



DOM in Sciter Script

home <https://sciter.com>
discussion <https://sciter.com/forums/>
github <https://github.com/c-smile/sciter-sdk>
wrappers <https://github.com/sciter-sdk>

DOM classes

Node text,comment,element
 Element:Node Element
 View DOM container - window

Predefined elements

self // current document
 view.root // window's root document

Element, creation

```
new Element("div" [, "...text..."])
Element.create { ... }
elem.insert { ... }
```

Element and Node traversal

```
elem.first // first child
elem.last // last child
elem.next // next element
elem.prior // previous element
```

```
elem.firstChild
elem.lastNode
node.nextSibling
node.priorNode
```

```
node.parent // parent element
elem.owner // owner element
elem.layoutParent
```

```
child = elem[int]; // nth element
child = elem.nodes()[int]; // nth node
```

```
nChildren = elem.length;
nChildNodes = elem.nodes().length;
```

```
for(var child in elem) {...}
for(var (index,child) in elem) {...}
```

Element attributes

```
elem.id
elem.tag
```

```
elem.attributes.length
elem.attributes[int|"name"]
elem.attributes.clear()
elem.attributes.remove(int|"name")
elem.attributes.exists(int|"name")
```

```
elem.attributes.addClass("name")
elem.attributes.removeClass("name")
elem.attributes.toggleClass("name", on)
elem.attributes.hasClass("name")
```

```
for(var (name, val) in el.attributes){...}
```

Element style, get/set

```
elem.style["background-color"]
elem.style#background-color
```

```
elem.style["background-color"] = "red";
elem.style#background-color = "red";
elem.style.set {
  background-color : color(255,0,0),
  color : color(255,255,255)
}
```

Element style extras

```
elem.style.backgroundImage : null|Image
elem.style.foregroundImage : null|Image
```

```
elem.style.all() // used style properties
elem.style.clear() // drop runtime styles
elem.style.rules() // applied style rules
self.style.documentRules()
```

```
elem.style.constant("name") // style const
elem.style.variable("name")
elem.style.variable("name", newVal)
elem.style.variables() : object
elem.style.variables{var1:val1,...};
```

Select DOM elements

```
elem.select("selector"):Element|null
elem.selectAll("selector"):Array
elem.selectParent("selector"):Element|null
elem.match("selector"):true|false
```

```
elem.$(selector):Element|null
elem.$$ (selector):Array
elem.$p(selector):Element|null
elem.$$p(selector):Array
elem.$o(selector):Element|null // owner
elem.$is(selector):true|false
```

Global selector functions:

```
$(selector):Element|null
$$ (selector):Array
```

DOM content and mutation

```
elem.text; elem.text = "new text";
elem.html; elem.html = "inner html";
elem.outerHtml; elem.outerHtml = "html";
```

```
elem.append("html"|elem|Array);
elem.prepend("html"|elem|Array);
```

```
elem.insert("html"|elem|Array, atIndex);
elem.content(elem1[, elem2,...]|Array);
```

```
elem.$append(html {expr});
elem.$prepend(html {expr});
elem.$content(html {expr});
elem.$after(html); elem.$before(html);
elem.$replace(outer html {expr});
```

```
elem.remove();
var temp = elem.detach();
//for frames as for other elements:
elem.load("url" [, headers]);
elem.load("html", "url");
```

DOM position

```
elem.belongsTo(parent[, useUITree[, andThis]])
elem.commonParent(otherElem): Element
elem.find(x,y): Element
elem.mapLocalToView(x,y):(xView,yView)
elem.mapViewToLocal(xView,yView):(x,y)
```

Popups and "windowed" elements

```
elem.popup(popupToShow, pLacement[, x,y]);
elem.closePopup();
```

```
elem.move(x,y[,w,h][, relto][, mode][, rerTo])
```

Timers, animations and updates

```
elem.timer(300ms, function);
elem.animate(stepFunc[, endFunc][, 300ms]);
```

```
elem.refresh([x,y,w,h]);
elem.update();
elem.update(stateUpdateFunc);
```

DOM Indicators

```
elem.index // in parent
node.nodeIndex // in parent.nodes()
```

```
elem.root:Element // parent document
elem.view:View // parent window
```

```
elem.isVisible :true|false
elem.isEnabled :true|false
node.isElement :true|false
node.isText :true|false
node.isComment :true|false
```

```
var v = elem.value;
elem.value = newVal;
```

```
elem.url(["relativePath"])
```

Layout position

```
elem.box(what, edge, relTo)
```

```
where what is:
#left returns x1 pos
#top returns y1 pos
#right returns x2 pos
#bottom returns y2 pos
#width
#height
#rect (x1,y1,x2,y2)
#rectw (x1,y1,width,height)
#dimension (width,height)
#position (x1,y1)
```

edge is one of:

```
#margin #border #padding #inner
#content #icon
```

relTo is one of:

```
#self #view #root #parent #screen
#content #container
```

Event handlers

```
elem.on("event" [, "selector"], func);
elem.off("event" [, "selector"] | func);
```

```
elem << event event [$(sel)] [(evt)]{...}
```

Free standing (global) event handler

```
event event [$(sel)] [(evt[, that])]{...}
```

```
elem.capture(onOff[, #strict]);
```

Event generation and post

```
elem.sendEvent("event" [, data]);
elem.postEvent("event" [, data]);
```

```
elem.post(func[, ifNotThere]);
```

```
elem.sendKeyEvent(eventDef);
elem.sendMouseEvent(eventDef);
```

Element state

```
elem.state.stateName
```

where stateName is a bool prop, one of:

```
link hover active focus ownsfocus
visited current checked unchecked
selected disabled readonly expanded
collapsed incomplete invalid
animating focusable anchor
synthetic ready popup ownspopup
tabfocus empty busy dragover
droptarget moving copying dragsource
pressed value screen
```