

Usage: query()

Use `query` to select one or more DOM elements based on a simple selector string. The `query` method is used to return *all* nodes that match your criteria unless the `firstOnly` arg is true.

```
var matchingNodes = YAHOO.util.Selector.query(
    "ul li a", "itemList");
```

Note: Will return all anchor elements within list-items of unordered lists who are descendants of the element whose id attribute is "itemList".

Usage: YAHOO.util.Selector.query()

```
YAHOO.util.Selector.query(string selector[, node
    | string startingNode, bool firstOnly])
```

Arguments:

- (1) **selector:** A string representing the CSS selector you want to target.
- (2) **startingNode:** The node at which to begin the search (defaults to *document*). Be as specific as possible in choosing your startingNode to maximize performance.
- (3) **firstOnly:** Whether or not to return only the first match.

Returns:

- (1) **Matching Node(s):** An array of nodes that match your selector criteria. If `firstOnly` is true, this returns a single node or null if no match.

Usage: YAHOO.util.Selector.filter()

```
YAHOO.util.Selector.filter(arr | nodeset nodes,
    string selector)
```

Arguments:

- (1) **nodes:** A nodeList or an array of nodes from which you want to select specific nodes that match your criteria.
- (2) **selector:** A CSS selector against which you want to test and filter the *nodes*.

Usage: YAHOO.util.Selector.test()

```
YAHOO.util.Selector.test(str | elRef node,
    string selector)
```

Arguments:

- (1) **node:** A node to test
- (2) **selector:** A CSS selector against which you want to test the *node*.

Note: returns `true` if the *node* matches the *selector*, otherwise `false`.

Pseudo-classes

The Selector Utility supports the use of the pseudo-classes listed here; for more info on these, see the W3C Selectors working draft (<http://www.w3.org/TR/css3-selectors/#pseudo-classes>).

Pseudo-class	Description
:root	The root of the document; in HTML 4.x, this is the HTML element.
:nth-child(<i>an+b</i>)	Starting from the <i>b</i> th child, match every <i>a</i> th element.
:nth-last-child(<i>an+b</i>)	An element that has <i>an + b</i> siblings after it.
:nth-of-type(<i>an+b</i>)	An element that has <i>an + b</i> siblings before it that share the same element name.
:nth-last-of-type(<i>an+b</i>)	An element that has <i>an + b</i> siblings after it that share the same element name.
:first-child	Same as :nth-child(1) — the first child of a given element.
:last-child	Same as :nth-last-child(1) — the last child of a given element.
:first-of-type	Same as :nth-of-type(1) — the first child of a given element with a given element name.
:last-of-type	Same as :nth-last-of-type(1) — the last child of a given type of the specified element.
:only-child	An element who is the only child of its parent node.
:only-of-type	An element whose element name is not shared by any sibling nodes.
:empty	An element that has no children.
:not()	The negation pseudo-class; takes a simple selector as an argument, representing an element not represented by the argument.
:contains()	An element whose textual contents contain the substring provided in the argument.
:checked	A radio button or checkbox that is in a checked state.

Notes regarding (an+b) notation:

Starting from the *b*th child, match every *a*th element. For example, "nth-child(2n+1)" starts from the first element and returns every other element. The "odd" and "even" keywords are supported, so "2n+1" is equivalent to "odd". "1n+2" and "n+2" are equivalent. "nth-child(0n+3)" is equivalent to "nth-child(3)". Zero value means no repeat matching, thus only the first *b*th element is matched. "3n+0" is equivalent to "3n".

Attribute Operators

att=val	equality	att^=val	value starts with val
att!=val	inequality	att\$=val	value ends with val
att~val	value matches one of space-delimited words in val	att*=val	value contains at least one occurrence of val
att =val	value starts with val or val-	att	test for the existence of the attribute

Solutions

```
Selector.query("#nav ul:first-of-type > li:not(.selected)"); //
Starting from the first "ul" inside of "nav", return all "li"
elements that do not have the "selected" class.

Selector.query("ul:first-of-type > li.selected", "nav", true); //
Starting from the first "ul" inside of "nav", return the first
"li" element that has the "selected" class.

Dom.addClass(Selector.query("#data tr:nth-child(odd)", "odd" ) //
add the class "odd" to all odd rows within the "data" element.
```

YAHOO.util.Selector Methods

query(string *selector*[, node | string *startingNode*, bool *firstOnly*]) the *startingNode* can be passed in as a string element ID or as an element reference and defaults to the document element; returns an array of matching nodes

filter(arr | nodeList *nodes*, string *selector*) returns any *nodes* that match the *selector*

test(str | elRef *node*, string *selector*) returns boolean indicating whether the *node* matches the *selector* criteria

Combinators

The Selector Utility supports the following four combinators:

" "	Descendant Combinator: "A B" represents an element B that has A as an ancestor.
>	Child Combinator: "A > B" represents an element B whose parent node is A.
+	Direct Adjacent Combinator: "A + B" represents an element B immediately following a sibling element A.
~	Indirect Adjacent Combinator: "A ~ B" represents an element B following (not necessarily immediately following) a sibling element A.

Dependencies

The Selector Utility requires only the YAHOO Global Object.