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 const continue debugger default delete do dynamic else false final finally for function if in instanceof interface is let 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 4 Identifier

4 4 **ReservedIdentifier**

PrimaryName

5 3 Identifier

6 4 NamespaceName **::** PropertyIdentifier

7 4 ParenExpression **::** PropertyIdentifier

NamespaceName
8 4 PrimaryName
9 4 StringLiteral

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

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

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

ObjectInitialiser$^{\text{noColon}}$
15 3 InitialiserAttribute **{** FieldList **}**

ObjectInitialiser$^{\text{allowColon}}$
16 3 InitialiserAttribute **{** FieldList **}**
17 4 InitialiserAttribute **{** FieldList **}** **:** RecordType
18 4 InitialiserAttribute **{** FieldList **}** **:** TypeName

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

Field
22 3 InitialiserAttribute FieldName **:** AssignmentExpression$^{\text{allowColon, allowIn}}$
23 3 **__proto__** **:** AssignmentExpression$^{\text{allowColon, allowIn}}$
24 4 **get** FieldName GetterSignature FunctionExpressionBody$^{\text{allowColon, allowIn}}$
25 4 **set** FieldName SetterSignature FunctionExpressionBody$^{\text{allowColon, allowIn}}$

InitialiserAttribute
26 3 «empty»
27 4 **const**
28 4 **var**

FieldName
29 3 [lookahead ![**__proto__**]] PrimaryName
30 3 **StringLiteral**
31 3 **NumberLiteral**
32 4 **ReservedIdentifier**

ArrayInitialiser$^{\text{noColon}}$
33 3 InitialiserAttribute **[** ArrayElements **]**

ArrayInitialiser$^{\text{allowColon}}$
34 3 InitialiserAttribute **[** ArrayElements **]**
35 4 InitialiserAttribute **[** ArrayElements **]** **:** ArrayType
36 4 InitialiserAttribute **[** ArrayElements **]** **:** TypeName

ArrayElements
37 3 ArrayElementList
38 4 ArrayComprehension

ArrayElementList
39 3 «empty»
40 3 ArrayElement
41 3 SpreadExpression
42 3 **,** ArrayElementList
43 3 ArrayElement **,** ArrayElementList

ArrayElement
44 3 AssignmentExpression$^{\text{allowColon, allowIn}}$

SpreadExpression
45 4 **...** AssignmentExpression$^{\text{allowColon, allowIn}}$

ArrayComprehension
46 4 ArrayElement ComprehensionExpression

ComprehensionExpression
47 4 **for** **(** TypedPattern$^{\text{noIn}}$ **in** CommaExpression$^{\text{allowColon, allowIn}}$ **)** ComprehensionClause

48 4 **for each (** TypedPattern^{noIn} **in** CommaExpression$^{allowColon, allowIn}$ **)** ComprehensionClause

ComprehensionClause
49 4 «empty»
50 4 **let** ParenExpression ComprehensionClause
51 4 **if** ParenExpression ComprehensionClause
52 4 ComprehensionExpression

PrimaryExpression$^{\alpha, \beta}$
53 3 **null**
54 3 **true**
55 3 **false**
56 3 **DoubleLiteral**
57 4 **DecimalLiteral**
58 3 **StringLiteral**
59 3 **RegExpInitialiser**
60 3 ArrayInitialiser$^{\alpha}$
61 3 ObjectInitialiser$^{\alpha}$
62 3 FunctionExpression$^{\alpha, \beta}$
63 3 ThisExpression
64 3 ParenExpression
65 4 LetExpression$^{\alpha, \beta}$
66 3 PrimaryName

ThisExpression
67 3 **this**
68 4 **this** [no line break] **function**
69 4 **this** [no line break] **generator**

LetExpression$^{\alpha, \beta}$
70 4 **let (** LetBindingList **)** CommaExpression$^{\alpha, \beta}$

LetBindingList
71 4 «empty»
72 4 NonemptyLetBindingList

NonemptyLetBindingList
73 4 VariableBinding^{allowIn}
74 4 VariableBinding^{allowIn} **,** NonemptyLetBindingList

SuperExpression
75 4 **super**
76 4 **super** ParenExpression

Arguments
77 3 **( )**
78 3 **(** SpreadExpression **)**
79 3 **(** ArgumentList **)**
80 3 **(** ArgumentList **,** SpreadExpression **)**

ArgumentList
81 3 AssignmentExpression$^{allowColon, allowIn}$
82 3 ArgumentList **,** AssignmentExpression$^{allowColon, allowIn}$

PropertyOperator
83 4 **.** **ReservedIdentifier**
84 3 **.** PrimaryName
85 3 BracketsOrSlice
86 4 TypeApplication

Brackets
87 3 **[** CommaExpression$^{allowColon, allowIn}$ **]**

BracketsOrSlice
88 3 **[** CommaExpression$^{noColon, allowIn}$ **]**
89 4 **[** SliceExpression **]**

SliceExpression
90 4 OptionalExpression **:** OptionalExpression
91 4 OptionalExpression **:** OptionalExpression **:** OptionalExpression
92 4 **::** OptionalExpression
93 4 OptionalExpression **::**

OptionalExpression
94 4 «empty»
95 4 CommaExpression$^{noColon, allowIn}$

TypeApplication

96 4 .< TypeExpressionList >

97 4 .< TypeExpressionList >> [leave > in the token stream]

98 4 .< TypeExpressionList >>> [leave >> in the token stream]

MemberExpression[α, β]

99 3 PrimaryExpression[α, β]

100 3 new MemberExpression[α, β] Arguments

101 4 SuperExpression PropertyOperator

102 3 MemberExpression[α, β] PropertyOperator

CallExpression[α, β]

103 3 MemberExpression[α, β] Arguments

104 3 CallExpression[α, β] Arguments

105 3 CallExpression[α, β] PropertyOperator

NewExpression[α, β]

106 3 MemberExpression[α, β]

107 3 new NewExpression[α, β]

LeftHandSideExpression[α, β]

108 3 NewExpression[α, β]

109 3 CallExpression[α, β]

PostfixExpression[α, β]

110 3 LeftHandSideExpression[α, β]

111 3 LeftHandSideExpression[α, β] [no line break] ++

112 3 LeftHandSideExpression[α, β] [no line break] --

UnaryExpression[α, β]

113 3 PostfixExpression[α, β]

114 3 delete PostfixExpression[α, β]

115 3 void UnaryExpression[α, β]

116 3 typeof UnaryExpression[α, β]

117 3 ++ PostfixExpression[α, β]

118 3 -- PostfixExpression[α, β]

119 3 + UnaryExpression[α, β]

120 3 - UnaryExpression[α, β]

121 3 ~ UnaryExpression[α, β]

122 3 ! UnaryExpression[α, β]

MultiplicativeExpression[α, β]

123 3 UnaryExpression[α, β]

124 3 MultiplicativeExpression[α, β] * UnaryExpression[α, β]

125 3 MultiplicativeExpression[α, β] / UnaryExpression[α, β]

126 3 MultiplicativeExpression[α, β] % UnaryExpression[α, β]

AdditiveExpression[α, β]

127 3 MultiplicativeExpression[α, β]

128 3 AdditiveExpression[α, β] + MultiplicativeExpression[α, β]

129 3 AdditiveExpression[α, β] - MultiplicativeExpression[α, β]

ShiftExpression[α, β]

130 3 AdditiveExpression[α, β]

131 3 ShiftExpression[α, β] << AdditiveExpression[α, β]

132 3 ShiftExpression[α, β] >> AdditiveExpression[α, β]

133 3 ShiftExpression[α, β] >>> AdditiveExpression[α, β]

RelationalExpression[α, β]

134 3 ShiftExpression[α, β]

135 3 RelationalExpression[α, β] < ShiftExpression[α, β]

136 3 RelationalExpression[α, β] > ShiftExpression[α, β]

137 3 RelationalExpression[α, β] <= ShiftExpression[α, β]

138 3 RelationalExpression[α, β] >= ShiftExpression[α, β]

139 3 RelationalExpression[α, β] [β == allowIn] in ShiftExpression[α, β]

140 3 RelationalExpression[α, β] instanceof ShiftExpression[α, β]

141 4 RelationalExpression[α, β] cast TypeExpression

142 4 RelationalExpression[α, β] is TypeExpression

143 4 RelationalExpression[α, β] like TypeExpression

EqualityExpression[α, β]

144 3 RelationalExpression[α, β]

145 3 EqualityExpression[α, β] == RelationalExpression[α, β]

146 3 EqualityExpression[α, β] != RelationalExpression[α, β]

147 3 EqualityExpression[α, β] === RelationalExpression[α, β]

148 3 EqualityExpression[α, β] !== RelationalExpression[α, β]

BitwiseAndExpression$^{\alpha,\beta}$

149 3 EqualityExpression$^{\alpha,\beta}$

150 3 BitwiseAndExpression$^{\alpha,\beta}$ **&** EqualityExpression$^{\alpha,\beta}$

BitwiseXorExpression$^{\alpha,\beta}$

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

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

BitwiseOrExpression$^{\alpha,\beta}$

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

154 3 BitwiseOrExpression$^{\alpha,\beta}$ **|** BitwiseXorExpression$^{\alpha,\beta}$

LogicalAndExpression$^{\alpha,\beta}$

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

156 3 LogicalAndExpression$^{\alpha,\beta}$ **&&** BitwiseOrExpression$^{\alpha,\beta}$

LogicalOrExpression$^{\alpha,\beta}$

157 3 LogicalAndExpression$^{\alpha,\beta}$

158 3 LogicalOrExpression$^{\alpha,\beta}$ **||** LogicalAndExpression$^{\alpha,\beta}$

ConditionalExpression$^{\alpha,\beta}$

159 4 UnaryTypeExpression

160 4 YieldExpression$^{\alpha,\beta}$

161 3 LogicalOrExpression$^{\alpha,\beta}$

162 3 LogicalOrExpression$^{\alpha,\beta}$ **?** AssignmentExpression$^{noColon,\beta}$

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

NonAssignmentExpression$^{\alpha,\beta}$

164 4 UnaryTypeExpression

165 4 YieldExpression$^{\alpha,\beta}$

166 3 LogicalOrExpression$^{\alpha,\beta}$

167 3 LogicalOrExpression$^{\alpha,\beta}$ **?** NonAssignmentExpression$^{noColon,\beta}$

168 3 **:** NonAssignmentExpression$^{\alpha,\beta}$

UnaryTypeExpression

169 4 **type** TypeExpression

YieldExpression$^{\alpha,\beta}$

170 4 **yield**

171 4 **yield** [no line break] AssignmentExpression$^{\alpha,\beta}$

AssignmentExpression$^{\alpha,\beta}$

172 3 ConditionalExpression$^{\alpha,\beta}$

173 3 Pattern$^{\alpha,\beta,allowExpr}$ **=** AssignmentExpression$^{\alpha,\beta}$

174 3 SimplePattern$^{\alpha,\beta,allowExpr}$ CompoundAssignmentOperator AssignmentExpression$^{\alpha,\beta}$

CompoundAssignmentOperator

175 3 ***=**

176 3 **/=**

177 3 **%=**

178 3 **+=**

179 3 **-=**

180 3 **<<=**

181 3 **>>=**

182 3 **>>>=**

183 3 **&=**

184 3 **^=**

185 3 **|=**

186 3 **&&=**

187 3 **||=**

CommaExpression$^{\alpha,\beta}$

188 3 AssignmentExpression$^{\alpha,\beta}$

189 3 CommaExpression$^{\alpha,\beta}$ **,** AssignmentExpression$^{\alpha,\beta}$

PATTERNS

γ = { allowExpr, noExpr }

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

190 3 SimplePattern$^{\alpha,\beta,\gamma}$

191 4 ObjectPattern$^{\alpha,\beta,\gamma}$

192 4 ArrayPattern$^{\gamma}$

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

193 3 Identifier

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

194 3 LeftHandSideExpression$^{\alpha,\beta}$

ObjectPattern$^{\gamma}$

195 4 **{** FieldListPattern$^{\gamma}$ **}**

FieldListPattern$^{\gamma}$

196 4 «empty»

197 4 FieldPattern$^{\gamma}$

198 4 FieldListPattern$^{\gamma}$ **,**

199 4 FieldListPattern$^{\gamma}$ **,** FieldPattern$^{\gamma}$

FieldPattern$^{\gamma}$

200 4 FieldName

201 4 FieldName **:** Pattern$^{\text{allowColon, allowIn, }\gamma}$

ArrayPattern$^{\gamma}$

202 4 **[** ElementListPattern$^{\gamma}$ **]**

ElementListPattern$^{\gamma}$

203 4 «empty»

204 4 ElementPattern$^{\gamma}$

205 4 **...** SimplePattern$^{\text{allowColon, allowIn, }\gamma}$

206 4 **,** ElementListPattern$^{\gamma}$

207 4 ElementPattern$^{\gamma}$ **,** ElementListPattern$^{\gamma}$

ElementPattern$^{\gamma}$

208 4 Pattern$^{\text{allowColon, allowIn, }\gamma}$

TypedIdentifier

209 3 Identifier

210 4 Identifier **:** TypeExpression

TypedPattern$^{\beta}$

211 3 Pattern$^{\text{allowColon, }\beta\text{, noExpr}}$

212 4 Pattern$^{\text{allowColon, }\beta\text{, noExpr}}$ **:** TypeExpression

LikenedPattern$^{\beta}$

213 4 Pattern$^{\text{allowColon, }\beta\text{, noExpr}}$ **like** TypeExpression

TYPE EXPRESSIONS

TypeExpression

214 4 BasicTypeExpression

215 4 **?** BasicTypeExpression

216 4 **!** BasicTypeExpression

BasicTypeExpression

217 4 *

218 4 **null**

219 4 **undefined**

220 4 TypeName

221 4 FunctionType

222 4 UnionType

223 4 RecordType

224 4 ArrayType

TypeName

225 4 PrimaryName

226 4 PrimaryName TypeApplication

FunctionType

227 4 **function** FunctionSignatureType

FunctionSignatureType

228 4 TypeParameters **(** ParametersType **)** ResultType

229 4 TypeParameters **(** **this** **:** TypeName **)** ResultType

230 4 TypeParameters **(** **this** **:** TypeName **,** NonemptyParametersType **)** ResultType

ParametersType

231 4 «empty»

232 4 NonemptyParametersType

233 4 NonemptyParametersType RestParameterType

NonemptyParametersType

234 4 ParameterType **,** NonemptyParametersType

235 4 ParameterType

236 4 OptionalParametersType

OptionalParametersType
237 4 OptionalParameterType
238 4 OptionalParameterType , OptionalParametersType

OptionalParameterType
239 4 ParameterType =

ParameterType
240 4 TypeExpression
241 4 Identifier : TypeExpression

RestParameterType
242 4 ...
243 4 ... Identifier

UnionType
244 4 (TypeUnionList)

TypeUnionList
245 4 «empty»
246 4 NonemptyTypeUnionList

NonemptyTypeUnionList
247 4 TypeExpression
248 4 TypeExpression | NonemptyTypeUnionList

RecordType
249 4 { FieldTypeList }

FieldTypeList
250 4 «empty»
251 4 FieldType
252 4 FieldType , FieldTypeList

FieldType
253 4 FieldName
254 4 FieldName : TypeExpression

ArrayType
255 4 [ElementTypeList]

ElementTypeList
256 4 «empty»
257 4 TypeExpression
258 4 ... TypeExpression
259 4 , ElementTypeList
260 4 TypeExpression , ElementTypeList

TypeExpressionList
261 4 TypeExpression
262 4 TypeExpressionList , TypeExpression

STATEMENTS

τ = { global, class, interface, local, statement }
ω = { abbrev, noShortIf, full I713}

Statement[τ, ω]
263 3 BlockStatement
264 3 BreakStatement Semicolon[ω]
265 3 ContinueStatement Semicolon[ω]
266 3 DoStatement Semicolon[ω]
267 3 ExpressionStatement Semicolon[ω]
268 3 ForStatement[ω]
269 3 IfStatement[ω]
270 3 LabeledStatement[τ]
271 4 LetBlockStatement
272 3 ReturnStatement Semicolon[ω]
273 3 SwitchStatement
274 4 SwitchTypeStatement
275 3 ThrowStatement Semicolon[ω]
276 3 TryStatement
277 3 WhileStatement[ω]
278 3 WithStatement[ω]

Substatement[w]

279 3 EmptyStatement

280 3 Statement[local, w]

281 3 VariableDefinition[noIn, statement]

Semicolon[abbrev]

282 3 **;**

283 3 **VirtualSemicolon**

284 3 «empty»

Semicolon[noShortIf]

285 3 **;**

286 3 **VirtualSemicolon**

287 3 «empty»

Semicolon[full]

288 3 **;**

289 3 **VirtualSemicolon**

EmptyStatement

290 3 **;**

ExpressionStatement

291 3 [lookahead ∉{ **{**, **const**, **function**, **let**, **var** }] CommaExpression[allowColon, allowIn]

BlockStatement[x]

292 3 Block[local]

LabeledStatement[w]

293 3 Identifier **:** Substatement[w]

LetBlockStatement

294 4 **let** **(** LetBindingList **)** Block[local]

IfStatement[abbrev]

295 3 **if** ParenExpression Substatement[abbrev]

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

IfStatement[full]

297 3 **if** ParenExpression Substatement[full]

298 3 **if** ParenExpression Substatement[noShortIf] **else** Substatement[full]

IfStatement[noShortIf]

299 3 **if** ParenExpression Substatement[noShortIf] **else** Substatement[noShortIf]

WithStatement[w]

300 3 **with** ParenExpression Substatement[w]

SwitchStatement

301 3 **switch** ParenExpression **{** CaseElements **}**

3 CaseElements

302 3 CaseClauses[full] DefaultClause[full] CaseClauses[abbrev]

303 3 CaseClauses[full] DefaultClause[abbrev]

304 3 CaseClauses[abbrev]

3 CaseClauses[w]

305 3 «empty»

306 3 CaseClauses[full] CaseClause[w]

3 CaseClause[w]

307 3 **case** CommaExpression[allowColon, allowIn] **:** Directives[local, w]

3 DefaultClause[w]

308 3 **default** **:** Directives[local, w]

SwitchTypeStatement

309 4 **switch** **type** ParenExpression **{** TypeCaseElements **}**

TypeCaseElements

310 4 TypeCaseElement

311 4 TypeCaseElements TypeCaseElement

TypeCaseElement

312 4 **case** **(** TypedPattern[allowColon, allowIn] **)** Block[local]

DoStatement

313 3 **do** Substatement[abbrev] **while** ParenExpression

WhileStatement[ω]
314 3 **while** ParenExpression Substatement[ω]

ForStatement[ω]
315 3 **for** **(** ForInitialiser **;** OptionalExpression **;** OptionalExpression **)** Substatement[ω]
316 3 **for** **(** ForInBinding **in** CommaExpression[allowColon, allowIn] **)** Substatement[ω]
317 4 **for** **each** **(** ForInBinding **in** CommaExpression[allowColon, allowIn] **)** Substatement[ω]

ForInitialiser
318 3 «empty»
319 3 CommaExpression[allowColon, noIn]
320 3 VariableDefinition[noIn, ν]

ForInBinding
321 3 Pattern[allowColon, noIn, allowExpr]
322 3 VariableDefinitionKind[local] VariableBinding[noIn]

ContinueStatement
323 3 **continue**
324 3 **continue** [no line break] Identifier

BreakStatement
325 3 **break**
326 3 **break** [no line break] Identifier

ReturnStatement
327 3 **return**
328 3 **return** [no line break] CommaExpression[allowColon, allowIn]

ThrowStatement
329 3 **throw** CommaExpression[allowColon, allowIn]

TryStatement
330 3 **try** Block[local] CatchClauses
331 3 **try** Block[local] CatchClauses **finally** Block[local]
332 3 **try** Block[local] **finally** Block[local]

CatchClauses
333 3 CatchClause
334 3 CatchClauses CatchClause

CatchClause
335 3 **catch** **(** Parameter **)** Block[local]

DIRECTIVES

Directives[τ]
336 3 «empty»
337 3 DirectivesPrefix[τ] Directive[τ, abbrev]

DirectivesPrefix[τ]
338 3 «empty»
339 4 DirectivesPrefix[τ] Pragma[ε]
340 3 DirectivesPrefix[τ] Directive[τ, full]

Directive[class, ω]
341 4 **static** [no line break] Block[local, ω]
342 4 AnnotatableDirective[class, ω]

Directive[interface, ω]
343 4 AnnotatableDirective[interface, ω]

Directive[τ, ω]
344 3 EmptyStatement
345 3 Statement[τ, ω]
346 3 AnnotatableDirective[τ, ω]

AnnotatableDirective[global, ω]
347 4 Attribute[global] [no line break] AnnotatableDirective[global, ω]
348 3 VariableDefinition[allowIn, global] Semicolon[ω]
349 3 FunctionDefinition[global, ω]
350 4 NamespaceDefinition Semicolon[ω]
351 4 ClassDeclaration Semicolon[ω]
352 4 ClassDefinition
353 4 InterfaceDeclaration Semicolon[ω]

354 4 InterfaceDefinition
355 4 TypeDeclaration Semicolon$^{\omega}$
356 4 TypeDefinition Semicolon$^{\omega}$

AnnotatableDirective$^{class,\ \omega}$
357 4 Attribute^{class} [no line break] AnnotatableDirective$^{class,\ \omega}$
358 3 VariableDefinition$^{allowIn,\ class}$ Semicolon$^{\omega}$
359 3 FunctionDefinition$^{class,\ \omega}$
360 4 NamespaceDefinition Semicolon$^{\omega}$
361 4 TypeDefinition Semicolon$^{\omega}$

AnnotatableDirective$^{interface,\ \omega}$
362 4 Attribute^{interface} [no line break] AnnotatableDirective$^{interface,\ \omega}$
363 4 FunctionDeclaration Semicolon$^{\omega}$

AnnotatableDirective$^{local,\ \omega}$
364 3 VariableDefinition$^{allowIn,\ local}$ Semicolon$^{\omega}$
365 3 FunctionDefinition$^{local,\ \omega}$

Attribute^{global}
366 4 NamespaceName
367 4 **dynamic**
368 4 **final**
369 4 **native**

Attribute^{class}
370 4 NamespaceName
371 4 **final**
372 4 **native**
373 4 **override**
374 4 **__proto__**
375 4 **static**

Attribute^{interface}
376 4 NamespaceName

DEFINITIONS

VariableDefinition$^{\beta,\ \tau}$
377 3 VariableDefinitionKind$^{\tau}$ VariableBindingList$^{\beta}$

VariableDefinitionKind^{statement}
378 3 **var**

VariableDefinitionKind$^{\tau}$
379 4 **const**
380 4 **let**
381 3 **var**

VariableBindingList$^{\beta}$
382 3 VariableBinding$^{\beta}$
383 3 VariableBindingList$^{\beta}$ **,** VariableBinding$^{\beta}$

VariableBinding$^{\beta}$
384 3 TypedIdentifier
385 3 TypedPattern$^{\beta}$ VariableInitialisation$^{\beta}$

VariableInitialisation$^{\beta}$
386 3 **=** AssignmentExpression$^{allowColon,\ \beta}$

FunctionDeclaration
387 4 **function** PropertyIdentifier FunctionSignatureType
388 4 **function** **get** PropertyIdentifier GetterSignature
389 4 **function** **set** PropertyIdentifier SetterSignature

FunctionDefinition$^{class,\ \omega}$
390 4 **function** Identifier [Identifier == outer classname] ConstructorSignature Block^{local}
391 4 **function** PropertyIdentifier FunctionSignature FunctionBody$^{allowIn,\ \omega}$
392 4 **function** **get** PropertyIdentifier FunctionSignature FunctionBody$^{allowIn,\ \omega}$
393 4 **function** **set** PropertyIdentifier FunctionSignature FunctionBody$^{allowIn,\ \omega}$

FunctionDefinition$^{local,\ \omega}$
394 4 **const** **function** PropertyIdentifier FunctionSignature FunctionBody$^{allowIn,\ \omega}$
395 4 **function** PropertyIdentifier FunctionSignature FunctionBody$^{allowIn,\ \omega}$

FunctionDefinition$^{\tau,\ \omega}$
396 4 **const** **function** PropertyIdentifier FunctionSignature FunctionBody$^{allowIn,\ \omega}$

397 4 **function** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn, ω}}$
398 4 **function get** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn, ω}}$
399 4 **function set** PropertyIdentifier FunctionSignature FunctionBody$^{\text{allowIn, ω}}$

FunctionSignature
400 3 TypeParameters **(** Parameters **)** ResultTypeOrLike
401 4 TypeParameters **(** **this** **:** TypeName **)** ResultTypeOrLike
402 4 TypeParameters **(** **this** **:** TypeName **,** NonemptyParameters **)** ResultTypeOrLike

GetterSignature
403 4 TypeParameters **(** **)** ResultTypeOrLike

SetterSignature
404 4 TypeParameters **(** Parameter **)** **:** ResultTypeVoid

FunctionBody$^{\alpha, \beta, \omega}$
405 3 Block$^{\text{local}}$
406 4 CommaExpression$^{\alpha, \beta}$ Semicolon$^{\omega}$

TypeParameters
407 3 «empty»
408 4 **.<** TypeParameterList **>**

TypeParameterList
409 4 Identifier
410 4 Identifier **,** TypeParameterList

Parameters
411 3 «empty»
412 RestParameter
413 3 NonemptyParameters
414 4 NonemptyParameters RestParameter

NonemptyParameters
415 3 Parameter **,** NonemptyParameters
416 3 Parameter
417 3 OptionalParameters

OptionalParameters
418 4 OptionalParameter
419 4 OptionalParameter **,** OptionalParameters

OptionalParameter
420 4 Parameter **=** NonAssignmentExpression$^{\text{allowIn}}$

Parameter
421 3 ParameterAttribute TypedPattern$^{\text{allowIn}}$
422 3 ParameterAttribute LikenedPattern$^{\text{allowIn}}$

ParameterAttribute
423 3 «empty»
424 4 **const**

RestParameter
425 4 **...**
426 4 **...** Identifier

ResultTypeOrLike
427 4 ResultType
428 4 **like** TypeExpression

ResultType
429 4 «empty»
430 4 **: void**
431 4 **:** TypeExpression

ResultTypeVoid
432 4 «empty»
433 4 **: void**

ResultTypeBoolean
434 4 «empty»
435 4 **: boolean**

ConstructorSignature
436 4 **(** Parameters **)**
437 4 **(** Parameters **) :** ConstructorInitialiser

ConstructorInitialiser
438 4 SettingList
439 4 SettingList SuperInitialiser
440 4 SuperInitialiser

SettingList
441 4 Setting
442 4 SettingList , Setting

Setting
443 4 Pattern$^{allowIn,\ allowExpr}$ VariableInitialisation^{allowIn}

SuperInitialiser
444 4 **super** Arguments

ClassDeclaration
445 **class** Identifier TypeSignature

ClassDefinition
446 4 **class** Identifier TypeSignature ClassInheritance ClassBody

TypeSignature
447 4 TypeParameters
448 4 TypeParameters **!**

ClassInheritance
449 4 «empty»
450 4 **extends** TypeName
451 4 **implements** TypeNameList
452 4 **extends** TypeName **implements** TypeNameList

TypeNameList
453 4 TypeName
454 4 TypeNameList , TypeName

ClassBody
455 4 Block^{class}

InterfaceDeclaration
456 4 **interface** Identifier TypeSignature

InterfaceDefinition
457 4 **interface** Identifier TypeSignature InterfaceInheritance InterfaceBody

InterfaceInheritance
458 4 «empty»
459 4 **extends** TypeNameList

InterfaceBody
460 4 Block^{interface}

TypeDeclaration
461 4 **type** Identifier TypeSignature

TypeDefinition
462 4 **type** Identifier TypeSignature TypeInitialisation

TypeInitialisation
463 4 **=** TypeExpression

NamespaceDefinition
464 4 **namespace** Identifier NamespaceInitialisation

NamespaceInitialisation
465 4 «empty»
466 4 **=** NamespaceName

PRAGMAS

Pragma^{s}
467 4 UsePragma^{s} Semicolon^{full}

UsePragma^{s}
468 4 **use** PragmaItems^{s}

PragmaItems^{s}

469 4 PragmaItem$^{\gamma}$

470 4 PragmaItems$^{\gamma}$, PragmaItem$^{\gamma}$

PragmaItem^{local}

471 4 **namespace** NamespaceName

472 4 **standard**

473 4 **strict**

PragmaItem$^{\gamma}$

474 4 **default namespace** NamespaceName

475 4 **namespace** NamespaceName

476 4 **standard**

477 4 **strict**

BLOCKS AND PROGRAMS

Block$^{\gamma}$

478 3 **{** Directives$^{\gamma}$ **}**

Program

479 3 Directives^{global}

Revision History:

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

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)

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