

Predefined action! values

Bitwise actions

and~

USAGE:

AND~ value1 value2

DESCRIPTION:

Returns the first value ANDed with the second.
AND~ is an action! value.

ARGUMENTS:

value1 [logic! integer! char! bitset! binary! typeset! pair! tuple!
vector!]
value2 [logic! integer! char! bitset! binary! typeset! pair! tuple!
vector!]

RETURNS:

[logic! integer! char! bitset! binary! typeset! pair! tuple! vector!]

complement

USAGE:

COMPLEMENT value

DESCRIPTION:

Returns the opposite (complementing) value of the input value.
COMPLEMENT is an action! value.

ARGUMENTS:

value [logic! integer! tuple! bitset! typeset! binary!]

RETURNS:

[logic! integer! tuple! bitset! typeset! binary!]

or~

USAGE:

OR~ value1 value2

DESCRIPTION:

Returns the first value ORed with the second.

OR~ is an action! value.

ARGUMENTS:

value1 [logic! integer! char! bitset! binary! typeset! pair! tuple!
vector!]

value2 [logic! integer! char! bitset! binary! typeset! pair! tuple!
vector!]

RETURNS:

[logic! integer! char! bitset! binary! typeset! pair! tuple! vector!]

xor~

USAGE:

XOR~ value1 value2

DESCRIPTION:

Returns the first value exclusive ORed with the second.

XOR~ is an action! value.

ARGUMENTS:

value1 [logic! integer! char! bitset! binary! typeset! pair! tuple!
vector!]

value2 [logic! integer! char! bitset! binary! typeset! pair! tuple!
vector!]

RETURNS:

[logic! integer! char! bitset! binary! typeset! pair! tuple! vector!]

I/O actions

close

USAGE:

CLOSE port

DESCRIPTION:

Closes a port.

CLOSE is an action! value.

ARGUMENTS:

port [port!]

create

USAGE:

CREATE port

DESCRIPTION:

Send port a create request.

CREATE is an action! value.

ARGUMENTS:

port [port! file! url! block!]

delete

USAGE:

DELETE file

DESCRIPTION:

Deletes the specified file or empty folder.

DELETE is an action! value.

ARGUMENTS:

file [file! port!]

open

USAGE:

OPEN port

DESCRIPTION:

Opens a port; makes a new port from a specification if necessary.

OPEN is an action! value.

ARGUMENTS:

port [port! file! url! block!]

REFINEMENTS:

/new => Create new file - if it exists, deletes it.

/read => Open for read access.

/write => Open for write access.

/seek => Optimize for random access.

/allow => Specifies right access attributes.

access [block!]

open?

USAGE:

OPEN? port

DESCRIPTION:

Returns TRUE if port is open.

OPEN? is an action! value.

ARGUMENTS:

port [port!]

query

USAGE:

QUERY target

DESCRIPTION:

Returns information about a file.

QUERY is an action! value.

ARGUMENTS:

target [file! port!]

read

USAGE:

READ source

DESCRIPTION:

Reads from a file, URL, or other port.

READ is an action! value.

ARGUMENTS:

source [file! url! port!]

REFINEMENTS:

/part => Partial read a given number of units (source relative).

length [number!]

/seek => Read from a specific position (source relative).

index [number!]

/binary => Preserves contents exactly.

/lines => Convert to block of strings.

/info =>

/as => Read with the specified encoding, default is 'UTF-8.

encoding [word!]

rename

USAGE:

RENAME from to

DESCRIPTION:

Rename a file.

RENAME is an action! value.

ARGUMENTS:

from [port! file! url!]

to [port! file! url!]

update

USAGE:

UPDATE port

DESCRIPTION:

Updates external and internal states (normally after read/write).

UPDATE is an action! value.

ARGUMENTS:

port [port!]

write

USAGE:

WRITE destination data

DESCRIPTION:

Writes to a file, URL, or other port.

WRITE is an action! value.

ARGUMENTS:

destination [file! url! port!]

data [any-type!]

REFINEMENTS:

/binary => Preserves contents exactly.

/lines => Write each value in a block as a separate line.

/info =>

/append => Write data at end of file.

/part => Partial write a given number of units.

length [number!]

/seek => Write at a specific position.

index [number!]

/allow => Specifies protection attributes.

access [block!]

/as => Write with the specified encoding, default is 'UTF-8.

encoding [word!]

General actions

form

USAGE:

FORM value

DESCRIPTION:

Returns a user-friendly string representation of a value.

FORM is an action! value.

ARGUMENTS:

value [any-type!]

REFINEMENTS:

/part => Limit the length of the result.

limit [integer!]

RETURNS:

[string!]

make

USAGE:

MAKE type spec

DESCRIPTION:

Returns a new value made from a spec for that value's type.

MAKE is an action! value.

ARGUMENTS:

type [any-type!] "The datatype, an example or prototype value."

spec [any-type!] "The specification of the new value."

RETURNS:

Returns the specified datatype.

[any-type!]

mold

USAGE:

MOLD value

DESCRIPTION:

Returns a source format string representation of a value.

MOLD is an action! value.

ARGUMENTS:

value [any-type!]

REFINEMENTS:

/only => Exclude outer brackets if value is a block.

/all => TBD: Return value in loadable format.

/flat => TBD: Exclude all indentation.

/part => Limit the length of the result.

limit [integer!]

RETURNS:

[string!]

random

USAGE:

RANDOM value

DESCRIPTION:

Returns a random value of the same datatype; or shuffles series.

RANDOM is an action! value.

ARGUMENTS:

value "Maximum value of result (modified when series)."

REFINEMENTS:

/seed => Restart or randomize.

/secure => Returns a cryptographically secure random number.

/only => Pick a random value from a series.

RETURNS:

[any-type!]

reflect

USAGE:

REFLECT value field

DESCRIPTION:

Returns internal details about a value via reflection.

REFLECT is an action! value.

ARGUMENTS:

value	[any-type!]
field	[word!] {spec, body, words, etc. Each datatype defines its own reflectors.}

to

USAGE:

TO type spec

DESCRIPTION:

Converts to a specified datatype.

TO is an action! value.

ARGUMENTS:

type	[any-type!] "The datatype or example value."
spec	[any-type!] "The attributes of the new value."

Series actions

append

USAGE:

APPEND series value

DESCRIPTION:

Inserts value(s) at series tail; returns series head.

APPEND is an action! value.

ARGUMENTS:

series	[series! bitset! port!]
value	[any-type!]

REFINEMENTS:

/part	=> Limit the number of values inserted.
length	[number! series!]
/only	=> Insert block types as single values (overrides /part).
/dup	=> Duplicate the inserted values.
count	[integer!]

RETURNS:

[series! port! bitset!]

at

USAGE:

AT series index

DESCRIPTION:

Returns a series at a given index.

AT is an action! value.

ARGUMENTS:

series	[series! port!]
index	[integer! pair!]

RETURNS:

[series! port!]

back

USAGE:

BACK series

DESCRIPTION:

Returns a series at the previous index.
BACK is an action! value.

ARGUMENTS:

series [series! port!]

RETURNS:

[series! port!]

change

USAGE:

CHANGE series value

DESCRIPTION:

Changes a value in a series and returns the series after the change.
CHANGE is an action! value.

ARGUMENTS:

series [series! port!] "Series at point to change."
value [any-type!] "The new value."

REFINEMENTS:

/part	=> Limits the amount to change to a given length or position.
range	[number! series!]
/only	=> Changes a series as a series.
/dup	=> Duplicates the change a specified number of times.
count	[number!]

clear

USAGE:

`CLEAR series`

DESCRIPTION:

Removes series values from current index to tail; returns new tail.
CLEAR is an action! value.

ARGUMENTS:

`series` `[series! port! bitset! map! none!]`

RETURNS:

`[series! port! bitset! map! none!]`

copy

USAGE:

`COPY value`

DESCRIPTION:

Returns a copy of a non-scalar value.
COPY is an action! value.

ARGUMENTS:

`value` `[series! any-object! bitset! map!]`

REFINEMENTS:

<code>/part</code>	=> Limit the length of the result.
<code>length</code>	<code>[number! series! pair!]</code>
<code>/deep</code>	=> Copy nested values.
<code>/types</code>	=> Copy only specific types of non-scalar values.
<code>kind</code>	<code>[datatype!]</code>

RETURNS:

`[series! any-object! bitset! map!]`

find

USAGE:

FIND series value

DESCRIPTION:

Returns the series where a value is found, or NONE.

FIND is an action! value.

ARGUMENTS:

series	[series! bitset! typeset! port! map! none!]
value	[any-type!]

REFINEMENTS:

/part	=> Limit the length of the search.
length	[number! series!]
/only	=> Treat a series search value as a single value.
/case	=> Perform a case-sensitive search.
/same	=> Use "same?" as comparator.
/any	=> TBD: Use * and ? wildcards in string searches.
/with	=> TBD: Use custom wildcards in place of * and ?.
wild	[string!]
/skip	=> Treat the series as fixed size records.
size	[integer!]
/last	=> Find the last occurrence of value, from the tail.
/reverse	=> Find the last occurrence of value, from the current index.
/tail	=> Return the tail of the match found, rather than the head.
/match	=> Match at current index only and return tail of match.

head

USAGE:

HEAD series

DESCRIPTION:

Returns a series at its first index.

HEAD is an action! value.

ARGUMENTS:

series	[series! port!]
--------	-----------------

RETURNS:

[series! port!]

head?

USAGE:

HEAD? series

DESCRIPTION:

Returns true if a series is at its first index.
HEAD? is an action! value.

ARGUMENTS:

series [series! port!]

RETURNS:

[logic!]

index?

USAGE:

INDEX? series

DESCRIPTION:

Returns the current index of series relative to the head, or of word in a context.
INDEX? is an action! value.

ARGUMENTS:

series [series! port! any-word!]

RETURNS:

[integer!]

insert

USAGE:

INSERT series value

DESCRIPTION:

Inserts value(s) at series index; returns series past the insertion.

INSERT is an action! value.

ARGUMENTS:

series	[series! port! bitset!]
value	[any-type!]

REFINEMENTS:

/part	=> Limit the number of values inserted.
length	[number! series!]
/only	=> Insert block types as single values (overrides /part).
/dup	=> Duplicate the inserted values.
count	[integer!]

RETURNS:

[series! port! bitset!]

length?

USAGE:

LENGTH? series

DESCRIPTION:

Returns the number of values in the series, from the current index to the tail.

LENGTH? is an action! value.

ARGUMENTS:

series	[series! port! bitset! map! tuple! none!]
--------	---

RETURNS:

[integer! none!]

move

USAGE:

MOVE origin target

DESCRIPTION:

Moves one or more elements from one series to another position or series.
MOVE is an action! value.

ARGUMENTS:

origin	[series! port!]
target	[series! port!]

REFINEMENTS:

/part	=> Limit the number of values inserted.
length	[integer!]

RETURNS:

[series! port!]

next

USAGE:

NEXT series

DESCRIPTION:

Returns a series at the next index.
NEXT is an action! value.

ARGUMENTS:

series	[series! port!]
--------	-----------------

RETURNS:

[series! port!]

pick

USAGE:

PICK series index

DESCRIPTION:

Returns the series value at a given index.

PICK is an action! value.

ARGUMENTS:

series [series! port! bitset! pair! tuple! money! date! time!]

index [scalar! any-string! any-word! block! logic! time!]

RETURNS:

[any-type!]

poke

USAGE:

POKE series index value

DESCRIPTION:

Replaces the series value at a given index, and returns the new value.

POKE is an action! value.

ARGUMENTS:

series [series! port! bitset!]

index [scalar! any-string! any-word! block! logic!]

value [any-type!]

RETURNS:

[series! port! bitset!]

put

USAGE:

```
PUT series key value
```

DESCRIPTION:

Replaces the value following a key, and returns the new value.
 PUT is an action! value.

ARGUMENTS:

```
series    [series! port! map! object!]
key       [scalar! any-string! any-word! binary!]
value     [any-type!]
```

REFINEMENTS:

```
/case      => Perform a case-sensitive search.
```

RETURNS:

```
[series! port! map! object!]
```

remove

USAGE:

```
REMOVE series
```

DESCRIPTION:

Returns the series at the same index after removing a value.
 REMOVE is an action! value.

ARGUMENTS:

```
series    [series! port! bitset! map! none!]
```

REFINEMENTS:

```
/part      => Removes a number of values, or values up to the given series
index.
```

```
length    [number! char! series!]
/key       => Removes a key in map.
key-arg    [scalar! any-string! any-word! binary! block!]
```

RETURNS:

```
[series! port! bitset! map! none!]
```

reverse

USAGE:

REVERSE series

DESCRIPTION:

Reverses the order of elements; returns at same position.
REVERSE is an action! value.

ARGUMENTS:

series [series! port! pair! tuple!]

REFINEMENTS:

/part => Limits to a given length or position.
 length [number! series!]
 /skip => Treat the series as fixed size records.
 size [integer!]

RETURNS:

[series! port! pair! tuple!]

select

USAGE:

SELECT series value

DESCRIPTION:

Find a value in a series and return the next value, or NONE.
SELECT is an action! value.

ARGUMENTS:

series [series! any-object! map! none!]
 value [any-type!]

REFINEMENTS:

/part => Limit the length of the search.
 length [number! series!]
 /only => Treat a series search value as a single value.
 /case => Perform a case-sensitive search.
 /same => Use "same?" as comparator.
 /any => TBD: Use * and ? wildcards in string searches.
 /with => TBD: Use custom wildcards in place of * and ?.
 wild [string!]
 /skip => Treat the series as fixed size records.
 size [integer!]
 /last => Find the last occurrence of value, from the tail.
 /reverse => Find the last occurrence of value, from the current index.

skip

USAGE:

SKIP series offset

DESCRIPTION:

Returns the series relative to the current index.
SKIP is an action! value.

ARGUMENTS:

series	[series! port!]
offset	[integer! pair!]

RETURNS:

[series! port!]

sort

USAGE:

SORT series

DESCRIPTION:

Sorts a series (modified); default sort order is ascending.
SORT is an action! value.

ARGUMENTS:

series	[series! port!]
--------	-----------------

REFINEMENTS:

/case	=> Perform a case-sensitive sort.
/skip	=> Treat the series as fixed size records.
size	[integer!]
/compare	=> Comparator offset, block (TBD) or function.
comparator	[integer! block! any-function!]
/part	=> Sort only part of a series.
length	[number! series!]
/all	=> Compare all fields (used with /skip).
/reverse	=> Reverse sort order.
/stable	=> Stable sorting.

swap

USAGE:

```
SWAP series1 series2
```

DESCRIPTION:

Swaps elements between two series or the same series.
SWAP is an action! value.

ARGUMENTS:

```
series1      [series! port!]  
series2      [series! port!]
```

tail

USAGE:

```
TAIL series
```

DESCRIPTION:

Returns a series at the index after its last value.
TAIL is an action! value.

ARGUMENTS:

```
series      [series! port!]
```

RETURNS:

```
[series! port!]
```

tail?

USAGE:

```
TAIL? series
```

DESCRIPTION:

Returns true if a series is past its last value.
TAIL? is an action! value.

ARGUMENTS:

```
series      [series! port!]
```

RETURNS:

```
[logic!]
```

take

USAGE:

TAKE series

DESCRIPTION:

Removes and returns one or more elements.

TAKE is an action! value.

ARGUMENTS:

series [series! port! none!]

REFINEMENTS:

/part => Specifies a length or end position.

length [number! series!]

/deep => Copy nested values.

/last => Take it from the tail end.

trim

USAGE:

TRIM series

DESCRIPTION:

Removes space from a string or NONE from a block.

TRIM is an action! value.

ARGUMENTS:

series [series! port!]

REFINEMENTS:

/head => Removes only from the head.

/tail => Removes only from the tail.

/auto => Auto indents lines relative to first line.

/lines => Removes all line breaks and extra spaces.

/all => Removes all whitespace.

/with str => Same as /all, but removes characters in 'str'.
[char! string! binary! integer!]

Scalar actions

absolute

USAGE:

ABSOLUTE value

DESCRIPTION:

Returns the non-negative value.

ABSOLUTE is an action! value.

ARGUMENTS:

value [number! money! char! pair! time!]

RETURNS:

[number! money! char! pair! time!]

add

USAGE:

ADD value1 value2

DESCRIPTION:

Returns the sum of the two values.

ADD is an action! value.

ARGUMENTS:

value1 [scalar! vector!] "The augend."

value2 [scalar! vector!] "The addend."

RETURNS:

The sum.

[scalar! vector!]

divide

USAGE:

```
DIVIDE value1 value2
```

DESCRIPTION:

Returns the quotient of two values.

DIVIDE is an action! value.

ARGUMENTS:

value1 [number! money! char! pair! tuple! vector! time!] "The dividend (numerator)."

value2 [number! money! char! pair! tuple! vector! time!] "The divisor (denominator)."

RETURNS:

The quotient.

[number! money! char! pair! tuple! vector! time!]

multiply

USAGE:

```
MULTIPLY value1 value2
```

DESCRIPTION:

Returns the product of two values.

MULTIPLY is an action! value.

ARGUMENTS:

value1 [number! money! char! pair! tuple! vector! time!] "The multiplicand."

value2 [number! money! char! pair! tuple! vector! time!] "The multiplier."

RETURNS:

The product.

[number! money! char! pair! tuple! vector! time!]

negate

USAGE:

NEGATE number

DESCRIPTION:

Returns the opposite (additive inverse) value.

NEGATE is an action! value.

ARGUMENTS:

number [number! money! bitset! pair! time!]

RETURNS:

[number! money! bitset! pair! time!]

power

USAGE:

POWER number exponent

DESCRIPTION:

Returns a number raised to a given power (exponent).

POWER is an action! value.

ARGUMENTS:

number [number!] "Base value."

exponent [integer! float!] "The power (index) to raise the base value by."

RETURNS:

[number!]

remainder

USAGE:

```
REMAINDER value1 value2
```

DESCRIPTION:

Returns what is left over when one value is divided by another.

REMAINDER is an action! value.

ARGUMENTS:

```
value1      [number! money! char! pair! tuple! vector! time!] "The dividend
(enumerator)."
```

```
value2      [number! money! char! pair! tuple! vector! time!] "The divisor
(denominator)."
```

RETURNS:

The remainder.

[number! money! char! pair! tuple! vector! time!]

round

USAGE:

```
ROUND n
```

DESCRIPTION:

Returns the nearest integer. Halves round up (away from zero) by default.

ROUND is an action! value.

ARGUMENTS:

```
n          [number! money! time! pair!]
```

REFINEMENTS:

```
/to          => Return the nearest multiple of the scale parameter.
  scale      [number! money! time!] "Must be a non-zero value."
/even        => Halves round toward even results.
/down        => Round toward zero, ignoring discarded digits. (truncate).
/half-down   => Halves round toward zero.
/floor       => Round in negative direction.
/ceiling     => Round in positive direction.
/half-ceiling => Halves round in positive direction.
```

subtract

USAGE:

SUBTRACT value1 value2

DESCRIPTION:

Returns the difference between two values.

SUBTRACT is an action! value.

ARGUMENTS:

value1 [scalar! vector!] "The minuend."

value2 [scalar! vector!] "The subtrahend."

RETURNS:

The difference.

[scalar! vector!]

even?

USAGE:

EVEN? number

DESCRIPTION:

Returns true if the number is evenly divisible by 2.

EVEN? is an action! value.

ARGUMENTS:

number [number! money! char! time!]

RETURNS:

[logic!]

odd?

USAGE:

ODD? number

DESCRIPTION:

Returns true if the number has a remainder of 1 when divided by 2.

ODD? is an action! value.

ARGUMENTS:

number [number! money! char! time!]

RETURNS:

[logic!]