W3C DOM Compatibility - Core

These compatibility tables detail support for the W3C DOM Core Level 1 and 2 modules in all modern browsers.

Want these Tables updated? Join the donation drive.

On this page I grouped the various W3C DOM methods and properties in nine tables. Basically you must know the first five tables by heart and you'll rarely need the last four tables.

1. Creating elements; HTML elements or text nodes.
2. Several ways of getting elements.
3. Node information. Once you found a node you need more information about it.
4. Walking the DOM tree. How to go from one node to another.
5. DOM traversal properties that ignore text nodes.
6. Node manipulation. How to move nodes through the document.
7. Manipulating data. Data is always text, and there are some specialized methods for dealing with it.
8. Manipulating attributes. Terrible browser incompatibilities.
9. Miscellaneous methods and properties. You'll rarely need one of them.
10. Microsoft extensions to Level 1 DOM Core. Generally not interesting.

Last major update on 18 July 2010.

This page has been translated into Romanian.

Mobile table.

Creating elements

These two methods create new HTML elements which you can then insert into the document.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
createElement()			yes				yes				yes		yes		yes		to be tested
Create a new element Test page	`var x = document.createElement('P')` Create a new HTML element node <P> and temporarily place it in x, which is later inserted into the document. • IE8 and below also support `('<P>')`.																
createTextNode()			yes				yes				yes		yes		yes		to be tested
Create a new text node Test page	`var x = document.createTextNode('text')` Create a text node with content `text` and temporarily place it in x, which is later inserted into the document.																

Getting elements

These methods are meant for getting the HTML elements you need from the document.

You must know these methods by heart.

Contents of this table

See also the key to my compatibility tables.

show page contents

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getElementById()	almost		yes		yes		yes			yes		yes		yes			to be tested

Get the element with this ID

```
var x = document.getElementById('test')
```

Take the element with `id="test"` (wherever it is in the document) and put it in `x`.

Test page

If there is more than one element with `id="test"`, the method selects the first in the document. All others are ignored.

Lower case 'd'!!

- IE5-7 also return the element with `name="test"`.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getElementsByClassName()	no		yes		yes		yes			yes		yes		yes			to be tested

Get a nodeList of the elements with this class.

```
document.getElementsByClassName('test')
document.getElementsByClassName('test test2')
```

Test page

The first expression returns a nodeList with all elements that have a `class` value that contains "test". The second one returns a nodeList will all elements that have a `class` value that contains both "test" and "test2" (in any order).

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getElementsByTagName()	in-complete		yes		yes		yes			yes		yes		yes			to be tested

Get all tags of this type

```
var x = document.getElementsByTagName('P')
```

Test page

Make `x` into a nodeList of all P's in the document, so `x[1]` is the second P etc.

```
var x = y.getElementsByTagName('P')
```

Gets all paragraphs that are descendants of node `y`.

- The * argument, which ought to select all elements in the document, doesn't work in IE 5.5.
- Custom tags are not returned in Konqueror.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
querySelectorAll()	no		yes	no		yes				yes		yes		yes			to be tested

Get a nodeList of elements by CSS selector

```
document.querySelectorAll('.testClass')
document.querySelectorAll('.testClass + p')
```

Test page

Returns a nodeList with all elements that have a `class` value that contains "testClass"; or a nodeList with all paragraphs directly following such an element.

Essentially, this method allows you to use CSS syntax to retrieve elements.

| Selector | IE 5.5 | IE 6 | IE 7 | IE8 | IE9 pr3 | FF 3.0 | FF 3.5 | FF 3.6 | FF 4b1 | Saf 4.0 Win | Saf 5.0 Win | Chrome 4 | Chrome 5 | Opera 10.10 | Opera 10.53 | Opera 10.60 | Konqueror 4.x |

Node information

These four properties give basic information about all nodes. What they return depends on the node type. They are read-only, except for `nodeValue`.

There are three basic node types: element nodes (HTML tags), attribute nodes and text nodes. I test these properties for all these three types and added a fourth node type: the document node (the root of all other nodes).

You must know these properties by heart.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
nodeName	in-complete		yes				yes			yes		yes			yes		to be tested

The name of the node in UPPER CASE

```
x.nodeName
```

Test page

The name of node `x`. The correct names are:

Element	Attribute	Text	Comments	Document
the UPPER CASE tag name	the attribute name	`#text`	`#comment`	`#document`

- IE 5.5 doesn't support a `nodeName` for attributes and the document.

- IE 5.5 reports the nodeName of a comment as !.
- Konqueror doesn't see comment nodes.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
nodeType The type of the node Test page	in-complete	yes			yes		yes		yes	yes		yes		yes			to be tested

`x.nodeType`

The type of node x. The correct types are:

Element	Attribute	Text	Comments	Document
1	2	3	8	9

- IE 5.5: attributes and document not defined; comment has nodeType 1
- Konqueror doesn't see comment nodes.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
nodeValue The value of the node, if any. Read/write Test page	in-complete	yes			yes		yes		yes	yes		yes		yes			to be tested

`x.nodeValue`

Get the value of node x

`x.nodeValue = 'Test'`

Set the value of node x

Element	Attribute	Text	Comments	Document
n/a	Value of attribute	Content of text node	Content of comment node	n/a

- IE 5.5 doesn't support nodeValue for attributes
- Konqueror doesn't see comment nodes.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
tagName The tag name of an element node Test page **Don't use**	almost				yes		yes		yes	yes		yes		yes			to be tested

`x.tagName`

Get the tag name of node x. Correct values are:

Element	Attribute	Text	Comments	Document
the UPPER CASE tag name	n/a	n/a	n/a	n/a

My advice is not to use tagName at all.
nodeName contains all functionalities of tagName, plus a few more. Therefore nodeName is always the better choice.

- In IE (all versions) the tagName of a comment node is !

The DOM tree

Five properties and two arrays for walking through the DOM tree. Using these properties, you can reach nodes that are close to the current node in the document structure.

In general you shouldn't use too many of these properties. As soon as you're doing something like

```
x.parentNode.firstChild.nextSibling.children[2]
```

your code is too complicated. Complex relationships between nodes can suddenly and unexpectedly change when you alter the document structure, and altering the document structure is the point of the W3C DOM. In general you should use only one or two of these properties per action.

You must know these properties by heart.

Contents of this table

See also the key to my compatibility tables.

show page contents

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
childNodes[]			yes				yes			yes		yes			yes		to be tested
An array with all child nodes of the node	`x.childNodes[1]`																
	Get the second child node of node `x`.																
Test page	The `childNodes` nodeList consists of all child nodes of the element, including (empty) text nodes and comment nodes.																
	• IE up to 8 does not count empty text nodes. Since this is the desired behaviour as far as I'm concerned, it still gets a Yes.																
	• Konqueror ignores comment nodes.																
children[]		almost		yes	no		yes			yes		yes			yes		to be tested
An array with all child element nodes of the node	`x.children[1]`																
	Get the second element child node of node `x`.																
Test page	Where `childNodes` holds all child nodes, `children` only holds those that are element nodes (HTML tags).																
	• IE up to 8 incorrectly counts comment nodes, too.																
firstChild			yes				yes			yes		yes			yes		to be tested
The first child node of the node	`x.firstChild`																
	Get the first child node of node `x`.																
Test page	• Konqueror ignores comment nodes.																
lastChild			yes				yes			yes		yes			yes		to be tested
The last child node of the node	`x.lastChild`																
	Get the last child of node `x`.																
Test page	• Konqueror ignores comment nodes.																

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
nextSibling			yes				yes			yes		yes			yes		to be tested
The next sibling node of the node	`x.nextSibling`																
	Get the next child of the parent of `x`.																
Test page	• Konqueror ignores comment nodes.																
parentNode			yes				yes			yes		yes			yes		to be tested
The parent node of the node	`x.parentNode`																
	Get the parent node of `x`.																
Test page																	
previousSibling			yes				yes			yes		yes			yes		to be tested
The previous sibling node of the node	`x.previousSibling`																
	Get the previous child of the parent of `x`.																
Test page	• Konqueror ignores comment nodes.																
sourceIndex		yes		incorrect			no			no		no			yes		to be tested
The index number of the node in the page source	`x.sourceIndex`																
	Get the `sourceIndex` of element `x`. This is also the index number for the element in the `document.getElementsByTagName('*')` array.																
Test page	• IE returns a `sourceIndex` that's one higher than in Opera. In IE8 and lower this refers to the correct element, but in IE9 it refers to the next element.																

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

DOM Traversal

A few useful properties that should have been in the DOM from the start but mysteriously weren't.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
childElementCount		no			yes	no		yes		yes		yes			yes		to be tested
The number of element children	`x.childElementCount`																
Test page																	
firstElementChild		no			yes	no		yes		yes		yes			yes		to be tested
The first child that is an element node	`x.firstElementChild`																
Test page																	
lastElementChild		no			yes	no		yes		yes		yes			yes		to be tested
The last child that is an element node	`x.lastElementChild`																
Test page																	
nextElementSibling		no			yes	no		yes		yes		yes			yes		to be tested
The next element node sibling	`x.nextElementSibling`																
Test page																	
previousElementSibling		no			yes	no		yes		yes		yes			yes		to be tested
The previous element node sibling	`x.previousElementSibling`																
Test page																	

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Node manipulation

These five methods allow you to restructure the document. The average DOM script uses at least two of these methods.

The changes in the document structure are applied immediately, the whole DOM tree is altered. The browser, too, will immediately show the changes.

You must know these methods by heart.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
appendChild()			yes				yes			yes		yes			yes		to be tested
Append a child node as the last node to an element	`x.appendChild(y)`																
	Make node `y` the last child of node `x`.																
Test page	If you append a node that's somewhere else in the document, it moves to the new position.																
cloneNode()			yes				yes			yes		yes			yes		to be tested
Clone a node	`x = y.cloneNode(true | false)`																
	Make node `x` a copy of node `y`. If the argument is `true`, the entire tree below `y` is copied, if it's `false` only the root node `y` is copied.																
Test page	Later you insert the clone into the document.																

Text data

These methods are for manipulating text data, i.e. the contents of text nodes.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
appendData()	no		yes		buggy			yes			yes		yes		yes		to be tested

Append data to a text node

```
x.appendData(' some extra text')
```

Appends the string `some extra text` to x, which must be a text node.

Test page

- IE8 and 9 appends the text, but doesn't show it until you click the element.

data			yes				yes				yes		yes		yes		to be tested

The content of a text node

```
x.data
```

The content of x, which must be a text node. The same as `x.nodeValue`.

Test page

Can also be set:

```
x.data = 'The new text'
```

deleteData()	no		yes				yes				yes		yes		yes		to be tested

Delete text from a text node

```
x.deleteData(4,3)
```

Delete some data from x, which must be a text node, starting at the fifth character and deleting three characters. Second argument is required.

Test page

insertData()	no		yes				yes				yes		yes		yes		to be tested

Insert text into a text node

```
x.insertData(4,' and now for some extra text ')
```

Insert the string `and now for some extra text` after the fourth character into x, which must be a text node.

Test page

normalize()	no		yes				yes				yes		yes		yes		to be tested

Merge adjacent text nodes into one node

```
x.normalize()
```

All child nodes of node x that are text nodes and have other text nodes as siblings, are merged. This is in fact the reverse of `splitText`: text nodes that were split, come together again.

Test page

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
replaceData()	no		yes				yes				yes		yes		yes		to be tested

Replace text in a text node

```
x.replaceData(4,3,' and for some new text ')
```

Replace three characters, beginning at the fifth one, of node x, which must be a text node, by the string `and for some new text`.

Test page

splitText()				buggy				yes			yes		yes		yes		to be tested

Split a text node into two text nodes

```
x.splitText(5)
```

Split the text node x at the 6th character. x now contains the first part (char. 0-5), while a new node is created (and becomes `x.nextSibling`) which contains the second part (char. 6-end) of the orginial text.

Test page

- IE handles the first `splitText()` fine, but after you've normalized the text IE doesn't split it any more.

substringData()	no		yes				yes				yes		yes		yes		to be tested

Take a substring of the text in the text node

```
x.substringData(4,3)
```

Takes a substring of x, which must be a text node, starting at the fifth character and with a length of three characters. Thus it's the same as the old substr() method of strings.

Test page

wholeText				no		yes	no		yes		yes		yes		yes		to be tested

The text of a text node *plus* the text in directly adjacent text nodes. Read only.

This read-only property is useful if you want to get the entire text at a certain point and don't want to be bothered by borders between text nodes.

Test page

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Attributes

A bloody mess. Try influencing attributes in this order:

1. Try getting or setting a specific property, like `x.id` or `y.onclick`.
2. If there is no specific property, use `getAttribute()` or `setAttribute()`.
3. If even that doesn't work, try any other method or property in the table below. Most have horrible browser incompatibility patterns, though.
4. Avoid `attributes[]`. It's worse than anything else.

In my view any method or property concerning attribute nodes should also work on the `style` attribute, event handlers and custom attributes. If not I judge the method or property incomplete.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
attributes[index] An array with the attributes of a node, accessed by index number, in the order they're defined in the source code. Test page	alternative			incorrect		incorrect				yes		yes			yes		to be tested

```
x.attributes[1]
```

This array consists of all defined attributes *in the source code order* .

- Firefox and IE8 try to create a list in source code order, but the order is off.
- IE5-7 takes the second *possible* attribute of node `x` (dataFld in the test), whether it's defined or not. Therefore it consists of all attributes that can *possibly* be defined on the node (84 all in all).
- IE 5.5 initially gives the value of the attribute; not the attribute object.

Do not use

Use `getAttribute()` instead

Do yourself a favour and don't use the indexed `attributes` array.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
attributes[key] An array with the attributes of a node, accessed by attribute name Test page	incorrect	almost		yes		yes		yes		yes		yes			yes		to be tested

```
x.attributes['align']
```

Get the align attribute object of node `x`. If the node has no align attribute, it returns `undefined` (except in IE, where it returns an attribute object that has no value.)

- IE5-7 doesn't return the value of a `style` attribute.
- IE 5.5 doesn't return custom attributes, and initially gives the attribute value instead of an attribute object.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
createAttribute() and setAttributeNode() Create a new attribute node and append it to an element node. Test page	no		yes			yes				yes		yes			yes		to be tested

```
z = document.createAttribute('title');
z.value = 'Test title';
x.setAttributeNode(z)
```

This creates a title attribute with a value and sets it on node `x`.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getAttribute() Get the value of an attribute Test page		almost		yes		yes				yes		yes			yes		to be tested

```
x.getAttribute('align')
```

Gives the value of the align attribute of node `x`. Upper case attribute names are also allowed.

- In IE5-7, accessing the style attribute gives an object, and accessing the onclick attribute gives an anonymous function wrapped around the actual content.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getAttributeNode()	no	almost		yes		yes				yes		yes			yes		to be tested

Get an attribute node

`x.getAttributeNode('align')`

Test page

Get the attribute object `align` of node `x`. This is an object, not a value.

- IE 6/7 don't allow you to access the value of `x.getAttributeNode('style')`.

hasAttribute()	no	yes	yes	yes	yes	yes	to be tested

Check if a node has a certain attribute

`x.hasAttribute('align')`

Returns `true` when node `x` has an align attribute, `false` when it hasn't.

Test page

hasAttributes()	no	yes	yes	yes	yes	yes	to be tested

Check if a node has attributes

`x.hasAttributes()`

Returns `true` when node `x` has attributes, `false` when it hasn't.

Test page

name	no	yes	yes	yes	yes	yes	to be tested

The name of an attribute

`x.name`

The name of attribute node `x`.

Test page

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
removeAttribute()		almost		yes		yes		yes		yes		yes		almost		to be tested	

Remove an attribute node

`x.removeAttribute('align')`

Remove the `align` attribute from node `x`.

Test page

- IE5-7 and Opera don't remove event handlers.
- Konqueror doesn't remove the align attribute.

removeAttributeNode()	no	minimal	incomplete	yes		yes		yes		yes		yes		almost		to be tested	

Remove an attribute node

```
x.removeAttributeNode(x.attributes['align'])
x.removeAttributeNode(x.attributes[1])
x.removeAttributeNode(x.getAttributeNode('align'))
```

Test page

Removes the attribute node. There is little difference with `removeAttribute()`, except in the argument.

- IE 6 does't remove anything, but doesn't give an error message either.
- IE 7 doesn't remove styles and event handlers.
- Opera doesn't remove event handlers.
- Konqueror doesn't remove the align attribute.

setAttribute()	incomplete		yes		yes		yes		yes		yes		yes		yes	to be tested	

Set the value of an attribute

`x.setAttribute('align','right')`

Test page

Set the align attribute of node `x` to `right`. The name and value are both strings.

- IE5-7 doesn't set styles and removes events when you try to set them.

setAttributeNode()

See `createAttribute()`

value	no	almost		yes		yes		yes		yes		yes		yes		to be tested	

The value of an attribute

`x.value`

The value of attribute `x`.

Test page

- IE6-7 gives `null` for style values.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Miscellaneous

A lot of miscellaneous methods and properties that you'll rarely need. I use only two of them in an actual script.

show page contents

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
compareDocumentPosition()		no			yes		yes			yes		yes			yes		to be tested

Gives the relative place of one element compared to another.

```
x.compareDocumentPosition(y)
```

Compares the document (DOM) position of element `y` to that of element `x` The method returns a bitmask:

Test page

- 1: Position disconnected
- 2: Precedes
- 4: Follows
- 8: Contains
- 16: Is contained by

All relevant numbers are added, and this sum is returned. So if `y` follows (4) and is contained by (16) `x`, the method returns 20.

contains()		yes			no		yes			yes		yes			yes		to be tested

Check whether an element contains another element

```
x.contains(y)
```

If node `y` is a descendant of node `x`, the method returns `true`, else `false`.

Test page

See this blog post for a Firefox workaround. (And yes, Firefox's constant refusal to implement this very useful method is becoming extremely annoying.)

createDocument()		no		incorrect		yes			yes		yes		yes		yes		to be tested

Create a new document

```
x = document.implementation.createDocument('','',null)
```

Test page

Creates a new XML document.

- IE9 creates an HTML document.

createDocumentFragment()		no			yes		yes			yes		yes			yes		to be tested

Create a document fragment

```
x = document.createDocumentFragment();
x.[fill with nodes];
document.[somewhere].appendChild(x);
```

Test page

Create a fragment, add a lot of nodes to it, and then insert it into the document. Note that the fragment itself is not inserted, only its child nodes.

documentElement		yes			yes		yes			yes		yes			yes		to be tested

The HTML tag

```
document.documentElement
```

Test page

Represents the root element of the XML document. In any HTML document, the `<html>` element is of course the root element.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
getElementsByName()		incorrect and incomplete			yes		yes			yes		yes		incorrect			to be tested

Get elements by their name attribute

```
var x = document.getElementsByName('test')
```

Create a nodeList with all elements that have `name="test"`. It should ignore elements with `id="test"`

Test page

On my test page the `<p>`, `<input>`, `<img>` and `<ppk>` tags have this name, while there's also a paragraph with `id="test"`. Ideally, all browsers should get the first four elements and ignore the fifth one.

- IE ignores the `<p>` and `<ppk>` tags with `name="test"`, but counts the `<div>` with `id="test"`
- Konqueror ignores the `<ppk>` tag.
- Opera counts the `<div>` with `id="test"`

hasChildNodes()		yes			yes		yes			yes		yes			yes		to be tested

Check if the node has child nodes

```
x.hasChildNodes()
```

Returns `true` when node `x` has child nodes; `false` when it hasn't.

Test page

show page contents

item()		yes			yes			yes			yes			yes			yes	to be tested

Access an item in an array

```
document.getElementsByTagName('P').item(0)
```

The same as `document.getElementsByTagName('P')[0]`.

Test page

Not necessary in JavaScript

The `item()` method is meant for other programming languages where nodeLists like those returned by getElementsByTagName are not conveniently accessible as if they were arrays.

You don't need `item()` at all in JavaScript.

ownerDocument	no		yes			yes			yes			yes			yes			yes	to be tested

The document that 'owns' the element

```
x.ownerDocument
```

Refers to the document object that 'owns' node `x`. This is the document node.

Test page

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x

Microsoft extensions

As usual Microsoft has extended the standard somewhat. Though sometimes its extensions are brilliant (`innerHTML` springs to mind), in the case of the DOM Core they aren't.

Note the difference between W3C and Microsoft methods. The W3C methods are owned by the parent element of the node you want to adjust, the Microsoft methods by the node itself.

Contents of this table

See also the key to my compatibility tables.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
applyElement()		yes				no				no		no		no			to be tested

Something with nodes

```
var y = document.createElement('i');
x.applyElement(y)
```

Test page

The `<i>` element is inserted into element `x`, around the text.

clearAttributes()		incomplete				no				no		no		no			to be tested

Remove all attributes from a node

```
x.clearAttributes()
```

Remove all attributes from node `x`.

Test page

- IE doesn't clear event handlers and inline styles.

mergeAttributes()		yes				no				no		no		no			to be tested

Copy all attributes of one node to another node

```
x.mergeAttributes(y)
```

Copy all of node `y`'s attributes to node `x`.

Test page

removeNode()		yes				no				no		no		no		yes	to be tested

Remove a node

```
x.removeNode(true | false)
```

Remove node `x` from the document. If you use the argument `true` its children are also removed; if you use `false` they aren't. Note that all text nodes count as children, too.

Test page

replaceNode()		yes				no				no		no		no			to be tested

Replace a node by another node

```
x.replaceNode(y)
```

Replace node `x` by node `y`.

Test page

swapNode()		yes				no				no		no		no			to be tested

Swap two nodes

```
x.swapNode(y)
```

Put node `x` in node `y`'s place and vice versa.

Test page

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x