MysterX: Using Windows COM Objects in Scheme

Version 4.0.1

June 22, 2008

MysterX is a toolkit for building Windows applications from ActiveX and COM components, using Scheme as glue code. Dynamic HTML (DHTML) is used for component presentation and event-handling.

```
(require mysterx)
```

Contents

1 Overview

1.1 Installation

MysterX requires Internet Explorer (IE) 4 or later to be installed. Distributed COM (DCOM) for your version of Windows is also required. Recent versions of Windows come with DCOM; DCOM packages for Windows 95 and 98 are made available separately.

Two Windows DLLs support low-level operations in MysterX: `"myspage.dll"` and `"myssink.dll"`. Both are installed in the registry (using `regsvr32.exe`) when Setup PLT runs the the MysterX post-installer. If you move the location of your PLT installation, you may need to re-run Setup PLT to make MysterX work. Neither of these DLLs is specific to a PLT Scheme version, so it's ok for one version of PLT Scheme to use the DLLs registered by another.

Prior to version 369.4, `"myssink.dll"` was version-specific. Its GUID was changed when it was made version-independent.

If you build a stand-alone executable that uses MysterX, you need to specifically include `"myspage.dll"` and `"myssink.dll"` with your distribution, and the DLLs will need to be registered on the end user's machine. One way to do that is to include the following little setup program (as an executable) in your distribution:

```
(module setup scheme/base
  (require mzlib/runtime-path
           mzlib/process)

  ; Ensure that DLLs are included with the distribution:
  (define-runtime-path myspage-dll '(so "myspage"))
  (define-runtime-path myssink-dll '(so "myssink"))

  ; Register the DLLs:
  (define regsvr32
    (path->string (find-executable-path "regsvr32.exe" #f)))
  (system* regsvr32 (path->string myspage-dll))
  (system* regsvr32 (path->string myssink-dll)))
```

1.2 Running a Demo

Try

```
(require mysterx/mxdemo)
```

The demo requires the MSCal Calendar control. The calendar control is normally installed with Microsoft Office, but it can also be downloaded from elsewhere; look for `"mscal.ocx"`.

1.3 Loading

Load the MysterX module with

```
(require mysterx)
```

Because some MysterX code relies on the `scheme/class` class system, you may also need

```
(require mzlib/class)
```

Several MysterX procedures take HTML strings as input. The `xml` library provides procedures that convert Scheme syntax into XML strings. You may find using these procedures useful in creating HTML strings for use by MysterX.

1.4 Remote COM servers (DCOM)

For the MysterX procedures `cocreate-instance-from-coclass` and `cocreate-instance-from-progid`, the optional *where* argument can be `'remote`. In that case, the server instance is run at the location given by the Registry key

HKEY_CLASSES_ROOT\AppID\⟨*CLSID*⟩\RemoteServerName

where ⟨*CLSID*⟩ is the CLSID of the application. This key may be set using the `dcomcnfg` utility. From `dcomcnfg`, pick the application to be run on the Applications tab, then click on the Properties button. On the Location tab, choose Run application on the following computer, and enter the machine name.

In order to run a COM remote server, the registry on the client machine must contain an entry at

HKEY_CLASSES_ROOT\CLSID\⟨*CLSID*⟩

where ⟨*CLSID*⟩ is the CLSID for the server. The server application itself need not be installed on the client machine.

There are a number of configuration issues relating to DCOM, which MysterX uses to invoke remote COM servers. The Web page

http://www.distribucon.com/dcom95.html

discusses how to setup client and server machines for DCOM.

2 COM

MysterX allows scripting of most COM components from Scheme. A COM component can be scripted in MysterX if it supports OLE Automation via the `IDispatch` interface, and if it publishes type information using the `ITypeInfo` interface.

2.1 COM Methods and Properties

MysterX allows scripting of most COM components from Scheme. A COM component can be scripted in MysterX if it supports OLE Automation via the `IDispatch` interface, and if it publishes type information using the `ITypeInfo` interface.

```
(com-all-coclasses) → (listof string?)
```

Returns a list of strings for all COM classes registered on a system.

```
(com-all-controls) → (listof string?)
```

Returns a list of strings for all COM classes registered on a system that have the `"Control"` subkey.

```
(cocreate-instance-from-coclass coclass
                                [where]) → com-object?
  coclass : string?
  where : (or/c (one-of/c 'local 'remote) string?) = 'local
(cci/coclass coclass [where]) → com-object?
  coclass : string?
  where : (or/c (one-of/c 'local 'remote) string?) = 'local
```

Returns an instance of *coclass*. This is useful for COM classes without a visual representation, or when a visual representation is not needed.

The optional argument *where* indicates a for running the instance, and may be `'local`, `'remote`, or a string indicating a machine name. See §1.4 "Remote COM servers (DCOM)" for more information.

```
(cocreate-instance-from-progid progid
                               [where]) → com-object?
  progid : string?
  where : (or/c (one-of/c 'local 'remote) string?) = 'local
(cci/progid progid [where]) → com-object?
```

```
  progid : string?
  where : (or/c (one-of/c 'local 'remote) string?) = 'local
```

Like cocreate-instance-from-coclass, but using a ProgID.

```
(coclass obj) → string?
  obj : com-object?
```

Returns a string that is the name of the COM class instantiated by *obj*, or raises an error if the COM class is not known.

```
(progid obj) → string?
  obj : com-object?
```

Returns a string that is the name of the ProgID instantiated by *obj*, or raises an error if the COM class is not known.

```
(set-coclass! obj coclass) → void?
  obj : com-object?
  coclass : string?
```

Sets the COM class for *obj* to *coclass*. This is useful when MysterX COM event-handling procedures can obtain only ambiguous information about the object's COM class.

```
(set-coclass-from-progid! obj progid) → void?
  obj : com-object?
  progid : string?
```

Like set-coclass!, but using a ProgID.

```
(com-methods obj/type) → (listof string?)
  obj/type : (or/c com-object? com-type?)
```

Returns a list of strings indicating the names of methods on *obj/type*.

```
(com-method-type obj/type method-name) → (listof symbol?)
  obj/type : (or/c com-object? com-type?)
  method-name : string?
```

Returns a list of symbols indicating the type of the specified method in *obj/type*. See §2.2 "COM Types" for information on the symbols.

```
(com-invoke obj method-name v) → any/c
  obj : com-object?
  method-name : string?
  v : any/c
```

Invokes *method-name* on *obj* with *v*s as the arguments. The special value `com-omit` may be used for optional arguments, which useful when values are supplied for arguments after the omitted argument(s).

```
(com-get-properties obj/type) → (listof string?)
  obj/type : (or/c com-object? com-type?)
```

Returns a list of strings indicating the names of readable properties in *obj/type*.

```
(com-get-property-type obj/type
                       property-name) → (listof symbol?)
  obj/type : (or/c com-object? com-type?)
  property-name : string?
```

Returns a list of symbols indicating the type of the specified property in *obj/type*. See §2.2 "COM Types" for information on the symbols.

```
(com-get-property obj property ...+) → any/c
  obj : com-object?
  property : string?
```

Returns the value of the final property by following the indicated path of *property*s, where each intermediate property is a COM object.

```
(com-set-properties obj/type) → (listof string?)
  obj/type : (or/c com-object? com-type?)
```

Returns a list of strings indicating the names of writeable properties in *obj/type*.

```
(com-set-property-type obj/type
                       property-name) → (listof symbol?)
  obj/type : (or/c com-object? com-type?)
  property-name : strig?
```

Returns a list of symbols indicating the type of the specified property in *obj/type*. See §2.2 "COM Types" for information on the symbols.

```
(com-set-property! obj string? ...+ v) → void?
  obj : com-object?
  string? : property
  v : any/c
```

Sets the value of the final property in *obj* to *v* by following the *property*s, where the value of each intermediate property is a COM object.

```
(com-help obj/type [topic]) → void?
  obj/type : (or/c com-object? com-type?)
  topic : string? = ""
```

Starts the Window Help system with help about the COM object or COM type. The optional *topic* is typically a method or property name.

```
(com-object-eq? obj1 obj2) → boolean?
  obj1 : com-object?
  obj2 : com-object?
```

Returns #t if the two COM objects are the same, #f otherwise.

```
(com-object? obj) → boolean?
  obj : com-object?
```

Returns #t if the argument is a COM object, #f otherwise.

```
(com-add-ref obj) → void?
  obj : com-object?
```

Increments the reference count for *obj*. This procedure should only be called when system-level errors occur due to a mismanaged COM object. Ordinarily, MysterX handles all COM reference-counting automatically.

```
(com-ref-count obj) → exact-nonnegative-integer?
  obj : com-object?
```

Returns a number indicating the current reference count for a COM object.

2.2 COM Types

In the result of a function like com-method-type, a type 'mx-any standards for a character, real number, string, boolean, COM currency (as in com-currency?), COM date (as in com-date?), COM scode value (as in com-scode?), COM IUnknown value (as in com-iunknown?, or COM object (as in com-object?).}

(com-object-type *obj*) → com-type?
  obj : com-object?

Returns a type for a COM object.

(com-is-a? *obj type*) → boolean?
  obj : com-object?
  type : com-type?

Return #t if *obj* is of the type *type*.

(com-currency? *v*) → boolean?
  v : any/c

Returns #t if *v* is a COM currency value, #f otherwise.

(com-currency->number *curr*) → real?
  curr : com-currency?

Returns a number for *curr*.

(number->com-currency *n*) → com-currency?
  n : real?

Converts a number to a COM currency value. A currency value is repsented with a 64-bit two's-complement integer, though *n* may contain decimal digits. If *n* is too large, an exception is raised.

(com-date? *v*) → boolean?
  v : any/c

Returns #t if *v* is a COM date value, #f otherwise.

```
(com-date->date d) → date?
  d : com-date?
```

Converts a COM date to an instance of the `date` structure type. In the result, the `dst?` field is always `#f`, and the `time-zone-offset` field is `0`.

```
(date->com-date d) → com-date?
  d : date?
```

Converts a `date` instance to a COM date value.

```
(com-scode? v) → boolean?
  v : any/c
```

Returns `#t` if *v* is a COM scode value, `#f` otherwise.

```
(com-scode->number sc) → integer?
  sc : com-scode?
```

Converts a COM scode value to an integer.

```
(number->com-scode n) → com-scode?
  n : integer?
```

Converts a number to a COM scode value. The number must be representable as a 32-bit two's-complement number, otherwise an exception is raised.

```
(com-iunknown? v) → boolean?
  v : any/c
```

Returns `#t` if *v* is a COM IUnknown value, `#f` otherwise.

```
com-omit : any/c
```

Used with `com-invoke` to represent an argument that is not provided.

2.3 COM Events

COM events are generated by COM objects. Unlike HTML events, there is no fixed set of COM events, though there are "stock" events that many COM objects support. MysterX

allows the programmer to write handlers for both stock and custom events.

```
(com-events obj/type) → (listof string?)
  obj/type : (or/c com-object? com-type?)
```

Returns a list of strings naming the events supported by *obj/type*.

If calling this procedure results in an error indicating that the COM object's coclass is ambiguous, try using either `set-coclass!` or `set-coclass-from-progid!`, then retry `com-events`.

```
(com-event-type obj/type ev) → (listof string?)
  obj/type : (or/c com-object? com-type?)
  ev : string?
```

Returns the type of an event handler for the event *ev* generated by the particular COM object/type *obj/type*. The return type of all COM event handlers is void.

See also `com-events` for dealing with a COM object that has an ambiguous class.

```
(com-register-event-handler obj ev f) → void?
  obj : com-object?
  ev : string?
  f : (any/c . -> . any)
```

Registers *f* as event handler for the event *ev* when generated by *obj*. The type of argument supplied to *f* depends on the event; the result of *f* is always discarded.

See also `com-events` for dealing with a COM object that has an ambiguous class.

```
(com-unregister-event-handler obj ev) → void?
  obj : com-object?
  ev : string?
```

Unregisters any event handler for the event *ev* that is generated by the COM object *obj*.

3 ActiveX and DHTML

A MysterX application consists of one or more browsers, which are instances of the class `mx-browser%`.

3.1 Browsers

`mx-browser%` : `class?`
  superclass: `object%`

```
(new mx-browser% [[label label]
                  [width width]
                  [height height]
                  [x x]
                  [y y]]
                  [style style])  → (is-a?/c mx-browser%)
  label : string? = "MysterX"
  width : (or/c exact-nonnegative-integer? (one-of/c 'default))
        = 'default
  height : (or/c exact-nonnegative-integer? (one-of/c 'default))
         = 'default
  x : (or/c exact-integer? (one-of/c 'default)) = 'default
  y : (or/c exact-integer? (one-of/c 'default)) = 'default
  style : (listof (any-of/c 'iconize 'maximize 'no-system-menu
                            'no-thick-border 'scrollbars))
```

Creates an instance of a MysterX browser. The *label* argument is a string for the document caption, with default . The *width*, *height*, *x*, and *y* give the size and placement of the browser window on the desktop, with defaults provided by Windows. When `style-list` includes `'scrollbars`, the vertical scrollbar is disabled if scrolling is unnecessary, and the horizontal scrollbar disappears if scrolling is unnecessary.

Although the browser window cannot be hidden initially, it can be iconized. The `restore` method can be used to restore an iconized browser to an ordinary window.

```
(send a-mx-browser current-document)
 → (is-a?/c mx-document<%>)
```

Returns the current document in the browser.

```
(send a-mx-browser print-document) → void?
```

Prints the document displayed by the browser to the default printer. As an unintentional side-effect, the browser window is minimized.

```
(send a-mx-browser show show?) → void?
  show? : any/c
```

If *show?* is `#f`, the browser window is hidden. Otherwise, the window is shown.

```
(send a-mx-browser navigate url) → string?
  url : string?
```

Navigates the browser to the URL given by *url*. Any DHTML changes to the page associated with the URL are not shown. Returns a string that is the actual URL navigated to.

```
(send a-mx-browser navigate/status url)
 → (list/c string? (or/c false/c integer? (one-of/c 'no-status)))
  url : string?
```

Navigates the browser to the URL given by *url*. Any DHTML changes to the page associated with the URL are not shown. Returns a list, whose first element string that is the actual URL navigated to, and whose second element is a status code, one of: `#f`, indicating no status could be obtained; a number, such as `200` or `404`, indicating the http status; or `'no-status`, indicating that *url* does not denote a URL with the "http" scheme.

```
(send a-mx-browser go-back) → string?
```

Navigates the browser back to a URL within its history list. Any DHTML changes to the page associated with the URL are not shown. Returns a string that is the actual URL navigated to.

```
(send a-mx-browser go-forward) → string?
```

Navigates the browser forward to a URL within its history list. Any DHTML changes to the page associated with the URL are not shown. Returns a string that is the actual URL navigated to.

```
(send a-mx-browser refresh) → boolean?
```

Refreshes the document in the browser. Returns `#t` if the refresh is successful, `#f` otherwise.

`(send a-mx-browser iconize) → void?`

Iconizes the browser window.

`(send a-mx-browser restore) → void?`

Restores the browser window, if it has been iconized.

`(send a-mx-browser current-url) → string?`

Returns a string indicating the currently displayed URL.

```
(send a-mx-browser register-event-handler elem
                                          f)   → void?
  elem : (is-a?/c mx-element%)
  f : ((is-a?/c mx-event<%>) . -> . any)
```

Registers an event handler for the HTML element *elem*. The result of *f* is discarded.

```
(send a-mx-browser unregister-event-handler elem) → void?
  elem : (is-a?/c mx-element%)
```

Unregisters an event handler for an HTML element in the browser.

`(send a-mx-browser handle-events) → void?`

Creates a thread to handle events using the registered event handlers.

`(send a-mx-browser stop-handling-events) → void?`

Kills the thread currently handling events for the browser.

`(block-while-browsers) → void?`

Blocks until all browser windows have been closed or hidden, using the show method of `mx-browser%`. This is useful when a MysterX program file is run as a script, to prevent `mzscheme` or `mred` from closing prematurely.

3.2 Documents

A browser contains one document at a time. If hyperlinks are clicked, or the navigation methods (navigate, go-forward, go-back) are used, the document changes.

`mx-document<%> : interface?`

`(send a-mx-document insert-html html) → void?`
`  html : string?`

Inserts the specified HTML string at the beginning of the document.

`(send a-mx-document append-html html) → void?`
`  html : string?`

Appends the specified HTML string at the end of the document.

`(send a-mx-document replace-html html) → void?`
`  html : string?`

Replace the current HTML in the document with the specified HTML string.

`(send a-mx-document objects) → (listof com-object?)`

Returns a list of COM objects, including ActiveX controls, that occur in the document. The order of the objects is the same as in the document.

```
(send a-mx-document insert-object-from-coclass  coclass
                                                width
                                                height
                                               [size])
 → com-object?
  coclass : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Inserts a COM object with class *coclass* at the beginning of the document. The optional *size* argument gives an interpretation for the width and height, where `'percent` indicates that the width and height are a fixed percentage of the document window size.

```
(send a-mx-document insert-object-from-progid progid
                                               width
                                               height
                                              [size])
 → com-object?
  progid : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Like insert-object-from-coclass, but with a ProgID instead of a COM class.

```
(send a-mx-document append-object-from-coclass coclass
                                               width
                                               height
                                              [size])
 → com-object?
  coclass : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Like insert-object-from-coclass, but adds to the end of the document.

```
(send a-mx-document append-object-from-progid progid
                                               width
                                               height
                                              [size])
 → com-object?
  progid : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Like insert-object-from-progid, but adds to the end of the document.

```
(send a-mx-document title) → string?
```

Returns a string indicating the document's title, that is, the text that appears within HTML TITLE tags. If the document has no title, the empty string is returned.

```
(send a-mx-document find-element  tag
                                  id
                                 [index]) → (is-a?/c mx-element%)
  tag : string?
  id : string?
  index : exact-nonnegative-integer? = 0
```

Returns an object that encapsulates an HTML element, where *tag* names an HTML tag, and *id* names the `"id"` attribute of the HTML element. The *index* is a nonnegative integer indicating the zero-based index of the element among all elements with the same *tag* and *id*. The ordering of elements is defined by Internet Explorer. The requested element must be within the document's `"body"` tags or the `"body"` element itself.

```
(send a-mx-document find-element-by-id-or-name  id
                                                [index])
 → (is-a?/c mx-element%)
  id : string?
  index : exact-nonnegative-integer? = 0
```

Returns an object that encapsulates an HTML element, where *id* names either the `"id"` or `"name"` attribute of the HTML element. The *index* is a nonnegative integer indicating the zero-based index of the element among all elements with the same `"id"` or `"name"`. The ordering of elements is defined by Internet Explorer. The requested element must be within the document's `"body"` tags or the `"body"` element itself.

```
(send a-mx-document elements-with-tag tag)
 → (listof (is-a?/c mx-element%))
  tag : string?
```

Returns a list of elements with the HTML tag given by *tag*. The requested elements must be within the document's `"body"` tags or the `"body"` element itself.

3.3 HTML Events

MysterX HTML events are generated by mouse and keyboard interaction with HTML elements in a document.

```
mx-event<%> : interface?
```

```
(send a-mx-event keypress?) → boolean?
(send a-mx-event keydown?) → boolean?
(send a-mx-event keyup?) → boolean?
(send a-mx-event mousedown?) → boolean?
(send a-mx-event mousemove?) → boolean?
(send a-mx-event mouseover?) → boolean?
(send a-mx-event mouseout?) → boolean?
(send a-mx-event mouseup?) → boolean?
(send a-mx-event click?) → boolean?
(send a-mx-event dblclick?) → boolean?
(send a-mx-event error?) → boolean?
```

Exactly one of these methods returns `#t` to indicate the type of a given event, and the others return `#f` for the event.

```
(send a-mx-event alt-key) → boolean?
```

Returns `#t` if the Alt key was pressed when the event was generated, `#f` otherwise.

```
(send a-mx-event ctrl-key) → boolean?
```

Returns `#t` if the Ctrl key was pressed when the event was generated, `#f` otherwise.

```
(send a-mx-event from-tag) → string?
```

Returns a string indicating the tag of the HTML element where the mouse is being moved from. The return value is valid only for events for which `mouseover?` or `mouseout?` produces `#t`.

```
(send a-mx-event from-id) → string?
```

Returns a string indicating the identifier of the HTML element where the mouse is being moved from. Return value is valid only for events for which `mouseover?` or `mouseout?` produces `#t`.

```
(send a-mx-event id) → string?
```

Returns a string indicating the identifier of the HTML element where the event occurred.

```
(send a-mx-event keycode) → exact-integer?
```

Returns a number indicating the keycode for the key that generated the event. Return value is valid only for events for which `keypress?`, `keydown?`, or `keyup?` produces `#t`.

`(send a-mx-event shift-key) → boolean?`

Returns `#t` if the Shift key was pressed when the event was generated, `#f` otherwise.

`(send a-mx-event tag) → string?`

Returns a string indicating the HTML tag of the element where the event occurred.

`(send a-mx-event to-tag) → string?`

Returns a string indicating the tag of the target HTML element where the mouse is being moved to. Return value is valid only for events for which `mouseover?` or `mouseout?` produces `#t`.

`(send a-mx-event to-id) → boolean?`

Returns a string indicating the identifier of the target HTML element where the mouse is being moved from. Return value is valid only for events for which `mouseover?` or `mouseout?` produces `#t`.

`(send a-mx-event x) → exact-integer?`

Returns an integer indicating the x-coordinate within the document where the event occurred.

`(send a-mx-event y) → exact-integer?`

Returns an integer indicating the y-coordinate within the document where the event occurred.

3.4 HTML and Dynamic HTML

The `mx-element%` class encapsulates HTML elements. By calling the methods of the class, you can change the appearance of elements, and place new HTML before or after the element. While the methods are described here, a good DHTML reference, such as Goodman's *Dynamic HTML* will have more complete information.

Many of the `mx-element%` methods have two variants, a version that takes or returns Scheme data, and another `-native` version that takes or returns a string. For methods that return

values of element properties, we assume two characteristics, which we do not mention in the methods' documentation: 1) Native methods return the empty string for properties that have not been set, and 2) non-native methods raise an error for properties that have not been set.

3.4.1 HTML Elements

`mx-element%` : `class?`
  superclass: `object%`

`(send a-mx-element get-html)` → `string?`

Returns a string containing all the HTML between the pair of tags represented by the element.

`(send a-mx-element get-text)` → `string?`

Returns a string containing just the text between the pair of tags represented by the element. Any nested HTML tags are not contained in the returned string.

`(send a-mx-element insert-html html)` → `void?`
  `html` : `string?`

Places the HTML given by the string `html` before the element.

`(send a-mx-element append-html html)` → `void?`
  `html` : `string?`

Places the HTML given by the string `html` after the element.

`(send a-mx-element replace-html html)` → `void?`
  `html` : `string?`

Replaces the HTML in the element with the string `html`. You must use the `find-element` or `find-element-by-id-or-name` methods of `mx-document<%>` to retrieve the updated element.

`(send a-mx-element insert-text txt)` → `void?`
  `txt` : `string?`

Places the text given by the string `txt` before the HTML element.

`(send a-mx-element append-text txt)` → `void?`
  `txt` : `string?`

Places the text given by the string *txt* after the HTML element.

```
(send a-mx-element insert-object-from-coclass  coclass
                                               width
                                               height
                                              [size])   → void?
  coclass : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Composes `coclass->html` with `insert-html`.

```
(send a-mx-element insert-object-from-progid  coclass
                                              width
                                              height
                                             [size])   → void?
  coclass : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Composes `progid->html` with `insert-html`.

```
(send a-mx-element append-object-from-coclass  coclass
                                               width
                                               height
                                              [size])   → void?
  coclass : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Composes `coclass->html` with `append-html`.

```
(send a-mx-element append-object-from-progid  coclass
                                              width
                                              height
                                             [size])   → void?
  coclass : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Composes `progid->html` with `append-html`.

```
(send a-mx-element focus) → void?
```

Sets the focus to the element. This method works only with Internet Explorer 5 and later.

```
(send a-mx-element selection) → string?
```

If the element has the `"select"` tag, returns a string indicating the value of the current selection. Otherwise, an exception s raised. The value of the selection may be different from the string visible in the dropdown list.

```
(send a-mx-element set-selection! val) → void?
  val : string?
```

If the element has the `"select"` tag, sets the selection to the entry with the value *val*, a string. Otherwise, an exception is raised. The value of the selection may be different from the string visible in the dropdown list.

```
(send a-mx-element attribute attr)
 → (or/c string? real? boolean?)
  attr : string?
```

Retrieves the attribute named by the string *attr*. The return value has a type that depends on the attribute.

```
(send a-mx-element set-attribute! attr val) → void?
  attr : string?
  val : (or/c string? real? boolean?)
```

Sets the attribute named by the string *attr*. The new value *val* has a type that depends on the attribute.

```
(send a-mx-element click) → void?
```

Simulates a mouse click on the element.

```
(send a-mx-element tag) → string?
```

Retrieves the element's HTML tag.

```
(send a-mx-element font-family) → (listof string?)
(send a-mx-element font-family-native) → string?
(send a-mx-element set-font-family! v) → void?
  v : (listof string?)
(send a-mx-element set-font-family-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `font-family` for the element.

```
(send a-mx-element font-style)
 → (one-of/c 'normal 'italic 'oblique)
(send a-mx-element font-style-native) → string?
(send a-mx-element set-font-style! v) → void?
  v : (one-of/c 'normal 'italic 'oblique)
(send a-mx-element set-font-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `font-style` for the element.

```
(send a-mx-element font-variant)
 → (one-of/c 'normal 'small-caps)
(send a-mx-element font-variant-native) → string?
(send a-mx-element set-font-variant! v) → void?
  v : (one-of/c 'normal 'small-caps)
(send a-mx-element set-font-variant-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `font-variant` for the element.

```
(send a-mx-element font-weight)
 → (one-of/c 'normal 'bold 'bolder 'lighter
             100 200 300 400 500 600 700 800 900)
(send a-mx-element font-weight-native) → string?
(send a-mx-element set-font-weight! v) → void?
  v : (one-of/c 'normal 'bold 'bolder 'lighter
                100 200 300 400 500 600 700 800 900)
(send a-mx-element set-font-weight-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `font-weight` for the element.

```
(send a-mx-element font-native) → string?
(send a-mx-element set-size-native! fs) → void?
  fs : string?
```

Retrieves or sets a string that encodes the CSS `font-style`, `font-variant`, `font-weight`, `font-size`, `line-height`, and `font-family` using the format

[⟨*font-style*⟩ | ⟨*font-variant*⟩ | ⟨*font-weight*⟩] ⟨*font-size*⟩ [⟨*line-height*⟩] ⟨*font-family*⟩

```
(send a-mx-element font-size)
 → (or/c
     (one-of/c
      'xx-small 'x-small 'small 'medium 'large 'x-large 'xx-large
      'larger 'smaller)
     css-length?
     css-percentage?)
(send a-mx-element font-size-native) → string?
(send a-mx-element set-font-size! v) → void?
  v : (or/c
       (one-of/c
        'xx-small 'x-small 'small 'medium 'large 'x-large 'xx-large
        'larger 'smaller)
       css-length?
       css-percentage?)
(send a-mx-element set-font-size-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `font-size` for the element.

```
(send a-mx-element background-native) → string?
(send a-mx-element set-background-native! b) → void?
  b : string
```

Gets or sets the element's CSS `background-color`, `background-image`, `background-repeat`, `background-attachment`, and `background-position` using the string *b*.

```
(send a-mx-element background-image)
 → (or/c (one-of/c 'none) string?)
(send a-mx-element background-image-native) → string?
(send a-mx-element set-background-image! v) → void?
  v : (or/c (one-of/c 'none) string?)
(send a-mx-element set-background-image-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `background-image` for the element.

```
(send a-mx-element background-repeat)
 → (one-of/c 'no-repeat 'repeat 'repeat-x 'repeat-y)
(send a-mx-element background-repeat-native) → string?
(send a-mx-element set-background-repeat! v) → void?
  v : (one-of/c 'no-repeat 'repeat 'repeat-x 'repeat-y)
(send a-mx-element set-background-repeat-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `background-repeat` for the element.

```
(send a-mx-element background-position)
 → (or/c
     css-length?
     css-percentage?
     (one-of/c 'left 'center 'right)
     (list/c
      (or/c css-length? css-percentage?
            (one-of/c 'left 'center 'right))
      (or/c css-length? css-percentage?
            (one-of/c 'left 'center 'right))))
(send a-mx-element background-position-native) → string?
(send a-mx-element set-background-position! v) → void?
  v : (or/c
       css-length?
       css-percentage?
       (one-of/c 'left 'center 'right)
       (list/c
        (or/c css-length? css-percentage?
              (one-of/c 'left 'center 'right))
        (or/c css-length? css-percentage?
              (one-of/c 'left 'center 'right))))
(send a-mx-element set-background-position-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `background-position` for the element.

```
(send a-mx-element text-decoration)
 → (listof (one-of/c 'none 'underline 'overline 'line-through 'blink))
(send a-mx-element text-decoration-native) → string?
(send a-mx-element set-text-decoration! v) → void?
  v : (listof (one-of/c 'none 'underline 'overline 'line-through 'blink))
(send a-mx-element set-text-decoration-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `text-decoration` for the element.

```
(send a-mx-element text-transform)
 → (one-of/c 'none 'capitalize 'uppercase 'lowercase)
(send a-mx-element text-transform-native) → string?
(send a-mx-element set-text-transform! v) → void?
  v : (one-of/c 'none 'capitalize 'uppercase 'lowercase)
(send a-mx-element set-text-transform-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `text-transform` for the element.

```
(send a-mx-element text-align)
 → (one-of/c 'left 'right 'center 'justify)
(send a-mx-element text-align-native) → string?
(send a-mx-element set-text-align! v) → void?
  v : (one-of/c 'left 'right 'center 'justify)
(send a-mx-element set-text-align-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `text-align` for the element.

```
(send a-mx-element margin) → (listof (or/c (one-of 'auto)
                                           css-length?
                                           css-percentage?))
(send a-mx-element margin-native) → string?
(send a-mx-element set-margin! v) → void?
  v : (listof (or/c (one-of 'auto)
                    css-length?
                    css-percentage?))
(send a-mx-element set-margin-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `margin` for the element.

A list representation contains one to four elements. A single element applies to all sides; two elements are top–bottom and left–right, respectively; four elements are top, left, bottom, and right, respectively.

```
(send a-mx-element padding)
 → (listof (or/c css-length? css-percentage?))
(send a-mx-element padding-native) → string?
(send a-mx-element set-padding! v) → void?
  v : (listof (or/c css-length? css-percentage?))
(send a-mx-element set-padding-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `padding` for the element.

The list contains one to four elements, which apply to sides as for `margin`.

```
(send a-mx-element border)
 → (listof (or/c (or/c (one-of/c 'medium 'thin 'thick) css-length?)
                 (one-of/c 'none 'dotted 'dashed 'solid 'double
                           'groove 'ridge 'inset 'outset)
                 (or/c symbol? string?)))
(send a-mx-element border-native) → string?
(send a-mx-element set-border! v) → void?
  v : (listof (or/c (or/c (one-of/c 'medium 'thin 'thick) css-length?)
                    (one-of/c 'none 'dotted 'dashed 'solid 'double
                              'groove 'ridge 'inset 'outset)
                    (or/c symbol? string?)))
(send a-mx-element set-border-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border` for the element.

Each element of the list describes a width, style, or color. A color is a symbol indicating a color or an RGB string.

```
(send a-mx-element border-top) → ....
(send a-mx-element border-top-native) → string?
(send a-mx-element set-border-top! v) → void?
  v : ....
(send a-mx-element set-border-top-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-top` for the element.

The non-string repersentation is the same as for `border`.

```
(send a-mx-element border-bottom) → ....
(send a-mx-element border-bottom-native) → string?
(send a-mx-element set-border-bottom! v) → void?
  v : ....
(send a-mx-element set-border-bottom-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-bottom` for the element.

The non-string repersentation is the same as for `border`.

```
(send a-mx-element border-left) → ....
(send a-mx-element border-left-native) → string?
(send a-mx-element set-border-left! v) → void?
  v : ....
(send a-mx-element set-border-left-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-left` for the element.

The non-string repersentation is the same as for `border`.

```
(send a-mx-element border-right) → ....
(send a-mx-element border-right-native) → string?
(send a-mx-element set-border-right! v) → void?
  v : ....
(send a-mx-element set-border-right-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-right` for the element.

The non-string repersentation is the same as for `border`.

```
(send a-mx-element border-color)
 → (listof (or/c symbol? string?))
(send a-mx-element border-color-native) → string?
(send a-mx-element set-border-color! v) → void?
  v : (listof (or/c symbol? string?))
(send a-mx-element set-border-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-color` for the element.

The list contains one to four elements, with side assignments as for `margin`.

```
(send a-mx-element border-width)
 → (listof (or/c css-length?
                 (one-of/c 'medium 'thin 'thick)))
(send a-mx-element border-width-native) → string?
(send a-mx-element set-border-width! v) → void?
  v : (listof (or/c css-length?
                  (one-of/c 'medium 'thin 'thick)))
(send a-mx-element set-border-width-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-width` for the element.

The list contains one to four elements, with side assignments as for `margin`.

```
(send a-mx-element border-style)
 → (one-of/c 'none 'dotted 'dashed 'solid 'double
             'groove 'ridge 'inset 'outset)
(send a-mx-element border-style-native) → string?
(send a-mx-element set-border-style! v) → void?
  v : (one-of/c 'none 'dotted 'dashed 'solid 'double
                'groove 'ridge 'inset 'outset)
(send a-mx-element set-border-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-style` for the element.

```
(send a-mx-element border-top-style) → ....
(send a-mx-element border-top-style-native) → string?
(send a-mx-element set-border-top-style! v) → void?
  v : ....
(send a-mx-element set-border-top-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-top-style` for the element.

The non-string repersentation is the same as for `border-style`.

```
(send a-mx-element border-bottom-style) → ....
(send a-mx-element border-bottom-style-native) → string?
(send a-mx-element set-border-bottom-style! v) → void?
  v : ....
(send a-mx-element set-border-bottom-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-bottom-style` for the element.

The non-string repersentation is the same as for `border-style`.

```
(send a-mx-element border-left-style) → ....
(send a-mx-element border-left-style-native) → string?
(send a-mx-element set-border-left-style! v) → void?
  v : ....
(send a-mx-element set-border-left-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-left-style` for the element.

The non-string repersentation is the same as for `border-style`.

```
(send a-mx-element border-right-style) → ....
(send a-mx-element border-right-style-native) → string?
(send a-mx-element set-border-right-style! v) → void?
  v : ....
(send a-mx-element set-border-right-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-right-style` for the element.

The non-string repersentation is the same as for `border-style`.

```
(send a-mx-element style-float)
 → (one-of/c 'none 'left 'right)
(send a-mx-element style-float-native) → string?
(send a-mx-element set-style-float! v) → void?
  v : (one-of/c 'none 'left 'right)
(send a-mx-element set-style-float-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `style-float` for the element.

```
(send a-mx-element clear)
 → (one-of/c 'none 'left 'right 'both)
(send a-mx-element clear-native) → string?
(send a-mx-element set-clear! v) → void?
  v : (one-of/c 'none 'left 'right 'both)
(send a-mx-element set-clear-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `clear` for the element.

```
(send a-mx-element display)
 → (one-of/c 'block 'none 'inline 'list-item
             'table-header-group 'table-footer-group)
(send a-mx-element display-native) → string?
(send a-mx-element set-display! v) → void?
  v : (one-of/c 'block 'none 'inline 'list-item
                'table-header-group 'table-footer-group)
(send a-mx-element set-display-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `display` for the element.

```
(send a-mx-element visibility)
 → (one-of/c 'inherit 'visible 'hidden)
(send a-mx-element visibility-native) → string?
(send a-mx-element set-visibility! v) → void?
  v : (one-of/c 'inherit 'visible 'hidden)
(send a-mx-element set-visibility-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `visibility` for the element.

```
(send a-mx-element list-style-type)
 → (one-of/c 'disc 'circle 'square 'decimal
             'lower-roman 'upper-roman
              'lower-alpha 'upper-alpha 'none)
(send a-mx-element list-style-type-native) → string?
(send a-mx-element set-list-style-type! v) → void?
  v : (one-of/c 'disc 'circle 'square 'decimal
                'lower-roman 'upper-roman
                 'lower-alpha 'upper-alpha 'none)
(send a-mx-element set-list-style-type-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `list-style-type` for the element.

```
(send a-mx-element list-style-position)
 → (one-of/c 'outside 'inside)
(send a-mx-element list-style-position-native) → string?
(send a-mx-element set-list-style-position! v) → void?
  v : (one-of/c 'outside 'inside)
(send a-mx-element set-list-style-position-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `list-style-position` for the element.

```
(send a-mx-element list-style-image)
 → (lambda (s)
     (and string?
          (regexp-match? #rx"^(none|url[(].*[)])$" s)))
(send a-mx-element list-style-image-native) → string?
(send a-mx-element set-list-style-image! v) → void?
  v : (lambda (s)
        (and string?
             (regexp-match? #rx"^(none|url[(].*[)])$" s)))
(send a-mx-element set-list-style-image-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `list-style-image` for the element.

```
(send a-mx-element list-style) → list?
(send a-mx-element list-style-native) → string?
(send a-mx-element set-list-style! v) → void?
  v : list?
(send a-mx-element set-list-style-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `list-style` for the element.

A list representation contains one to three elements, which have the same representations as for `list-style-type`, `list-style-position`, and `list-style-image`. The values may appear in any order.

```
(send a-mx-element position)
 → (one-of/c 'absolute 'relative 'static)
(send a-mx-element position-native) → string?
(send a-mx-element set-position! v) → void?
  v : (one-of/c 'absolute 'relative 'static)
(send a-mx-element set-position-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `position` for the element.

```
(send a-mx-element overflow)
 → (one-of/c 'visible 'scroll 'hidden 'auto)
(send a-mx-element overflow-native) → string?
(send a-mx-element set-overflow! v) → void?
  v : (one-of/c 'visible 'scroll 'hidden 'auto)
(send a-mx-element set-overflow-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `overflow` for the element.

```
(send a-mx-element pagebreak-before)
 → (one-of/c 'always 'auto 'none)
(send a-mx-element pagebreak-before-native) → string?
(send a-mx-element set-pagebreak-before! v) → void?
  v : (one-of/c 'always 'auto 'none)
(send a-mx-element set-pagebreak-before-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `pagebreak-before` for the element.

```
(send a-mx-element pagebreak-after)
 → (one-of/c 'always 'auto 'none)
(send a-mx-element pagebreak-after-native) → string?
(send a-mx-element set-pagebreak-after! v) → void?
  v : (one-of/c 'always 'auto 'none)
(send a-mx-element set-pagebreak-after-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `pagebreak-after` for the element.

```
(send a-mx-element css-text-native) → string?
(send a-mx-element set-css-text-native! txt) → void?
  txt : string?
```

Retrieves or sets a string describing the CSS `text` for the element.

```
(send a-mx-element cursor)
 → (one-of/c 'auto 'crosshair 'default
             'hand 'move 'n-resize 'ne-resize 'nw-resize 's-resize
             'se-resize 'sw-resize 'e-resize 'w-resize 'text 'wait
             'help)
(send a-mx-element cursor-native) → string?
(send a-mx-element set-cursor! v) → void?
  v : (one-of/c 'auto 'crosshair 'default
                'hand 'move 'n-resize 'ne-resize 'nw-resize 's-resize
                'se-resize 'sw-resize 'e-resize 'w-resize 'text 'wait
                'help)
(send a-mx-element set-cursor-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `cursor` for the element.

```
(send a-mx-element clip)
 → (or/c (one-of/c 'auto)
         (list/c (or/c (one-of/c 'auto)
                       css-length?)
                 (or/c (one-of/c 'auto)
                       css-length?)
                 (or/c (one-of/c 'auto)
                       css-length?)
                 (or/c (one-of/c 'auto)
                       css-length?)))
(send a-mx-element clip-native) → string?
(send a-mx-element set-clip! v) → void?
  v : (or/c (one-of/c 'auto)
            (list/c (or/c (one-of/c 'auto)
                          css-length?)
                    (or/c (one-of/c 'auto)
                          css-length?)
                    (or/c (one-of/c 'auto)
                          css-length?)
                    (or/c (one-of/c 'auto)
                          css-length?)))
(send a-mx-element set-clip-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `clip` for the element.

```
(send a-mx-element filter)
 → (cons/c symbol? (listof (list/c symbol? any/c)))
(send a-mx-element filter-native) → string?
(send a-mx-element set-filter! v) → void?
  v : (cons/c symbol? (listof (list/c symbol? any/c)))
(send a-mx-element set-filter-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `filter` for the element.

For a filter value that combines a symbol with a list, the symbol is a filter name, and the list maps symbol option names to values. The table below shows the possible options and value types for each possible filter name.

```
filter              option              value
'alpha              'enabled            boolean?
                    'finish-opacity     (integer-in 0 100)
                    'opacity            (integer-in 0 100)
                    'start-x            exact-integer?
                    'start-y            exact-integer?
                    'finish-x           exact-integer?
                    'finish-y           exact-integer?
                    'style              (one-of/c 'uniform
                                                  'linear
                                                  'radial
                                                  'rectangular)
'blend-trans        'enable             boolean?
                    'duration           real?
                    'status             (one-of/c 'stopped
                                                  'applied
                                                  'playing)
'blur               'add                boolean?
                    'enabled            boolean?
                    'direction          (one-of/c 0 45 90
                                                  135 180
                                                  225 270
                                                  315)
                    'strength           (integer-in 1 100)
'chroma             'enabled            boolean?
                    'color              string?
'drop-shadow        'enabled            boolean?
                    'off-x              exact-integer?
                    'off-y              exact-integer?
'flip-horizontal    'enabled            boolean?
'flip-vertical      'enabled            boolean?
'glow               'enabled            boolean?
                    'color              string?
                    'strength           (integer-in 1 100)
'gray               'enabled            boolean?
'invert             'enabled            boolean?
'light              'enabled            boolean?
'mask               'enabled            boolean?
                    'color              string?
'redirect           'enabled            boolean?
'reveal-trans       'enabled            boolean?
                    'duration           real?
                    'status             (one-of/c 'stopped
                                                  'applied
                                                  'playing)
'shadow             'enabled            boolean?
                    'color              string?
                    'direction          (one-of/c 0 45 90
                                                  135 180
                                                  225 270
                                                  315)
'wave               'enabled            boolean?
                    'freq               (and/c real?
                                               (not/c negative?))
                    'light-strength     (integer-in 1 100)
'x-ray              'enabled            boolean?
```

```
(send a-mx-element style-string) → string?
```

Retrieves a string describing the complete CSS description for the element.

```
(send a-mx-element text-decoration-none) → boolean?
(send a-mx-element set-text-decoration-none! v) → void?
  v : any/c
```

Retrieves or sets the CSS `text-decoration-none` for the element.

```
(send a-mx-element text-decoration-underline) → boolean?
(send a-mx-element set-text-decoration-underline! v) → void?
  v : any/c
```

Retrieves or sets the CSS `text-decoration-underline` for the element.

```
(send a-mx-element text-decoration-overline) → boolean?
(send a-mx-element set-text-decoration-overline! v) → void?
  v : any/c
```

Retrieves or sets the CSS `text-decoration-overline` for the element.

```
(send a-mx-element text-decoration-linethrough) → boolean?
(send a-mx-element set-text-decoration-linethrough! v) → void?
  v : any/c
```

Retrieves or sets the CSS `text-decoration-linethrough` for the element.

```
(send a-mx-element text-decoration-blink) → boolean?
(send a-mx-element set-text-decoration-blink! v) → void?
  v : any/c
```

Retrieves or sets the CSS `text-decoration-blink` for the element.

```
(send a-mx-element pixel-top) → exact-integer?
(send a-mx-element set-pixel-top! v) → void?
  v : exact-integer?
```

Retrieves or sets the CSS `pixel-top` for the element.

```
(send a-mx-element pixel-left) → exact-integer?
(send a-mx-element set-pixel-left! v) → void?
  v : exact-integer?
```

Retrieves or sets the CSS `pixel-left` for the element.

```
(send a-mx-element pixel-width) → exact-integer?
(send a-mx-element set-pixel-width! v) → void?
  v : exact-integer?
```

Retrieves or sets the CSS `pixel-width` for the element.

```
(send a-mx-element pixel-height) → exact-integer?
(send a-mx-element set-pixel-height! v) → void?
  v : exact-integer?
```

Retrieves or sets the CSS `pixel-height` for the element.

```
(send a-mx-element pos-top) → real?
(send a-mx-element set-pos-top! v) → void?
  v : real?
```

Retrieves or sets the CSS `pos-top` for the element.

```
(send a-mx-element pos-left) → real?
(send a-mx-element set-pos-left! v) → void?
  v : real?
```

Retrieves or sets the CSS `pos-left` for the element.

```
(send a-mx-element pos-width) → real?
(send a-mx-element set-pos-width! v) → void?
  v : real?
```

Retrieves or sets the CSS `pos-width` for the element.

```
(send a-mx-element pos-height) → real?
(send a-mx-element set-pos-height! v) → void?
  v : real?
```

Retrieves or sets the CSS `pos-height` for the element.

```
(send a-mx-element color) → (or/c symbol? string?)
(send a-mx-element color-native) → string?
(send a-mx-element set-color! v) → void?
  v : (or/c symbol? string?)
(send a-mx-element set-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `color` for the element.

```
(send a-mx-element background-color) → (or/c symbol? string?)
(send a-mx-element background-color-native) → string?
(send a-mx-element set-background-color! v) → void?
  v : (or/c symbol? string?)
(send a-mx-element set-background-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `background-color` for the element.

```
(send a-mx-element background-position-x)
 → (or/c css-length? css-percentage?
          (one-of/c 'left 'center 'right))
(send a-mx-element background-position-x-native) → string?
(send a-mx-element set-background-position-x! v) → void?
  v : (or/c css-length? css-percentage?
            (one-of/c 'left 'center 'right))
(send a-mx-element set-background-position-x-native! str)
 → void?
  str : string?
```

Retrieves or sets a value describing the CSS `background-position-x` for the element.

```
(send a-mx-element background-position-y)
 → (or/c css-length? css-percentage?
          (one-of/c 'left 'center 'right))
(send a-mx-element background-position-y-native) → string?
(send a-mx-element set-background-position-y! v) → void?
  v : (or/c css-length? css-percentage?
            (one-of/c 'left 'center 'right))
(send a-mx-element set-background-position-y-native! str)
 → void?
  str : string?
```

Retrieves or sets a value describing the CSS `background-position-y` for the element.

```
(send a-mx-element letter-spacing)
 → (or/c css-length? (one-of/c 'normal))
(send a-mx-element letter-spacing-native) → string?
(send a-mx-element set-letter-spacing! v) → void?
  v : (or/c css-length? (one-of/c 'normal))
(send a-mx-element set-letter-spacing-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `letter-spacing` for the element.

```
(send a-mx-element vertical-align)
 → (one-of/c 'baseline 'sub 'super 'top 'middle
             'bottom 'text-top 'text-bottom)
(send a-mx-element vertical-align-native) → string?
(send a-mx-element set-vertical-align! v) → void?
  v : (one-of/c 'baseline 'sub 'super 'top 'middle
                'bottom 'text-top 'text-bottom)
(send a-mx-element set-vertical-align-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `vertical-align` for the element.

```
(send a-mx-element text-indent)
 → (or/c css-length? css-percentage?)
(send a-mx-element text-indent-native) → string?
(send a-mx-element set-text-indent! v) → void?
  v : (or/c css-length? css-percentage?)
(send a-mx-element set-text-indent-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `text-indent` for the element.

```
(send a-mx-element line-height)
 → (or/c css-length? css-percentage?
         (one-of/c 'normal))
(send a-mx-element line-height-native) → string?
(send a-mx-element set-line-height! v) → void?
  v : (or/c css-length? css-percentage?
           (one-of/c 'normal))
(send a-mx-element set-line-height-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `line-height` for the element.

```
(send a-mx-element margin-top)
 → (or/c css-length? css-percentage?
        (one-of/c 'auto))
(send a-mx-element margin-top-native) → string?
(send a-mx-element set-margin-top! v) → void?
  v : (or/c css-length? css-percentage?
          (one-of/c 'auto))
(send a-mx-element set-margin-top-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `margin-top` for the element.

```
(send a-mx-element margin-bottom)
 → (or/c css-length? css-percentage?
         (one-of/c 'auto))
(send a-mx-element margin-bottom-native) → string?
(send a-mx-element set-margin-bottom! v) → void?
  v : (or/c css-length? css-percentage?
           (one-of/c 'auto))
(send a-mx-element set-margin-bottom-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `margin-bottom` for the element.

```
(send a-mx-element margin-left)
 → (or/c css-length? css-percentage?
         (one-of/c 'auto))
(send a-mx-element margin-left-native) → string?
(send a-mx-element set-margin-left! v) → void?
  v : (or/c css-length? css-percentage?
           (one-of/c 'auto))
(send a-mx-element set-margin-left-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `margin-left` for the element.

```
(send a-mx-element margin-right)
 → (or/c css-length? css-percentage?
         (one-of/c 'auto))
(send a-mx-element margin-right-native) → string?
(send a-mx-element set-margin-right! v) → void?
  v : (or/c css-length? css-percentage?
           (one-of/c 'auto))
(send a-mx-element set-margin-right-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `margin-right` for the element.

```
(send a-mx-element padding-top)
 → (or/c css-length? css-percentage?)
(send a-mx-element padding-top-native) → string?
(send a-mx-element set-padding-top! v) → void?
  v : (or/c css-length? css-percentage?)
(send a-mx-element set-padding-top-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `padding-top` for the element.

```
(send a-mx-element padding-bottom)
 → (or/c css-length? css-percentage?)
(send a-mx-element padding-bottom-native) → string?
(send a-mx-element set-padding-bottom! v) → void?
  v : (or/c css-length? css-percentage?)
(send a-mx-element set-padding-bottom-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `padding-bottom` for the element.

```
(send a-mx-element padding-left)
 → (or/c css-length? css-percentage?)
(send a-mx-element padding-left-native) → string?
(send a-mx-element set-padding-left! v) → void?
  v : (or/c css-length? css-percentage?)
(send a-mx-element set-padding-left-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `padding-left` for the element.

```
(send a-mx-element padding-right)
 → (or/c css-length? css-percentage?)
(send a-mx-element padding-right-native) → string?
(send a-mx-element set-padding-right! v) → void?
  v : (or/c css-length? css-percentage?)
(send a-mx-element set-padding-right-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `padding-right` for the element.

```
(send a-mx-element border-top-color) → (or/c symbol? string?)
(send a-mx-element border-top-color-native) → string?
(send a-mx-element set-border-top-color! v) → void?
  v : (or/c symbol? string?)
(send a-mx-element set-border-top-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-top-color` for the element.

```
(send a-mx-element border-bottom-color)
 → (or/c symbol? string?)
(send a-mx-element border-bottom-color-native) → string?
(send a-mx-element set-border-bottom-color! v) → void?
  v : (or/c symbol? string?)
(send a-mx-element set-border-bottom-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-bottom-color` for the element.

```
(send a-mx-element border-left-color) → (or/c symbol? string?)
(send a-mx-element border-left-color-native) → string?
(send a-mx-element set-border-left-color! v) → void?
  v : (or/c symbol? string?)
(send a-mx-element set-border-left-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-left-color` for the element.

```
(send a-mx-element border-right-color)
 → (or/c symbol? string?)
(send a-mx-element border-right-color-native) → string?
(send a-mx-element set-border-right-color! v) → void?
  v : (or/c symbol? string?)
(send a-mx-element set-border-right-color-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-right-color` for the element.

```
(send a-mx-element border-top-width)
 → (or/c css-length?
         (one-of/c 'medium 'thin 'thick))
(send a-mx-element border-top-width-native) → string?
(send a-mx-element set-border-top-width! v) → void?
  v : (or/c css-length?
           (one-of/c 'medium 'thin 'thick))
(send a-mx-element set-border-top-width-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-top-width` for the element.

```
(send a-mx-element border-bottom-width)
 → (or/c css-length?
         (one-of/c 'medium 'thin 'thick))
(send a-mx-element border-bottom-width-native) → string?
(send a-mx-element set-border-bottom-width! v) → void?
  v : (or/c css-length?
            (one-of/c 'medium 'thin 'thick))
(send a-mx-element set-border-bottom-width-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-bottom-width` for the element.

```
(send a-mx-element border-left-width)
 → (or/c css-length?
         (one-of/c 'medium 'thin 'thick))
(send a-mx-element border-left-width-native) → string?
(send a-mx-element set-border-left-width! v) → void?
  v : (or/c css-length?
            (one-of/c 'medium 'thin 'thick))
(send a-mx-element set-border-left-width-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-left-width` for the element.

```
(send a-mx-element border-right-width)
 → (or/c css-length?
         (one-of/c 'medium 'thin 'thick))
(send a-mx-element border-right-width-native) → string?
(send a-mx-element set-border-right-width! v) → void?
  v : (or/c css-length?
            (one-of/c 'medium 'thin 'thick))
(send a-mx-element set-border-right-width-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `border-right-width` for the element.

```
(send a-mx-element width) → (or/c css-length? css-percentage?
                                  (one-of/c 'auto))
(send a-mx-element width-native) → string?
(send a-mx-element set-width! v) → void?
  v : (or/c css-length? css-percentage?
            (one-of/c 'auto))
(send a-mx-element set-width-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `width` for the element.

```
(send a-mx-element height) → (or/c css-length? css-percentage?
                                   (one-of/c 'auto))
(send a-mx-element height-native) → string?
(send a-mx-element set-height! v) → void?
  v : (or/c css-length? css-percentage?
            (one-of/c 'auto))
(send a-mx-element set-height-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `height` for the element.

```
(send a-mx-element top) → (or/c css-length? css-percentage?
                                 (one-of/c 'auto))
(send a-mx-element top-native) → string?
(send a-mx-element set-top! v) → void?
  v : (or/c css-length? css-percentage?
            (one-of/c 'auto))
(send a-mx-element set-top-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `top` for the element.

```
(send a-mx-element left) → (or/c css-length? css-percentage?
                                  (one-of/c 'auto))
(send a-mx-element left-native) → string?
(send a-mx-element set-left! v) → void?
  v : (or/c css-length? css-percentage?
            (one-of/c 'auto))
(send a-mx-element set-left-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `left` for the element.

```
(send a-mx-element z-index)
 → (or/c exact-integer? (one-of/c 'auto))
(send a-mx-element z-index-native) → string?
(send a-mx-element set-z-index! v) → void?
  v : (or/c exact-integer? (one-of/c 'auto))
(send a-mx-element set-z-index-native! str) → void?
  str : string?
```

Retrieves or sets a value describing the CSS `z-index` for the element.

3.4.2 Generating ActiveX HTML

```
(coclass->html name width height [size]) → string?
  name : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
(progid->html name width height [size]) → string?
  name : string?
  width : exact-integer?
  height : exact-integer?
  size : (one-of/c 'pixels 'percent) = 'pixels
```

Returns a string containing HTML which when inserted into a document loads the COM object with the COM class or ProgID given by *name*. This procedure is suitable for placing ActiveX controls within complex HTML. The optional *size* argument gives an interpretation for the *width* and *height* arguments; by default, *size* is `'pixels`, but may also be `'percent`, indicating that the width and height are a fixed percentage of the document window size.

3.4.3 CSS

In the `mx-element%` method descriptions, "CSS" refers to the Cascading Style Sheets specification. A CSS length is string consisting of a decimal integer number followed by one of the units `px` (pixels), `em` (font height), `ex` (height of an "x"), `in` (inches), `cm` (centimeters), `mm` (millimeters), `pc` (picas), or `pt` (points). A CSS percentage is a string consisting of a decimal real number followed by `%`. When using `-native` methods, CSS lengths and percentages are given as strings. For use by non-native methods, the `css-percentage` and `css-length` structures have been defined.

```
(struct css-percentage (num))
```

```
  num : real?
(struct css-length (num units))
  num : real?
  units : (symbols em ex cm mm in pt pc px)
```

3.4.4 Colors

Many element properties represent colors. In HTML, colors may be represented by an RGB string, which contains 7 characters. The first character is `#`, the rest are hexadecimal digits (`0`-`9` and `a`-`f` or `A`-`F`); the first two digits are for the red component of the color, the middle two for the green component, and the last two for the blue component. For example, `"#FFFFFF"` is white, `"#000000"` is black, and `"#00FF00"` is green.

There are also predefined color names. The `-native` methods use these names in strings, while their nonnative counterpart methods use the names as symbols.

The predefined color names are:

```
aliceblue antiquewhite aqua aquamarine azure
beige bisque black blanchedalmond blue
blueviolet brown burlywood cadetblue chartreuse
chocolate coral cornflower cornsilk crimson cyan
darkblue darkcyan darkgoldenrod darkgray
darkgreen darkkhaki darkmagenta darkolivegreen
darkorange darkorchid darkred darksalmon
darkseagreen darkslateblue darkslategray
darkturquoise darkviolet deeppink deepskyblue
dimgray dodgerblue firebrick floralwhite
forestgreen fuchsia gainsboro ghostwhite gold
goldenrod gray green greenyellow honeydew
hotpink indianred indigo ivory khaki lavender
lavenderblush lawngreen lemonchiffon lightblue
lightcoral lightcyan lightgoldenrodyellow
lightgreen lightgray lightpink lightsalmon
lightseagreen lightskyblue lightslategray
lightsteelblue lightyellow lime limegreen linen
magenta maroon mediumaquamarine mediumblue
mediumorchid mediumpurple mediumseagreen
mediumslateblue mediumspringgreen
mediumturquoise mediumvioletred midnightblue
mintcream mistyrose moccasin navajowhite navy
oldlace olive olivedrab orange orangered orchid
palegoldenrod palegreen paleturquoise
palevioletred papayawhip peachpuff peru pink
plum powderblue purple red rosybrown royalblue
```

```
saddlebrown salmon sandybrown seagreen seashell
sienna silver skyblue slateblue slategray snow
springgreen steelblue tan teal thistle tomato
turquoise violet wheat white whitesmoke yellow
yellowgreen
```

4 Version

`(mx-version)` → `string?`

Returns a string indicating the version of MysterX.

Index