Learning to program with F#

Jon Sporring

September 8, 2016

Part V

Appendix

Appendix E

Language Details

This appendix lists various language details.

E.1 Arithmetic operators on basic types

Operator	leftOp	rightOp	Expression	Result	Description
leftOp + rightOp	ints	ints	5 + 2	7	Addition
	floats	floats	5.0 + 2.0	7.0	
	chars	chars	'a' + 'b'	'\195'	Addition of codes
	strings	strings	"ab" + "cd"	"abcd"	Concatenation
leftOp - rightOp	ints	ints	5 - 2	3	Subtraction
	floats	floats	5.0 - 2.0	3.0	
leftOp * rightOp	ints	ints	5 * 2	10	Multiplication
	floats	floats	5.0 * 2.0	10.0	
leftOp / rightOp	ints	ints	5 / 2	2	Integer division
	floats	floats	5.0 / 2.0	2.5	Division
leftOp % rightOp	ints	ints	5 % 2	1	Remainder
	floats	floats	5.0 % 2.0	1.0	
leftOp ** rightOp	floats	floats	5.0 ** 2.0	25.0	Exponentiation
leftOp && rightOp	bool	bool	true && false	false	boolean and
leftOp || rightOp	bool	bool	true || false	false	boolean or
leftOp &&& rightOp	ints	ints	0b1010 &&& 0b1100	0b1000	bitwise bool and
leftOp ||| rightOp	ints	ints	0b1010 ||| 0b1100	0b1110	bitwise boolean or
leftOp ^^^ rightOp	ints	ints	0b1010 ^^^ 0b1101	0b0111	bitwise boolean exclusive or
leftOp <<< rightOp	ints	ints	0b00001100uy <<< 2	0b00110000uy	bitwise shift left
leftOp >>> rightOp	ints	ints	0b00001100uy >>> 2	0b00000011uy	bitwise and
+op	ints		+3	3	identity
	floats		+3.0	3.0	
-op	ints		-3	-3	negation
	floats		-3.0	-3.0	
not op	bool		not true	false	boolean negation
~~op	ints		~~~0b00001100uy	0b11110011uy	bitwise boolean negation

Table E.1: Arithmetic operators on basic types. Ints, floats, chars, and strings means all built-in integer types etc.. Note that for the bitwise operations, digits 0 and 1 are taken to be true and false.

Operator	`leftOp`	`rightOp`	Expression	Result	Description
`leftOp < rightOp`	bool	bool	`true < false`	`false`	Less than
	ints	ints	`5 < 2`	`false`	
	floats	floats	`5.0 < 2.0`	`false`	
	chars	chars	`'a' < 'b'`	`true`	
	strings	strings	`"ab" < "cd"`	`true`	
`leftOp > rightOp`	bool	bool	`true > false`	`true`	Greater than
	ints	ints	`5 > 2`	`true`	
	floats	floats	`5.0 > 2.0`	`true`	
	chars	chars	`'a' > 'b'`	`false`	
	strings	strings	`"ab" > "cd"`	`false`	
`leftOp = rightOp`	bool	bool	`true = false`	`false`	Equal
	ints	ints	`5 = 2`	`false`	
	floats	floats	`5.0 = 2.0`	`false`	
	chars	chars	`'a' = 'b'`	`false`	
	strings	strings	`"ab" = "cd"`	`false`	
`leftOp <= rightOp`	bool	bool	`true <= false`	`false`	Less than or equal
	ints	ints	`5 <= 2`	`false`	
	floats	floats	`5.0 <= 2.0`	`false`	
	chars	chars	`'a' <= 'b'`	`true`	
	strings	strings	`"ab" <= "cd"`	`true`	
`leftOp >= rightOp`	bool	bool	`true >= false`	`true`	Greater than or equal
	ints	ints	`5 >= 2`	`true`	
	floats	floats	`5.0 >= 2.0`	`true`	
	chars	chars	`'a' >= 'b'`	`false`	
	strings	strings	`"ab" >= "cd"`	`false`	
`leftOp <> rightOp`	bool	bool	`true <> false`	`true`	Not Equal
	ints	ints	`5 <> 2`	`true`	
	floats	floats	`5.0 <> 2.0`	`true`	
	chars	chars	`'a' <> 'b'`	`true`	
	strings	strings	`"ab" <> "cd"`	`true`	

Table E.2: Comparison operators on basic types. Ints, floats, chars, and strings means all built-in integer types etc..

E.2 Basic arithmetic functions

Type	Function name	Example	Result	Description
Ints and floats	`abs`	`abs -3`	`3`	Absolute value
Floats	`acos`	`acos 0.8`	`0.6435011088`	Inverse cosine
Floats	`asin`	`asin 0.8`	`0.927295218`	Inverse sinus
Floats	`atan`	`atan 0.8`	`0.6747409422`	Inverse tangent
Floats	`atan2`	`atan2 0.8 2.3`	`0.3347368373`	Inverse tangentvariant
Floats	`ceil`	`ceil 0.8`	`1.0`	Ceiling
Floats	`cos`	`cos 0.8`	`0.6967067093`	Cosine
Floats	`cosh`	`cosh 0.8`	`1.337434946`	Hyperbolic cosine
Floats	`exp`	`exp 0.8`	`2.225540928`	Natural exponent
Floats	`floor`	`floor 0.8`	`0.0`	Floor
Floats	`log`	`log 0.8`	`-0.2231435513`	Natural logarithm
Floats	`log10`	`log10 0.8`	`-0.09691001301`	Base-10 logarithm
Ints, floats, chars, and strings	`max`	`max 3.0 4.0`	`4.0`	Maximum
Ints, floats, chars, and strings	`min`	`min 3.0 4.0`	`3.0`	Minimum
Ints	`pown`	`pown 3 2`	`9`	Integer exponent
Floats	`round`	`round 0.8`	`1.0`	Rounding
Ints and floats	`sign`	`sign -3`	`-1`	Sign
Floats	`sin`	`sin 0.8`	`0.7173560909`	Sinus
Floats	`sinh`	`sinh 0.8`	`0.8881059822`	Hyperbolic sinus
Floats	`sqrt`	`sqrt 0.8`	`0.894427191`	Square root
Floats	`tan`	`tan 0.8`	`1.029638557`	Tangent
Floats	`tanh`	`tanh 0.8`	`0.6640367703`	Hyperbolic tangent

Table E.3: Predefined functions for arithmetic operations

Name	Example	Description
`fst`	`fst (1, 2)`	
`snd`	`snd (1, 2)`	
`failwith`	`failwith`	
`invalidArg`	`invalidArg`	
`raise`	`raise`	
`reraise`	`reraise`	
`ref`	`ref`	
`ceil`	`ceil`	

Table E.4: Built-in functions.

E.3 Precedence and associativity

Operator	Associativity	Description
`+op`, `-op`, `~~~op`	Left	Unary identity, negation, and bitwise negation operator
`f x`	Left	Function application
`leftOp ** rightOp`	Right	Exponent
`leftOp * rightOp`, `leftOp / rightOp`, `leftOp % rightOp`	Left	Multiplication, division and remainder
`leftOp + rightOp`, `leftOp - rightOp`	Left	Addition and subtraction binary operators
`leftOp ^^^ rightOp`	Right	bitwise exclusive or
`leftOp < rightOp`, `leftOp <= rightOp`, `leftOp > rightOp`, `leftOp >= rightOp`, `leftOp = rightOp`, `leftOp <> rightOp`, `leftOp <<< rightOp`, `leftOp >>> rightOp`, `leftOp &&& rightOp`, `leftOp ||| rightOp`,	Left	Comparison operators, bitwise shift, and bitwise 'and' and 'or'.
`&&`	Left	Boolean and
`||`	Left	Boolean or

Table E.5: Some common operators, their precedence, and their associativity. Rows are ordered from highest to lowest precedences, such that `leftOp * rightOp` has higher precedence than `leftOp + rightOp`. Operators in the same row has same precedence. Full table is given in Table E.6.

· boolean or
· boolean and

Operator	Associativity	Description
`ident "<"types ">"`	Left	High-precedence type application
`ident "("expr ")"`	Left	High-predence application
`"."`	Left	
`prefixOp`	Left	All prefix operators
`""` rule|	Left	Pattern matching rule
`ident expr`, `"lazy'' expr`, `"assert'' epxr`	Left	
`"**"opChar`	Right	Exponent like
`"*"opChar`, `"/"opChar`, `"%"opChar`	Left	Infix multiplication like
`"−"opChar`, `"+"opChar`	Left	Infix addition like
`":?''`	None	
`"::''`	Right	
`"^'' opChar`	Right	
`"!="opChar`, `"<"opChar`, `">"opChar`, `"="`, `"|"opChar`, `"&"opChar`, `"$"opChar`	Left	Infix addition like
`":>"`, `":?>"`	Right	
`"&"`, `"&&"`	Left	Boolean and like
`"or"`, `"||"`	Left	Boolean or like
`","`	None	
`":="`	Right	
`"−>"`	Right	
`"if"`	None	
`"function"`, `"fun"`, `"match"`, `"try"`	None	
`"let"`	None	
`";"`	Right	
`"|"`	Left	
`"when"`	Right	
`"as"`	Right	

Table E.6: Precedence and associativity of operators. Operators in the same row has same precedence. See Listing 6.1 for the definition of `prefixOp`

E.4 Lightweight Syntax

To appear later.[1]

[1] Todo: **See Lightweight Syntax, Spec-4.0 Chapter 15.1**