunctionBody$^{\text{allowIn}, \omega}$

386 4 **native** FunctionDeclaration$^{\text{class}}$

FunctionDefinition$^{\text{local}, \omega}$

387 4 **const function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}, \omega}$

388 3 **function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}, \omega}$

FunctionDefinition$^{\tau, \omega}$

389 4 **const function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}, \omega}$

390 3 **function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn}, \omega}$

391 4 **function get** PropertyIdentifier GetterSignature FunctionBody$^{\text{allowIn}, \omega}$

392 4 **function set** PropertyIdentifier SetterSignature FunctionBody$^{\text{allowIn}, \omega}$

393 4 **native** FunctionDeclaration$^{\tau}$

FunctionSignature

394 3 TypeParameters **( )** ResultTypeOrLike

395 3 TypeParameters **(** Parameters **)** ResultTypeOrLike

396 4 TypeParameters **(** **this** **:** TypeName **)** ResultTypeOrLike

397 4 TypeParameters **(** **this** **:** TypeName **,** Parameters **)** ResultTypeOrLike

GetterSignature

398 4 **( )** ResultTypeOrLike

SetterSignature

399 4 **(** Parameter **)** ResultTypeVoid

FunctionBody$^{\alpha, \beta, \omega}$

400 3 **{** Directives$^{\text{local}}$ **}**

401 4 [lookahead !{ **{** }] CommaExpression$^{\alpha, \beta}$ Semicolon$^{\omega}$

TypeParameters

402 3 «empty»

403 4 **.<** TypeParameterList **>**

TypeParameterList

404 4 Identifier

405 4 TypeParametersList **,** Identifier

Parameters

406 4 RestParameter

407 3 NonRestParameters

408 4 NonRestParameters **,** RestParameter

NonRestParameters

409 3 Parameter **,** NonRestParameters

410 3 Parameter

411 3 OptionalParameters

~~OptionalParameter~~

~~413~~ ~~4~~ ~~Parameter = NonAssignmentExpression~~$^{\text{allowIn}}$

~~Parameter~~

~~414~~ ~~3~~ ~~ParameterAttribute TypedPattern~~$^{\text{allowIn}}$

~~415~~ ~~4~~ ~~ParameterAttribute LikenedPattern~~$^{\text{allowIn}}$

~~ParameterAttribute~~

~~416~~ ~~3~~ ~~«empty»~~

~~417~~ ~~4~~ ~~**const**~~

~~RestParameter~~

~~418~~ ~~4~~ ~~**...**~~

~~419~~ ~~4~~ ~~**...** Identifier~~

~~ResultTypeOrLike~~

~~420~~ ~~3~~ ~~ResultType~~

~~421~~ ~~4~~ ~~**like** TypeExpression~~

~~ResultType~~

~~422~~ ~~3~~ ~~«empty»~~

~~423~~ ~~4~~ ~~**: void**~~

~~424~~ ~~4~~ ~~**:** TypeExpression~~

~~ResultTypeVoid~~

~~425~~ ~~4~~ ~~«empty»~~

~~426~~ ~~4~~ ~~**: void**~~

~~ConstructorSignature~~

~~427~~ ~~4~~ ~~**( )** ConstructorInitialiser~~

~~428~~ ~~4~~ ~~**(** Parameters **)** ConstructorInitialiser~~

~~ConstructorInitialiser~~

~~429~~ ~~4~~ ~~«empty»~~

~~430~~ ~~4~~ ~~SettingList~~

~~431~~ ~~4~~ ~~SettingList **,** SuperInitialiser~~

~~432~~ ~~4~~ ~~SuperInitialiser~~

~~SettingList~~

~~433~~ ~~4~~ ~~Setting~~

~~434~~ ~~4~~ ~~SettingList **,** Setting~~

~~Setting~~

~~435~~ ~~4~~ ~~Pattern~~$^{\text{allowIn, allowExpr}}$ ~~VariableInitialisation~~$^{\text{allowIn}}$

~~SuperInitialiser~~

~~436~~ ~~4~~ ~~**super** Arguments~~

~~ClassDeclaration~~

~~437~~ ~~4~~ ~~**class** Identifier TypeSignature~~

~~ClassDefinition~~

OptionalParameters

412 4 OptionalParameter

413 4 OptionalParameter   OptionalParameters

OptionalParameter

414 4 Parameter **=** NonAssignmentExpression^{allowIn}

Parameter

415 3 ParameterAttribute TypedPattern^{allowIn}

416 4 ParameterAttribute LikenedPattern^{allowIn}

ParameterAttribute

417 3 «empty»

418 4 **const**

RestParameter

419 4

420 4 **...** Identifier

ResultTypeOrLike

421 3 ResultType

422 4 **like** TypeExpression

ResultType

423 3 «empty»

424 4 **: void**

425 4 **:** TypeExpression

ResultTypeVoid

426 4 «empty»

427 4 **: void**

ConstructorSignature

428 4 **( )** ConstructorInitialiser

429 4 **(** Parameters **)** ConstructorInitialiser

ConstructorInitialiser

430 4 «empty»

431 4 SettingList

432 4 SettingList **,** SuperInitialiser

433 4 SuperInitialiser

SettingList

434 4 Setting

435 4 SettingList **,** Setting

Setting

436 4 Pattern$^{allowIn, allowExpr}$ VariableInitialisation^{allowIn}

SuperInitialiser

437 4 **super** Arguments

~~438~~ ~~4~~ ~~**class** Identifier TypeSignature ClassInheritance ClassBody~~

~~TypeSignature~~

~~439~~ ~~4~~ ~~TypeParameters~~

~~440~~ ~~4~~ ~~TypeParameters **!**~~

~~ClassInheritance~~

~~441~~ ~~4~~ ~~«empty»~~

~~442~~ ~~4~~ ~~**extends** TypeName~~

~~443~~ ~~4~~ ~~**implements** TypeNameList~~

~~444~~ ~~4~~ ~~**extends** TypeName **implements** TypeNameList~~

~~TypeNameList~~

~~445~~ ~~4~~ ~~TypeName~~

~~446~~ ~~4~~ ~~TypeNameList , TypeName~~

~~ClassBody~~

~~447~~ ~~4~~ ~~**{** $\text{Directives}^{\text{class}}$ **}**~~

~~InterfaceDeclaration~~

~~448~~ ~~4~~ ~~**interface** Identifier TypeSignature~~

~~InterfaceDefinition~~

~~449~~ ~~4~~ ~~**interface** Identifier TypeSignature InterfaceInheritance InterfaceBody~~

~~InterfaceInheritance~~

~~450~~ ~~4~~ ~~«empty»~~

~~451~~ ~~4~~ ~~**extends** TypeNameList~~

~~InterfaceBody~~

~~452~~ ~~4~~ ~~**{** $\text{Directives}^{\text{interface}}$ **}**~~

~~TypeDeclaration~~

~~453~~ ~~4~~ ~~**type** Identifier TypeSignature~~

~~TypeDefinition~~

~~454~~ ~~4~~ ~~**type** Identifier TypeSignature TypeInitialisation~~

~~TypeInitialisation~~

~~455~~ ~~4~~ ~~**=** TypeExpression~~

~~NamespaceDefinition~~

~~456~~ ~~4~~ ~~**namespace** Identifier NamespaceInitialisation~~

~~NamespaceInitialisation~~

~~457~~ ~~4~~ ~~«empty»~~

~~458~~ ~~4~~ ~~**=** NamespaceExpression~~

~~**PRAGMAS**~~

~~Pragma^{τ}~~

~~459~~ ~~4~~ ~~UsePragma^{τ} $\text{Semicolon}^{\text{full}}$~~

ClassDeclaration

438 4 **class** Identifier TypeSignature

ClassDefinition

439 4 **class** Identifier TypeSignature ClassInheritance ClassBody

TypeSignature

440 4 TypeParameters

441 4 TypeParameters **!**

ClassInheritance

442 4 «empty»

443 4 **extends** TypeName

444 4 **implements** TypeNameList

445 4 **extends** TypeName **implements** TypeNameList

TypeNameList

446 4 TypeName

447 4 TypeNameList **,** TypeName

ClassBody

448 4 **{** Directives^{class} **}**

InterfaceDeclaration

449 4 **interface** Identifier TypeSignature

InterfaceDefinition

450 4 **interface** Identifier TypeSignature InterfaceInheritance InterfaceBody

InterfaceInheritance

451 4 «empty»

452 4 **extends** TypeNameList

InterfaceBody

453 4 **{** Directives^{interface} **}**

TypeDeclaration

454 4 **type** Identifier TypeSignature

TypeDefinition

455 4 **type** Identifier TypeSignature TypeInitialisation

TypeInitialisation

456 4 **=** TypeExpression

NamespaceDefinition

457 4 **namespace** Identifier NamespaceInitialisation

NamespaceInitialisation

458 4 «empty»

459 4 **=** NamespaceExpression

PRAGMAS

~~UsePragma$^{\tau}$~~

~~460~~ ~~4~~ ~~**use** PragmaItems$^{\tau}$~~

~~PragmaItems$^{\tau}$~~

~~461~~ ~~4~~ ~~PragmaItem$^{\tau}$~~

~~462~~ ~~4~~ ~~PragmaItems$^{\tau}$, PragmaItem$^{\tau}$~~

~~PragmaItem^{local}~~

~~463~~ ~~4~~ ~~**namespace** NamespaceExpression~~

~~464~~ ~~4~~ ~~**strict**~~

~~PragmaItem^{global}~~

~~465~~ ~~4~~ ~~**default namespace** NamespaceExpression~~

~~466~~ ~~4~~ ~~**namespace** NamespaceExpression~~

~~467~~ ~~4~~ ~~**standard**~~

~~468~~ ~~4~~ ~~**strict**~~

~~PragmaItem$^{\tau}$~~

~~469~~ ~~4~~ ~~**default namespace** NamespaceExpression~~

~~470~~ ~~4~~ ~~**namespace** NamespaceExpression~~

~~471~~ ~~4~~ ~~**strict**~~

~~PROGRAMS~~

~~Program~~

~~472~~ ~~9~~ ~~Directives^{global}~~

Pragma$^{\tau}$

460 4 UsePragma$^{\tau}$ Semicolon$^{\text{full}}$

UsePragma$^{\tau}$

461 4 **use** PragmaItems$^{\tau}$

PragmaItems$^{\tau}$

462 4 PragmaItem$^{\tau}$

463 4 PragmaItems$^{\tau}$, PragmaItem$^{\tau}$

PragmaItem$^{\text{local}}$

464 4 **namespace** NamespaceExpression

465 4 **strict**

PragmaItem$^{\text{global}}$

466 4 **default namespace** NamespaceExpression

467 4 **namespace** NamespaceExpression

468 4 **standard**

469 4 **strict**

PragmaItem$^{\tau}$

470 4 **default namespace** NamespaceExpression

471 4 **namespace** NamespaceExpression

472 4 **strict**

PROGRAMS

Program

473 3 Directives$^{\text{global}}$

~~Revision History:~~

16-May-2008: Fix various entries in the edition column (38, 354, 355, 363, 387, 389, 415, 420, 422, 436); Allow parameter-less constructor definitions (427-428, 429-431)

10-May-2008: Add alpha to OptionalExpression (79-82, 83-84, 307); Replace inadvertently erased definition of LetBindingList; Replace ParenExpression with LetBindingList in ComprehensionExpression (45); Remove hack to handle >> and >>> in .< expressions (86, 87); Move lookahead restriction on __proto__ from NameExpression to ReservedIdentifier in FieldName (27, 30); Change allowColon to allowIn in TypedPattern and LikenedPattern (202-205); Add explicit syntax for native functions to FunctionDefinition (386-389, 392-395); Remove TypeParameter from GetterSignature and SetterSignature (400, 401); Change FunctionSignature to GetterSignature and SetterSignature in FunctionDefinition (388, 389, 394, 395); Insert comma in ConstructorInitialiser (433); Restrict use of 'use standard' to global code (470); Add use of EmptyStatement to Statement (255-270); Remove use of EmptyStatement from Substatement and Directive (272, 339, 344); Move unary 'type' expression to UnaryExpression and earse definition and uses of UnaryTypeExpression (104-113, 150, 155, 160); Remove tau parameter from Statement (255-270, 272, 341, 345)

05-May-2008: Remove paren expression qualifier from PrimaryName (7); Rename NamespaceName to NamespaceExpression (6, 8, 9, 366, 370, 376, 466, 471, 474, 475); Remove Brackets (); Rename BracketsOrSlice to Brackets (); Rename PrimaryName to NameExpression (); Replace TypeName with TypeExpression in initialiser annotations (17, 35); Remove structual type annotation on array and object initialisers (18, 36); Add InitialiserAttribute to getter and setter syntax in object initialisers (24, 25); Inline ArrayElement (40, 43, 46); Replace use of NonemptyLetBindingList with VariableBindingList (72); Erase definition of NonemptyLetBindingList (73, 74); Refactor FunctionTypeSignature and FunctionSignature to allow rest after this parameter (230-233, 400-402, 411-415); Replace occurances of Block with { Directives } (292, 294, 312, 330, 331, 332, 335, 455, 460); Remove definition of Block (478); Erase errant ':' (404); Remove unused ResultTypeBoolean (434-435); Add SuperStatement and Directive for constructor contexts; Allow Pragma wherever Directive is allowed (339, 341-346); Consolidate Attribute non-terminals (347, 357, 362, 366-376)

29-Apr-2008: Define NamespaceName; Use NamespaceName from 'use namespace', 'use default namespace', NamespaceInitialisation, qualifier expressions and Attribute (6, 359, 363, 369, 456, 462, 465, 466); Define ClassDeclaration, InterfaceDeclaration and TypeDeclaration and allow them in global code (343-349); Moved 'const', 'dynamic', 'final', 'interface', 'let', 'namespace', 'native', 'override', 'prototoype', 'static', 'use', and 'yield' from ContextuallyReservedIdentifier to ReservedIdentifier (lexical 1, 2); Rename TypeReference to TypeName and TypeReferenceList to TypeNameList (223, 224, 445, 446); Replace all uses of TypeReference, TypeReferenceList, and PrimaryName that are type names with TypeName (16, 34, 218, 227, 228, 394, 395, 442-446, 450); Rename 'prototype' to '__proto__' in Attribute (367); Move '__proto__' from ContextuallyReservedIdentifier to ReservedIdentifier (lexical: 1, 2); Remove [look ahead...] conditions in Attribute (359, 363); Add LetBlockStatement to Statement (261-275)

26-Apr-2008: Remove ambiguous production '. ParenExpression :: QualifiedNameIdentifer' in PropertyOperator (82); Remove stale use of PackageDefinition in AnnotatableDirective (349); Remove ParameterType without trailing '=' from OptionalParameterType (237); Refactored Parameters and ParametersType to allow a rest parameter as the only parameter (340, 407); Remove namespace and type definitions from local blocks (359, 360); Add Directive for class and interface blocks; Add DecimalLiteral to PrimaryExpression (55); Add lookahead condition to disambiguate PrimaryName from explicit identifiers in Attributes (361, 365); Replace FunctionName with Identifier in FunctionDeclaration (384); Add productions for getters and setters in FunctionDeclaration (384); Remove 'import' from ContextuallyReservedIdentifiers (2, lexical); Remove restriction disallowing 'let' in classes (374, 375); Allow ReservedIdentifiers as function identifiers (11, 384-394); Disallow 'use default namespace' in local blocks (336, 459-466); Remove the use of StringLiteral and NumberLiteral in QualifiedNameIdentifier and rename to PropertyIdentifier (5, 6); Move ! in TypeSignature from prefix to postfix position (441)

19-Apr-2008: Remove Qualifier non-terminal (3, 4); Remove PrimaryName that begins with Qualifier (4); Remove definition of ReservedNamespace (5-8); Replace uses of NamspaceAttribute with PrimaryName (378, 382, 388,); Remove definition of NamespaceAttribute (389-396); Add [no line break] to ReturnStatement (342); Move definition of gamma parameters to Patterns section; Add 'meta', 'reflect', 'intrinsic', 'iterator' and __proto__ to ContextuallyReservedIdentifiers (3, 4: lexical); Remove duplicate productions in RelationalExpression by adding an inline condition for beta == allowIn (150-158, 145); Allow Pragma anywhere in DirectivesPrefix (353); Remove definition of Pragmas (484, 485); Remove lingering use of ImportPragma in Pragma (487)

18-Apr-2008: Remove TypeParameter from ConstructorSignature (452, 453); Remove Brackets in QualifiedNameIdentifier (13); Change argument to Block in BlockStatement to 'local' (304); Removed lingering uses of 'external' from NamespaceAttributes (388, 394); Remove lingering E4X punctuators **</** and **/>** from (6, lexical); Change let and function expression forms to use CommaExpression instead of AssignmentExpression (22, 76, 423); Add productions for handling **>>** and **>>>** in TypeApplication (101); Add productions for handling **::** in SliceExpression (98); Disallow 'let' in class bodies (398)

Revision History:

08-Jul-2008: Change operand of unary 'delete', '++' and '--' from PostfixExpression to UnaryExpression (105, 108, 109); Add lookahead constraint to FunctionExpressionBody and FunctionBody (13, 401); Remove redundant parens on SuperStatement (326); Add 'type' to forbidden lookahead tokens in ExpressionStatement (281)

30-May-2008: Make TypeParametersList left recursive (404); Change omega parameter in DoWhileSatement from abbrev to full (303); Rename AnnotatableDirective to AttributedDirective (343-361); Change parameter to Semicolon after DoWhileStatement to abbrev (256); Remove '>==' from Punctuator (lexical 3); Remove getter and setter declarations from interface definitions (359, 378-380, 385, 392)

16-May-2008: Fix various entries in the edition column (38, 354, 355, 363, 387, 389, 415, 420, 422, 436); Allow parameter-less constructor definitions (427-428, 429-431)

10-May-2008: Add alpha to OptionalExpression (79-82, 83-84, 307); Replace inadvertently erased definition of LetBindingList; Replace ParenExpression with LetBindingList in ComprehensionExpression (45); Remove hack to handle >> and >>> in .< expressions (86, 87); Move lookahead restriction on __proto__ from NameExpression to ReservedIdentifier in FieldName (27, 30); Change allowColon to allowIn in TypedPattern and LikenedPattern (202-205); Add explicit syntax for native functions to FunctionDefinition (386-389, 392-395); Remove TypeParameter from GetterSignature and SetterSignature (400, 401); Change FunctionSignature to GetterSignature and SetterSignature in FunctionDefinition (388, 389, 394, 395); Insert comma in ConstructorInitialiser (433); Restrict use of 'use standard' to global code (470); Add use of EmptyStatement to Statement (255-270); Remove use of EmptyStatement from Substatement and Directive (272, 339, 344); Move unary 'type' expression to UnaryExpression and earse definition and uses of UnaryTypeExpression (104-113, 150, 155, 160); Remove tau parameter from Statement (255-270, 272, 341, 345)

05-May-2008: Remove paren expression qualifier from PrimaryName (7); Rename NamespaceName to NamespaceExpression (6, 8, 9, 366, 370, 376, 466, 471, 474, 475); Remove Brackets (); Rename BracketsOrSlice to Brackets (); Rename PrimaryName to NameExpression (); Replace TypeName with TypeExpression in initialiser annotations (17, 35); Remove structual type annotation on array and object initialisers (18, 36); Add InitialiserAttribute to getter and setter syntax in object initialisers (24, 25); Inline ArrayElement (40, 43, 46); Replace use of NonemptyLetBindingList with VariableBindingList (72); Erase definition of NonemptyLetBindingList (73, 74); Refactor FunctionTypeSignature and FunctionSignature to allow rest after this parameter (230-233, 400-402, 411-415); Replace occurances of Block with { Directives } (292, 294, 312, 330, 331, 332, 335, 455, 460); Remove definition of Block (478); Erase errant ':' (404); Remove unused ResultTypeBoolean (434-435); Add SuperStatement and Directive for constructor contexts; Allow Pragma wherever Directive is allowed (339, 341-346); Consolidate Attribute non-terminals (347, 357, 362, 366-376)

29-Apr-2008: Define NamespaceName; Use NamespaceName from 'use namespace', 'use default namespace', NamespaceInitialisation, qualifier expressions and Attribute (6, 359, 363, 369, 456, 462, 465, 466); Define ClassDeclaration, InterfaceDeclaration and TypeDeclaration and allow them in global code (343-349); Moved 'const', 'dynamic', 'final', 'interface', 'let', 'namespace', 'native', 'override', 'prototoype', 'static', 'use', and 'yield' from ContextuallyReservedIdentifier to ReservedIdentifier (lexical 1, 2); Rename TypeReference to TypeName and TypeReferenceList to TypeNameList (223, 224, 445, 446); Replace all uses of TypeReference, TypeReferenceList, and PrimaryName that are type names with TypeName (16, 34, 218, 227, 228, 394, 395, 442-446, 450); Rename 'prototype' to '__proto__' in Attribute (367); Move '__proto__' from ContextuallyReservedIdentifier to ReservedIdentifier (lexical: 1, 2); Remove [look ahead...] conditions in Attribute (359, 363); Add LetBlockStatement to Statement (261-275)

26-Apr-2008: Remove ambiguous production '. ParenExpression :: QualifiedNameIdentifer' in PropertyOperator (82); Remove stale use of PackageDefinition in AnnotatableDirective (349); Remove ParameterType without trailing '=' from OptionalParameterType (237); Refactored Parameters and ParametersType to allow a rest parameter as the only parameter (340, 407); Remove namespace and type definitions from local blocks (359, 360); Add Directive for class and interface blocks; Add DecimalLiteral to PrimaryExpression (55); Add lookahead condition to disambiguate PrimaryName from explicit identifiers in Attributes (361, 365); Replace FunctionName with Identifier in FunctionDeclaration (384); Add productions for getters and setters in FunctionDeclaration (384); Remove 'import' from ContextuallyReservedIdentifiers (2, lexical); Remove restriction disallowing 'let' in classes (374, 375); Allow ReservedIdentifiers as function identifiers (11, 384-394); Disallow 'use default namespace' in local blocks (336, 459-466); Remove the use of StringLiteral and NumberLiteral in QualifiedNameIdentifier and rename to PropertyIdentifier (5, 6); Move ! in TypeSignature from prefix to postfix position (441)

19-Apr-2008: Remove Qualifier non-terminal (3, 4); Remove PrimaryName that begins with Qualifier (4); Remove definition of ReservedNamespace (5-8); Replace uses of NamspaceAttribute with PrimaryName (378, 382, 388,); Remove definition of NamespaceAttribute (389-396); Add [no line break] to ReturnStatement (342); Move definition of gamma parameters to Patterns section; Add 'meta', 'reflect', 'intrinsic', 'iterator' and __proto__ to ContextuallyReservedIdentifiers (3, 4: lexical); Remove duplicate productions in RelationalExpression by adding an inline condition for beta == allowIn (150-158, 145); Allow Pragma anywhere in DirectivesPrefix (353); Remove definition of Pragmas (484, 485); Remove lingering use of ImportPragma in Pragma (487)

17-Apr-2008: Rename ElementComprehension to ArrayComprehension; Allow empty body of 'let' clause in ArrayComprehension; Add 'standard' as a pragma; Fix obligatory ',' bug in ArrayType; Allow only SimplePattern in RestParameter; Remove PackageDefinition; Remove ImportPragma; Remove 'external' from ReservedIdentifier and ReservedNamespace; Add 'Identifier : TypeExpression' to ParameterType; Replace TypeExpression with Identifier in RestParameterType; Removed 'meta::' productions from ObjectInitialiser; Remove ContextuallyReservedIdentifiers 'package', and 'xml'; (Re)-add ContextuallyReservedIdentifier 'standard'; Replace uses of QualifiedName with PrimaryName; Remove QualifiedName;

10-Apr-2008: Removed reserved E4X syntax; Rename and update object and array initialisers to match latest proposals; Rename SplatExpression to SpreadExpression; Add signatures for getters and setters; Add void and boolean result types; Move 'internal', 'private', 'protected', 'public' from ReservedIdentifier to ContextuallyReservedIdentifier; Rename various "Literal" non-terminal to "Initialiser" with corresponding changes to their constituents; Change argument to CommaExpression in BracketOrSlice from allowColon to noColon; Allow FieldType with ': TypeExpression' elided; Remove getters and setters from local blocks; Change signature of FunctionDeclaration to FunctionSignatureType; Include nested let, if and for-in expressions in ElementComprehension; Allow 'const' attribute on parameters; Require optional parameters to follow obigatory ones; Replace SimplePattern in TypedIdentifier with Identifier; Refactor CaseElements; Add 'const' and 'var' to the lookahead set of ExpressionStatement

09-Apr-2008: Remove description of triple quoted strings; Rename LikedPattern to LikenedPattern; Allow trailing comma in RecordType and ObjectPattern; Add [no line break] to ThisExpression; Add reference to "line continuations" spec in lexical section; Limit syntax of annotations on object and array literals; Replace PrimaryName... in TypeExpression with TypeReference; Refactor class Block to only allow a static block statements; Added description of source text handling; Allow VariableDefinition in Substatement

03-Apr-2008: Remove reserved identifiers 'wrap' and 'has'; Replace use of PropertyName with PrimaryName in PropertyOperator; Remove definition of PropertyName; Remove 'enum' from ReservedIdentifiers; Move 'extends' from ReservedIdentifiers to ContextuallyReservedIdentifiers; Add FieldKind to getters and setter in LiteralField; Remove omega from VariableDefinition in AnnotatableDirective (Global...); Add Semicolon the other occurances of VariableDefinition in AnnotatableDirective; Add Semicolon to occurances of TypeDefinition and NamespaceDefinition in AnnotatableDirectives; Remove TypeDefinition from InterfaceDefinition; Fix various arguments in RelationalExpression; Fix argument in AnnotatableDirective (class); Add Semicolon to FunctionDeclaration production in AnnotatableDirective (interface); Add interface argument to NamespaceAttribute in Attribute (interface); Add NamespaceAttribute (interface); Replace 'intrinsic' with 'external' in NamespaceAttribute rules; Remove Attribute (local); Remove definition and use of OverloadedOperator; Rename InitialiserList to SettingList and Initialiser to Setting; Make TypeReferenceList left recursive; Rename PackageAttributes to PackageAttribute

30-Mar-2008: Rename ListExpression to CommaExpression; Make CommaExpression a binary expression in the AST; Change ParenExpression to ParenListExpression in SuperExpression; Rename ParenListExpression to ParenExpression; Remove Path qualified PropertyNames; Mark reserved/deferred features with 'x'; Remove 'wrap'; Remove 'like' as a type; Add 'like' as a binary type operator; Remove LetStatement; Remove UnitDefinition; Fold NullableTypeExpression into TypeExpression; Remove OverloadedOperator from QualifiedNameIdentifier; Add distinguishing syntax for tuples and array types in ArrayType; Add SplatExpression to arguments and array literals; Add RestPattern to array patterns; Add to ReservedIdentifiers 'type'; Add to ContextuallyReservedIdentifiers 'external'; Removed from ContextuallyReservedIdentifiers 'decimal', 'double', 'generic', 'int', 'Number', 'precision', 'rounding', 'standard', 'to', 'uint', 'unit'; Add LikedPattern to Parameter; Add LikePredicate to ResultType; Remove ParameterKind and use in Parameter

20-Mar-2008: Use noColon parameter before : in ConditionalExpression and NonAssignmentExpression; Swapped [PropertyName, QualifiedName] => [QualifiedName, PropertyName]; Removed . AttributeName from PropertyOperator; Add AttributeName to PrimaryName; Rename Brackets to BracketsOrSlice; Add Brackets, without slice; Change Brackets in PropertyOperator to BracketsOrSlice; Add TypeUnionList etc to allow for | list separators and empty unions; Move LetExpression from ConditionalExpression to PrimaryExpression; Move the UnaryTypeExpression from PostfixExpression to ConditionalExpression and NonAssignmentExpression; Replace TypedExpression with ParenListExpression; Remove TypedExpression; Remove import aliasing; Add ReservedNamespace to PrimaryExpression; Add ".*" syntax to PropertyOperator for E4X compatibility; Remove "intrinsic" from ReservedNamesapce and ContextuallyReservedIdentifiers; Add TypeApplication syntax to BasicTypeExpression (got dropped by ealier refactoring); Refactored CaseElementsPrefix; Change PrimaryNameList to TypeReferenceList in InterfaceInheritance (typo)

04-Dec-2007: Add productins for AnnotattableDirective(class,...)

31-Oct-2007: Add 'wrap' to ReservedIdenifiers; Move 'is' and 'cast' from ContextuallyReservedIdentifiers to ReservedIdentifiers; Add version number for which each production applies

23-Oct-2007: Add 'wrap' operation to RelationalExpression; Add 'like' type expression; Rename root type expression from NullableType to TypeExpression

18-Apr-2008: Remove TypeParameter from ConstructorSignature (452, 453); Remove Brackets in QualifiedNameIdentifier (13); Change argument to Block in BlockStatement to 'local' (304); Removed lingering uses of 'external' from NamespaceAttributes (388, 394); Remove lingering E4X punctuators **</** and **/>** from (6, lexical); Change let and function expression forms to use CommaExpression instead of AssignmentExpression (22, 76, 423); Add productions for handling **>>** and **>>>** in TypeApplication (101); Add productions for handling **::** in SliceExpression (98); Disallow 'let' in class bodies (398)

17-Apr-2008: Rename ElementComprehension to ArrayComprehension; Allow empty body of 'let' clause in ArrayComprehension; Add 'standard' as a pragma; Fix obligatory ',' bug in ArrayType; Allow only SimplePattern in RestParameter; Remove PackageDefinition; Remove ImportPragma; Remove 'external' from ReservedIdentifier and ReservedNamespace; Add 'Identifier : TypeExpression' to ParameterType; Replace TypeExpression with Identifier in RestParameterType; Removed 'meta::' productions from ObjectInitialiser; Remove ContextuallyReservedIdentifiers 'package', and 'xml'; (Re)-add ContextuallyReservedIdentifier 'standard'; Replace uses of QualifiedName with PrimaryName; Remove QualifiedName;

10-Apr-2008: Removed reserved E4X syntax; Rename and update object and array initialisers to match latest proposals; Rename SplatExpression to SpreadExpression; Add signatures for getters and setters; Add void and boolean result types; Move 'internal', 'private', 'protected', 'public' from ReservedIdentifier to ContextuallyReservedIdentifier; Rename various "Literal" non-terminal to "Initialiser" with corresponding changes to their constituents; Change argument to CommaExpression in BracketOrSlice from allowColon to noColon; Allow FieldType with ': TypeExpression' elided; Remove getters and setters from local blocks; Change signature of FunctionDeclaration to FunctionSignatureType; Include nested let, if and for-in expressions in ElementComprehension; Allow 'const' attribute on parameters; Require optional parameters to follow obigatory ones; Replace SimplePattern in TypedIdentifier with Identifier; Refactor CaseElements; Add 'const' and 'var' to the lookahead set of ExpressionStatement

09-Apr-2008: Remove description of triple quoted strings; Rename LikedPattern to LikenedPattern; Allow trailing comma in RecordType and ObjectPattern; Add [no line break] to ThisExpression; Add reference to "line continuations" spec in lexical section; Limit syntax of annotations on object and array literals; Replace PrimaryName... in TypeExpression with TypeReference; Refactor class Block to only allow a static block statements; Added description of source text handling; Allow VariableDefinition in Substatement

03-Apr-2008: Remove reserved identifiers 'wrap' and 'has'; Replace use of PropertyName with PrimaryName in PropertyOperator; Remove definition of PropertyName; Remove 'enum' from ReservedIdentifiers; Move 'extends' from ReservedIdentifiers to ContextuallyReservedIdentifiers; Add FieldKind to getters and setter in LiteralField; Remove omega from VariableDefinition in AnnotatableDirective (Global...); Add Semicolon the other occurances of VariableDefinition in AnnotatableDirective; Add Semicolon to occurances of TypeDefinition and NamespaceDefinition in AnnotatableDirectives; Remove TypeDefinition from InterfaceDefinition; Fix various arguments in RelationalExpression; Fix argument in AnnotatableDirective (class); Add Semicolon to FunctionDeclaration production in AnnotatableDirective (interface); Add interface argument to NamespaceAttribute in Attribute (interface); Add NamespaceAttribute (interface); Replace 'intrinsic' with 'external' in NamespaceAttribute rules; Remove Attribute (local); Remove definition and use of OverloadedOperator; Rename InitialiserList to SettingList and Initialiser to Setting; Make TypeReferenceList left recursive; Rename PackageAttributes to PackageAttribute

30-Mar-2008: Rename ListExpression to CommaExpression; Make CommaExpression a binary expression in the AST; Change ParenExpression to ParenListExpression in SuperExpression; Rename ParenListExpression to ParenExpression; Remove Path qualified PropertyNames; Mark reserved/deferred features with 'x'; Remove 'wrap'; Remove 'like' as a type; Add 'like' as a binary type operator; Remove LetStatement; Remove UnitDefinition; Fold NullableTypeExpression into TypeExpression; Remove OverloadedOperator from QualifiedNameIdentifier; Add distinguishing syntax for tuples and array types in ArrayType; Add SplatExpression to arguments and array literals; Add RestPattern to array patterns; Add to ReservedIdentifiers 'type'; Add to ContextuallyReservedIdentifiers 'external'; Removed from ContextuallyReservedIdentifiers 'decimal', 'double', 'generic', 'int', 'Number', 'precision', 'rounding', 'standard', 'to', 'uint', 'unit'; Add LikedPattern to Parameter; Add LikePredicate to ResultType; Remove ParameterKind and use in Parameter

20-Mar-2008: Use noColon parameter before : in ConditionalExpression and NonAssignmentExpression; Swapped [PropertyName, QualifiedName] => [QualifiedName, PropertyName]; Removed . AttributeName from PropertyOperator; Add AttributeName to PrimaryName; Rename Brackets to BracketsOrSlice; Add Brackets, without slice; Change Brackets in PropertyOperator to BracketsOrSlice; Add TypeUnionList etc to allow for | list separators and empty unions; Move LetExpression from ConditionalExpression to PrimaryExpression; Move the UnaryTypeExpression from PostfixExpression to ConditionalExpression and NonAssignmentExpression; Replace TypedExpression with ParenListExpression; Remove TypedExpression; Remove import aliasing; Add ReservedNamespace to PrimaryExpression; Add ".*" syntax to PropertyOperator for E4X compatibility; Remove "intrinsic" from ReservedNamesapce and ContextuallyReservedIdentifiers; Add TypeApplication syntax to BasicTypeExpression (got dropped by ealier refactoring); Refactored CaseElementsPrefix; Change PrimaryNameList to TypeReferenceList in InterfaceInheritance (typo)

04-Dec-2007: Add productins for AnnotattableDirective(class,...)

31-Oct-2007: Add 'wrap' to ReservedIdenifiers; Move 'is' and 'cast' from ContextuallyReservedIdentifiers to ReservedIdentifiers; Add version number for which each production applies

17-Oct-2007: Change 'this callee' to 'this function'; Remove 'callee' from ContextuallyReservedIdentifiers; Add TypeReference and TypeReferenceList; Replace use of PrimaryName and PrimaryNameList in ClassInheritance and InterfaceInheritance with TypeReference and TypeReferenceList; Remove [No newline] contraint in ReturnStatement; Add Semicolon after DoStatement; Minor reordering of productions in PrimaryExpression; Rename ObjectType to RecordType; Initial definition of mapping from concrete to abstract syntax

14-Oct-2007: Remove 'type' TypeExpression from UnaryExpr; Add UnaryTypeExpression; Change uses of TypeExpression to NullableTypeExpression for symmetry with TypeDefinitions; Restore use of 'undefined' in TypeExpression (although ambiguous, provides clarity); update 'use decimal' pragma; Rename DestructuringField* to Field*Pattern and DestructuringElement* to Element*Pattern; Change "Path . Identifier" in NamespaceAttribute to PrimaryName; Remove Identifier from NamespaceAttribute

04-Oct-2007: Replace Identifier with NonAttributeQualifiedIdentifier in FieldName; Add ReservedNamespace to Qualifier; Change arguments to Pattern in Initialiser to allowIn, allowExpr; Remove Semicolon after DoStatement; Add TypeApplication to PropertyIdentifier; Remove PropertyName; Rename NonAttributeIdentifier to PropertyName; Remove default from TypeCaseElement; Remove duplicate production for XMLElementContent

22-Aug-2007: Fix several cases of missing rule arguments; Move use of Semicolon out of VariableDefinition

21-Aug-2007: Remove '*' from QualifiedNameIdentifier; Rename use of AttributeIdentifier to AttributeName in PrimaryExpression; Add SwitchTypeStatement to Statement; Replace ClassName with Identifier TypeSignature in InterfaceDefinition and FunctionDefinition; Replace ParameterisedTypeName with Identifier TypeSignature in TypeDefinition; Fix various other typos found by E. Suen

20-Aug-2007: Remove LiteralField without value; Add FieldName without pattern to DestructuringField; Move null and undefined from NullableTypeExpression to TypeExpression; Erase ToSignature; Distinguish FunctionExpressionBody from FunctionBody; Move Semicolon into specific definition rules that use them; Add UnitDefinition; Fix use unit pragma; Factor out ClassSignature from ClassName (now just Identifier); Replace use of SimpleQualifiedName with PrimaryName in NamespaceInitialiser; Rename RecordType to ObjectType; Change String to StringLiteral; Number to NumberLiteral in QualifiedNameIdentifier; Remove ambiguous ReservedNamespace in Qualifier; Remove 'undefined' from TypeExpression; Add 'callee' and 'generator' to ContextuallyReservedIdentifiers

23-Jul-2007: Require Block body in LetStatement; Fixed missed renames of *Identifier to *Name; Allow trailing common in ObjectLiteral; Make 'debugger' a reserved identifier; Add 'this callee' and 'this generator' as a primary expressions; Simplified TypedPattern; Change prefix of type application from TypeExpression to ParenListExpression; Remove 'null' and 'undefined' from TypeExpression; Require semicolon after braceless function body; Various fixes to the beta argument; Add alpha parameter to indicate contexts which allow annotations on object and array literals; Fix missed replacement of PrimaryIdentifier with PrimaryName; Add Unit pragmas; Relax rules that packages must come before any other directive (make PackageDefinition a Directive)

29-May-2007: Add types 'null' and 'undefined' to TypeExpression; Rename Identifier to Name; add non-terminal QualifiedNameIdentifier to hold various kinds of identifiers; Add TypedExpression and use in head of WithStatement and SwitchTypeStatement; Change name of get and set fields to FieldName; Eliminate distinction between NullableTypeExpression and TypeExpression;

23-May-2007: Fix list comprehensions; Remove 'debugger' and 'include' from ContextuallyReservedIdentifier; Change body of yield, let and function expressions from ListExpression to AssignmentExpression; Remove use of the alpha parameter to distinguish allowList from noList uses of yield, let and function expressions; Add optional Qualifier to FieldName

10-Apr-2007: Fix several typos; Add to SimpleQualifiedIdentifier syntax for calling global intrinsic overloadable operators

06-Apr-2007: Replace errant references to TypeIdentifier with PropertyIdentifier; Move from ReservedIdentifiers to ContextuallyReservedIdentifiers: cast const implements import interface internal intrinsic is let package private protected public to use; Remove ReservedIdentifier: as; Add missing allowIn argument to uses of FunctionBody; Remove lexical non-terminal PackageIdentifiers

30-Mar-2007: Replace TypeIdentifier in PrimaryExpression with PrimaryIdentifier; Inline PropertyIdentifier production; Rename TypeIdentifier to PropertyIdentifier; Remove function names with embedded *

29-Mar-2007: Revert previous restriction that 'use default namespace' argument must be a particular reserved namespace; Add tau parameter to BlockStatement and Block to allow top-level blocks with hoisted definitions; Rename ParameterisedClassName to ParameterisedTypeName; Change Identifier in TypeDefinition to ParameterisedTypeName; Replace the lexeme PackageIdentifier with the nonterminal Path, which gets resolved to a PackageName or an object referece by the definer; Move the ListExpression form of function body into FunctionBody; Add PrimaryIdentifier production and move Path qualified references out of TypeIdentifier to PrimaryIdentifier; Change right side of PropertyOperator from QualifiedIdentifier to TypeIdentifier; Add 'has' to the ContextuallyReservedIdentifiers; Update FunctionName to include 'call' and 'has' functions; Remove 'invoke' from ContextuallyReservedIdentifiers

23-Oct-2007: Add 'wrap' operation to RelationalExpression; Add 'like' type expression; Rename root type expression from NullableType to TypeExpression

17-Oct-2007: Change 'this callee' to 'this function'; Remove 'callee' from ContextuallyReservedIdentifiers; Add TypeReference and TypeReferenceList; Replace use of PrimaryName and PrimaryNameList in ClassInheritance and InterfaceInheritance with TypeReference and TypeReferenceList; Remove [No newline] contraint in ReturnStatement; Add Semicolon after DoStatement; Minor reordering of productions in PrimaryExpression; Rename ObjectType to RecordType; Initial definition of mapping from concrete to abstract syntax

14-Oct-2007: Remove 'type' TypeExpression from UnaryExpr; Add UnaryTypeExpression; Change uses of TypeExpression to NullableTypeExpression for symmetry with TypeDefinitions; Restore use of 'undefined' in TypeExpression (although ambiguous, provides clarity); update 'use decimal' pragma; Rename DestructuringField* to Field*Pattern and DestructuringElement* to Element*Pattern; Change "Path . Identifier" in NamespaceAttribute to PrimaryName; Remove Identifier from NamespaceAttribute

04-Oct-2007: Replace Identifier with NonAttributeQualifiedIdentifier in FieldName; Add ReservedNamespace to Qualifier; Change arguments to Pattern in Initialiser to allowIn, allowExpr; Remove Semicolon after DoStatement; Add TypeApplication to PropertyIdentifier; Remove PropertyName; Rename NonAttributeIdentifier to PropertyName; Remove default from TypeCaseElement; Remove duplicate production for XMLElementContent

22-Aug-2007: Fix several cases of missing rule arguments; Move use of Semicolon out of VariableDefinition

21-Aug-2007: Remove '*' from QualifiedNameIdentifier; Rename use of AttributeIdentifier to AttributeName in PrimaryExpression; Add SwitchTypeStatement to Statement; Replace ClassName with Identifier TypeSignature in InterfaceDefinition and FunctionDefinition; Replace ParameterisedTypeName with Identifier TypeSignature in TypeDefinition; Fix various other typos found by E. Suen

20-Aug-2007: Remove LiteralField without value; Add FieldName without pattern to DestructuringField; Move null and undefined from NullableTypeExpression to TypeExpression; Erase ToSignature; Distinguish FunctionExpressionBody from FunctionBody; Move Semicolon into specific definition rules that use them; Add UnitDefinition; Fix use unit pragma; Factor out ClassSignature from ClassName (now just Identifier); Replace use of SimpleQualifiedName with PrimaryName in NamespaceInitialiser; Rename RecordType to ObjectType; Change String to StringLiteral; Number to NumberLiteral in QualifiedNameIdentifier; Remove ambiguous ReservedNamespace in Qualifier; Remove 'undefined' from TypeExpression; Add 'callee' and 'generator' to ContextuallyReservedIdentifiers

23-Jul-2007: Require Block body in LetStatement; Fixed missed renames of *Identifier to *Name; Allow trailing common in ObjectLiteral; Make 'debugger' a reserved identifier; Add 'this callee' and 'this generator' as a primary expressions; Simplified TypedPattern; Change prefix of type application from TypeExpression to ParenListExpression; Remove 'null' and 'undefined' from TypeExpression; Require semicolon after braceless function body; Various fixes to the beta argument; Add alpha parameter to indicate contexts which allow annotations on object and array literals; Fix missed replacement of PrimaryIdentifier with PrimaryName; Add Unit pragmas; Relax rules that packages must come before any other directive (make PackageDefinition a Directive)

29-May-2007: Add types 'null' and 'undefined' to TypeExpression; Rename Identifier to Name; add non-terminal QualifiedNameIdentifier to hold various kinds of identifiers; Add TypedExpression and use in head of WithStatement and SwitchTypeStatement; Change name of get and set fields to FieldName; Eliminate distinction between NullableTypeExpression and TypeExpression;

23-May-2007: Fix list comprehensions; Remove 'debugger' and 'include' from ContextuallyReservedIdentifier; Change body of yield, let and function expressions from ListExpression to AssignmentExpression; Remove use of the alpha parameter to distinguish allowList from noList uses of yield, let and function expressions; Add optional Qualifier to FieldName

10-Apr-2007: Fix several typos; Add to SimpleQualifiedIdentifier syntax for calling global intrinsic overloadable operators

06-Apr-2007: Replace errant references to TypeIdentifier with PropertyIdentifier; Move from ReservedIdentifiers to ContextuallyReservedIdentifiers: cast const implements import interface internal intrinsic is let package private protected public to use; Remove ReservedIdentifier: as; Add missing allowIn argument to uses of FunctionBody; Remove lexical non-terminal PackageIdentifiers

30-Mar-2007: Replace TypeIdentifier in PrimaryExpression with PrimaryIdentifier; Inline PropertyIdentifier production; Rename TypeIdentifier to PropertyIdentifier; Remove function names with embedded *

13-Mar-2007: Add SuperInitialiser to as optional final constituent of ConstructorInitialiser; Erase SuperStatement; Erase "const function" from the class context (all methods are const); Restrict use default namespace argument to public, internal and intrinsic; Remove 'in' from ContextuallyReservedIdentifiers; Define 'function to' so that no return type is allowed; Remove 'construct' from ContextuallyReservedIdentifiers; Add 'invoke' to ContextuallyReservedIdentifiers

02-Mar-2007: Erase gamma parameter from TypedPattern (always noExpr), Add syntax for array comprehension; Rename ElementList to Elements; Rename FieldList to Fields; Rename NonemptyFieldList to FieldList; Add "const function" definition syntax; Change PropertyIdentifier to * in function call definitions; Rename call to invoke in non-catchall definitions; Remove 'construct' function; Update PackageIdentifier; Remove '^^' and '^^=' punctuators; Fork FunctionSignatureType from FunctionSignature; Fix bug which allowed "this : T ," in FunctionSignature; Make 'null' and 'undefined' NullableTypeExpressions; Add 'undefined' to ContextuallyReservedIdentifiers

18-Jan-2007: Add syntactic parameter τ to distinguish between contexts that allow / exclude certain kinds of definitions; Add syntax for constructor definitions, including ConstructorInitialiser; Add syntax to FunctionSignature to constrain type of 'this'; Dinstinguish between nullable/nonnullable and orther type expression; Allow any TypeExpression in TypedPattern

08-Dec-2006: Add FieldKind to LiteralField; Change NonAttributeQualifiedIdentifier to PropertyIdentifier in FieldName; Remove [no line break] constraint from FunctionName; Add to FunctionName productions for 'construct' and for 'call' and 'to' without a name; Add 'construct' to ContextuallyReservedIdentifiers

06-Dec-2006: Add BlockStatement non-terminal, minor refactoring of the Program productions; Rename PackageDefinition as Package; Change NonAttributeQualifiedIdentifier to FieldName in DestructuringField; Change SwitchTypeStatement to take a ListExpression and TypeExpression in its head rather than a binding form; Merge LogicalAssignmnetOperator into CompoundAssignmentOperator; Rename Inheritance to ClassInheritance; Rename ExtendsList to InterfaceInheritance; Refactor InterfaceDefinition to have a more specific syntax;

29-Nov-2006: Update AST nodes for VariableDefinition; Update AST nodes for Pragmas; Change rhs of SimplePattern from PostfixExpression to LeftHandSideExpression; Tighten the syntax of definition attributes that are reference to namespaces; Add AST nodes for SwitchStatement and SwitchTypeStatement

21-Nov-2006: Make the 'cast' operator a peer of the infx 'to' operator; Propagate the α parameter to FunctionExpression; Unify TypedIdentifier and TypedPattern, and lhs postfix expressions and Pattern; Remove logical xor operator; Add 'precision' to PragmaIdentifier and ContextuallyReservedIdentifier; Add AST node types for expressions; Refactor slice syntax; Remove empty bracket syntax

14-Nov-2006: Move 'yield' from Reserved to contextually reserved; Add ReservedIdentifier after '::' in ExpressionQualifiedIdentifier; Refactor RestParameter; Remove abstract function declaration from FunctionCommon; Add accessors to ObjectLiteral; Move TypedIdentifier and TypedPattern to the Expressions section; Remove FieldName : ParenExpression; Remove ExpressionClosure; Add expression closure syntax to FunctionExpression; Propagate the β parameter down to FunctionExpression; Distinguish between RecordType and ArrayType in TypedPattern; Rename noLet and allowLet to noList and allowList, respectively; Add «empty» to DestructuringFieldList; Added links to 'triple quotes' and 'extend regexp' proposals

26-Sep-2006: Add ReservedIdentifier after '::'; Parameterise productions to restrict the context where LetExpression and YieldExpression can be used; Change the body of LetExpression and YieldExpression from AssignmentExpression to ListExpression

21-Sep-2006: Rename lexical non-terminals 'String' to 'StringLiteral' and 'Number' to 'NumberLiteral'; Remove infix 'cast' expressions; Remove prefix 'to' expressions; Change the rhs of 'to' to be a TypeExpression; Move 'yield' to 'AssignmentExpression' (again); Replace Arguments with ParenExpression in SuperExpression

15-Sep-2006: Add rules for tagging an object or array literal with a structural type; Add “decimal”, “double”, “int”, “uint”, “Number”, “rounding”, “strict”, and “standard” to the list of ContextuallyReservedIdentifiers; Fix capitalisation of PackageIdentifier (409); Add definition of lexical Identifier; Remove redundant productions referring to ContextuallyReservedIdentifier; Add "Number" as a PragmaArgument; Refactor YieldExpression to be used by MultiplicativeExpression and use UnaryExpression

30-Aug-2006: Remove 'native' from ReservedIdentifier; Add lexical non-terminals for missing literal forms and VirtualSemicolon; Replace productions for Identifier with one that uses lexical symbol ContextuallyReservedIdentifiers; Replace RestParameters with RestParameter (57); Replace Expression with ListExpression (94,99,101,106); Replace NonAssignmentExpression with LogicalOrExpression (219); Remove unused production for DestructuringAssignmentExpression (250); Remove Statement production for SwitchTypeStatement (291); Sort Statement productions; Remove unused productions for Substatements and SubstatementsPrefix; Replace use of VariableInitialiser with AssignmetExpression (441); Replace uses of TypeName with TypeIdentifier (462,463); Rename TypeNameList as TypeIdentifierList

29-Mar-2007: Revert previous restriction that 'use default namespace' argument must be a particular reserved namespace; Add tau parameter to BlockStatement and Block to allow top-level blocks with hoisted definitions; Rename ParameterisedClassName to ParameterisedTypeName; Change Identifier in TypeDefinition to ParameterisedTypeName; Replace the lexeme PackageIdentifier with the nonterminal Path, which gets resolved to a PackageName or an object referece by the definer; Move the ListExpression form of function body into FunctionBody; Add PrimaryIdentifier production and move Path qualified references out of TypeIdentifier to PrimaryIdentifier; Change right side of PropertyOperator from QualifiedIdentifier to TypeIdentifier; Add 'has' to the ContextuallyReservedIdentifiers; Update FunctionName to include 'call' and 'has' functions; Remove 'invoke' from ContextuallyReservedIdentifiers

13-Mar-2007: Add SuperInitialiser to as optional final constituent of ConstructorInitialiser; Erase SuperStatement; Erase "const function" from the class context (all methods are const); Restrict use default namespace argument to public, internal and intrinsic; Remove 'in' from ContextuallyReservedIdentifiers; Define 'function to' so that no return type is allowed; Remove 'construct' from ContextuallyReservedIdentifiers; Add 'invoke' to ContextuallyReservedIdentifiers

02-Mar-2007: Erase gamma parameter from TypedPattern (always noExpr), Add syntax for array comprehension; Rename ElementList to Elements; Rename FieldList to Fields; Rename NonemptyFieldList to FieldList; Add "const function" definition syntax; Change PropertyIdentifier to * in function call definitions; Rename call to invoke in non-catchall definitions; Remove 'construct' function; Update PackageIdentifier; Remove '^^' and '^^=' punctuators; Fork FunctionSignatureType from FunctionSignature; Fix bug which allowed "this : T ," in FunctionSignature; Make 'null' and 'undefined' NullableTypeExpressions; Add 'undefined' to ContextuallyReservedIdentifiers

18-Jan-2007: Add syntactic parameter τ to distinguish between contexts that allow / exclude certain kinds of definitions; Add syntax for constructor definitions, including ConstructorInitialiser; Add syntax to FunctionSignature to constrain type of 'this'; Dinstinguish between nullable/nonnullable and orther type expression; Allow any TypeExpression in TypedPattern

08-Dec-2006: Add FieldKind to LiteralField; Change NonAttributeQualifiedIdentifier to PropertyIdentifier in FieldName; Remove [no line break] constraint from FunctionName; Add to FunctionName productions for 'construct' and for 'call' and 'to' without a name; Add 'construct' to ContextuallyReservedIdentifiers

06-Dec-2006: Add BlockStatement non-terminal, minor refactoring of the Program productions; Rename PackageDefinition as Package; Change NonAttributeQualifiedIdentifier to FieldName in DestructuringField; Change SwitchTypeStatement to take a ListExpression and TypeExpression in its head rather than a binding form; Merge LogicalAssignmnetOperator into CompoundAssignmentOperator; Rename Inheritance to ClassInheritance; Rename ExtendsList to InterfaceInheritance; Refactor InterfaceDefinition to have a more specific syntax;

29-Nov-2006: Update AST nodes for VariableDefinition; Update AST nodes for Pragmas; Change rhs of SimplePattern from PostfixExpression to LeftHandSideExpression; Tighten the syntax of definition attributes that are reference to namespaces; Add AST nodes for SwitchStatement and SwitchTypeStatement

21-Nov-2006: Make the 'cast' operator a peer of the infx 'to' operator; Propagate the α parameter to FunctionExpression; Unify TypedIdentifier and TypedPattern, and lhs postfix expressions and Pattern; Remove logical xor operator; Add 'precision' to PragmaIdentifier and ContextuallyReservedIdentifier; Add AST node types for expressions; Refactor slice syntax; Remove empty bracket syntax

14-Nov-2006: Move 'yield' from Reserved to contextually reserved; Add ReservedIdentifier after '::' in ExpressionQualifiedIdentifier; Refactor RestParameter; Remove abstract function declaration from FunctionCommon; Add accessors to ObjectLiteral; Move TypedIdentifier and TypedPattern to the Expressions section; Remove FieldName : ParenExpression; Remove ExpressionClosure; Add expression closure syntax to FunctionExpression; Propagate the β parameter down to FunctionExpression; Distinguish between RecordType and ArrayType in TypedPattern; Rename noLet and allowLet to noList and allowList, respectively; Add «empty» to DestructuringFieldList; Added links to 'triple quotes' and 'extend regexp' proposals

26-Sep-2006: Add ReservedIdentifier after '::'; Parameterise productions to restrict the context where LetExpression and YieldExpression can be used; Change the body of LetExpression and YieldExpression from AssignmentExpression to ListExpression

21-Sep-2006: Rename lexical non-terminals 'String' to 'StringLiteral' and 'Number' to 'NumberLiteral'; Remove infix 'cast' expressions; Remove prefix 'to' expressions; Change the rhs of 'to' to be a TypeExpression; Move 'yield' to 'AssignmentExpression' (again); Replace Arguments with ParenExpression in SuperExpression

15-Sep-2006: Add rules for tagging an object or array literal with a structural type; Add “decimal”, “double”, “int”, “uint”, “Number”, “rounding”, “strict”, and “standard” to the list of ContextuallyReservedIdentifiers; Fix capitalisation of PackageIdentifier (409); Add definition of lexical Identifier; Remove redundant productions referring to ContextuallyReservedIdentifier; Add "Number" as a PragmaArgument; Refactor YieldExpression to be used by MultiplicativeExpression and use UnaryExpression

15-Jun-2006: Add 'yield' expression without subexpression; Remove Semicolon after PragmaItems in UsePragma; Remove parens around PragmaARgument in PragmaItem; Change SimpleQualifiedIdentifier to SimpleTypeIdentifier in PragmaArgument; Add SimpleTypeIdentifier to NamespaceInitialisation

07-Jun-2006: Remove AttributeCombination from Attributes; Remove true and false from Attributes (they are a carryover from the NS proposal and have never been proposed here); Added comment on the creation of a lexical PackageIdentifier from a syntactic PackageName; Allow 'let' on VariableDefinition and FunctionDefinition; Merge SwitchType into SwitchStatement; Add 'call' to context keywords and syntactic identifier; Replace ListExpression in Arguments with ArgumentList; Reuse VariableBinding for LetBinding; Add ParameterAttributes to Pattern in Parameter; Add TypedParameter to RestParameter; Change Identifier to TypedIdentifier in RestParameter; Add TypedPattern to TypeCaseElement; Rename 'private' to 'internal' in PackageAttributes

01-Jun-2006: Add '!' to ClassName; Remove 'as'; Replace TypeExpression on the rhs of 'is' and 'to' with ShiftExpression; Rename AttributeQualifiedIdentifier to AttributeIdentifier; Add 'type' operator to UnaryExpression; Change yield construct from YieldStatement to YieldExpression; Add 'yield' to the list of reserved identifiers; Add TypedPattern everywhere that TypedIdentifier is used to defined a variable, except in switch-type; Define the meaning of the lexical symbol PackageIdentifier; Add primary expression for "to" and binary expression for "cast"

23-May-2006: Add 'super' to reserved words; Refactor TypeIdentifier; Use simpler E3 syntax for PostfixExpression; Rename LPattern and children to Pattern etc.; Move DestructuringAssignmentExpression out of AssignmentExpresion; Move LetExpression to AssignmentExpression; Remove attribute blocks; Remove variable initialiser with multiple attributes on the rhs; Add parens around pragma arguments; Add prama identifiers 'default namespace' and 'default package'; Add PackageAttribute to PackageDefinition; Sort rules for readability

16-May-2006: Added '.' before '<…>' in type definitions; removed ReservedNamespace from PrimaryExpression since it is already include via QualifiedIdentifier; simplified PostfixExpression; changed qualifier on ExpressionQualifiedIdentifier from ParenExpression to ParentListExpression; Refactored TypeIdentifier; replaced QualifiedIdentifier with TypeIdentifier and added AttributeQualifiedIdentifier in PrimaryExpression; made .< a token rather than two; Redefined TypeParameters to include the .< and > delimiters

15-May-2006: Moved 'PackageIdentifier . Identifier' from PrimaryExpression to QualifiedIdenfier; Added dot to left angle brace for parameterized type expressions in TypeExpression

12-May-2006: Initial draft. First attempt to capture the whole grammar of ES4. Current with the latest proposals

30-Aug-2006: Remove 'native' from ReservedIdentifier; Add lexical non-terminals for missing literal forms and VirtualSemicolon; Replace productions for Identifier with one that uses lexical symbol ContextuallyReservedIdentifiers; Replace RestParameters with RestParameter (57); Replace Expression with ListExpression (94,99,101,106); Replace NonAssignmentExpression with LogicalOrExpression (219); Remove unused production for DestructuringAssignmentExpression (250); Remove Statement production for SwitchTypeStatement (291); Sort Statement productions; Remove unused productions for Substatements and SubstatementsPrefix; Replace use of VariableInitialiser with AssignmetExpression (441); Replace uses of TypeName with TypeIdentifier (462,463); Rename TypeNameList as TypeIdentifierList

15-Jun-2006: Add 'yield' expression without subexpression; Remove Semicolon after PragmaItems in UsePragma; Remove parens around PragmaARgument in PragmaItem; Change SimpleQualifiedIdentifier to SimpleTypeIdentifier in PragmaArgument; Add SimpleTypeIdentifier to NamespaceInitialisation

07-Jun-2006: Remove AttributeCombination from Attributes; Remove true and false from Attributes (they are a carryover from the NS proposal and have never been proposed here); Added comment on the creation of a lexical PackageIdentifier from a syntactic PackageName; Allow 'let' on VariableDefinition and FunctionDefinition; Merge SwitchType into SwitchStatement; Add 'call' to context keywords and syntactic identifier; Replace ListExpression in Arguments with ArgumentList; Reuse VariableBinding for LetBinding; Add ParameterAttributes to Pattern in Parameter; Add TypedParameter to RestParameter; Change Identifier to TypedIdentifier in RestParameter; Add TypedPattern to TypeCaseElement; Rename 'private' to 'internal' in PackageAttributes

01-Jun-2006: Add '!' to ClassName; Remove 'as'; Replace TypeExpression on the rhs of 'is' and 'to' with ShiftExpression; Rename AttributeQualifiedIdentifier to AttributeIdentifier; Add 'type' operator to UnaryExpression; Change yield construct from YieldStatement to YieldExpression; Add 'yield' to the list of reserved identifiers; Add TypedPattern everywhere that TypedIdentifier is used to defined a variable, except in switch-type; Define the meaning of the lexical symbol PackageIdentifier; Add primary expression for "to" and binary expression for "cast"

23-May-2006: Add 'super' to reserved words; Refactor TypeIdentifier; Use simpler E3 syntax for PostfixExpression; Rename LPattern and children to Pattern etc.; Move DestructuringAssignmentExpression out of AssignmentExpresion; Move LetExpression to AssignmentExpression; Remove attribute blocks; Remove variable initialiser with multiple attributes on the rhs; Add parens around pragma arguments; Add prama identifiers 'default namespace' and 'default package'; Add PackageAttribute to PackageDefinition; Sort rules for readability

16-May-2006: Added '.' before '<…>' in type definitions; removed ReservedNamespace from PrimaryExpression since it is already include via QualifiedIdentifier; simplified PostfixExpression; changed qualifier on ExpressionQualifiedIdentifier from ParenExpression to ParentListExpression; Refactored TypeIdentifier; replaced QualifiedIdentifier with TypeIdentifier and added AttributeQualifiedIdentifier in PrimaryExpression; made .< a token rather than two; Redefined TypeParameters to include the .< and > delimiters

15-May-2006: Moved 'PackageIdentifier . Identifier' from PrimaryExpression to QualifiedIdenfier; Added dot to left angle brace for parameterized type expressions in TypeExpression

12-May-2006: Initial draft. First attempt to capture the whole grammar of ES4. Current with the latest proposals