**Symbol and Operator Reference (F#)**

[**Table of Symbols and Operators**](javascript:void(0))

The following table describes symbols used in the F# language, provides links to topics that provide more information, and provides a brief description of some of the uses of the symbol. Symbols are ordered according to the ASCII character set ordering.

|  |  |  |
| --- | --- | --- |
| **Symbol or operator** | **Links** | **Description** |
| **!** | [Reference Cells (F#)](http://msdn.microsoft.com/en-us/library/dd233186.aspx)  [Computation Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233182.aspx) | * Dereferences a reference cell. * After a keyword, indicates a modified version of the keyword's behavior as controlled by a workflow. |
| **!=** | Not applicable. | * Not used in F#. Use **<>** for inequality operations. |
| **"** | [Literals (F#)](http://msdn.microsoft.com/en-us/library/dd233193.aspx)  [Strings (F#)](http://msdn.microsoft.com/en-us/library/dd323829.aspx) | * Delimits a text string. |
| **"""** | [Strings (F#)](http://msdn.microsoft.com/en-us/library/dd323829.aspx) | Delimits a verbatim text string. Differs from **@"..."** in that a you can indicate a quotation mark character by using a single quote in the string. |
| **#** | [Compiler Directives (F#)](http://msdn.microsoft.com/en-us/library/dd233195.aspx)  [Flexible Types (F#)](http://msdn.microsoft.com/en-us/library/dd233198.aspx) | * Prefixes a preprocessor or compiler directive, such as **#light**. * When used with a type, indicates a *flexible type*, which refers to a type or any one of its derived types. |
| **$** | No more information available. | * Used internally for certain compiler-generated variable and function names. |
| **%** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx)  [Code Quotations (F#)](http://msdn.microsoft.com/en-us/library/dd233212.aspx) | * Computes the integer modulus. Used for splicing quotations. |
| **%?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | Computes the integer modulus, when the right side is a nullable type. |
| **&** | [Match Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233242.aspx) | * Computes the address of a mutable value, for use when interoperating with other languages. * Used in AND patterns. |
| **&&** | [Boolean Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469492.aspx) | * Computes the Boolean AND operation. |
| **&&&** | [Bitwise Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469495.aspx) | * Computes the bitwise AND operation. |
| **'** | [Literals (F#)](http://msdn.microsoft.com/en-us/library/dd233193.aspx)  [Automatic Generalization (F#)](http://msdn.microsoft.com/en-us/library/dd233183.aspx) | * Delimits a single-character literal. * Indicates a generic type parameter. |
| **``...``** | No more information available. | * Delimits an identifier that would otherwise not be a legal identifier, such as a language keyword. |
| **( )** | [Unit Type (F#)](http://msdn.microsoft.com/en-us/library/dd483472.aspx) | * Represents the single value of the unit type. |
| **(...)** | [Tuples (F#)](http://msdn.microsoft.com/en-us/library/dd233200.aspx)  [Operator Overloading (F#)](http://msdn.microsoft.com/en-us/library/dd233204.aspx) | * Indicates the order in which expressions are evaluated. * Delimits a tuple. * Used in operator definitions. |
| **(\*...\*)** |  | * Delimits a comment that could span multiple lines. |
| **(|...|)** | [Active Patterns (F#)](http://msdn.microsoft.com/en-us/library/dd233248.aspx) | * Delimits an active pattern. Also called *banana clips*. |
| **\*** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx)  [Tuples (F#)](http://msdn.microsoft.com/en-us/library/dd233200.aspx)  [Units of Measure (F#)](http://msdn.microsoft.com/en-us/library/dd233243.aspx) | * When used as a binary operator, multiplies the left and right sides. * In types, indicates pairing in a tuple. * Used in units of measure types. |
| **\*?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Multiplies the left and right sides, when the right side is a nullable type. |
| **\*\*** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Computes the exponentiation operation (x \*\* y means x to the power of y). |
| **+** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * When used as a binary operator, adds the left and right sides. * When used as a unary operator, indicates a positive quantity. (Formally, it produces the same value with the sign unchanged.) |
| **+?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Adds the left and right sides, when the right side is a nullable type. |
| **,** | [Tuples (F#)](http://msdn.microsoft.com/en-us/library/dd233200.aspx) | * Separates the elements of a tuple, or type parameters. |
| **-** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * When used as a binary operator, subtracts the right side from the left side. * When used as a unary operator, performs a negation operation. |
| **-** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Subtracts the right side from the left side, when the right side is a nullable type. |
| **->** | [Functions (F#)](http://msdn.microsoft.com/en-us/library/dd233229.aspx)  [Match Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233242.aspx) | * In function types, delimits arguments and return values. * Yields an expression (in sequence expressions); equivalent to the **yield** keyword. * Used in match expressions |
| **.** | [Members (F#)](http://msdn.microsoft.com/en-us/library/dd233244.aspx)  [Primitive Types (F#)](http://msdn.microsoft.com/en-us/library/dd233210.aspx) | * Accesses a member, and separates individual names in a fully qualified name. * Specifies a decimal point in floating point numbers. |
| **..** | [Loops: for...in Expression (F#)](http://msdn.microsoft.com/en-us/library/dd233227.aspx) | * Specifies a range. |
| **.. ..** | [Loops: for...in Expression (F#)](http://msdn.microsoft.com/en-us/library/dd233227.aspx) | * Specifies a range together with an increment. |
| **.[...]** | [Arrays (F#)](http://msdn.microsoft.com/en-us/library/dd233214.aspx) | * Accesses an array element. |
| **/** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx)  [Units of Measure (F#)](http://msdn.microsoft.com/en-us/library/dd233243.aspx) | * Divides the left side (numerator) by the right side (denominator). * Used in units of measure types. |
| **/?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Divides the left side by the right side, when the right side is a nullable type. |
| **//** |  | * Indicates the beginning of a single-line comment. |
| **///** | [XML Documentation (F#)](http://msdn.microsoft.com/en-us/library/dd233217.aspx) | * Indicates an XML comment. |
| **:** | [Functions (F#)](http://msdn.microsoft.com/en-us/library/dd233229.aspx) | * In a type annotation, separates a parameter or member name from its type. |
| **::** | [Lists (F#)](http://msdn.microsoft.com/en-us/library/dd233224.aspx)  [Match Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233242.aspx) | * Creates a list. The element on the left side is appended to the list on the right side. * Used in pattern matching to separate the parts of a list. |
| **:=** | [Reference Cells (F#)](http://msdn.microsoft.com/en-us/library/dd233186.aspx) | * Assigns a value to a reference cell. |
| **:>** | [Casting and Conversions (F#)](http://msdn.microsoft.com/en-us/library/dd233220.aspx) | * Converts a type to type that is higher in the hierarchy. |
| **:?** | [Match Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233242.aspx) | * Returns **true** if the value matches the specified type; otherwise, returns **false**(type test operator). |
| **:?>** | [Casting and Conversions (F#)](http://msdn.microsoft.com/en-us/library/dd233220.aspx) | * Converts a type to a type that is lower in the hierarchy. |
| **;** | [Verbose Syntax (F#)](http://msdn.microsoft.com/en-us/library/dd233199.aspx)  [Lists (F#)](http://msdn.microsoft.com/en-us/library/dd233224.aspx)  [Records (F#)](http://msdn.microsoft.com/en-us/library/dd233184.aspx) | * Separates expressions (used mostly in verbose syntax). * Separates elements of a list. * Separates fields of a record. |
| **<** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Computes the less-than operation. |
| **<?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | Computes the less than operation, when the right side is a nullable type. |
| **<<** | [Functions (F#)](http://msdn.microsoft.com/en-us/library/dd233229.aspx) | * Composes two functions in reverse order; the second one is executed first (backward composition operator). |
| **<<<** | [Bitwise Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469495.aspx) | * Shifts bits in the quantity on the left side to the left by the number of bits specified on the right side. |
| **<-** | [Values (F#)](http://msdn.microsoft.com/en-us/library/dd233185.aspx) | * Assigns a value to a variable. |
| **<...>** | [Automatic Generalization (F#)](http://msdn.microsoft.com/en-us/library/dd233183.aspx) | * Delimits type parameters. |
| **<>** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Returns **true** if the left side is not equal to the right side; otherwise, returns false. |
| **<>?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Computes the "not equal" operation when the right side is a nullable type. |
| **<=** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Returns **true** if the left side is less than or equal to the right side; otherwise, returns false. |
| **<=?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Computes the "less than or equal to" operation when the right side is a nullable type. |
| **<|** | [Functions (F#)](http://msdn.microsoft.com/en-us/library/dd233229.aspx) | * Passes the result of the expression on the right side to the function on left side (backward pipe operator). |
| **<||** | [Operators.( <|| )<'T1,'T2,'U> Function (F#)](http://msdn.microsoft.com/en-us/library/ee370400.aspx) | * Passes the tuple of two arguments on the right side to the function on left side. |
| **<|||** | [Operators.( <||| )<'T1,'T2,'T3,'U> Function (F#)](http://msdn.microsoft.com/en-us/library/ee370477.aspx) | * Passes the tuple of three arguments on the right side to the function on left side. |
| **<@...@>** | [Code Quotations (F#)](http://msdn.microsoft.com/en-us/library/dd233212.aspx) | * Delimits a typed code quotation. |
| **<@@...@@>** | [Code Quotations (F#)](http://msdn.microsoft.com/en-us/library/dd233212.aspx) | * Delimits an untyped code quotation. |
| **=** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Returns **true** if the left side is equal to the right side; otherwise, returns false. |
| **=?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Computes the "equal" operation when the right side is a nullable type. |
| **==** | Not applicable. | * Not used in F#. Use **=** for equality operations. |
| **>** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Returns **true** if the left side is greater than the right side; otherwise, returns false. |
| **>?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Computes the "greather than" operation when the right side is a nullable type. |
| **>>** | [Functions (F#)](http://msdn.microsoft.com/en-us/library/dd233229.aspx) | * Composes two functions (forward composition operator). |
| **>>>** | [Bitwise Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469495.aspx) | * Shifts bits in the quantity on the left side to the right by the number of places specified on the right side. |
| **>=** | [Arithmetic Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469493.aspx) | * Returns **true** if the right side is greater than or equal to the left side; otherwise, returns false. |
| **>=?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Computes the "greater than or equal" operation when the right side is a nullable type. |
| **?** | [Parameters and Arguments (F#)](http://msdn.microsoft.com/en-us/library/dd233213.aspx) | * Specifies an optional argument. * Used as an operator for dynamic method and property calls. You must provide your own implementation. |
| **? ... <- ...** | No more information available. | * Used as an operator for setting dynamic properties. You must provide your own implementation. |
| **?>=** , **?>**, **?<=**, **?<**, **?=**, **?<>**, **?+**, **?-**, **?\***, **?/** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Equivalent to the corresponding operators without the ? prefix, where a nullable type is on the left. |
| **>=?** , **>?**, **<=?**, **<?**, **=?**, **<>?**,**+?**, **-?**, **\*?**, **/?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Equivalent to the corresponding operators without the ? suffix, where a nullable type is on the right. |
| **?>=?** , **?>?**, **?<=?**, **?<?**, **?=?**, **?<>?**, **?+?**, **?-?**, **?\*?**, **?/?** | [Nullable Operators (F#)](http://msdn.microsoft.com/en-us/library/hh370984.aspx) | 1. Equivalent to the corresponding operators without the surrounding question marks, where both sides are nullable types. |
| **@** | [Lists (F#)](http://msdn.microsoft.com/en-us/library/dd233224.aspx)  [Strings (F#)](http://msdn.microsoft.com/en-us/library/dd323829.aspx) | * Concatenates two lists. * When placed before a string literal, indicates that the string is to be interpreted verbatim, with no interpretation of escape characters. |
| **[...]** | [Lists (F#)](http://msdn.microsoft.com/en-us/library/dd233224.aspx) | * Delimits the elements of a list. |
| **[|...|]** | [Arrays (F#)](http://msdn.microsoft.com/en-us/library/dd233214.aspx) | * Delimits the elements of an array. |
| **[<...>]** | [Attributes (F#)](http://msdn.microsoft.com/en-us/library/dd233179.aspx) | * Delimits an attribute. |
| **\** | [Strings (F#)](http://msdn.microsoft.com/en-us/library/dd323829.aspx) | * Escapes the next character; used in character and string literals. |
| **^** | [Statically Resolved Type Parameters (F#)](http://msdn.microsoft.com/en-us/library/dd548046.aspx)  [Strings (F#)](http://msdn.microsoft.com/en-us/library/dd323829.aspx) | * Specifies type parameters that must be resolved at compile time, not at runtime. * Concatenates strings. |
| **^^^** | [Bitwise Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469495.aspx) | * Computes the bitwise exclusive OR operation. |
| **\_** | [Match Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233242.aspx)  [Generics (F#)](http://msdn.microsoft.com/en-us/library/dd233215.aspx) | * Indicates a wildcard pattern. * Specifies an anonymous generic parameter. |
| **`** | [Automatic Generalization (F#)](http://msdn.microsoft.com/en-us/library/dd233183.aspx) | * Used internally to indicate a generic type parameter. |
| **{...}** | [Sequences (F#)](http://msdn.microsoft.com/en-us/library/dd233209.aspx)  [Records (F#)](http://msdn.microsoft.com/en-us/library/dd233184.aspx) | * Delimits sequence expressions and computation expressions. * Used in record definitions. |
| **|** | [Match Expressions (F#)](http://msdn.microsoft.com/en-us/library/dd233242.aspx) | * Delimits individual match cases, individual discriminated union cases, and enumeration values. |
| **||** | [Boolean Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469492.aspx) | * Computes the Boolean OR operation. |
| **|||** | [Bitwise Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469495.aspx) | * Computes the bitwise OR operation. |
| **|>** | [Functions (F#)](http://msdn.microsoft.com/en-us/library/dd233229.aspx) | * Passes the result of the left side to the function on the right side (forward pipe operator). |
| **||>** | [Operators.( ||> )<'T1,'T2,'U> Function (F#)](http://msdn.microsoft.com/en-us/library/ee340237.aspx) | * Passes the tuple of two arguments on the left side to the function on the right side. |
| **|||>** | [Operators.( |||> )<'T1,'T2,'T3,'U> Function (F#)](http://msdn.microsoft.com/en-us/library/ee370269.aspx) | 1. Passes the tuple of three arguments on the left side to the function on the right side. |
| **~~** | [Operator Overloading (F#)](http://msdn.microsoft.com/en-us/library/dd233204.aspx) | * Used to declare an overload for the unary negation operator. |
| **~~~** | [Bitwise Operators (F#)](http://msdn.microsoft.com/en-us/library/dd469495.aspx) | * Computes the bitwise NOT operation. |
| **~-** | [Operator Overloading (F#)](http://msdn.microsoft.com/en-us/library/dd233204.aspx) | * Used to declare an overload for the unary minus operator. |
| **~+** | [Operator Overloading (F#)](http://msdn.microsoft.com/en-us/library/dd233204.aspx) | * Used to declare an overload for the unary plus operator. |

[**Operator Precedence**](javascript:void(0))

The following table shows the order of precedence of operators and other expression keywords in the F# language, in order from lowest precedence to the highest precedence. Also listed is the associativity, if applicable.

|  |  |
| --- | --- |
| **Operator** | **Associativity** |
| **as** | Right |
| **when** | Right |
| **|** (pipe) | Left |
| **;** | Right |
| **let** | Nonassociative |
| **function** , **fun**, **match**, **try** | Nonassociative |
| **if** | Nonassociative |
| **->** | Right |
| **:=** | Right |
| **,** | Nonassociative |
| **or** , **||** | Left |
| **&** , **&&** | Left |
| **<** *op*, **>***op*, **=**, **|***op*, **&***op* | Left |
| **&&&** , **|||**, **^^^**, **~~~**, **<<<**, **>>>** | Left |
| **^** *op* | Right |
| **::** | Right |
| **:?>** , **:?** | Nonassociative |
| **-** *op*, **+***op*, (binary) | Left |
| **\*** *op*, **/***op*, **%***op* | Left |
| **\*\*** *op* | Right |
| **f x** (function application) | Left |
| **|** (pattern match) | Right |
| prefix operators (+*op*, -*op*, %, %%, &, &&, !*op*, ~*op*) | Left |
| **.** | Left |
| **f(x)** | Left |
| **f<** *types* **>** | Left |

F# supports custom operator overloading. This means that you can define your own operators. In the previous table, *op* can be any valid (possibly empty) sequence of operator characters, either built-in or user-defined. Thus, you can use this table to determine what sequence of characters to use for a custom operator to achieve the desired level of precedence. Leading **.** characters are ignored when the compiler determines precedence.