ECMAScript 4th Edition Grammar

ID **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 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 continue debugger default delete do else false finally for function if in instanceof is new null return super switch this throw true try type typeof var void while with**

ContextuallyReservedIdentifier [one of]

2 **const dynamic each eval extends final generator get implements import interface internal intrinsic iterator let meta namespace native override private protected prototype public reflect set standard static strict undefined use yield __proto__** [__proto__, internal, intrinsic, iterator, meta private, protected, public, and reflect do not appear in the grammar but are contextually reserved because they have special meaning. __proto__ has special meaning when used as a FieldName in ObjectInitialiser; internal, intrinsic, iterator, meta private, protected, public, and reflect cannot be redefined by user code, and intrinsic, iterator, meta and reflect can only qualify names bound by system code]

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]

NumberLiteral

8 [see Ecma-262 section 7.8.3] todo: type suffixes

RegExpInitialiser

9 [see Ecma-262 section 7.8.5]

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

11 [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**

QualifiedNameIdentifier

3 4 Identifier

4 4 **ReservedIdentifier**

5 4 **StringLiteral**

6 4 **NumberLiteral**

PrimaryName

7 3 Identifier

8 4 PrimaryName **::** QualifiedNameIdentifier
9 4 ParenExpression **::** QualifiedNameIdentifier

ParenExpression
10 3 **(** CommaExpression$^{allowColon, allowIn}$ **)**

FunctionExpression$^{\alpha, \beta}$
11 3 **function** Identifier FunctionSignature FunctionExpressionBody$^{\alpha, \beta}$
12 3 **function** FunctionSignature FunctionExpressionBody$^{\alpha, \beta}$

FunctionExpressionBody$^{\alpha, \beta}$
13 3 Block^{local}
14 4 CommaExpression$^{\alpha, \beta}$

ObjectInitialiser^{noColon}
15 3 InitialiserAttribute **{** FieldList **}**

ObjectInitialiser^{allowColon}
16 3 InitialiserAttribute **{** FieldList **}**
17 4 InitialiserAttribute **{** FieldList **}** **:** RecordType
18 4 InitialiserAttribute **{** FieldList **}** **:** TypeReference

FieldList
19 3 «empty»
20 3 Field
21 3 Field **,** FieldList

Field
22 3 InitialiserAttribute FieldName **:** AssignmentExpression$^{allowColon, allowIn}$
23 4 **get** FieldName GetterSignature FunctionExpressionBody$^{allowColon, allowIn}$
24 4 **set** FieldName SetterSignature FunctionExpressionBody$^{allowColon, allowIn}$

InitialiserAttribute
25 3 «empty»
26 4 **const**
27 4 **var**

FieldName
28 3 PrimaryName
29 3 **StringLiteral**
30 3 **NumberLiteral**
31 4 **ReservedIdentifier**

ArrayInitialiser^{noColon}
32 3 InitialiserAttribute **[** ArrayElements **]**

ArrayInitialiser^{allowColon}
33 3 InitialiserAttribute **[** ArrayElements **]**
34 4 InitialiserAttribute **[** ArrayElements **]** **:** ArrayType
35 4 InitialiserAttribute **[** ArrayElements **]** **:** TypeReference

ArrayElements
36 3 ArrayElementList
37 4 ArrayComprehension

ArrayElementList
38 3 «empty»
39 3 ArrayElement
40 3 SpreadExpression
41 3 **,** ArrayElementList
42 3 ArrayElement **,** ArrayElementList

ArrayElement
43 3 AssignmentExpression$^{allowColon, allowIn}$

SpreadExpression
44 4 **...** AssignmentExpression$^{allowColon, allowIn}$

ArrayComprehension
45 4 ArrayElement **for** **(** TypedPattern^{noIn} **in** CommaExpression$^{allowColon, allowIn}$ **)** ComprehensionClause
46 4 ArrayElement **for each** **(** TypedPattern^{noIn} **in** CommaExpression$^{allowColon, allowIn}$ **)** ComprehensionClause

ComprehensionClause
47 4 «empty»
48 4 **let** ParenExpression ComprehensionClause
49 4 **if** ParenExpression ComprehensionClause
50 4 ComprehensionExpression ComprehensionClause

PrimaryExpression[α, β]

51 3 **null**
52 3 **true**
53 3 **false**
54 3 **NumberLiteral**
55 3 **StringLiteral**
56 3 **RegExpInitialiser**
57 3 ArrayInitialiser[α]
58 3 ObjectInitialiser[α]
59 3 FunctionExpression[α, β]
60 3 ThisExpression
61 3 ParenExpression
62 4 LetExpression[α, β]
63 3 PrimaryName

ThisExpression

64 3 **this**
65 4 **this** [no line break] **function**
66 4 **this** [no line break] **generator**

LetExpression[α, β]

67 4 **let** **(** LetBindingList **)** CommaExpression[α, β]

LetBindingList

68 4 «empty»
69 4 NonemptyLetBindingList

NonemptyLetBindingList

70 4 VariableBinding[allowIn]
71 4 VariableBinding[allowIn] **,** NonemptyLetBindingList

SuperExpression

72 4 **super**
73 4 **super** ParenExpression

Arguments

74 3 **( )**
75 3 **(** SpreadExpression **)**
76 3 **(** ArgumentList **)**
77 3 **(** ArgumentList **,** SpreadExpression **)**

ArgumentList

78 3 AssignmentExpression[allowColon, allowIn]
79 3 ArgumentList **,** AssignmentExpression[allowColon, allowIn]

PropertyOperator

80 4 **.** **ReservedIdentifier**
81 3 **.** PrimaryName
82 4 **.** ParenExpression **::** QualifiedNameIdentifier
83 3 BracketsOrSlice
84 4 TypeApplication

Brackets

85 3 **[** CommaExpression[allowColon, allowIn] **]**

BracketsOrSlice

86 3 **[** CommaExpression[noColon, allowIn] **]**
87 4 **[** SliceExpression **]**

SliceExpression

88 4 OptionalExpression **:** OptionalExpression
89 4 OptionalExpression **:** OptionalExpression **:** OptionalExpression
90 4 **::** OptionalExpression
91 4 OptionalExpression **::**

OptionalExpression

92 4 «empty»
93 4 CommaExpression[noColon, allowIn]

TypeApplication

94 4 **.<** TypeExpressionList **>**
95 4 **.<** TypeExpressionList **>>** [leave **>** in the token stream]
96 4 **.<** TypeExpressionList **>>>** [leave **>>** in the token stream]

MemberExpression[α, β]

97 3 PrimaryExpression[α, β]

98 3 **new** MemberExpression$^{\alpha,\beta}$ Arguments
99 4 SuperExpression PropertyOperator
100 3 MemberExpression$^{\alpha,\beta}$ PropertyOperator

CallExpression$^{\alpha,\beta}$
101 3 MemberExpression$^{\alpha,\beta}$ Arguments
102 3 CallExpression$^{\alpha,\beta}$ Arguments
103 3 CallExpression$^{\alpha,\beta}$ PropertyOperator

NewExpression$^{\alpha,\beta}$
104 3 MemberExpression$^{\alpha,\beta}$
105 3 **new** NewExpression$^{\alpha,\beta}$

LeftHandSideExpression$^{\alpha,\beta}$
106 3 NewExpression$^{\alpha,\beta}$
107 3 CallExpression$^{\alpha,\beta}$

PostfixExpression$^{\alpha,\beta}$
108 3 LeftHandSideExpression$^{\alpha,\beta}$
109 3 LeftHandSideExpression$^{\alpha,\beta}$ [no line break] **++**
110 3 LeftHandSideExpression$^{\alpha,\beta}$ [no line break] **--**

UnaryExpression$^{\alpha,\beta}$
111 3 PostfixExpression$^{\alpha,\beta}$
112 3 **delete** PostfixExpression$^{\alpha,\beta}$
113 3 **void** UnaryExpression$^{\alpha,\beta}$
114 3 **typeof** UnaryExpression$^{\alpha,\beta}$
115 3 **++** PostfixExpression$^{\alpha,\beta}$
116 3 **--** PostfixExpression$^{\alpha,\beta}$
117 3 **+** UnaryExpression$^{\alpha,\beta}$
118 3 **-** UnaryExpression$^{\alpha,\beta}$
119 3 **~** UnaryExpression$^{\alpha,\beta}$
120 3 **!** UnaryExpression$^{\alpha,\beta}$

MultiplicativeExpression$^{\alpha,\beta}$
121 3 UnaryExpression$^{\alpha,\beta}$
122 3 MultiplicativeExpression$^{\alpha,\beta}$ ***** UnaryExpression$^{\alpha,\beta}$
123 3 MultiplicativeExpression$^{\alpha,\beta}$ **/** UnaryExpression$^{\alpha,\beta}$
124 3 MultiplicativeExpression$^{\alpha,\beta}$ **%** UnaryExpression$^{\alpha,\beta}$

AdditiveExpression$^{\alpha,\beta}$
125 3 MultiplicativeExpression$^{\alpha,\beta}$
126 3 AdditiveExpression$^{\alpha,\beta}$ **+** MultiplicativeExpression$^{\alpha,\beta}$
127 3 AdditiveExpression$^{\alpha,\beta}$ **-** MultiplicativeExpression$^{\alpha,\beta}$

ShiftExpression$^{\alpha,\beta}$
128 3 AdditiveExpression$^{\alpha,\beta}$
129 3 ShiftExpression$^{\alpha,\beta}$ **<<** AdditiveExpression$^{\alpha,\beta}$
130 3 ShiftExpression$^{\alpha,\beta}$ **>>** AdditiveExpression$^{\alpha,\beta}$
131 3 ShiftExpression$^{\alpha,\beta}$ **>>>** AdditiveExpression$^{\alpha,\beta}$

RelationalExpression$^{\alpha,\beta}$
132 3 ShiftExpression$^{\alpha,\text{allowIn}}$
133 3 RelationalExpression$^{\alpha,\text{allowIn}}$ **<** ShiftExpression$^{\alpha,\text{allowIn}}$
134 3 RelationalExpression$^{\alpha,\text{allowIn}}$ **>** ShiftExpression$^{\alpha,\text{allowIn}}$
135 3 RelationalExpression$^{\alpha,\text{allowIn}}$ **<=** ShiftExpression$^{\alpha,\text{allowIn}}$
136 3 RelationalExpression$^{\alpha,\text{allowIn}}$ **>=** ShiftExpression$^{\alpha,\text{allowIn}}$
137 3 RelationalExpression$^{\alpha,\text{allowIn}}$ [β == allowIn] **in** ShiftExpression$^{\alpha,\text{allowIn}}$
138 3 RelationalExpression$^{\alpha,\text{allowIn}}$ **instanceof** ShiftExpression$^{\alpha,\text{allowIn}}$
139 4 RelationalExpression$^{\alpha,\text{allowIn}}$ **cast** TypeExpression
140 4 RelationalExpression$^{\alpha,\text{allowIn}}$ **is** TypeExpression
141 4 RelationalExpression$^{\alpha,\text{allowIn}}$ **like** TypeExpression

EqualityExpression$^{\alpha,\beta}$
142 3 RelationalExpression$^{\alpha,\beta}$
143 3 EqualityExpression$^{\alpha,\beta}$ **==** RelationalExpression$^{\alpha,\beta}$
144 3 EqualityExpression$^{\alpha,\beta}$ **!=** RelationalExpression$^{\alpha,\beta}$
145 3 EqualityExpression$^{\alpha,\beta}$ **===** RelationalExpression$^{\alpha,\beta}$
146 3 EqualityExpression$^{\alpha,\beta}$ **!==** RelationalExpression$^{\alpha,\beta}$

BitwiseAndExpression$^{\alpha,\beta}$
147 3 EqualityExpression$^{\alpha,\beta}$
148 3 BitwiseAndExpression$^{\alpha,\beta}$ & EqualityExpression$^{\alpha,\beta}$

BitwiseXorExpression$^{\alpha,\beta}$
149 3 BitwiseAndExpression$^{\alpha,\beta}$
150 3 BitwiseXorExpression$^{\alpha,\beta}$ **^** BitwiseAndExpression$^{\alpha,\beta}$

BitwiseOrExpression$^{\alpha,\beta}$

151 3 BitwiseXorExpression$^{\alpha,\beta}$

152 3 BitwiseOrExpression$^{\alpha,\beta}$ **|** BitwiseXorExpression$^{\alpha,\beta}$

LogicalAndExpression$^{\alpha,\beta}$

153 3 BitwiseOrExpression$^{\alpha,\beta}$

154 3 LogicalAndExpression$^{\alpha,\beta}$ **&&** BitwiseOrExpression$^{\alpha,\beta}$

LogicalOrExpression$^{\alpha,\beta}$

155 3 LogicalAndExpression$^{\alpha,\beta}$

156 3 LogicalOrExpression$^{\alpha,\beta}$ **||** LogicalAndExpression$^{\alpha,\beta}$

ConditionalExpression$^{\alpha,\beta}$

157 4 UnaryTypeExpression

158 4 YieldExpression$^{\alpha,\beta}$

159 3 LogicalOrExpression$^{\alpha,\beta}$

160 3 LogicalOrExpression$^{\alpha,\beta}$ **?** AssignmentExpression$^{noColon,\beta}$

161 **:** AssignmentExpression$^{\alpha,\beta}$

NonAssignmentExpression$^{\alpha,\beta}$

162 4 UnaryTypeExpression

163 4 YieldExpression$^{\alpha,\beta}$

164 3 LogicalOrExpression$^{\alpha,\beta}$

165 3 LogicalOrExpression$^{\alpha,\beta}$ **?** NonAssignmentExpression$^{noColon,\beta}$

166 3 **:** NonAssignmentExpression$^{\alpha,\beta}$

UnaryTypeExpression

167 4 **type** TypeExpression

YieldExpression$^{\alpha,\beta}$

168 4 **yield**

169 4 **yield** [no line break] AssignmentExpression$^{\alpha,\beta}$

AssignmentExpression$^{\alpha,\beta}$

170 3 ConditionalExpression$^{\alpha,\beta}$

171 3 Pattern$^{\alpha,\beta,allowExpr}$ **=** AssignmentExpression$^{\alpha,\beta}$

172 3 SimplePattern$^{\alpha,\beta,allowExpr}$ CompoundAssignmentOperator AssignmentExpression$^{\alpha,\beta}$

CompoundAssignmentOperator

173 3 ***=**

174 3 **/=**

175 3 **%=**

176 3 **+=**

177 3 **-=**

178 3 **<<=**

179 3 **>>=**

180 3 **>>>=**

181 3 **&=**

182 3 **^=**

183 3 **|=**

184 3 **&&=**

185 3 **||=**

CommaExpression$^{\alpha,\beta}$

186 3 AssignmentExpression$^{\alpha,\beta}$

187 3 CommaExpression$^{\alpha,\beta}$ **,** AssignmentExpression$^{\alpha,\beta}$

PATTERNS

γ = { allowExpr, noExpr }

Pattern$^{\alpha,\beta,\gamma}$

188 3 SimplePattern$^{\alpha,\beta,\gamma}$

189 4 ObjectPattern$^{\alpha,\beta,\gamma}$

190 4 ArrayPattern$^{\gamma}$

SimplePattern$^{\alpha,\beta,noExpr}$

191 3 Identifier

SimplePattern$^{\alpha,\beta,allowExpr}$

192 3 LeftHandSideExpression$^{\alpha,\beta}$

ObjectPattern$^{\gamma}$

193 4 **{** FieldListPattern$^{\gamma}$ **}**

FieldListPattern$^{\gamma}$

194 4 «empty»
195 4 FieldPattern$^{\gamma}$
196 4 FieldListPattern$^{\gamma}$ **,**
197 4 FieldListPattern$^{\gamma}$ **,** FieldPattern$^{\gamma}$

FieldPattern$^{\gamma}$
198 4 FieldName
199 4 FieldName **:** Pattern$^{\text{allowColon, allowIn, }\gamma}$

ArrayPattern$^{\gamma}$
200 4 **[** ElementListPattern$^{\gamma}$ **]**

ElementListPattern$^{\gamma}$
201 4 «empty»
202 4 ElementPattern$^{\gamma}$
203 4 **...** SimplePattern$^{\text{allowColon, allowIn, }\gamma}$
204 4 **,** ElementListPattern$^{\gamma}$
205 4 ElementPattern$^{\gamma}$ **,** ElementListPattern$^{\gamma}$

ElementPattern$^{\gamma}$
206 4 Pattern$^{\text{allowColon, allowIn, }\gamma}$

TypedIdentifier
207 3 Identifier
208 4 Identifier **:** TypeExpression

TypedPattern$^{\beta}$
209 3 Pattern$^{\text{allowColon, }\beta\text{, noExpr}}$
210 4 Pattern$^{\text{allowColon, }\beta\text{, noExpr}}$ **:** TypeExpression

LikenedPattern$^{\beta}$
211 4 Pattern$^{\text{allowColon, }\beta\text{, noExpr}}$ **like** TypeExpression

TYPE EXPRESSIONS

TypeExpression
212 4 BasicTypeExpression
213 4 **?** BasicTypeExpression
214 4 **!** BasicTypeExpression

BasicTypeExpression
215 4 *
216 4 **null**
217 4 **undefined**
218 4 TypeReference
219 4 FunctionType
220 4 UnionType
221 4 RecordType
222 4 ArrayType

TypeReference
223 4 PrimaryName
224 4 PrimaryName TypeApplication

FunctionType
225 4 **function** FunctionSignatureType

FunctionSignatureType
226 4 TypeParameters **(** ParametersType **)** ResultType
227 4 TypeParameters **(** **this** **:** PrimaryName **)** ResultType
228 4 TypeParameters **(** **this** **:** PrimaryName **,** NonemptyParametersType **)** ResultType

ParametersType
229 4 «empty»
230 4 NonemptyParametersType

NonemptyParametersType
231 4 ParameterType **,** NonemptyParametersType
232 4 ParameterType
233 4 OptionalParametersType
234 4 RestParameterType

OptionalParametersType
235 4 OptionalParameterType
236 4 OptionalParameterType **,** OptionalParametersType

OptionalParameterType

237 4 ParameterType
238 4 ParameterType **=**

ParameterType
239 4 TypeExpression
240 4 Identifier **:** TypeExpression

RestParameterType
241 4 **...**
242 4 **...** Identifier

UnionType
243 4 **(** TypeUnionList **)**

TypeUnionList
244 4 «empty»
245 4 NonemptyTypeUnionList

NonemptyTypeUnionList
246 4 TypeExpression
247 4 TypeExpression **|** NonemptyTypeUnionList

RecordType
248 4 **{** FieldTypeList **}**

FieldTypeList
249 4 «empty»
250 4 FieldType
251 4 FieldType **,** FieldTypeList

FieldType
252 4 FieldName
253 4 FieldName **:** TypeExpression

ArrayType
254 4 **[** ElementTypeList **]**

ElementTypeList
255 4 «empty»
256 4 TypeExpression
257 4 **...** TypeExpression
258 4 **,** ElementTypeList
259 4 TypeExpression **,** ElementTypeList

TypeExpressionList
260 4 TypeExpression
261 4 TypeExpressionList **,** TypeExpression

STATEMENTS

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

Statement$^{\tau,\omega}$
262 3 BlockStatement$^{\tau}$
263 3 BreakStatement Semicolon$^{\omega}$
264 3 ContinueStatement Semicolon$^{\omega}$
265 3 DoStatement Semicolon$^{\omega}$
266 3 ExpressionStatement Semicolon$^{\omega}$
267 3 ForStatement$^{\omega}$
268 3 IfStatement$^{\omega}$
269 3 LabeledStatement$^{\omega}$
270 3 ReturnStatement Semicolon$^{\omega}$
271 3 SwitchStatement
272 4 SwitchTypeStatement
273 3 ThrowStatement Semicolon$^{\omega}$
274 3 TryStatement
275 3 WhileStatement$^{\omega}$
276 3 WithStatement$^{\omega}$

Substatement$^{\omega}$
277 3 EmptyStatement
278 3 Statement$^{local,\ \omega}$
279 3 VariableDefinition$^{noIn,\ statement}$

Semicolon^{abbrev}
280 3 **;**

281 3 **VirtualSemicolon**
282 3 «empty»

Semicolon[noShortIf]
283 3 **;**
284 3 **VirtualSemicolon**
285 3 «empty»

Semicolon[full]
286 3 **;**
287 3 **VirtualSemicolon**

EmptyStatement
288 3 **;**

ExpressionStatement
289 3 [lookahead ![**{**, **const**, **function**, **var**]] CommaExpression[allowColon, allowIn]

BlockStatement[ω]
290 3 Block[local]

LabeledStatement[ω]
291 3 Identifier **:** Substatement[ω]

IfStatement[abbrev]
292 3 **if** ParenExpression Substatement[abbrev]
293 3 **if** ParenExpression Substatement[noShortIf] **else** Substatement[abbrev]

IfStatement[full]
294 3 **if** ParenExpression Substatement[full]
295 3 **if** ParenExpression Substatement[noShortIf] **else** Substatement[full]

IfStatement[noShortIf]
296 3 **if** ParenExpression Substatement[noShortIf] **else** Substatement[noShortIf]

WithStatement[ω]
297 3 **with** ParenExpression Substatement[ω]

SwitchStatement
298 3 **switch** ParenExpression **{** CaseElements **}**

3 CaseElements
299 3 CaseClauses[full] DefaultClause[full] CaseClauses[abbrev]
300 3 CaseClauses[full] DefaultClause[abbrev]
301 3 CaseClauses[abbrev]

3 CaseClauses[ω]
302 3 «empty»
303 3 CaseClauses[full] CaseClause[ω]

3 CaseClause[ω]
304 3 **case** CommaExpression[allowColon, allowIn] **:** Directives[local, ω]

3 DefaultClause[ω]
305 3 **default** **:** Directives[local, ω]

SwitchTypeStatement
306 4 **switch** **type** ParenExpression **{** TypeCaseElements **}**

TypeCaseElements
307 4 TypeCaseElement
308 4 TypeCaseElements TypeCaseElement

TypeCaseElement
309 4 **case** **(** TypedPattern[allowColon, allowIn] **)** Block[local]

DoStatement
310 3 **do** Substatement[abbrev] **while** ParenExpression

WhileStatement[ω]
311 3 **while** ParenExpression Substatement[ω]

ForStatement[ω]
312 3 **for** **(** ForInitialiser **;** OptionalExpression **;** OptionalExpression **)** Substatement[ω]
313 3 **for** **(** ForInBinding **in** CommaExpression[allowColon, allowIn] **)** Substatement[ω]
314 4 **for** **each** **(** ForInBinding **in** CommaExpression[allowColon, allowIn] **)** Substatement[ω]

ForInitialiser

315 3 «empty»

316 3 CommaExpression[allowColon, noIn]

317 3 VariableDefinition[noIn, ~]

ForInBinding

318 3 Pattern[allowColon, noIn, allowExpr]

319 3 VariableDefinitionKind[local] VariableBinding[noIn]

ContinueStatement

320 3 **continue**

321 3 **continue** [no line break] Identifier

BreakStatement

322 3 **break**

323 3 **break** [no line break] Identifier

ReturnStatement

324 3 **return**

325 3 **return** [no line break] CommaExpression[allowColon, allowIn]

ThrowStatement

326 3 **throw** CommaExpression[allowColon, allowIn]

TryStatement

327 3 **try** Block[local] CatchClauses

328 3 **try** Block[local] CatchClauses **finally** Block[local]

329 3 **try** Block[local] **finally** Block[local]

CatchClauses

330 3 CatchClause

331 3 CatchClauses CatchClause

CatchClause

332 3 **catch** **(** Parameter **)** Block[local]

DIRECTIVES

Directives[~]

333 3 «empty»

334 3 DirectivesPrefix[~] Directive[~, abbrev]

DirectivesPrefix[~]

335 3 «empty»

336 4 DirectivesPrefix[~] Pragma

337 3 DirectivesPrefix[~] Directive[~, full]

Directive[class, ω]

338 4 **static** [no line break] Block[local, ω]

Directive[~, ω]

339 3 EmptyStatement

340 3 Statement[~, ω]

341 3 AnnotatableDirective[~, ω]

AnnotatableDirective[global, ω]

342 4 Attribute[global] [no line break] AnnotatableDirective[global, ω]

343 3 VariableDefinition[allowIn, global] Semicolon[ω]

344 3 FunctionDefinition[global, ω]

345 4 ClassDefinition

346 4 InterfaceDefinition

347 4 NamespaceDefinition Semicolon[ω]

348 4 TypeDefinition Semicolon[ω]

349 4 PackageDefinition

AnnotatableDirective[class, ω]

350 4 Attribute[class] [no line break] AnnotatableDirective[class, ω]

351 3 VariableDefinition[allowIn, class] Semicolon[ω]

352 3 FunctionDefinition[class, ω]

353 4 NamespaceDefinition Semicolon[ω]

354 4 TypeDefinition Semicolon[ω]

AnnotatableDirective[interface, ω]

355 4 Attribute[interface] [no line break] AnnotatableDirective[interface, ω]

356 4 FunctionDeclaration Semicolon[ω]

AnnotatableDirective[local, ω]

357 3 VariableDefinition[allowIn, local] Semicolon[ω]
358 3 FunctionDefinition[local, ω]
359 4 NamespaceDefinition Semicolon[ω]
360 4 TypeDefinition Semicolon[ω]

Attribute[global]
361 4 PrimaryName
362 4 **dynamic**
363 4 **final**
364 4 **native**

Attribute[class]
365 4 PrimaryName
366 4 **final**
367 4 **native**
368 4 **override**
369 4 **prototype**
370 4 **static**

Attribute[interface]
371 4 PrimaryName

DEFINITIONS

VariableDefinition[β, τ]
372 3 VariableDefinitionKind[τ] VariableBindingList[β]

VariableDefinitionKind[statement]
373 3 **var**

VariableDefinitionKind[class]
374 4 **const**
375 4 **var**

VariableDefinitionKind[τ]
376 4 **const**
377 4 **let**
378 3 **var**

VariableBindingList[β]
379 3 VariableBinding[β]
380 3 VariableBindingList[β] **,** VariableBinding[β]

VariableBinding[β]
381 3 TypedIdentifier
382 3 TypedPattern[β] VariableInitialisation[β]

VariableInitialisation[β]
383 3 **=** AssignmentExpression[allowColon, β]

FunctionDeclaration
384 4 **function** FunctionName FunctionSignatureType

FunctionDefinition[class, ω]
385 4 **function** Identifier [identifier == outer classname] ConstructorSignature Block[local]
386 4 **function** Identifier FunctionSignature FunctionBody[allowIn, ω]
387 4 **function get** Identifier GetterSignature FunctionBody[allowIn, ω]
388 4 **function set** Identifier SetterSignature FunctionBody[allowIn, ω]

FunctionDefinition[local, ω]
389 4 **const function** Identifier FunctionSignature FunctionBody[allowIn, ω]
390 4 **function** Identifier FunctionSignature FunctionBody[allowIn, ω]

FunctionDefinition[τ, ω]
391 4 **const function** Identifier FunctionSignature FunctionBody[allowIn, ω]
392 4 **function** Identifier FunctionSignature FunctionBody[allowIn, ω]
393 4 **function get** Identifier GetterSignature FunctionBody[allowIn, ω]
394 4 **function set** Identifier SetterSignature FunctionBody[allowIn, ω]

FunctionSignature
395 3 TypeParameters **(** Parameters **)** ResultTypeOrLike
396 4 TypeParameters **(** **this** **:** PrimaryName **)** ResultTypeOrLike
397 4 TypeParameters **(** **this** **:** PrimaryName **,** NonemptyParameters **)** ResultTypeOrLike

GetterSignature
398 4 TypeParameters **(** **)** ResultTypeOrLike

SetterSignature
399 4 TypeParameters **(** Parameter **)** **:** ResultTypeVoid

FunctionBody[α, β, ω]
400 3 Block[local]
401 4 CommaExpression[α, β] Semicolon[ω]

TypeParameters
402 3 «empty»
403 4 **.<** TypeParameterList **>**

TypeParameterList
404 4 Identifier
405 4 Identifier **,** TypeParameterList

Parameters
406 3 «empty»
407 3 NonemptyParameters

NonemptyParameters
408 3 Parameter **,** NonemptyParameters
409 3 Parameter
410 3 OptionalParameters
411 4 RestParameter

OptionalParameters
412 4 OptionalParameter
413 4 OptionalParameter , OptionalParameters

OptionalParameter
414 4 Parameter **=** NonAssignmentExpression[allowIn]

Parameter
415 3 ParameterAttribute TypedPattern[allowIn]
416 3 ParameterAttribute LikenedPattern[allowIn]

ParameterAttribute
417 3 «empty»
418 4 **const**

RestParameter
419 4 **...**
420 4 **...** SimplePattern[allowColon, allowIn, noExpr]

ResultTypeOrLike
421 4 ResultType
422 4 **like** TypeExpression

ResultType
423 4 «empty»
424 4 **: void**
425 4 **:** TypeExpression

ResultTypeVoid
426 4 «empty»
427 4 **: void**

ResultTypeBoolean
428 4 «empty»
429 4 **: boolean**

ConstructorSignature
430 4 **(** Parameters **)**
431 4 **(** Parameters **) :** ConstructorInitialiser

ConstructorInitialiser
432 4 SettingList
433 4 SettingList SuperInitialiser
434 4 SuperInitialiser

SettingList
435 4 Setting
436 4 SettingList **,** Setting

Setting
437 4 Pattern[allowIn, allowExpr] VariableInitialisation[allowIn]

SuperInitialiser
438 4 **super** Arguments

ClassDefinition
439 4 **class** Identifier TypeSignature ClassInheritance ClassBody

TypeSignature
440 4 TypeParameters
441 4 **!** TypeParameters

ClassInheritance
442 4 «empty»
443 4 **extends** TypeReference
444 4 **implements** TypeReferenceList
445 4 **extends** TypeReference **implements** TypeReferenceList

TypeReferenceList
446 4 TypeReference
447 4 TypeReferenceList , TypeReference

ClassBody
448 4 Block[class]

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

InterfaceInheritance
450 4 «empty»
451 4 **extends** TypeReferenceList

InterfaceBody
452 4 Block[interface]

TypeDefinition
453 4 **type** Identifier TypeSignature TypeInitialisation

TypeInitialisation
454 4 **=** TypeExpression

NamespaceDefinition
455 4 **namespace** Identifier NamespaceInitialisation

NamespaceInitialisation
456 4 «empty»
457 4 **= StringLiteral**
458 4 **=** PrimaryName

PRAGMAS

Pragma
459 4 UsePragma Semicolon[full]

UsePragma
460 4 **use** PragmaItems

PragmaItems
461 4 PragmaItem
462 4 PragmaItems **,** PragmaItem

PragmaItem
463 4 **default namespace** PrimaryName
464 4 **namespace** PrimaryName
465 4 **standard**
466 4 **strict**

BLOCKS AND PROGRAMS

Block'
467 3 **{** Directives' **}**

Program
468 3 Directives[global]

Revision History:

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', 'reflet', '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)

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

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

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

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