Opera Mini™

Client-side Content Folding

Opera Software ASA

Last revision July 21, 2009

1 Introduction

Opera Mini includes an extension for folding and unfolding content on the client-side. The main advantage of this feature is the ability to change the displayed content without the need of a server roundtrip which thus enhances the user experience.

The folding and unfolding can be triggered using the `mini:onClick` event or the `href` attribute, both restricted to anchor tags. It is possible to fold and unfold any content that can be put in a block level container like `<div>` - including links, images and nested folds. A block is marked as a fold segment if it has a `-o-mini-fold` property set in CSS. If a block is marked as folded or unfolded it is a fold segment. If it is marked as folded it is hidden in its initial state.

2 Syntax

The CSS property is called `-o-mini-fold` and accepts one of the following two values: `folded` and `unfolded`. It sets the initial state and is used to identify foldable elements. Property definition as per CSS1 (REC-CSS-20080411) syntax:

Name: -o-mini-fold
Value: none | folded | unfolded
Initial: none
Applies to: block-level elements
Inherited: no
Percentage values: N/A

The HTML id attribute for the block that defines the fold will be used to identify the fold. Thus the fold information is the y-coordinates for fold top and bottom (start and stop), initial fold state (hidden or shown) and its id. Folds are not allowed to overlap or have the same id, and the behaviour if that occurs is not defined.

To control the fold state a new URL protocol "`fold`" is introduced, which will list what fold elements on the page should change state and how. The elements can either be explicit id tokens, or global expressions matching several of the id tokens. As a special case an asterix can be used to indicate that all folds should be folded or unfolded. The URL will contain two sections, separated by a semicolon, the first being blocks to be folded and the second blocks to become unfolded.

Definition of fold protocol using ABNF (RFC 2234) where the the Name, NameChar and Letter products are from the XML specification (REC-xml-20060816):

```
FOLD_PROTOCOL = "fold:" HIDE ";" SHOW
HIDE          = ID_LIST
SHOW          = ID_LIST
ID_LIST       = ELEMENT *("," ELEMENT)
ELEMENT       = Name / 1*("*" / "?" / Letter / "_" / ":") *("*" / "?" / NameChar)
```

For the basic use of folds, to create the same effect as with folding link areas, a fold URL is used as the target for a normal HTML anchor tag (as an alternative to `href`, the `mini:onClick` attribute can be used, but is also restricted to anchor tags). The links are encoded as FOCUSDATA with CONTENTFOLDINGTARGET as TARGET.

The content (un)folding is applied to all matching ids that either have `-o-mini-fold` in their style attribute or in one of their CSS classes.

3 Examples

All of the following examples are stripped of their `<html>` and `<body>` tags as well as of their complete `<head>`. They assume the following CSS code (either in their `<head>` or in a linked file):

```
.folded { -o-mini-fold: folded; }
.unfolded { -o-mini-fold: unfolded; }
```

3.1 Basic Usage

This example demonstrates the folding mechanism by folding and unfolding a single line of text.

```
<p>
  The text "content"is foldable using the following links:<br />
  <a mini:onClick="fold:sample;">Fold</a><br />
  <a mini:onClick="fold:;sample">Unfold</a>
</p>
<div id="sample" class="unfolded">content</div>
```

Note that when removing `class="unfolded"`, the example wouldn't work.

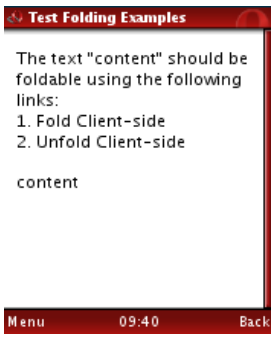

Figure 1: Unfolded text

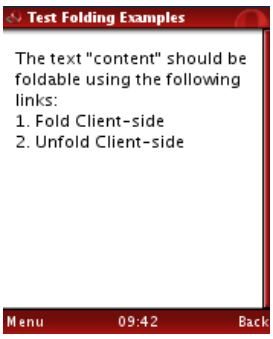

Figure 2: Folded text

3.2 Wildcards

This example illustrates the possibility of folding and unfolding more than one item at a time. In this case, the folded elements constitute some text and an image.

```
<p>
  Clicking the following link folds/ unfolds all foldables on the page.<br />
  <a mini:onClick="fold:*;">Fold</a><br />
  <a mini:onClick="fold:;*">Unfold</a>
</p>

<div id="text" class="unfolded" style="border:1px solid black;">some text</div>
<div id="image" class="unfolded"><img src="opera.png" alt="Opera" /></div>
```

Figure 3: Unfolded content

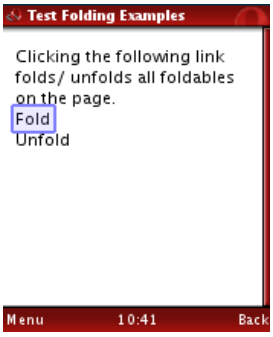

Figure 4: Folded content

3.3 Nested Folds

This example demonstrates the possibility of having a folding area inside another folding folding area.

```
<p>
  <a mini:onClick="fold:container;">Fold</a>|
  <a mini:onClick="fold:;container">Unfold</a>container
</p>

<div id="container" class="unfolded" style="border:1px solid; padding:0.5em;">
  <a mini:onClick="fold:img;">Fold</a>|
  <a mini:onClick="fold:;img">Unfold</a>image<br />
  <a mini:onClick="fold:text;">Fold</a>|
  <a mini:onClick="fold:;text">Unfold</a>text<br />
  <img id="img" class="unfolded" src="opera.png" alt="Opera Software" />
  <p id="text" class="unfolded" ><b>Opera Software</b></p>
</div>
```

Figure 5: Everything is unfolded

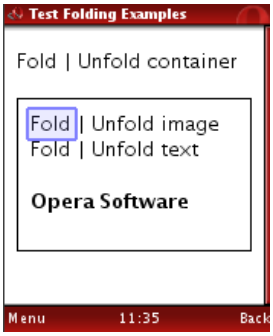

Figure 6: Nested image is folded

3.4 Folding multiple elements at a time

This final example shows that it is possible to (un)fold any number of elements on a single click.

```
<p>
  <a mini:onClick="fold:*;">Fold</a>|
```

```
  <a mini:onClick="fold:;*">Unfold</a>  all<br />
  <a mini:onClick="fold:i*;">Fold</a>|
  <a mini:onClick="fold:;i*">Unfold</a>  Items 1-3<br />
  <a mini:onClick="fold:i2,j;">Fold</a>|
  <a mini:onClick="fold:;i2,j">Unfold</a>  Items 2, j<br />
</p>

<div id="i1" class="unfolded">Item 1</div>
<div id="i2" class="unfolded">Item 2</div>
<div id="i3" class="unfolded">Item 3</div>
<div id="j" class="unfolded">Item j</div>
```

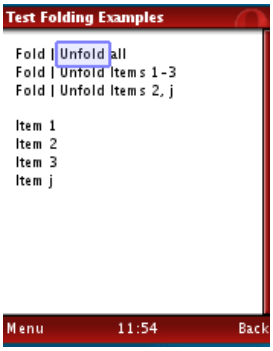

Figure 7: Everything is unfolded

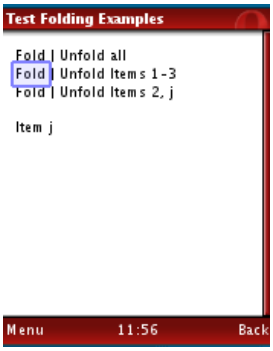

Figure 8: Items 1-3 are folded

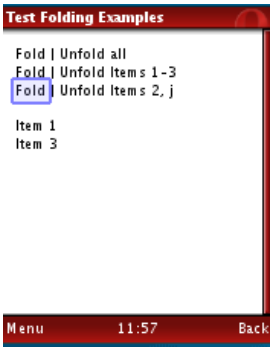

Figure 9: Items 2 and j are folded (after unfolding all)

3.5 Programmatically Set mini:onclick

It is also possible to programmatically add and remove the trigger for content folding using JavaScript. The following example has the same result as example 3.1 on page 2.

```
<script type="text/javascript">
window.addEventListener("load", init, false);
function init () {
  l1 = document.getElementById("l1");
  l2 = document.getElementById("l2");
  l1.setAttribute("mini:onClick","fold:sample;");
  l2.setAttribute("mini:onClick","fold:;sample");
}
</script>
<p>
  The text "content"is foldable using the following links:<br />
  <a id="l1">Fold</a><br />
  <a id="l2">Unfold</a>
</p>
<div id="sample" class="unfolded">content</div>
```

4 Example Application - Carousel

One of the many possible applications of this extension is a simple carousel-like menu. The menubar on top of the page can be used to open the connected entries and like when switching open tabs, no page load is required. The arrows allow to change the visible options and overflow at the end, effectively creating a carousel. See appendix A on page 6 for a listing of the PHP code that makes up this example.

Figure 10: A "carousel" menu

Figure 11: Another tab is visible

The trick is to simply create the menu once for each possible state and then fold all but one of those menus. The same applies for the content.

5 Limitations

Due to the complexity in implementing a client/server folding mechanism, certain basic limitations for the markup apply. Generally, problems can be avoided by not using the CSS property "overflow" inside of foldable containers and by using "padding" instead of "margin".

A PHP Code of the Sample Application

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN""http://www.w3.
org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"xml:lang="en">

<head>
  <meta content="application/xhtml+xml"/>
  <title>Example Application</title>
  <link rel="stylesheet"type="text/css"href="sample-app.css"/>
</head>

<body>

<?php
  // to extend this example, just add your entries to the array
  // and create a name.php file as well as an im/name.png icon
  $entries = array("news","translate","bookmarks","im","pictures","calendar",
  "settings");
  $last = sizeof($entries) - 1;

  $line = "\n";
  foreach ($entries as $i => $name) {
    $state = "folded";
    $go_right = $i + 1;
    $go_left = $i - 1;
    // dealing with menu overflows
    if ($i == 0) {
      $state = "unfolded";
      $go_left = $last;
    }
    if ($i <= $last-2) {
      $left = $name;
      $middle = $entries[$i+1];
      $right = $entries[$i+2];
    } elseif ($i == $last - 1) {
      $left = $name;
      $middle = $entries[$last];
      $right = $entries[0];
    } elseif ($i == $last) {
      $left = $name;
      $middle = $entries[0];
      $right = $entries[1];
      $go_right = 0;
    }
    // a tad hard to read, sorry
    echo '<div id="menu'.$i.'" class="'.$state.'  menu">'.$line;
```

```
    echo '  <a mini:onClick="fold:menu*;menu'.$go_left.'">'.$line;
    echo '    <img src="im/left.png" alt="left" /></a>'.$line;
    echo '  <a mini:onClick="fold:c*;c_'.$left.'">'.$line;
    echo '    <img src="im/'.$left.'.png" alt="'.$left.'" /></a>'.$line;
    echo '  <a mini:onClick="fold:c*;c_'.$middle.'">'.$line;
    echo '    <img src="im/'.$middle.'.png" alt="'.$middle.'" /></a>'.$line;
    echo '  <a mini:onClick="fold:c*;c_'.$right.'">'.$line;
    echo '    <img src="im/'.$right.'.png" alt="'.$right.'" /></a>'.$line;
    echo '  <a mini:onClick="fold:menu*;menu'.$go_right.'">'.$line;
    echo '    <img src="im/right.png" alt="right" /></a>'.$line;
    echo '</div>'.$line.$line;
  }

  foreach ($entries as $i => $name) {
    $state = "folded";
    if ($i == 0) {
      $state = "unfolded";
    }
    echo '<div id="c_'.$name.'" class="'.$state.'  content">'.$line;
    $fileName = $name.".php";
    $error = "The widget you chose is currently not available.";
    if (file_exists($fileName)) {
      include($fileName);
    } else {
      echo $error;
    }
    echo '</div>'.$line.$line;
  }
?>

</body>
</html>
```