Introduction

# Grammar

Grammar is mostly modelled after Visual Basic. See http://msdn.microsoft.com/en-us/library/aa711636%28v=vs.71%29.aspx (Visual Basic .Net Grammar Summary)

## Expressions

**Expression** ::=

SimpleExpression |

TypeExpression |

MemberAccessExpression |

DictionaryAccessExpression |

IndexExpression |

NewExpression |

CastExpression |

OperatorExpression

**ConstantExpression** ::= Expression

**SimpleExpression** ::=

LiteralExpression |

ParenthesizedExpression |

InstanceExpression |

SimpleNameExpression |

AddressOfExpression

**LiteralExpression** ::= Literal

**ParenthesizedExpression** ::= ( Expression )

**InstanceExpression** ::= Me | MyClass | MyBase

**SimpleNameExpression** ::= Identifier

**AddressOfExpression** ::= AddressOf Expression

**MemberAccessExpression** ::=

[ [ MemberAccessBase ] . ] IdentifierOrKeyword

MemberAccessBase ::= Expression | BuiltInTypeName

**DictionaryAccessExpression** ::= [ Expression ] ! IdentifierOrKeyword

**InvocationExpression** ::= Expression [ ( [ ArgumentList ] ) ]

ArgumentList ::=

PositionalArgumentList , NamedArgumentList |

PositionalArgumentList |

NamedArgumentList

**PositionalArgumentList** ::=

Expression |

PositionalArgumentList , [ Expression ]

**NamedArgumentList** ::=

IdentifierOrKeyword : = Expression |

NamedArgumentList , IdentifierOrKeyword : = Expression

**IndexExpression** ::= Expression ( [ ArgumentList ] )

**NewExpression** ::=

ObjectCreationExpression |

ArrayCreationExpression |

DelegateCreationExpression

**ObjectCreationExpression** ::= New TypeName [ ( [ ArgumentList ] ) ]

**ArrayCreationExpression** ::=

New TypeName [ ArraySizeInitializationModifier ]

ArrayElementInitializer

**DelegateCreationExpression** ::= New TypeName ( Expression )

**OperatorExpression** ::=

ArithmeticOperatorExpression |

RelationalOperatorExpression |

LikeOperatorExpression |

ConcatenationOperatorExpression |

ShortCircuitLogicalOperatorExpression |

LogicalOperatorExpression |

ShiftOperatorExpression

**ArithmeticOperatorExpression** ::=

UnaryPlusExpression |

UnaryMinusExpression |

AdditionOperatorExpression |

SubtractionOperatorExpression |

MultiplicationOperatorExpression |

DivisionOperatorExpression |

ModuloOperatorExpression |

ExponentOperatorExpression

UnaryPlusExpression ::= + Expression

UnaryMinusExpression ::= - Expression

AdditionOperatorExpression ::= Expression + Expression

SubtractionOperatorExpression ::= Expression - Expression

MultiplicationOperatorExpression ::= Expression \* Expression

DivisionOperatorExpression ::= FPDivisionOperatorExpression | IntegerDivisionOperatorExpression

FPDivisionOperatorExpression ::= Expression / Expression

IntegerDivisionOperatorExpression ::= Expression \ Expression

ModuloOperatorExpression ::= Expression Mod Expression

ExponentOperatorExpression ::= Expression ^ Expression

RelationalOperatorExpression ::=

Expression = Expression |

Expression < > Expression |

Expression < Expression |

Expression > Expression |

Expression < = Expression |

Expression > = Expression

LikeOperatorExpression ::= Expression Like Expression

ConcatenationOperatorExpression ::= Expression & Expression

LogicalOperatorExpression ::=

Not Expression |

Expression And Expression |

Expression Or Expression |

Expression Xor Expression

ShortCircuitLogicalOperatorExpression ::=

Expression AndAlso Expression |

Expression OrElse Expression

ShiftOperatorExpression ::=

Expression << Expression |

Expression >> Expression