The "Unknown:"s below indicate that an entry is incomplete.

- either the entry exist in the language, and please tell.
- either the entry doesn't exist in the language, and please tell so. The entry will be marked as such and won't appear as missing anymore.

- Category: Object Oriented, Dynamically typed

- Various

`nothing needed`	breaking lines (useful when end-of-line and/or indentation has a special meaning)
`/* ... */`	commenting (nestable)
`%`	commenting (until end of line)
`< > =< >=`	comparison
`Min / Max`	comparison (min / max (binary or more))
`== \=`	equality / inequality (deep)
`System.gcDo`	force garbage collection
`( ... )`	grouping expressions
`Compiler.evalExpression or Compiler.parseOzVirtualString`	runtime evaluation
`=`	variable assignment or declaration (assignment)
`local V1 = e V2 = e2 in ... end`	variable assignment or declaration (declaration)

Unknown:

documentation comment
information about the current line and file
tokens (case-sensitivity (keywords, variable identifiers...))
tokens (what is the standard way for scrunching together multiple words)
tokens (variable identifier regexp)
comparison (returns 3 values (i.e. inferior, equal or superior))

- Functions

`fun {$ A B} ... end(1)`	anonymous function
`{f a b}`	function call
`{f}`	function call (with no parameter)
`fun { F Para1 Para2 } ... end`	function definition

`proc { F Para1 Para2 } ... end`	function definition (procedures)
`no syntax needed`(2)	function return value (function body is the result)

Unknown:

runtime inspecting the caller information

- Control Flow

`try a catch exn then ... end`	exception (catching)
`raise ... end`	exception (throwing)
`if c then ... end`	if_then
`if c then b1 elseif c2 then b2 else b3 end`	if_then_else

Unknown:

multiple selection (switch)
loop (forever loop)
loop (while condition do something)
loop (do something until condition)
loop (for each value in a numeric range, 1 increment (see also the entries about ranges))
loop (for each value in a numeric range, 1 decrement)
loop (for each value in a numeric range, free increment)
loop (for "a la C" (while + initialisation))

- Object Oriented & Reflexivity

Unknown:

method invocation
method invocation (with no parameter)
object creation
object cloning
manually call an object's destructor
class declaration
testing class membership
get the type/class corresponding to an object/instance/value
methods available
inheritance
has the method
current instance
accessing parent method

- Package, Module

Unknown:

package scope
declare
import

- Strings

`&z`	character "z"
`== \=`	string equality & inequality
`Length`	string size
`"..."`	strings (with no interpolation of variables)
`ToUpper / ToLower`	upper / lower case character

Unknown:

strings (with interpolation of variables)
strings (end-of-line (without writing the real CR or LF character))
multi-line
convert something to a string (see also string interpolation)
serialize (marshalling)
unserialize (un-marshalling)
sprintf-like
simple print
string concatenation
duplicate n times
uppercase / lowercase / capitalized string
ascii to character
character to ascii
accessing n-th character
extract a substring
locate a substring
locate a substring (starting at the end)

- Booleans

`false`	false value
`Not`	logical not
`Or / And(3)`	logical or / and (non short circuit (always evaluates both arguments))
`orelse / andthen`	logical or / and (short circuit)
`true`	true value

- Bags and Lists

`|`	adding an element at the beginning (list cons) (return the new list (no side-effect))
`FoldL`	f(... f(f(init, e1), e2) ..., en)
`FoldR`	f(e1, f(e2, ... f(en, init) ...))
`ForAll`	for each element do something

`Member`	is an element in the list
`Some`	is the predicate true for an element
`All`	is the predicate true for every element
`forAllInd`	iterate with index
`Filter`	keep elements (matching)
`Last`	last element
`Append`	list concatenation
`[ a b c ]`	list constructor
`Flatten`	list flattening (one level depth)
`Length`	list size
`Nth`	list/array indexing
`Partition`	partition a list: elements matching, elements non matching
`Reverse`	reverse
`Sort`	sort
`Map`	transform a list (or bag) in another one
`Zip`	transform two lists in parallel

Unknown:

adding an element at index
adding an element at the end
first element
all but the first element
get the first element and remove it
get the last element and remove it
smallest / biggest element
join a list of strings in a string using a glue string
remove duplicates

- Various Data Types

`List.number A B Step`	range (inclusive .. inclusive)
`.`	record (selector)
`Assign`	reference (pointer) (assigning (when dereferencing doesn't give a lvalue))
`NewCell`	reference (pointer) (creation)

`Access`	reference (pointer) (dereference)

Unknown:

tuple constructor
computable tuple (these are a kind of immutable lists playing a special role in parameter passing) (empty tuple)
computable tuple (these are a kind of immutable lists playing a special role in parameter passing) (1-uple)
computable tuple (these are a kind of immutable lists playing a special role in parameter passing) (using a tuple for a function call)
optional value (null value)
optional value (value)
optional value (null coalescing)
dictionary (constructor)
dictionary (access)
dictionary (has the key ?)
dictionary (remove by key)
dictionary (list of keys)
dictionary (list of values)
dictionary (merge)

- Mathematics

`Pow`	exponentiation (power)
`Log`	logarithm (base e)
`mod`	modulo (modulo of -3 / 2 is -1)
`~`	negation
`1000., 1.E3`	numbers syntax (floating point)
`0b1, 07, 0xf`(4)	numbers syntax (integers in base 2, octal and hexadecimal)
`1000`	numbers syntax (integers)
`Sqrt / Exp / Abs`	square root / e-exponential / absolute value
`Sin / Cos / Tan`	trigonometry (basic)
`Asin / Acos / Atan`	trigonometry (inverse)
`/ Round / Floor / Ceil`	truncate / round / floor / ceil

Unknown:

addition / subtraction / multiplication / division
random (random number)
random (seed the pseudo random generator)
operator priorities and associativities
logarithm (base 10)
bitwise operators

Remarks

- (1) also works for procedures: proc {$ A B} ... end
- (2) in Matlab, only for anonymous function
- (3) in Oz, simple functions, not operators
- (4) 0b1 new in PHP 5.4

Pixel

Generated from syntax-across-languages.html.pl
$Id: syntax-across-languages.html.pl 408 2008-08-29 08:32:23Z pixel $