

Series 40 Web Apps Developer's Guide and API Reference

NOKIA Developer

Document created on 25 October 2011

Version 1.1



Table of contents

1.	Overview	5
1.1	Nokia Browser for Series 40	5
1.2	Versions of Series 40 web apps	6
1.3	Other documents	6
1.4	Platform architecture	7
2.	Standards support	8
3.	About the phones	9
4.	Events	10
4.1	Unsupported events	11
4.2	Synthetic events	11
4.2.1	Syntax for registering synthetic events	11
4.2.2	Synthetic event example	12
4.3	Adding events	13
4.3.1	Inline example	13
4.4	Mixed events	13
5.	Creating interactive applications	14
5.1	JavaScript in a distributed web browser	14
5.2	Adding on-phone interactivity	14
5.3	Animated transitions	14
5.4	Descriptions of MWL methods	16
5.4.1	addClass()	18
5.4.2	removeClass()	20
5.4.3	toggleClass()	22
5.4.4	switchClass()	24
5.4.5	iterateClass()	26
5.4.6	setGroupTarget()	29
5.4.7	setGroupNext()	31
5.4.8	show()	33
5.4.9	hide()	35
5.4.10	toggle()	37
5.4.11	scrollTo()	39
5.4.12	loadURL()	41
5.4.13	setInputValue()	42
5.4.14	timer()	44
5.4.15	stopTimer()	47
6.	Geolocation API	49
6.1	API feature support	49
6.2	Code Example	50
6.3	Privacy	51
6.4	Location accuracy	51
6.5	Related third-party APIs	51
7.	About the widget object	52
7.1	Other issues related to W3C widget standards	53
7.2	Preferences property	53
7.3	Web app version	53
8.	Image caching	55

9. Terms and abbreviations	56
Appendix A HTML support	57
Appendix B CSS support	63

Tables

Table 1: Events supported	10
Table 2: Synthetic event registration parameters	11
Table 3: Methods for registering synthetic events	12
Table 4: Transition style properties	15
Table 5: MWL methods	16
Table 6: addClass parameters	18
Table 7: removeClass parameters	20
Table 8: toggleClass parameters	22
Table 9: switchClass parameters	24
Table 10: iterateClass parameters	26
Table 11: setGroupTarget parameters	29
Table 12: setGroupNext parameters	31
Table 13: show parameters	33
Table 14: hide parameters	35
Table 15: toggle parameters	37
Table 16: scrollTo parameters	39
Table 17: loadURL parameters	41
Table 18: setInputValue parameters	42
Table 19: timer parameters	45
Table 20: stopTimer parameters	47
Table 21: The support for objects/interfaces and methods of the W3C geolocation API are listed.	49
Table 22: Widget object properties and methods	52
Table 23: Common attribute and property groups	57
Table 24: HTML tag support	58
Table 25: CSS property support	63

Figures

Figure 1: Nokia Browser for Series 40 overview	5
Figure 2: Platform stack	7

Examples

Example 1: Sample code for synthetic events 12

Example 2: **className** style 15

Example 3: Sample code for **addClass**..... 19

Example 4: Sample code for **removeClass**..... 21

Example 5: Sample code for **toggleClass**..... 23

Example 6: Sample code for **switchClass**..... 25

Example 7: Examples of **iterateClass** calls..... 27

Example 8: Sample code for **iterateClass** 28

Example 9: Sample code for **setGroupTarget** 30

Example 10: Sample code for **setGroupNext** 32

Example 11: Sample code for **show** 34

Example 12: Sample code for **hide** 36

Example 13: Sample code for **toggle**..... 38

Example 14: Sample code for **scrollTo** 40

Example 15: Sample code for **loadURL** 41

Example 16: Sample code for **setInputValue**..... 43

Example 17: Sample code for **timer** 46

Example 18: Sample code for **stopTimer**..... 48

Example 19: Example of how to implement location acquisition using the geolocation API is shown..... 50

Change history

7 April 2011	1.0	Initial document release
27 October 2011	1.1	Updated to provide information on new features added in Series 40 web apps 1.5.

1. Overview

This document contains the information you need to develop web apps for Nokia Browser for Series 40. Using this technology you can create interactive web apps using web standards such as XHTML, cascading style sheets (CSS), and the JavaScript™ programming language. You can package each web app into a single compressed file following the W3C widget standards, then install it on a Nokia Browser for Series 40 Proxy server, and run it in the Nokia Browser for Series 40 Client on a phone. Within these web apps you can create rich, interactive pages easily, pages that run well even on phones with limited resources.

1.1 Nokia Browser for Series 40

Nokia Browser for Series 40 is a distributed or proxy-based web browser that supports full web page rendering on phones with limited processing power and memory (such as some Series 40 phones). A Nokia Browser for Series 40 Proxy server executes scripts and does other processing for the phone client. The server communicates with websites on behalf of the client. It also sends the client optimised web pages, reduced in payload and streamlined for efficient rendering. This interaction is illustrated in Figure 1.

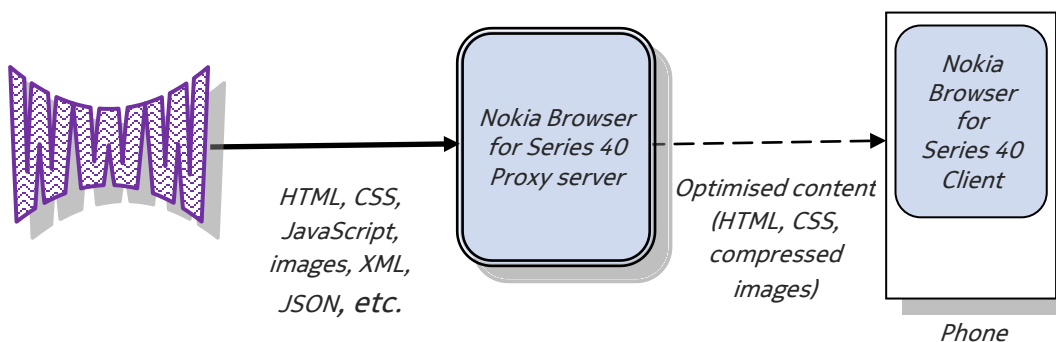


Figure 1: Nokia Browser for Series 40 overview

The goal of Series 40 web apps is to move most of the processing (especially the custom JavaScript code) from the phone to the proxy server. The server further minimises the amount of transmitted data by compressing images, sending optimised mark-up, and performing page updates by sending only the changed HTML and graphics rather than the full pages. The client has a JavaScript library called MWL (Mobile Web Library), which contains code to support application-like interaction on the phone. MWL processing is the only JavaScript that executes on the phone.

1.2 Versions of Series 40 web apps

The first version of Series 40 web apps was released in April 2011 with support for submitting web apps to Nokia Store added in May 2011. Version 1.5 was released in August 2011 and added the following features:

- improved UI with modern floating icons to activate generic web apps features in the Touch and Type UI.
- support for the core features of the W3C geolocation API, with location information provided by network-based positioning.
- web app image caching that stores all images defined in the web app on the phone for faster browsing.
- an SMS URI enabling users to create SMS messages by selecting links in web apps.

This document describes Series 40 web apps version 1.5.

1.3 Other documents

See these other documents for more information:

- [Series 40 Web Apps Platform Overview.](#)
- [Series 40 Web Apps Best Practices Guide.](#)
- [Series 40 Web Apps Publishing Guide.](#)
- [Series 40 Web Apps UX Guidelines.](#)

1.4 Platform architecture

Figure 2 depicts the platform stack for Series 40 web apps.

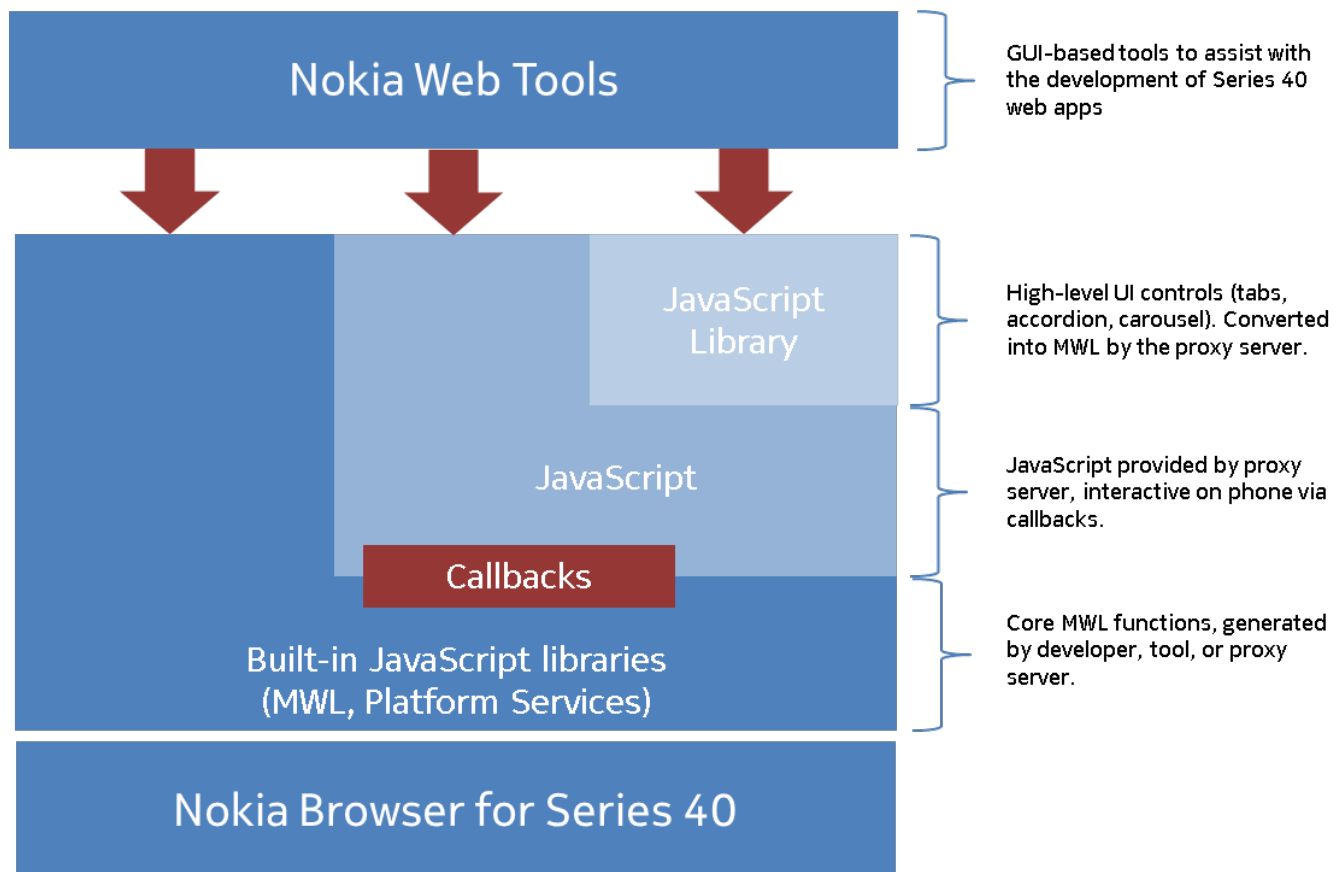


Figure 2: Platform stack

To simplify the task of implementing Series 40 web apps, Nokia services automatically handle several tasks. This includes the following functions:

- The server component reducing the amount of data sent to the Nokia Browser for Series 40 Client from the server by compressing images, using optimised HTML and only sending page changes when performing web page updates.
- All communication between the client browser and the Nokia Browser for Series 40 Proxy, including session connection and the web app **preferences** attribute.
- Colour reduction for graphics that use a large number of colours.

2. Standards support

Developers write Series 40 web apps based upon the following web standards, except as described in this document. *All URLs link to the most recent information available as of this document's publication date.*

- HTML 4.01 (defined in <http://www.w3.org/TR/1999/REC-html401-19991224/>).
- JavaScript 1.6 (defined in <https://developer.mozilla.org/en/JavaScript/Reference> and https://developer.mozilla.org/en/JavaScript/New_in_JavaScript/1.6).
- CSS Mobile Profile 2 (defined in <http://www.w3.org/TR/2008/CR-css-mobile-20081210/>).
- W3C widget packaging (defined in <http://www.w3.org/TR/2010/WD-widgets-20101005/>).
- SMS URI scheme (defined in <http://tools.ietf.org/html/rfc5724>).
- A subset of CSS3 for supporting animated transitions (see Chapter 5.3, 'Animated transitions' for more information on this as well as <http://www.w3.org/TR/2009/WD-css3-animations-20090320/>)

Please see Chapter 4.1, 'Unsupported events'; Chapter 4.2, 'Synthetic events'; Chapter 5.3, 'Animated transitions'; and Chapter 7, 'About the widget object' for information on non-standard system processing. For information on the supported HTML tags and CSS elements, see Appendix A: 'HTML support' and Appendix B, 'CSS support'.

3. About the phones

Nokia expects the set of phones that support Series 40 web apps to be continually expanding. This means that the supported phones have a variety of input methods (touch and non-touch), screen sizes, screen orientations (portrait or landscape), and so forth. It's up to you to determine how to handle these variations. You could create one application for all phones or different versions for different phones. For details of the specifications of the supported phones, please see the [supporting phones list](#) in the Devices section of [Nokia Developer](#).

One aspect of the differences between phones that will affect location based applications is the variation in location acquisition available on Series 40 phones. Most phones implement a single-cell location calculation method, while a limited number of phones use a cell-triangulation method. These different methods will have a marked effect on the accuracy of the location available to a web app. For more information, see Chapter 6, 'Geolocation API'

Development should consider also the constraints imposed by these phones. Limited memory and processing power can make it difficult for the client browser to process relatively large, complex web pages with numerous graphics. There is a time lag whenever the client has to communicate with the server, and the client doesn't cache HTML and images retrieved from the internet after the user closes a page. However, from runtime 1.5 any images in the web app are cached on the phone, for more information see Chapter 8, 'Image caching'. (The client keeps the HTML and image files cached temporarily while a web app page is open, which allows the proxy server to send the changed elements only when it updates the page.) Ultimately web app design needs to find an appropriate trade-off between use of phone resources and the desire to minimise round-trip communication between the client and server.

4. Events

Nokia Browser for Series 40 supports the DOM level 2 event model, with some exceptions. Unless specified otherwise, events occur synchronously and in the order specified.

Event	Comments
onload	This event can only be applied to the body element. You can process this event with MWL and non-MWL JavaScript.
onunload	Can only be applied to the body element. You can use it with MWL events and should not invoke an action that requires making a network request. Process this event with MWL JavaScript only.
onresize	Can only be applied to the body element. Process this event with MWL JavaScript only.
onfocus	On non-touch phones this event is fired when the cursor hovers over a node. On touch phones this event is fired when a touch fires. Process this event with MWL JavaScript only.
onblur	On non-touch phones this event is fired when the cursor departs a node. On touch phones this event is fired when a touch is removed. Process this event with MWL JavaScript only.
onclick	Fired when a user clicks the “fire” button on a 4-way navigation phone, or when the user taps on the target element on a touch-screen phone. You can process this event with MWL and non-MWL JavaScript.
onchange	This event is supported by the HTML form and select tags. See <i>Appendix A: HTML support</i> (page 57) for more information. You can process this event with MWL and non-MWL JavaScript.
onshow/onhide	These events can only appear in the body tag of the page or be added via the widget object. These events fire when the web application becomes active or releases the active application state. Examples of this event include switching browser tabs, changing from minimized to full-screen, being overlaid by a browser popup, and the browser coming out of or going into background processing. Process this event with MWL JavaScript only.

Table 1: Events supported



Note: Some of these events can be referenced as attributes in certain HTML tags. For more information on this, please see Appendix A, ‘HTML support’. Look for an attribute group called *event-attributes*.

4.1 Unsupported events

Some DOM level 2 events are only useful when combined with JavaScript code to interpret the effect of the action. Nokia Browser for Series 40 runtime environment doesn't support the following events:

- onkeydown, onkeypress, onkeyup
- onmousedown, onmousemove, onmouseup
- ontouchdown, ontouchmove, ontouchend

4.2 Synthetic events

In addition to supporting standard HTML DOM events, Nokia Browser for Series 40 allows a series of *synthetic* DOM events to support some gestures and keypad events. These events act just like standard events.



Note: The underlying phone platform manages gesture interpretation to provide a consistent experience. The web app author cannot modify the interpretation of a gesture.

4.2.1 Syntax for registering synthetic events

```
<static> mwl.add<Event>Listener(targetNode, listener)
```

The following table describes the parameters used to register synthetic events.


Parameter	Type	Description
targetNode	String	This is the selector of the node to which to add an event/listener.
listener	String	The command(s) to run when the event fires. <i>Note:</i> The system only supports MWL statements here.


Table 2: Synthetic event registration parameters

These are the methods for registering the synthetic events, which all use the syntax shown above:

Methods	Comments
addSwipeLeftListener, addSwipeRightListener, addSwipeUpListener, addSwipeDownListener	These apply to touch navigation only. They fire when the user swipes in an orthogonal direction inside the target element.
addNavLeftListener, addNavRightListener, addNavUpListener, addNavDownListener	This applies to keypad navigation only. They fire when the user presses a 4-way directional key inside the target element. <i>Note:</i> Take care when using these. They should capture only horizontal or only vertical navigation events. Otherwise there is no way to escape the element.
addLongPressListener	This fires an event after the user presses on the display (or on the fire button) for a certain period of time.

Table 3: Methods for registering synthetic events

- 

Note: The code for registering a synthetic event must be in a specific location in the HTML file. It has to be at the very end of the page body (immediately before the `</body>` tag).
- 

Note: These events can be referenced as attributes in certain HTML tags. For more information on this, please see *Appendix A: 'HTML support'*. Look for an attribute group called *event-attributes*.

4.2.2 Synthetic event example

The following code registers a listener for a left swipe event. It appears at the very end of the body. (This code calls an MWL method called `setGroupNext()`, which iterates to the next or previous element in a given block. For more information on this, please see Chapter 5.4.7, 'setGroupNext().')

```
<script type="text/javascript">
  mwl.addSwipeLeftListener('#group', "mwl.setGroupNext('#group', 'show', 'hide',
    'next')");
</script>
</body>
</html>
```

Example 1: Sample code for synthetic events

4.3 Adding events

You may add events using standard DOM Level 2 methods. You can add them either in line with the HTML, or register them via JavaScript. The system does not support registering non-synthetic events, either via JavaScript or inline.



Note: Due to the distributed nature of Nokia Browser for Series 40, JavaScript registered events can only be applied during the **onload** event phase, or for other events only to newly-inserted HTML. Web apps cannot register new events on existing HTML content.

4.3.1 Inline example

Events can appear inline on an element event handler:

```
<tag onEvent="mwl.EventHandler1(arg1, arg2);mwl.EventHandler2(arg1)">
```

4.4 Mixed events

Events can invoke both MWL methods and non-MWL methods under certain conditions.

- Interactive events – MWL and non-MWL events should not be fired by the same handler.
- **onload**, **onclick**, **onchange** – Mixed MWL and non-MWL statements are supported on these events. The Nokia Browser for Series 40 Proxy executes non-MWL statements, while MWL statements are processed on the phone. For **onclick** and **onchange**, this means that the MWL statements are processed before the non-MWL calls. For **onload**, however, non-MWL statements execute on the server before the page loads in the client, so the non-MWL calls are processed first.

5. Creating interactive applications

The client-server nature of Nokia Browser for Series 40 requires you to make some considerations to provide the best web application user experience. The following sections detail how to author web applications that fully take advantage of the capabilities of Nokia Browser for Series 40.

5.1 JavaScript in a distributed web browser

In the distributed Nokia Browser for Series 40 environment, JavaScript executes on the Nokia Browser for Series 40 Proxy. This generally means that invoking events (button clicks, timers, etc.) in the client browser will result in a network request to the server, which executes the JavaScript and returns a local page update. The server optimizes this update to minimize the amount of network traffic and payload. It sends new image assets to Nokia Browser for Series 40 client only and updates modified parts of the web application only. The only JavaScript operations that take place in the phone client should be the following, all via MWL:

- JavaScript that directly modifies CSS properties. Classes can be added/removed/switched on for elements, and new classes can be created.
- JavaScript that modifies displayable elements for a controlled amount of time (using the timer operations).

For more information on the methods that run in the phone client, please see Chapter 5.4, ‘Descriptions of MWL methods’.

5.2 Adding on-phone interactivity

With the Nokia Browser for Series 40 web runtime environment, MWL allows you to provide rich interactive applications that do not require network access. The MWL JavaScript library is built into the Nokia Browser for Series 40 Client and provides a set of optimized methods for creating interactive user interfaces and graphical transitions. When the Nokia Browser for Series 40 Proxy encounters MWL methods, it knows to defer execution of the method to the Nokia Browser for Series 40 Client. The MWL library also exists on the proxy server as a standalone JavaScript file for use and compatibility with other web browsers.

With MWL, developers can create user interface controls like tab pages, slideshows, etc. that execute directly on the client without server interaction.

5.3 Animated transitions

Nokia Browser for Series 40 supports graphical animations in web applications through partial support of CSS 3 module level 3 transitions. This allows you to specify that certain CSS style properties should smoothly transition

from one value to another instead of making a discrete change. To specify a transition on a node, add the following style rule:

```
.className {
    -webkit-transition-property: property-to-animate;
    -webkit-transition-duration: duration of the transition in seconds;
    -webkit-transition-timing-function: name of transition timing function;
    -webkit-transition-delay: wait time before starting transition in
seconds;
}
```

Example 2: className style

This table summarizes the style properties that support CSS 3 transitions in this version of the Nokia Browser for Series 40 web runtime environment.

Style property	Action
width	This smoothly changes the width of a block element. You can use it with the overflow:hidden style to hide or show content with a horizontal blind effect.
height	Smoothly changes the height of a block element. You can use it with the overflow:hidden style to hide or show content with a vertical blind effect.
margin-left	Changes the left margin position of an element. You can use it with the overflow:hidden style to provide carousel and slideshow effects. Margin-left values may be positive or negative pixels. (For more information on using negative margins, please see Series 40 Web Apps Best Practices Guide .)
margin-top	Changes the top margin position of an element. You can use it with the overflow:hidden style to provide carousel and slideshow effects. Margin-top values may be positive or negative pixels.

Table 4: Transition style properties

This version of the Nokia Browser for Series 40 web runtime environment supports the following **transition-timing-function** values:

- linear
- ease
- ease-in
- ease-out
- ease-in-out

CSS 3 transitions normally execute asynchronously. However, when a web app modifies CSS properties via the MWL API, they execute synchronously and the invoked MWL methods block until all the transitioning with the changed class properties completes.



Note: Some Nokia Browser for Series 40 phones cannot support transitions. In these situations, the browser ignores the transitions feature and the target CSS properties undergo discrete changes.

5.4 Descriptions of MWL methods

This section contains detailed reference information for the methods you can use to make interactive web applications for Series 40, as listed in Table 5. All of these methods are associated with the **mw** namespace, and all of them run in the client browser (in the phone).

Table 5: MWL methods

Name	Page #	Description
addClass()	18	Adds a specified CSS class to an element.
removeClass()	20	Removes a specified single CSS class from an element.
toggleClass()	22	Toggles a specified class attribute of an element.
switchClass()	24	Combines the removal of one element class and the addition of another in a single operation.
iterateClass()	26	Increments or decrements the specified class prefix name and applies it to the target node.
setGroupTarget()	29	Targets one node in a group to have a distinct class from the rest of its siblings.
setGroupNext()	31	Iterates to the next or previous element in a given block.
show()	33	Makes an element visible on the page.
hide()	35	Hides an element on the page.
toggle()	37	Controls the visibility of an element, making it visible if hidden and hidden if visible.

Name	Page #	Description
scrollTo()	39	Scrolls the browser to the block with the specified ID.
loadURL()	41	Breaks out of the web application and loads a specified resource directly into the root window of Nokia Browser for Series 40.
setInputValue()	42	Updates the value of a specified HTML input.
timer()	44	Executes commands in response to a timer firing.
stopTimer()	47	Stops running timers.

5.4.1 addClass()

5.4.1.1 Description

The **addClass** method adds a specified CSS class to a *targetNode* element. This only adds the class to the current class name attribute of the element. See *switchClass()* (page 24) to add and remove classes at the same time.

This is a synchronous method.

5.4.1.2 Syntax

```
<static> mw1.addClass(targetNode, className)
```

5.4.1.3 Parameters

The following table describes the parameters used in an **addClass** call.

Parameter	Type	Description
targetNode	String	This is the element to which to add a class.
className	String	This is the name of the class to add.

Table 6: **addClass** parameters

5.4.1.4 Return value

The **addClass** method does not return a value.

5.4.1.5 Example code

The following code sample adds a class to change the appearance of a line of text.

```
style type="text/css">
  .red_underline {
    color:red;
    text-decoration: underline;
  }
</style>

<div id="addClass_test">
  <a onclick="mw1.addClass('#sample_text ', 'red_underline'); return false;" href="#">
    Click to change text below to red & underline.
  </a>
  <div id="sample_text ">
    This is the text that will change
  </div>
</div>
```

Example 3: Sample code for **addClass**

5.4.2 removeClass()

5.4.2.1 Description

The **removeClass** method removes a specified single CSS class from a *targetNode* element.

This is a synchronous method.

5.4.2.2 Syntax

```
<static> mwl.removeClass(targetNode, className)
```

5.4.2.3 Parameters

The following table describes the parameters used in a **removeClass** call.

Parameter	Type	Description
targetNode	String	This is the element from which to remove a class.
className	String	This is the name of the class to remove. This can be a full class name or a simple expression: You may add a '*' to the end of the class name string to match a class-name prefix.

Table 7: **removeClass** parameters

5.4.2.4 Return value

The **removeClass** method does not return a value.

5.4.2.5 Example code

The following code sample uses **removeClass()** to change the characteristics of a line of text. Note the second parameter's use of the wildcard character **"*"**.

```
<style type="text/css">
  .red_underline {color:red; text-decoration: underline;}
  .underline_blue {color:blue; text-decoration: underline;}
  .underline_no_color {text-decoration: underline;}
  .bold {font-weight: bold;}
</style>

<p> <div id="removeClass_test_wildcard">
  <a onclick="mw1.removeClass('#sample_text_wildcard',
    'underline*');return false;" href="#">
    Click to remove blue & underline from text
  </a>
  <div id="sample_text_wildcard" class="underline_blue
    underline_no_color bold ">
    This is the text that will change
  </div>
</div>
</p>
```

Example 4: Sample code for **removeClass**

5.4.3 toggleClass()

5.4.3.1 Description

The **toggleClass** method toggles a specified class attribute of an element.

This is a synchronous method.

5.4.3.2 Syntax

```
<static> mwl.toggleClass(targetNode, className)
```

5.4.3.3 Parameters

The following table describes the parameters used in a **toggleClass** call.

Parameter	Type	Description
targetNode	String	This is the element whose class to toggle.
className	String	This is the name of the class to toggle.

Table 8: **toggleClass** parameters

5.4.3.4 Return value

The **toggleClass** method does not return a value.

5.4.3.5 Example code

The following code sample demonstrates several mwl methods – in addition to **toggleClass**, it uses **setGroupNext()** and **setInputValue()**. The code displays a graphical “switch” containing two small rectangular regions, one of which is grey. Clicking on the grey area causes different colours to scroll into the rectangular regions. A display field below shows whether the switch is in an ON or OFF state.

```

<style>
  body {padding:5px;}
  .container {width:100px; height:50px; border: solid; overflow:hidden;}
  .sw_sz {width:100px; height:50px;background-color:#00FF00;
    -webkit-transition:margin-left 0.5s linear;}
  .sw_off {margin-left:+50px;}
  .sw_on {margin-left:0px;}
  .hide {background-color:#aaaaaa; width:50px; height:50px;
display:none;}
  .show {background-color:#aaaaaa; width:50px; height:50px;
display:block;}
</style>

<div id="sw_bg" class="container">
  <div class="sw_sz sw_off" id="slide">
    <div class="show" id="is_off"
onclick="mwl.setInputValue('#sw_state','ON');
      mwl.toggleClass('#sw_bg','cont_on');
      mwl.switchClass('#slide','sw_off','sw_on');
      mwl.setGroupNext('#slide','show','hide','next');">
    </div>
    <div class="hide" id="is_on" onclick="mwl.setInputValue('#sw_state',
'OFF');
      mwl.toggleClass('#sw_bg','cont_on');
      mwl.switchClass('#slide','sw_on','sw_off');
      mwl.setGroupNext('#slide','show','hide','next');">
    </div>
  </div>
</div>

<div>Current state: <input type="text" id="sw_state" size="20" value="OFF"></div>

```

Example 5: Sample code for **toggleClass**

5.4.4 switchClass()

5.4.4.1 Description

The **switchClass** method combines the removal of one element class and the addition of another in a single operation.

This is a synchronous method.

5.4.4.2 Syntax

```
<static> mwl.switchClass(targetNode, removeClass, addClass)
```

5.4.4.3 Parameters

The following table describes the parameters used in a **switchClass** call.

Parameter	Type	Description
targetNode	String	This is the element to which to add one class and remove another.
removeClass	String	This is the name of the class to remove. This can be a full class name or a simple expression: You may add a '*' to the end of the class name string to match a class-name prefix.
addClass	String	This is the name of the class to add.

Table 9: **switchClass** parameters

5.4.4.4 Return value

The **switchClass** method does not return a value.

5.4.4.5 Example code

The following code sample is from a page that displays multiple panels in a container, with links that scroll from panel to panel horizontally. The `switchClass` calls appear at the end of the page body.

```
<style>
  .container {width: 240px; height: 320px; overflow:hidden; border-style:solid;}

  .panel {width: 240px; height: 320px;}

  #panels {width: 720px; -webkit-transition: margin-left 1.0s linear;}

  .panell1 {margin-left: 0px;}
  .panel2 {margin-left: -240px;}
  .panel3 {margin-left: -480px;}
</style>

<div id="page_wrapper" class="container">
  <div id="panels" class="panell1">
    <table>
      <tr>
        <td id="login_panel" class="panel">
          Panel 1
        </td>
        <td id="home_panel" class="panel">
          Panel 2
        </td>
        <td id="user_detail_panel" class="panel">
          Panel 3
        </td>
      </tr>
    </table>
  </div>
</div>

<a href="#" onclick="mwl.switchClass('#panels', 'panel*', 'panell1');">Panel1</a>
<a href="#" onclick="mwl.switchClass('#panels', 'panel*', 'panel2');">Panel2</a>
<a href="#" onclick="mwl.switchClass('#panels', 'panel*', 'panel3');">Panel3</a>
```

Example 6: Sample code for `switchClass`

5.4.5 iterateClass()

5.4.5.1 Description

Given a series of class names having a common prefix, the **iterateClass** method either increments or decrements the specified prefix and applies it to the target node. For example, if the class name is **frame5** and you want to increment, **iterateClass** removes the **frame5** class from the target node and applies the **frame6** class. Optionally, if the current class is at the lower or upper bound of the series, this method changes it to the other boundary.

This is a synchronous method.

5.4.5.2 Syntax

```
<static> mwl.iterateClass(targetNode, classPrefix, direction, numClasses, loop,
                          boundaryAction)
```

5.4.5.3 Parameters

The following table describes the parameters used in an **iterateClass** call.

Parameter	Type	Description
targetNode	String	This selector identifies the node to change.
classPrefix	String	The prefix of the class to iterate over.
direction	String	Use next to increment the class number or prev to decrement it.
numClasses	Number	The total number of classes in the series.
loop	Boolean	If true, calling this method when the current class is the upper or lower bound results in looping back to the other end. If false, this looping does not occur.
boundaryAction	String	This is the method to call if the increment or decrement operation sets the class prefix to the upper or lower bound. If you do not want to specify a method, use <code>""</code> . Also please see an important note below this table.

Table 10: **iterateClass** parameters



Note: If the **boundaryAction** parameter specifies a non-MWL method and the **iterateClass** call is part of a series of method calls being attached to an event handler, the **iterateClass** call must be the last of the series. See the following example code snippets, in which a non-MWL method called **example** is passed as the **boundaryAction** (other parameters are omitted for simplicity).

```
onclick="mwl.toggle(..);mwl.show(..);mwl.iterateClass(..,example());" // Works  
onclick="mwl.toggle(..);mwl.iterateClass(..,example());mwl.show(..);" // Doesn't work
```

Example 7: Examples of **iterateClass** calls

5.4.5.4 Return value

The **iterateClass** method does not return a value.

5.4.5.5 Example code

The following HTML uses `iterateClass` to display a series of several graphics in a container:

```
<style>
  body { padding:5px }
  .container { width:225px; height:140px; overflow:hidden; border:solid; }
  #images { width:675px; -webkit-transition:margin-left 0.5s linear; }
  .im0 { margin-left:0px; }
  .im1 { margin-left:-225px; }
  .im2 { margin-left:-450px; }
</style>

<div id="prev" onclick="mwl.iterateClass('#images','im','prev','3','true',
  "");">Previous image</div>
<div id="next" onclick="mwl.iterateClass('#images','im','next','3','true',
  "");">Next image</div>
<div class="container">
  <div class="im1" id="images">
    <table cellpadding="0" cellspacing="0" border="0">
      <tr>
        <td id="t2"></td>
        <td id="t1"></td>
        <td id="t3"></td>
      </tr>
    </table>
  </div>
</div>
```

Example 8: Sample code for `iterateClass`

5.4.6 setGroupTarget()

5.4.6.1 Description

The **setGroupTarget** method targets one node in a group to have a distinct class from the rest of its siblings. Given a group of nodes contained within an element identified by a *groupName* selector, this method applies the *targetClass* to the target node. Any sibling nodes that had the *targetClass* will replace this with the *defaultClass*. This method is useful for implementing UI controls like tab pages and accordions.

This is a synchronous method.

5.4.6.2 Syntax

```
<static> mwl.setGroupTarget(groupNode, targetNode, targetClass, defaultClass)
```

5.4.6.3 Parameters

The following table describes the parameters used in a **setGroupTarget** call.

Parameter	Type	Description
groupName	String	This is the selector of the group to be changed.
targetNode	String	The selector of the sibling element to add a class.
targetClass	String	The class to apply to the <i>targetNode</i> . The method removes this class from all other sibling nodes.
defaultClass	String	The class to apply to all other sibling nodes. This can be a class name or the prefix of a class name when appended with '*'.

Table 11: **setGroupTarget** parameters

5.4.6.4 Return value

The **setGroupTarget** method does not return a value.

5.4.6.5 Example code

The following code sample shows a very simple accordion page that uses `setGroupTarget`.

```

<style type="text/css">
  .fold {
    width:200px;
    background-color:#FF0000;
    overflow:hidden;
  }
  .hide {
    height:0px;
  }
  .show {
    height:200px;
  }
</style>

<div id="accord1">
  <div><a onclick="mwl.setGroupTarget('#accord1','#fold1','show','hide');return false;"
    href="#">Fold 1</a></div>
    <div id="fold1" class="show fold">
      This is the content of the first fold
    </div>
  <div><a onclick="mwl.setGroupTarget('#accord1','#fold2','show','hide');return false;"
    href="#">Fold 2</a></div>
    <div id="fold2" class="hide fold">
      This is the content of the second fold
    </div>
  <div><a onclick="mwl.setGroupTarget('#accord1','#fold3','show','hide');return false;"
    href="#">Fold 3</a></div>
    <div id="fold3" class="hide fold">
      This is the content of the third fold
    </div>
</div>

```

Example 9: Sample code for `setGroupTarget`

5.4.7 setGroupNext()

5.4.7.1 Description

You can use the **setGroupNext** method to iterate to the next or previous element in a given block. This method replaces the *targetClass* with the *defaultClass* on the current node and applies the *targetClass* (while removing the *defaultClass*) to the next or previous sibling node. When the method reaches the end of the list of siblings, it cycles back to the beginning of the list.

This is a synchronous method.

5.4.7.2 Syntax

```
<static> mwl.setGroupNext(groupNode, targetClass, defaultClass, direction)
```

5.4.7.3 Parameters

The following table describes the parameters used in a **setGroupNext** call.

Parameter	Type	Description
groupNode	String	This is the ID of the group to change.
targetClass	String	The class to apply to the new current node.
defaultClass	String	The class to apply to all sibling nodes other than the new current node.
direction	String	The direction to move in the list of sibling nodes. This can be either "next" or "prev". If the direction is some other value, the browser ignores the whole command.

Table 12: **setGroupNext** parameters

5.4.7.4 Return value

The **setGroupNext** method does not return a value.

5.4.7.5 Example code

The following code sample demonstrates several MWL methods – in addition to **setGroupNext**, it uses **toggleClass()** and **setInputValue()**. The code displays a graphical “switch” containing two small rectangular

regions, one of which is grey. Clicking on the grey area causes different colours to scroll into the rectangular regions. A display field below shows whether the switch is in an ON or OFF state.

```

<style>
  body {padding:5px;}
  .container {width:100px; height:50px; border: solid; overflow:hidden;}
  .sw_sz {width:100px; height:50px;background-color:#00FF00;
    -webkit-transition:margin-left 0.5s linear;}
  .sw_off {margin-left:+50px;}
  .sw_on {margin-left:0px;}
  .hide {background-color:#aaaaaa; width:50px; height:50px;
display:none;}
  .show {background-color:#aaaaaa; width:50px; height:50px;
display:block;}
</style>

<div id="sw_bg" class="container">
  <div class="sw_sz sw_off" id="slide">
    <div class="show" id="is_off"
onclick="mwl.setInputValue('#sw_state','ON');
        mwl.toggleClass('#sw_bg','cont_on');
        mwl.switchClass('#slide','sw_off','sw_on');
        mwl.setGroupNext('#slide','show','hide','next');">
      </div>
    <div class="hide" id="is_on" onclick="mwl.setInputValue('#sw_state',
'OFF');
        mwl.toggleClass('#sw_bg','cont_on');
        mwl.switchClass('#slide','sw_on','sw_off');
        mwl.setGroupNext('#slide','show','hide','next');">
      </div>
    </div>
  </div>
</div>

<div>Current state: <input type="text" id="sw_state" size="20" value="OFF"></div>

```

Example 10: Sample code for `setGroupNext`

5.4.8 show()

5.4.8.1 Description

The **show** method uses the CSS **display** property to make an element visible on the page. Regardless of the current value, the display property changes to 'block', which makes the element visible. If this method is invoked while the element is already in the shown state, the browser ignores the command.

This is a synchronous method.

5.4.8.2 Syntax

```
<static> mwl.show(targetNode)
```

5.4.8.3 Parameters

The following table describes the parameters used in a **show** call.

Parameter	Type	Description
targetNode	String	This is the selector of the element to show.

Table 13: **show** parameters

5.4.8.4 Return value

The **show** method does not return a value.

5.4.8.5 Example code

In the following code sample, additional content appears when the user clicks on a link.

```
<div id="show_test">
  <a onclick="mwl.show('#bookmarks_sample');return false;" href="#">
    Click here to show more content
  </a>
  <div class="MWLGroupItem" id="bookmarks_sample" style="display: none">
    <ul>
      <li><a href="http://www.nokia.com">Nokia</a>
      <li><a href="http://www.facebook.com">Facebook</a>
      <li><a href="http://www.twitter.com">Twitter</a>
    </ul>
  </div>
</div>
```

Example 11: Sample code for [show](#)

5.4.9 hide()

5.4.9.1 Description

The **hide** method uses the CSS **display** property to hide an element on the page. Regardless of the current value, the display property changes to 'none', which makes the element invisible. If this method is invoked while the element is already in the hidden state, the browser ignores the command.

This is a synchronous method.

5.4.9.2 Syntax

```
<static> mwl.hide(targetNode)
```

5.4.9.3 Parameters

The following table describes the parameters used in a **hide** call.

Parameter	Type	Description
targetNode	String	This is the selector of the element to hide.

Table 14: **hide** parameters

5.4.9.4 Return value

The **hide** method does not return a value.

5.4.9.5 Example code

In the following code sample, additional content disappears when the user clicks on a link.

```
<div id="hide_test">
  <a onclick="mwl.hide('#bookmarks');return false;" href="#">
    Click here to hide content
  </a>
  <div class="MWLGroupItem" id="bookmarks" style="display: block">
    <ul>
      <li><a href="http://www.nokia.com">Nokia</a>
      <li><a href="http://www.facebook.com">Facebook</a>
      <li><a href="http://www.twitter.com">Twitter</a>
    </ul>
  </div>
</div>
```

Example 12: Sample code for **hide**

5.4.10 toggle()

5.4.10.1 Description

The **toggle** method controls the visibility of the element with *targetNode*, changing its CSS display property to “none” if it is currently “block” and to “block” if it is any other value. (If the display property is “block”, the element is visible. Otherwise, the element is hidden.)

This is a synchronous method.

5.4.10.2 Syntax

```
<static> mwl.toggle(targetNode)
```

5.4.10.3 Parameters

The following table describes the parameters used in a **toggle** call.

Parameter	Type	Description
targetNode	String	This is the selector of the element to toggle.

Table 15: **toggle** parameters

5.4.10.4 Return value

The **toggle** method does not return a value.

5.4.10.5 Example code

In the following code sample, additional content alternately appears and disappears when the user clicks on a link.


```
<div id="toggle_test">
  <a id="bm_open_foc" onclick="mwl.toggle('#bookmarks');return false;" href="#">
    Click here to toggle content
  </a>
  <div class="MWLGroupItem" id="bookmarks" style="display: none">
    <ul>
      <li><a href="http://www.nokia.com">Nokia</a>
      <li><a href="http://www.facebook.com">Facebook</a>
      <li><a href="http://www.twitter.com">Twitter</a>
    </ul>
  </div>
</div>
```

Example 13: Sample code for **toggle**

5.4.11 scrollTo()

5.4.11.1 Description

The **scrollTo** method scrolls the browser to the block with the specified ID. Depending on the browser capabilities, the scrolling action may be smooth or a direct jump. For phones that can zoom, this also zooms in to the scroll area. If the target element is already fully in the display screen, no action takes place. If scrolling does occur, the method places the element at the top left of the screen, with the focus set on it.



Note: If you want an event to trigger this method, attach it directly to the event rather than calling it from a subroutine attached to the event. See the following examples.

<i>Do this:</i>	<i>Do not do this:</i>
<pre><body> Scroll to bottom <div> Text filler
Text filler
 Text filler
Text filler
 </div> <div id="btm"></div></pre>	<pre>function myRoutine(){ mwl.scrollTo("#btm") } </script> </head> <body> Scroll to bottom <div> Text filler
Text filler
 Text filler
Text filler
 </div> <div id="btm"></div></pre>

This is a synchronous method.

5.4.11.2 Syntax

```
<static> mwl.scrollTo(targetNode)
```

5.4.11.3 Parameters

The following table describes the parameters used in a **scrollTo** call.

Parameter	Type	Description
targetNode	String	This is the selector of the block to scroll the display to.

Table 16: **scrollTo** parameters

5.4.11.4 Return value

The **scrollTo** method does not return a value.

5.4.12 loadURL()

5.4.12.1 Description

The **loadURL** method breaks out of the web application and loads a specified resource directly into the root window of Nokia Browser for Series 40.

This is a synchronous method.

5.4.12.2 Syntax

```
<static> mwl.loadURL(URLtoLoad)
```

5.4.12.3 Parameters

The following table describes the parameters used in a **loadURL** call.

Parameter	Type	Description
URLtoLoad	String	This is the URL to load.

Table 17: loadURL parameters

5.4.12.4 Return value

The **loadURL** method does not return a value.

5.4.12.5 Example code

The following code sample simply opens a new URL in the browser.

```
<div>
  <a id="bm_closed_foc" onclick="mwl.loadURL('http://www.google.com');
    return false;" href="#">
    Click here to load Google.com
  </a>
</div>
```

Example 15: Sample code for loadURL

5.4.13 setInputValue()

5.4.13.1 Description

The **setInputValue** method updates the value of a specified HTML input. The input field must be marked with an ID attribute. The method supports the following input field types: *text*, *radio*, *checkbox*, and *hidden*. For example, with a text field, you could use this method to display something like “Enter text here” in the box before the user clicks in the field.

This is a synchronous method.

5.4.13.2 Syntax

```
<static> mwl.setInputValue(targetNode, value)
```

5.4.13.3 Parameters

The following table describes the parameters used in a **setInputValue** call.

Parameter	Type	Description
targetNode	String	This is the ID of the input field.
value	String	The value to give to the input field.

Table 18: **setInputValue** parameters

5.4.13.4 Return value

The **setInputValue** method does not return a value.

5.4.13.5 Example code

The following code sample demonstrates several MWL methods – in addition to **setInputValue**, it uses **toggleClass()** and **setGroupNext()**. The code displays a graphical “switch” containing two small rectangular regions, one of which is grey. Clicking on the grey area causes different colours to scroll into the rectangular regions. A display field below shows whether the switch is in an ON or OFF state.

```
<style>
  body {padding:5px;}
  .container {width:100px; height:50px; border: solid; overflow:hidden;}
  .sw_sz {width:100px; height:50px;background-color:#00FF00;
    -webkit-transition:margin-left 0.5s linear;}
  .sw_off {margin-left:+50px;}
  .sw_on {margin-left:0px;}
  .hide {background-color:#aaaaaa; width:50px; height:50px;
display:none;}
  .show {background-color:#aaaaaa; width:50px; height:50px;
display:block;}
</style>

<div id="sw_bg" class="container">
  <div class="sw_sz sw_off" id="slide">
    <div class="show" id="is_off"
onclick="mwl.setInputValue('#sw_state','ON');
      mwl.toggleClass('#sw_bg','cont_on');
      mwl.switchClass('#slide','sw_off','sw_on');
      mwl.setGroupNext('#slide','show','hide','next');">
    </div>
    <div class="hide" id="is_on" onclick="mwl.setInputValue('#sw_state',
'OFF');
      mwl.toggleClass('#sw_bg','cont_on');
      mwl.switchClass('#slide','sw_on','sw_off');
      mwl.setGroupNext('#slide','show','hide','next');">
    </div> </div> </div>

<div>Current state: <input type="text" id="sw_state" size="20" value="OFF"></div>
```

Example 16: Sample code for **setInputValue**

5.4.14 timer()

5.4.14.1 Description

The **timer** method causes commands to be executed in response to a timer firing. The browser automatically stops timers when a page is unloaded or via the **stopTimer()** method (see page 47). If this method is invoked with the same label as an existing timer, the method stops the running timer and replaces it with the new one.



Note: The interval time is the time elapsed between the start of the firing of each event. It does not account for the time to process the *methodString* actions. To create a sequence that fires timers after the completion of each *methodString*, set *numberOfTimes* to 1 and then invoke a new timer as the last method in the *methodString*.



Note: If you want an event to trigger this method, attach it directly to the event rather than calling it from a subroutine attached to the event. See the following examples.

Do this:

```
function callFunc() {
// Process event here...
}
</script>

</head> <body onload="mwl.timer
('timer1', 5000, 0, 'callFunc()');">
```

Do not do this:

```
function callFunc() {
// Process event here...
}
</script>

function myRoutine(){
mwl.timer('timer1', 5000, 0,
'callFunc()')
}
</script>

</head> <body onload="myRoutine();">
```

This is a synchronous method.

5.4.14.2 Syntax

```
<static> mwl.timer(label, interval, numberOfTimes, methodString)
```

5.4.14.3 Parameters

The following table describes the parameters used in a **timer** call.

Parameter	Type	Description
label	String	This string identifies the timer.
interval	Number	The time that elapses before firing the timer. This is in milliseconds; the granularity is phone-dependent.
numberOfTimes	Number	The number of times to repeat the timer. Setting this to 0 results in the timer running with no upper bound, in which case it repeats as long as the browser renders the page.
methodString	String	This is a list of semicolon-separated JavaScript statements to execute when the timer fires. This can be any combination of MWL and non-MWL statements. With non-MWL code, remember that it runs on the proxy server rather than the phone, which causes a client-server communications lag.

Table 19: **timer** parameters

5.4.14.4 Return value

The **timer** method does not return a value.

5.4.14.5 Example code

The following code sample starts 3 timers: One that fires indefinitely every 3 seconds, a second that fires indefinitely every 5 seconds, and a third that fires just twice with a 2-second interval.

```
<div id="addClass_test">
  <a onclick="mwl.timer('timer1', 3000, 0, 'alert(\'timer1\')');
    mwl.timer('timer2', 5000, 0, 'alert(\'timer2\');return false;" href="#">
    Click to add two unlimited timers (timer1 3sec & timer2 5sec)
  </a>
</div>
<div>
  <a onclick="mwl.timer('timer3', 2000, 2, 'alert(\'timer3\');return false;"
    href="#">
    Click to add a timer that will run twice (timer3 2sec).
  </a>
</div>
```

Example 17: Sample code for **timer**

5.4.15 stopTimer()

5.4.15.1 Description

The **stopTimer** method stops running timers. If you invoke the method with no label (see the parameters below), it stops *all* timers associated with the document. If you invoke the method with the same label as an existing timer, the method stops the running timer. (The browser also stops timers when unloading a page.)



Note: If you want an event to trigger this method, attach it directly to the event rather than calling it from a subroutine attached to the event. See the following examples.

Do this:

```
<body>

// Code for a slideshow here...

<a href="#" onclick="mwl.stopTimer();" >
  Click here to stop slideshow</a>
```

Do not do this:

```
function myRoutine(){
  mwl.stopTimer()
}
</script>
</head> <body>

// Code for a slideshow here...

<a href="#" onclick="myRoutine();" >
  Click here to stop slideshow</a>
```

This is a synchronous method.

5.4.15.2 Syntax

```
<static> mwl.stopTimer([label])
```

5.4.15.3 Parameters

The following table describes the parameters used in a **stopTimer** call. *Parameters in brackets are optional.*

Parameter	Type	Description
[label]	String	This string identifies the timer. If the label is unrecognized, the browser ignores the command. If the parameter is omitted, the browser stops all timers associated with the page.

Table 20: **stopTimer** parameters

5.4.15.4 Return value

The **stopTimer** method does not return a value.

5.4.15.5 Example code

The following code sample starts two timers, then stops the first one, the second one, or both respectively when the user clicks various links.

```
<div id="addClass_test ">
  <a onclick="mwl.timer('timer1', 3000, 0, 'alert(\'timer1\')');
  mwl.timer('timer2', 5000, 0, 'alert(\'timer2\');return false;" href="#">
    Click to add two unlimited timers (timer1 3sec & timer2 5sec)
  </a>
</div>

<div id="clear">
  <a onclick="mwl.stopTimer('timer1');return false;" href="#">
    Click to stop timer1
  </a> <br>
  <a onclick="mwl.stopTimer('timer2');return false;" href="#">
    Click to stop timer2
  </a> <br>
  <a onclick="mwl.stopTimer();return false;" href="#">
    Click to stop all timers
  </a>
</div>
```

Example 18: Sample code for **stopTimer**

6. Geolocation API

Introduced in version 1.5 of Series 40 web app, support for the W3C geolocation API enables web apps to query a phone's location. This chapter provides information on the supported features of the W3C geolocation API and the location information a web app can acquire.

6.1 API feature support

Table 21 lists the W3C geolocation API objects, interfaces, and methods supported in Series 40 web apps 1.5.

Table 21: The support for objects/interfaces and methods of the W3C geolocation API are listed.

Object/interface	methods	support
Geolocation	<code>getCurrentPosition()</code>	Yes
	<code>watchPosition()</code>	No
	<code>clearWatch()</code>	No
PositionOptions	<code>enableHighAccuracy</code>	No
	<code>timeout</code>	Yes
	<code>maximumAge</code>	Yes
Position	Coordinates	Yes
	timestamp	Yes
Coordinates	latitude	Yes
	longitude	Yes
	altitude	Yes, but the values returned may not be meaningful.
	accuracy	Yes

Object/interface	methods	support
	altitudeAccuracy	No
	heading	No
	speed	No
PositionError	PERMISSION_DENIED	Yes
	POSITION_UNAVAILABLE	Yes
	TIMEOUT	Yes

6.2 Code Example

The code in Example 19 shows how the geolocation API can be used to acquire a phone's position.

```
function getPosition()
{
  if(navigator.geolocation){
    navigator.geolocation.getCurrentPosition(geoSuccess, geoFailure);
  }
  else{

// geolocation API is not available, handle the case here
  }

}
function geoSuccess(position){
/*
* handle success here
* use: position.coords.latitude, position.coords.longitude
*           position.coords.accuracy and position.timeStamp
*/
}

function geoFailure(error){
/*
* handle failure here
* use: error.code and error.message
*/
}
}
```

Example 19: Example of how to implement location acquisition using the geolocation API is shown.

6.3 Privacy

By default all web apps start by prompting the user to allow the web app to acquire the phone's position. The user then has the option to set the location acquisition on a per web app basis as:

- **always ask** if location information can be accessed.
- **always allow** access to location information.
- **never allow** access to location information.

For more information, see [Series 40 Web Apps UX Guidelines](#).

6.4 Location accuracy

Series 40 web apps can acquire a location on Series 40 phones using network-based positioning systems, the system used depends on the Series 40 platform version the phone is based on:

- The [Nokia C2-02 Touch and Type](#), [Nokia C2-03 Touch and Type](#), [Nokia C2-05 phone](#), [Nokia C2-06 Touch and Type](#), [Nokia C2-07 Touch and Type](#), [Nokia C2-08 Touch and Type](#), and [Nokia X2-05](#) phone offer a multi-cell (triangulation) positioning system. The accuracy of this system is dependent on the density of cell towers in the phone's location, the more towers the greater the accuracy. The accuracy of the position will therefore vary from a few 10s of meters, where cell tower density is high, to a few 100s of meters, where the density of cell towers is low.
- phones based on Series 40 5th Edition, Feature Pack 1 and earlier offer single-cell positioning. The accuracy of this method will depend on the coverage offered by a cell tower and can vary from 100s of meters in urban locations to 1000s of meters in rural locations.



Warning: The Cell id information to enable a Series 40 phone to determine its location may not be available in all areas or from all operators.

6.5 Related third-party APIs

While the Geolocation API provides your web apps with location details, you will probably want to use a third-party API to add mapping or similar information. In such cases, the use of APIs that associate all the requests to a developer API key, rather than per IP address, are recommended.

APIs that restrict the number of requests on a per IP address basis will probably consume any access limit quickly, as the requests from all instances of a Series 40 web app are routed through the Proxy server and therefore a single IP address.

7. About the widget object

This chapter describes the support for the W3C widget interface in Series 40 web apps.

The Nokia Browser for Series 40 runtime environment supports the W3C widget interface. This is exposed via the JavaScript widget object.



Note: Series 40 web apps deviate from some parts of the W3C **widget** standard as shown in the following table, and also as described in Chapter 7.1, ‘Other issues related to W3C widget standards’. (This refers to the W3C standards in effect as of this document’s publication date. See the W3C widget standards for more information.)

The JavaScript widget object has the following properties:

Table 22: Widget object properties and methods

Property/method	Description
author	This is the name of the web app author.
authorEmail	The e-mail address of the web app author.
authorHref	URL provided by the web app author.
description	This is a text description of the web app.
id	A unique web app identifier.
name	The full name of the web app. <i>Note:</i> There <i>must</i> be one or more name elements (this is a deviation from W3C widget standards).
shortName	A short name for the web app. <i>Note:</i> A shortName attribute <i>must</i> exist for each name element (this is a deviation from W3C widget standards).
preferences	This is name/value storage for the web app. For more information on this, please see <i>Preferences property</i> (page 53).
version	The web app version.
height	This returns the web app’s viewport height.

Property/method	Description
width	This returns the web app's viewport width.
openURL()	Opens the requested URL in the browser. <i>Note:</i> This occurs in the Nokia Browser for Series 40 Proxy rather than in the Nokia Browser for Series 40 Client, with no round-trip communication with the client.



Note: No phone APIs are supported in this version of the Series 40 web apps API. Also, this version of the Series 40 web apps API does not support home screen widgets. It does support localization in compliance with the W3C widget specification.

7.1 Other issues related to W3C widget standards

Series 40 web apps do *not* deviate from the following W3C widget standards:

- *Icons.* You do *not* need to specify an icon or include an icon file. If you do not package an icon with the web app, the system will insert a default icon (as the WDE does).
- *The content element.* A **start-file** must exist and it must be a default named **start-file** (e.g. index.html) AND/OR be specified by name in the content element `src` attribute.



Note: The start file name must be provided in lower case, mixed case or upper case file names are not recognised.

7.2 Preferences property

Please see <http://www.w3.org/TR/widgets-apis/#the-preferences-attribute> for a good example on how to use the **preferences** property.

7.3 Web app version

Web applications are identified by a **feature** tag stored in the **config.xml** file. Various web application consumers can use this tag to identify the type and version of the web application.



Note: You *must* provide one or more **feature** elements in your web app (this is a deviation from W3C widget standards).

The **feature** element format is:

```
<feature name="nokia://s40.nokia.com/webAppType/versionNumber" required="true"/>
```

Nokia Browser for Series 40 identifies itself with the type and version settings for the web applications it can support. `webAppType` then describes the minimum runtime version the browser needs to support to run the web application. The valid version numbers are 1.0 or 1.5.



Note: Web apps should be created to support the latest version of the browser/web apps environment, that is 1.5. If the user attempts to run a web app that requires a higher version than that supported by their client, the user will be prompted to update their browser client. If the user declines to update their client, the web app will not run.

8. Image caching

To enhance the performance of web apps, from web apps runtime 1.5 images within the `*.wgt` files are cached onto a phone when the web app is first used.



Note: Images from the internet are not cached onto the phone.

To ensure the best use of this image caching feature you need to ensure that:

1. all static images used in the web app are included in the `*.wgt` file.
2. there are no unused images included in the web app's `*.wgt` file. The proxy server sends all images in the `*.wgt` file for caching on the phone, so any unused images consume unnecessary bandwidth when they are cached and consume space on the device needlessly.
3. all images in the `*.wgt` file are scaled to the size used by the web app. If these images are scaled in the web app's code — for example by use of attributes (such as `width` and `height`) or by CSS value — they are scaled by the server and sent to the client as they are needed. This makes the cached version of the image redundant. Where the same image is needed at difference sizes in the web app, it's better to provide two versions of the image.

9. Terms and abbreviations

Term or abbreviation	Meaning
MWL	Mobile Web Library – a Nokia JavaScript library that runs in the phone client
Nokia web tools for Series 40	The programming environment used to make web apps for Series 40 (sometimes called Web Development Environment or WDE)
Nokia Browser for Series 40 Client (or simply client)	The browser, the MWL and the other software that runs on the phone, supported by the Nokia Browser for Series 40 Proxy
Nokia Browser for Series 40 Proxy server	The Nokia Browser for Series 40 component that doesn't run on the phone, which processes web content on behalf of the client
Proxy	A server that acts as an intermediary for requests from clients seeking resources from other servers
Web application	An application containing HTML, CSS, and JavaScript elements

Appendix A HTML support

Nokia Browser for Series 40 supports the HTML tags shown below.



Note: This list shows compatibility only. Some of the elements are not implemented as defined by W3C. The browser may convert some of them to an equivalent representation.

Also, some of these are supported but not recommended for Series 40 web apps. See the [Series 40 Web Apps Best Practices Guide](#) for more information on what Nokia recommends for this environment.

To reduce repetition, the HTML tag table (Table 24) refers to certain *groups* of attributes and properties, which are defined in Table 23.

Table 23: Common attribute and property groups

Attribute/property groups used in Table 24	Attributes and properties in group
<i>event-attributes</i>	onblur, onchange, onclick, onfocus, onswipeleft, onswiperight, onswipeup, onswipedown, navleft, navright, navup, navdown, longpress
<i>border-properties</i>	border-color, border-style, border-top-color, border-right-color, border-bottom-color, border-left-color, border-top-style, border-right-style, border-bottom-style, border-left-style, border-top-width, border-right-width, border-bottom-width, border-left-width, border-width
<i>margin-properties</i>	margin-top, margin-right, margin-bottom, margin-left, margin
<i>padding-properties</i>	padding-top, padding-right, padding-bottom, padding-left, padding
<i>transition-properties</i>	-webkit-transition-delay, -webkit-transition-duration, -webkit-transition-property, -webkit-transition-timing-function

Table 24 is the main HTML support table, showing the combinations of tags, attributes, and CSS properties that you can use.

Table 24: HTML tag support

Element	Supported HTML attributes	Supported CSS properties
a	<i>event-attributes</i> , href, id, name	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
area	alt, coords, href, id, shape	
b	class, id, lang, onclick, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
base	href	
bdo	<i>event-attributes</i> , id	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
blockquote	cite, class, dir, id, lang, onclick, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
body	onload, onresize, onunload	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, unicode-bidi, white-space, width
br		
button	accesskey, class, dir, disabled, id, lang, name, onblur, onclick, onfocus, style, tabindex, title, type, value	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , <i>padding-properties</i> , text-decoration, vertical-align, width

Element	Supported HTML attributes	Supported CSS properties
dir	class, dir, id, lang, onclick, style, title	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, list-style-type, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
div	<i>event-attributes</i> , id, class, style	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
em	class, dir, id, lang, onclick, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
font	class, color, dir, face, id, lang, size, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
form	action, <i>event-attributes</i> , id, method, name	background-color, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
h1, h2, h3, h4, h5, h6	align, class, dir, id, lang, onclick, style, title	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
head		
html		
i	class, id, lang, onclick, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space

Element	Supported HTML attributes	Supported CSS properties
img	alt, <i>event-attributes</i> , height, id, src, usemap, width	background-color, <i>border-properties</i> , direction, display, height, <i>margin-properties</i> , <i>padding-properties</i> , unicode-bidi, vertical-align, width
input	alt, checked, disabled, <i>event-attributes</i> , height (for input type image only), id, maxlength, name, readonly, size, src, type, usemap, value, width (for input type image only)	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , <i>padding-properties</i> , text-decoration, vertical-align, width
li	<i>event-attributes</i> , id, value (used for ordered list)	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, list-style-type, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
map	id, name	
menu	class, dir, id, lang, onclick, style, title	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, list-style-type, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
ol	<i>event-attributes</i> , id	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
option	disabled, id, readonly, selected, value	
p	align, class, dir, id, lang, onclick, style, title	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width
progress	id, name, value	

Element	Supported HTML attributes	Supported CSS properties
q	cite, class, dir, id, lang, onclick, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
select	disabled, <i>event-attributes</i> , id, multiple, readonly, size (if multiple is specified)	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , <i>padding-properties</i> , text-decoration, vertical-align, width
sms	phone number, action	n/a
span	<i>event-attributes</i> , id	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
table	border, cellpadding, cellspacing, <i>event-attributes</i> , id	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , direction, display, height, <i>margin-properties</i> , min-height, <i>padding-properties</i> , <i>transition-properties</i> , width
td	colspan, <i>event-attributes</i> , id, rowspan	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, unicode-bidi, vertical-align, white-space, width
textarea	cols, disabled, <i>event-attributes</i> , id, name, readonly, rows	background-color, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, <i>margin-properties</i> , <i>padding-properties</i> , text-decoration, vertical-align, width
th	abbr, align, axis, bgcolor, char, charoff, class, colspan, dir, headers, height, id, lang, nowrap, onclick, rowspan, scope, style, title, valign, width	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, font-size, font-style, font-weight, height, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, unicode-bidi, vertical-align, white-space, width
title		
tr	id	

Element	Supported HTML attributes	Supported CSS properties
u	class, id, lang, onclick, style, title	background-color, color, direction, display, font-size, font-style, font-weight, text-decoration, unicode-bidi, vertical-align, white-space
ul	<i>event-attributes</i> , id	background-color, background-image, background-position, background-repeat, <i>border-properties</i> , color, direction, display, font-size, font-style, font-weight, height, list-style-type, <i>margin-properties</i> , min-height, overflow, <i>padding-properties</i> , text-align, text-decoration, text-overflow-mode, <i>transition-properties</i> , unicode-bidi, white-space, width

Appendix B CSS support

Nokia Browser for Series 40 supports the following CSS properties and values.



Note: This list shows compatibility only. Some of the elements are not implemented as defined by W3C. The browser may convert some of them to an equivalent representation.

Also, some of these are supported but not recommended for Series 40 web apps. See the [Series 40 Web Apps Best Practices Guide](#) for more information on what Nokia recommends for this environment.

Table 25: CSS property support

CSS property	Values/units	Default
background <i>See following background elements</i>		
background-color	[<i>hex-color</i> transparent]	#FFFFFF (white) for <body>; gray gradient for <input type=button>, <input type=submit>, and <input type=reset> elements; #FFFFFF (white) for other active <input> elements; #E8E8E8 (gray) for other inactive <input> elements; transparent for other elements
background-image	<i>url</i>	None
background-position	[<i>x-pixels</i> <i>x-percentage</i>] [<i>y-pixels</i> <i>y-percentage</i>]	OPX
background-repeat	[repeat repeat-x repeat-y no-repeat]	repeat
border <i>See following border elements</i>		
border-color	[<i>hex-color</i> transparent]	#ADAFB0 (gray) if <input> border #000000 (black) for other borders
border-style	[none dotted dashed solid]	none

CSS property	Values/units	Default
border-top border-bottom border-left border-right	[border-width border-style 'border-top-color']	
border-top-color border-right-color border-bottom-color border-left-color	[<i>hex-color</i> transparent]	border-color , if specified otherwise: #ADAFB0 (gray) if <input> border #000000 (black) for other borders
border-top-style border-right-style border-bottom-style border-left-style	[none dotted dashed solid]	border-style , if specified otherwise: none
border-top-width border-right-width border-bottom-width border-left-width	<i>pixel-width</i>	border-width , if specified otherwise: 3PX
border-width	<i>pixel-width</i>	3PX
color	<i>hex-color</i>	#0000FF (blue) for text in links #000000 (black) for other text
direction	[ltr rtl]	ltr
display	[inline block none]	Depends on the HTML element. See HTML 4 default style sheet (http://www.w3.org/TR/CSS21/sample.html).
font <i>See following font elements</i>		
font-size	[small medium large]	small
font-style	[normal italic]	normal
font-weight	[normal bold]	normal

CSS property	Values/units	Default
height	[<i>pixel-height</i> <i>percent-height</i>]	auto
list-style <i>See following list-style elements</i>		
list-style-type	[disc circle square none]	disc
margin	[<i>pixel-width</i> <i>percent-width</i> auto]	0PX
margin-top margin-right margin-bottom margin-left	[<i>pixel-width</i> <i>percent-width</i> auto]	margin , if specified otherwise: 0PX
min-height	[<i>pixel-height</i> <i>percent-height</i>]	0PX
outline	[solid none]	solid
outline-style	[none solid dashed dotted]	none
overflow	[visible hidden]	visible
padding	[<i>pixel-width</i> <i>percent-width</i>]	0PX
padding-top padding-right padding-bottom padding-left	[<i>pixel-width</i> <i>percent-width</i>]	padding , if specified otherwise: 0PX
text-align	[left right center]	left if direction is LTR right if direction is RTL
text-decoration	[none underline]	underline for text in links none for other text

CSS property	Values/units	Default
text-overflow-mode	[clip ellipsis]	clip
text-transform	[capitalize uppercase lowercase none]	none
transition <i>See -webkit-transition below</i>		
transition-delay <i>See -webkit-transition-delay below</i>		
transition-duration <i>See -webkit-transition-duration below</i>		
transition-property <i>See -webkit-transition-property below</i>		
transition-timing-function <i>See -webkit-transition-timing-function below</i>		
unicode-bidi	[normal embed bidi-override]	normal
vertical-align	[baseline top middle bottom]	baseline
white-space	[normal nowrap]	normal
width	[<i>pixel-width</i> <i>percent-width</i>]	auto
-webkit-transition <i>See following -webkit-transition elements</i>		
-webkit-transition-delay	<i>delay-seconds</i>	0S
-webkit-transition-duration	<i>duration-seconds</i>	0S

CSS property	Values/units	Default
<code>-webkit-transition-property</code>	[width height margin-left margin-top none]	none
<code>-webkit-transition-timing-function</code>	linear	linear

Copyright © 2011 Nokia Corporation. All rights reserved.

Nokia and Nokia Developer are trademarks or registered trademarks of Nokia Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other product and company names mentioned herein may be trademarks or trade names of their respective owners.

Disclaimer

The information in this document is provided 'as is' with no warranties whatsoever, including any warranty of merchantability, fitness for any particular purpose, or any warranty otherwise arising out of any proposal, specification, or sample. This document is provided for informational purposes only.

Nokia Corporation retains the right to make changes to this document at any time, without notice.

Furthermore, information provided in this document is preliminary, and may be changed substantially prior to final release.

Licence

A licence is hereby granted to download and print a copy of this document for personal use only. No other licence to any other intellectual property rights is granted herein.