QWebEditor v3.13

Developer Manual

http://www.qwebeditor.com/

Table of Content

License

When you purchased this product, QWebEditor, you agreed the terms of this agreement. This agreement is a legal contract between you and the developer of QWebEditor, Q-Surf Computing Solutions. Unless you received a different license agreement, installation or use of this software indicates your acceptance of the license and warranty limitation terms contained in this agreement. If you do not agree, you have to delete and remove the software from your system.

You receive one of the following license when you purchase this product:

Single Domain License - This license allows you to use QWebEditor for a web site. A web site is defined as content stored on a single domain name. A sub-domain is treated as another web site.

Single Server License - This license allows you to use QWebEditor for a single server. Unlimited number of web sites (and domains) under the same server are allowed to use QWebEditor.

Distribution License - A Distribution License allows you to distribute QWebEditor within a single commercial application. The License is valid for implementation ONLY by the Organization/Company/Licensee who had purchased the license.

License to Redistribute: Distributing the software and/or documentation with other products (commercial or otherwise) by any means without prior written permission from Q-Surf Computing Solutions is forbidden (except with distribution license). All rights to the QWebEditor software and documentation not expressly granted under this Agreement are reserved to 'Q-Surf Computing Solutions'. You are allowed to modify QWebEditor source code (except "license.js") to suit your applications. You can use the modified QWebEditor to your web site but the modified version of QWebEditor is still the property of Q-Surf Computing Solutions. You are not allowed to modify license.js in any circumstances. All copyrights in the source codes must be retained.

Disclaimer of Warranty: THIS SOFTWARE AND ACCOMPANYING DOCUMENTATION ARE PROVIDED "AS IS" AND WITHOUT WARRANTIES AS TO PERFORMANCE OF MERCHANTABILITY OR ANY OTHER WARRANTIES WHETHER EXPRESSED OR IMPLIED. BECAUSE OF THE VARIOUS HARDWARE AND SOFTWARE ENVIRONMENTS INTO WHICH QWebEditor MAY BE USED, NO WARRANTY OF FITNESS FOR A PARTICULAR PURPOSE IS OFFERED. THE USER MUST ASSUME THE ENTIRE RISK OF USING THIS PROGRAM. ANY LIABILITY OF QWebEditor WILL BE LIMITED EXCLUSIVELY TO PRODUCT REPLACEMENT OR REFUND OF PURCHASE PRICE. IN NO CASE SHALL QWebEditor BE LIABLE FOR ANY INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES OR LOSS, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR THE INABILITY TO USE EQUIPMENT OR ACCESS DATA, WHETHER SUCH DAMAGES ARE BASED UPON A BREACH OF EXPRESS OR IMPLIED WARRANTIES, BREACH OF CONTRACT, NEGLIGENCE, STRICT TORT, OR ANY OTHER LEGAL THEORY. THIS IS TRUE EVEN IF QWebEditor IS ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN NO CASE WILL QWebEditor OR LIABILITY EXCEED THE AMOUNT OF THE LICENSE FEE ACTUALLY PAID BY LICENSEE TO QWebEditor.

Introduction

QWebEditor brings the power of word processor to browsers. It is an online WYSIWYG HTML editor and perfect for content management system or any web sites require asking users to enter formatted text. It uses JavaScript and DHTML only and it is easy to be integrated into your web sites, no matter which web servers you are using. A PHP and ASP helper library is provided too to help you quickly incorporate QWebEditor.

History

2005-10-25 (version 3.13)

Enhancements:

- New Safari support (not all features available)
- New paste from Word and paste plain text features
- Now submitted content automatically convert characters only Windows character set to characters available in iso8859-1 character set
- New user manual (access through clicking the "About" button)
- Improved theming support and new sample themes

Bugs Fix:

- Context menu works reliably now under Firefox

2005-09-26 (version 3.12)

Enhancements:

- New cell merge function for HTML table (right click on a table cell to access the feature)
- New api to create custom toolbar buttons (refer to testeditor6.* for implementation sample)
- New increase/decrease font size function
- Updated javascript API and configuration allows greater degree of customization and easier usage of editor
- Paragraph drop down list box can now be customized and class attribute can be specified for each paragraph style
- The editor now takes multiple stylesheet (pass an array of css file name instead of a single css file to original api)
- Now properties box will not be blocked by popup up blocker when double click on image or hyperlink under IE
- Initialization of the editor is much faster now, even with multiple editors on a single web page.

Bugs Fix:

- Now <base href> should work reliably across different browsers

2005-04-11 (version 3.11b)

Bugs Fix:

- More HTML source editing fixed. JavaScript editing should be handled better
- Fixed a javascript error when external css file cannot be loaded
- Fixed HtmlEditIsLoaded() breakage introduced in 3.11
- Fixed a problem related to inserting hyperlink with target
- Fixed the editor to use "old" tags instead of css for formatting under Gecko based browsers. It was broken since 3.11.

2005-04-03 (version 3.11a)

Bugs Fix:

- Improve compatibility of old image dialog. Problems are caused by new insert image code.
- Improve insert html code used by Firefox during page edit mode.
- Fix javascript error in browse image dialog box under certain file name.

2005-03-28 (version 3.11)

Enhancements:

- Better page editing enhancements:
- load multiple style sheets properly into editor used by the page
- set up base href specified by the page
- preserve some meta tag in the page
- <base href> is better supported now.
- Both PHP and ASP image browser scripts now allow uploading non-image files. The editor can now correctly add <img> tag for image file, <embed> tag for flash file and <a href> for other files.
- Added javascript function "HtmlEditIsModified()" to query whether the content of the editor is changed. Please refer to cms examples on how to use it.
- "Remove All Formatting" now removes class attribute too
- CMS example has been enhanced to use several new features in this release
- Several timeout values are configurable now
- Updated tool bar icons

Bugs Fix:

- Fixed the problem that content of second editor got erased under Gecko based browsers under Mac OSX.
- Changed the drop down list box to have fixed width. The change overcame problems of Gecko based browsers that does not handle dynamic resizing well.

2005-02-06 (version 3.10)

Enhancements:

- Much better table operations handling especially for table with merged cells
- Add "Remove All Formats" operation that cleanly remove all span tags and style attributes for the whole document

Bugs fix:

- Fixed an occasionally javascript error when inserting and deleting a column or row
- Removed workaround code for Gecko initialization
- Fixed Gecko source editing problems
- in source mode, spaces in front of each line will be converted to " " after changed back to edit mode
- in source mode, "abcd[NEWLINE]efgh" will be converted to "abcdefgh" instead of "abcd efgh" after changed back to edit mode

2004-01-27 (version 3.09)

Bugs fix:

- Reenable background color function for Gecko based browser.
- Fixed a problem of setting style sheet for Gecko based browser and the style sheet is not available.

2005-01-09 (version 3.08)

Enhancements:

- Now able to use "Page Properties" to set margins.
- New JavaScript function HtmlEditIsLoaded() to query whether the editor is fully initialized (remote page and css) or not.

Bugs fix:

- Problem of removing table column under certain situation.
- Problem of unsetting certain HTML attributes after setting the attribute to empty in popup dialog box

2004-12-20 (version 3.07)
Enhancements:
- Optimized editor initialization

Bugs fix:
- Right click and double click on edit area for Gecko based browsers under Linux
- Right click and double click on edit area for MSIE 5.5
- Fixed couple problems related page editing
    - Current bgcolor, alink, vlink is not set properly in page properties dialog box
    - Removed page properties menu item in context menu during source editing mode
    - Reset body text color to black in source editing mode (in case text color is overridden)
- Background color in all popup dialog box uses theme color now
- Background color of toolbar drop down list boxes is changed to use window color
- Changed some buttons to traditional looks
- Fixed occasional crash when closing a popup dialog

2004-11-15 (version 3.06)
New features:
- Added JavaScript method to override symbol menu items.
- Editor now set width and height attribute with browseimages.php
- Added a new option to tell editor to generate safe HTML (skipped script tag and on** handlers) to avoid XSS security problem.

Bugs fix:
- Fixed the problem under firefox that the editor may be resized if toolbar dropdown listbox item is changed through javascript.
- Fixed the problem that content of one of the editors disappeared in a page containing multiple editors displayed in Netscape 7.2 for MacOSX.
- Fixed the problem that "undefined" may be set to image attributes

2004-08-28 (version 3.05)
New features:
- Added double clicking on hyperlinks and images to popup properties boxes
- Fixed minor bug for XHTML source generation code under Mozilla
- If XHTML source output is enabled, all script elements are removed automatically (to avoid XSS security problem).
- Table and Cell properites popup now use faster color picker code (instead of using another SLOW popup).
- Minor style changes to all dialog boxes.
- Updated browseimages.php to detect GD library capabilities automatically. ie. less setup to use browseimages.php
- Now allowed to use new browseimages2.php as the default image properties dialog (avoid popping up another dialog)
- Added optional callback after editor is completely initialized.

Bugs fix:
- Fixed IE resize code that is used for the editor in popup window
- Fixed two access violation bugs related to load page from url function

- Fixed error that right click on nested object in HTML page wont popup the proper dialog (eg. right click on an unordered list in a table wont popup the table properties dialog box)

2004-05-04 (version 3.04)

- Added ASP version of Browse Images script
- QWebEditor now automatically convert "www.something" to "http://www.something" if entered in link dialog box.
- Fixed a minor bug in browseimages.php script
- Fixed the bug that QWebEditor treated MSIE for Mac as one of the supported browser under JavaScript interface.

2004-03-22 (version 3.03)

- New theme support for QWebEditor to customize it to your site look and feel.
- Added functions/flags to enable and disable almost all buttons in the toolbar (except the about box).
- Fixed problems in PHP and ASP browser sniffer code that detects Netscape 7.
- XHTML output and using DIV tag under IE are changed from default to optional features.
- Changed to reset the cursor position to beginning of document after clicking the HTML source button under IE.

2004-02-15 (Version 3.02)

- Fixed the problem of XHTML generator if source has invalid structure (closing HTML tag without corresponding opening tag)
- Version 3.00 introduced using tag instead of <p> tag. Editor showed with one line spacing but browser displays nothing for the tags. This version changed all to   .
- Added delete image function and various UI improvements to browseimages.php script
- Fixed some spelling mistakes in English string resource.
- Translated some text in Chinese string resource.

2004-02-15 (Version 3.02)

- Fixed the problem of XHTML generator if source has invalid structure (closing HTML tag without corresponding opening tag)
- Version 3.00 introduced using tag instead of <p> tag. Editor showed with one line spacing but browser displays nothing for the tags. This version changed all to   .
- Added delete image function and various UI improvements to browseimages.php script
- Fixed some spelling mistakes in English string resource.
- Translated some text in Chinese string resource.

2004-02-08 (Version 3.01)

- Require new license.js
- Added to display error message in case popup blocker blocks creation of popup dialog
- HTML source output is XHTML compatible now by adding function to trace the DOM tree

- Under IE, sections are added inside <td> section if a new table is added. This change force IE to use instead of <p> tag and avoid the extra linefeed after a paragraph.

2004-01-27 (Version 3.00)

- Moved remaining text to language files.
- Included user contributed spanish language file (Provided by Cesar Sirvent. Thanks!)
- Changed browseimages.php to detect images URI automatically
- Return complete URL to the image (Mozilla needs it)

2004-01-22 (Version 3.00pre4.2)

- Completed developer document
- Setting editor margins in JavaScript interface
- potential lockup when 2 editors are used due to code changes
- better handling to force editor to use tag
- hide hilite color button under mozilla.
- Remove calling any command during initialization. As a result, no editor will receive focus unintentionally.

2004-01-19 (Version 3.00pre4.1)

- Fixed PHP interface that allows to use multple QWebEditor PHP object to create several editors.
- Fixed minor problem in PHP interface related to displaying previous entered value if default content is empty.
- Fixed QWebEditor from displaying "displaying insecure items in secure page" error message under IE.
- Force QWebEditor to use format if editor content is empty

2004-01-18 (Version 3.00pre4)

- Examples have been updated for new features. New Simple CMS examples for ASP and PHP interfaces are added too.
- QWebEditor now uses tag instead of <p> if possible to avoid extra spacing after a paragraph under IE.
- QWebEditor can now displayed previously entered content when using back button to revisit the same page with the editor (not working if content is set by URL)
- Fixed minor bugs in htmledit.php
    - correct browser sniffing code for Netscape
    - force an temporary variable to be global
- Changed the help icon to Q-Surf icon to avoid confusion to end user (it displays license information anyway)
- Modified font size dialog box to display "reference" point size
- Added Edit Page mode. By default, QWebEditor edit HTML code fragment rather than HTML page. By enabling edit page mode, option will be provided to edit various page properties (eg. title, text color) and return full page data instead.
- Added setting editor content by URL
- Added support of using <label for=editorid accesskey="k"> tag for editor focusing (checkout html_examples/testeditor03.html)
- Added tabbed indexing (checkout html_examples/testeditor03.html)

2003-12-31 (Version 3.00pre3)

- Added quite a few useful word processing/usability features:
    - Better context menu handling (Now trace back in DOM tree instead of using the clicked element.)
    - Added "Delete Table" in context menu
    - Added "list properties" in context menu
    - Added table alignment option in table dialog box
    - Mozilla can now display URL in hyperlink properties
    - Added back code to get around Mozilla problem. The problem happens when QWebEditor first loaded under Mozilla, backspace and delete wont work until any other key are pressed. Additional code is added to get around the problem (only work for Mozilla 1.5+).
- Added an additional flag to determine whether to display plaintext or HTML code in textarea box if the client's browser does not support QWebEditor.
- HTML source listing has improved formatting now.
- Changed default list of fonts if font enumeration is not used.
- Fixed HTML source formatting. Linefeed are now properly applied to some closing tags.

2003-12-10 (Version 3.00pre2)

- Slightly reduced code size
- Some codes moved to license.js
- Style dropdown list now show currently applied style
- Fixed htmledit_ta.js (included when JavaScript interface determined that client does not support QWebEditor) to support newer JavaScript interface.
- Fixed IE6 system fonts enumeration caused by initialization optimization

2003-11-20 (Version 3.00pre1)

- Replaced toolbar icons with new icons set.
- Slightly optimized initialization.
- Drop down list box for applying styles with class name started with "."
- Load editor content by URL

2003-10-23 (Version 2.05)

- Support setting external CSS file for editor
- New JavaScript interface for creating editor to avoid creating new function for new feature in the future
- Updated PHP and ASP interface to use the new JavaScript interface and new CSS feature
- Changed some arguments (strFormName) from required to optional

2003-10-20 (Version 2.04)

- Changed to use new command available in Mozilla 1.5 or later to implement insert html function. (better table, image, symbol insertion for Gecko based browsers).
- Changed to use toolbar button state to display view source status instead of using status bar
- Added implementation to view table border outline. The option can be toggle on/off by users and default is on.
- If "strElement" is the element name of QWebEditor, an additional "strElement_bHtmlEdit" is specified with a value of "1" when the browser supports QWebEditor.

2003-10-06 (Version 2.03)

- Added HTML and PHP examples for image browsing feature (an additional browse button is displayed next to the image URL text box in the image dialog box)
- Changed PHP examples to use image-browsing feature
- Fixed a minor bug in popup dialog implementation (caused by last optimization)

2003-10-04 (Version 2.02)

- Double number of colors in color selection
- Remove some codes for inserting codes for Gecko Mida (not working properly). The code is originally for forcing the cursor to locate after the inserted data.
- Fixed some translation in Chinese string resources
- Changed to use the same rich text editor to edit source rather than the old (and slow) popup window.
- Added "Html Source" and "Insert/Overwrite" indicators in status bar
- Fixed IE focus problems after clicking a command in toolbar

2003-09-30 (Version 2.01)

- Fixed bugs in image properties
- Fixed bugs in cell properties under Gecko Midas
- Changed dlgfrm.html frame scrolling attribute to auto (for long browsing images page)
- Minor changes of outlook in image properties box

2003-09-29 (Version 2.0)

- Supports Mozilla Midas, available in Netscape 7.1, Mozilla 1.5 and Mozilla Firebird since 20030924
- Added ASP helper class
- JavaScript version tried to detect browser version and generates backup codes if the browser does not support QWebEditor
- ASP and PHP helper class provides better browser detection and generates better backup codes.
- Insert Symbol drop down list box is now configurable.
- Better server sided and client sided codes to detect browser configuration
- Faster popup windows for foreground windows and background windows color

2003-08-18

- Changed testeditorXX.html examples to use flag names instead of a numeric value.
- Fixed mistakes in documentation

2003-07-14

- Added missing width and height attributes for couple buttons.

2003-07-06

- Fixed the border problem
- Much much faster initialization (changed to tracing children nodes to attach btn events instead of using 'all' collection)

2003-06-26

- Fixed table cell valign processing (used wrong property "valign" in code. should be "vAlign")
- Disable help button

2003-06-01 (Version 1.0)

Requirements

Browser requirements

- Windows version of Internet Explorer 5.5 or later, Netscape 7.1 or later, Mozilla 1.4 or later, Mozilla Firebird 20030924 or later, or other updated Gecko based browsers.
- JavaScript must be enabled.

Server requirements

None. QWebEditor uses JavaScript and DHTML only. However, a PHP library and an ASP library are provided for simpler integration to your existing system. To use all features provided by QWebEditor:

PHP

- At least PHP4 under different OSes
- Able to set directory permission (to allow image browser script to manage uploaded images)
- With GD extension enabled and access to system temp directory (to manage uploaded images and generation of thumbnails)

ASP

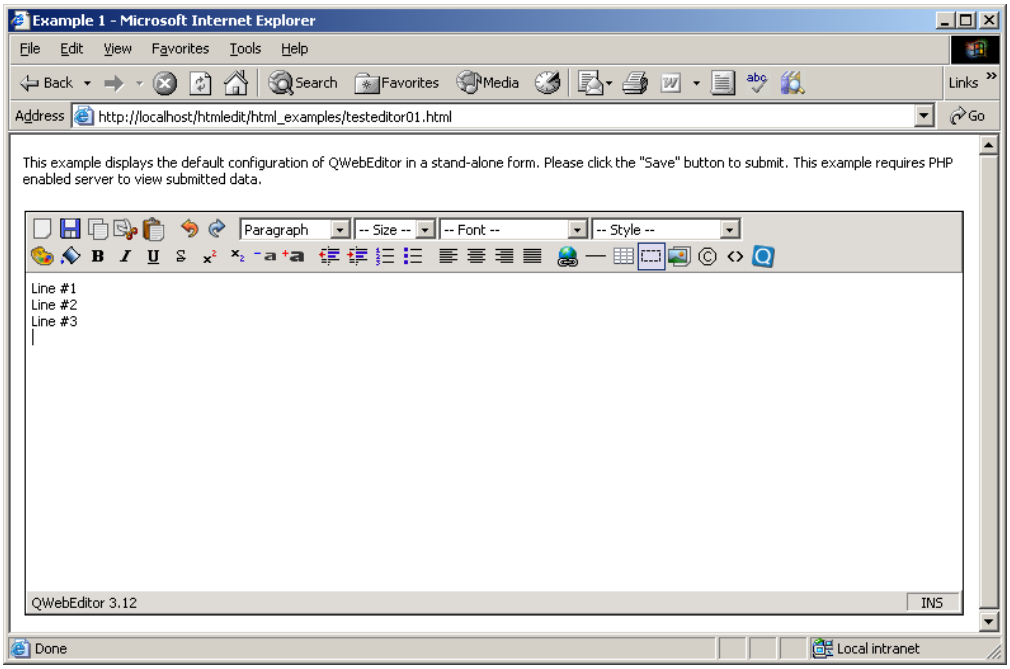
- At least using IIS 4
- Able to set directory permission (to allow image browser script to manage uploaded images)
- Able to register a COM object (to generate thumbnails)

Installation

- Extract the package qwebeditor_vx_xx.zip to your local computer.
- Upload the htmledit directory in the extracted directory into the document root directory of your web server.
- Review htmledit_cfg.js and customize it.
- To test the installation, view

  http://www.yourwebsite.com/htmledit/html_examples/testeditor01.html

  If QWebEditor is displayed in your browser, QWebEditor is successfully installed in your system.



Note 1: QWebEditor can be installed in an alternative directory (in case you cannot install QWebEditor in the document root directory.) Please refer to the programming guide for instructions.

Basic Usage

Using QWebEditor as a Form Element

QWebEditor can be used as a form element, or <textarea> replacement. You should provide the <form> tag and the submit button in this mode. To create QWebEditor as a form element:

JavaScript version:

```
<form action="viewresult.php" method="post">
<script language="javascript"><!--
HtmlEditCreateControlFromObj({
        strElementName:"pagecontent1",
        })
//--></script>
<input type="submit" value="Submit" /></form>
```

PHP version:

```
<form action="viewresult.php" method="post">
<?php
        $html = new CQWebEditor;
        $html->SetElementName('myelement');
        $html->CreateControl();
?>
<input type="submit" value="Submit" /></form>
```

ASP version:

```
<form action="viewresult.php" method="post">
<%
    dim html
    set html = new QWebEditor                ' Create the form object
    html.SetElementName "myelement"          ' Set the form element name (required)
    html.CreateControl                       ' Output the code to create the control
%>
<input type="submit" value="Submit" /></form>
```

Using QWebEditor as a Standalone Form

QWebEditor can be used as a standalone form. In this mode, a save button is provided and the editor is submitted if the button is clicked. You do not need to output any <form> tag for this mode. To create QWebEditor as a standalone form:

JavaScript version:

```
<script language="javascript"><!--
HtmlEditCreateControlFromObj({
        strElementName:"myelement",
        strAction:" viewresult.php",
        lFlags: g_lHeCBorder | g_lHeCModeStandaloneForm,
        strValue:""})
//--></script>
```

PHP version:

```
<?php
        $html = new CQWebEditor;
        $html->SetFlags(HeCBorder | HeCModeStandaloneForm);
        $html->SetElementName('myelement');
        $html->SetFormActionUrl('viewresult.php');
        $html->CreateControl();
?>
```

ASP version:

```
<%
    dim html
    set html = new QWebEditor                ' Create the form object
    html.SetFlags SetFlags(HeCBorder | HeCModeStandaloneForm
                                             ' Control will be acted as a standalone form
                                             ' (rather than form element)
    html.SetElementName "myelement"          ' Set the form element name (required)
    html.SetFormActionUrl "viewresult.asp"   ' Set the action attribute of the form
                                             ' (required)
    html.CreateControl                       ' Output the code to create the control
%>
```

Using QWebEditor as a Popup Editor

You can use QWebEditor as a popup editor. This mode of operation is available for JavaScript interface only. To use this mode, use either the HtmlEditOpenEditor() or HtmlEditOpenEditorFromObj() functions.

Setting the initial content of the editor

QWebEditor does not provide any method to read from database or local file to set the initial content of the editor. To set the initial content of the editor:

- specify the content in obj.strValue for JavaScript version, or
- call SetContent method for PHP or ASP version

Setting the initial content of the editor by URL

The previous method of setting content can only specify the content between the <body> tags. If you want to edit existing page content including body tag attributes and title tag, you should set the content by URL. To set the content of the editor by URL:

- specify the URL in obj.strPageSrc for JavaScript version, or
- call SetContentFromUrl() method for PHP or ASP version

Edit a complete HTML page

By default, QWebEditor edits HTML code between <body> tags instead of complete HTML page. It is fine for CMS (content management system) like online store that uses QWebEditor to edit product description. However, some CMS uses QWebEditor to maintain web pages of a web site. Then, QWebEditor should be able to edit a complete page (including attributes in body tag and page title). To set QWebEditor to edit a complete web page, you should:

- specify "g_lHeCEditPage" flag in obj.lFlags for JavaScript version
- call EnableEditPage() method before creating the control for PHP or ASP version

Determine whether source is plain text or HTML automatically

You may want to use QWebEditor for an existing system. However, original stored data is plain text format and therefore the existing text is displayed without linefeed inside QWebEditor. In this case, you can ask QWebEditor to determine the source content format and convert plain text formatted to HTML formatted content (basically linefeed to
 tag and spaces to). To do so:

- specify "g_lHeCDetectPlainText" flag in obj.lFlags for JavaScript version, or;
- call EnableDetectPlainText() method before creating the control for PHP or ASP version

Specify what to display if client browser does not support QWebEditor

By default, a <textarea> with the HTML content will be displayed if client browser does not support QWebEditor. This behavior preserves all text and formatting during editing. However, displaying HTML source is not user friendly and therefore you may want to display plain text instead. To ask QWebEditor to display plain text in this case:

- specify "g_lHeCToTextIfFail" flag in obj.lFlags for JavaScript version, or;
- call EnableToTextIfFail() method before creating the control for PHP or ASP version

Specify an external style sheet for the editor

You can specify an existing style sheet in the editor to match your site look and feel. To do so:

- specify URL in obj.strCssStyle for JavaScript version, or;
- call SetEditorCssFile () method before creating the control for PHP or ASP version

Using theme

QWebEditor supports theme. You can change to use other theme by setting the g_strHtmlEditThemeUrl variable in the htmledit_cfg.js file. QWebEditor provides several theme files: theme_xp.js (previous default), theme_03.js (current default), theme_crystal.js and theme_blue.js. A theme file is a javascript file which define different color values and styles used by the editor. You can use the provided theme files as template to create you own them.

Adding custom buttons to QWebEditor toolbar

You can add custom buttons to toolbar now. A common usage is to insert custom text to the editor. testeditor06.html, testeditor06.php and testeditor06.asp demonstrate how to create custom buttons to popup a drop down list box and dialog.

Add/remove buttons from toolbar, or customize drop down list boxes

From QWebEditor 3.12, different styles of editor can be defined under htmledit_styles.js. To customize the features and layout of the editor, you can create a new style and put it under htmledit_styles.js. Then you can create the editor to refer to this new style. Please refer to testeditor01.* part 4 for sample codes.

JavaScript Interface

Basic Configuration

You can use JavaScript alone to incorporate QWebEditor into your HTML pages. Following is the listing for the simplified version of testeditor01.html.

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=ISO-8859-1" />
<title>Example 1</title>

<!-- QWebEditor 3.12 Initialization Start
--><script language="javascript"><!--
// variables to match your qwebeditor installation
var g_strHtmlEditPath="/htmledit/"                    // uri to qwebeditor directory
function ijs(file){document.write("<scr"+"ipt language=\"javascript\" src=\""
        +g_strHtmlEditPath+file+"\"></scr"+"ipt>")}
// load utility libraries and configuration file
ijs("browserSniffer.js");ijs("utils.js");ijs("mydlg.js");ijs("htmledit_cfg.js")
//--></script><script language="javascript"><!--
// load necessary file if browser supports the editor
ijs(g_strHtmlEditLangFile)
if(has_htmledit){ijs(g_strHtmlEditThemeUrl);ijs("license.js");ijs("htmledit.js");ijs("htm
ledit_styles.js")}
// browser does not support the editor. load compatibility file
else {ijs("htmledit_ta.js")}
//--></script><!--
QWebEditor 3.12 Initialization End -->

</head>
<body>
<p>This example displays the default configuration of QWebEditor in a stand-alone form.
Please click the "Save" button to submit. This example requires PHP enabled server to
view submitted data.</p>

<form name="myform" action="../php_examples/viewresult2.php" method="post">

<script language="javascript"><!--
HtmlEditCreateControlFromObj({strElementName:"myelement"})
//--></script>
<input type="submit" name="submit" value="submit" />
</form>

</body>
</html>
```

The initialization code is used to include all required external QWebEditor JavaScript files to the current page. If you install QWebEditor to the default location in your server, you do not need change anything in the initialization code. Otherwise, your have to change all occurrences of "/htmledit/" to the URI that access your QWebEditor source directory. You can override some defaults value in htmledit_cfg.js by putting the values you want to override before the line ijs(g_strHtmlEditLangFile). Please refer to html_examples/testeditor05.html for details.

After initialization, you can call HtmlEditCreateControlFromObj() to create an editor. This function takes an object as parameter and this object holds different properties to control the outlook and behavior of editor.

Some JavaScript functions are also available and thus you can use client sided scripts to access the QWebEditor in run time. Those functions take at least one parameter, ID of the control, to access the control. Some useful functions include:

- HtmlEditFocus() – focus the QWebEditor control;
- HtmlEditInsertCode() – Insert HTML code to QWebEditor control current cursor location;
- HtmlEditPrepareUpdate() – Ask QWebEditor to refresh its toolbar buttons;
- HtmlEditGetContent() – Get the content within the <body> tag;
- HtmlEditGetDefContent() – If current editing more is page mode, returns the content of a complete page. Otherwise, returns content within the <body> tag;
- HtmlEditSetContent() – Set the content within the <body> tag.
- HtmlEditSetInsertPoint() – Set the cursor position of the editor.
- HtmlEditExecCmd() – Send control command string to the editor.
- HtmlEditIsLoaded() – Used to find out whether the editor is fully initialized.
- HtmlEditIsModified() – determine whether editor content is changed.

JavaScript Functions

HTMLEditCreateControlFromObj

HtmlEditCreateControlFromObj(obj)

Create a QWebEditor control.

Obj

It should be a JavaScript object created by "new Object()". Properties should be set to specify the features of the created QWebEditor control. The following properties can be specified:

obj.strId (optional) – Specifies the name of control and it is used to set the ID attribute of the control. This property should be specified if you need to access the control during run time.

obj.strWidth (optional) – Specifies the width of control in CSS length unit. Default is "100%".

obj.strHeight (optional) – Specifies the height of control in CSS length unit. Default is "100px".

obj.strValue (optional) – Specifies the initial value of the control.

obj.lFlags (optional) – Specifies features of the control.

obj.strFormName (optional) – Specifies the name of the form if QWebEditor is used as a standalone form.

obj.strElementName (required) – Specifies the element name (content is submitted with this name).

obj.strAction (optional) – Specifies the action attribute of the form if QWebEditor is used as a standalone form.

obj.strTarget (optional) – Specifies the target attribute of the form if QWebEditor is used as a standalone form

obj.strCssStyle (optional) – Specifies a string of URL (or an array of multiple URLs) to the external CSS file(s) used by the editor. It must be a complete URL (with protocol and hostname). The stylesheet(s) must have the same domain as the page using the editor. Otherwise, browsers prohibits QWebEditor from enumerating the CSS rules because of security reasons.

obj.strPageSrc (optional) – content of the editor is specified by loading a page from the URL specified by this property. This property will override the value from obj.strValue.

obj.onLoad (optional) – specify JavaScript statements that should be called after the editor is initialized.
eg. obj.onLoad = new Function("HtmlEditFocus('myctrl')")

obj.customBtns (optional) – specify an array of objects and each object represents a custom button. The custom button object should contain:
title – (strng) alt text of button
funcname – (string) callback function when the button is clicked.
imgname – (string) image used for the button and it should be located in the htmleditimgs directory.

obj.className (optional) – specify the style of the editor during creation. If this property is omitted, the "default" style is used. Default style is defined in htmledit.js. 2 additional styles "simple" and "example" are defined in htmledit_styles.js. You can use the default style as a template and put it it htmledit_styles.js to create your own style of the editor.

Possible values for obj.lFlags:

g_lHeCModeFormElement	control will be used as a form element. You have to output the form tag and other necessary form elements (eg. submit button). You should specify the name attribute in form tag and specify the 'formname' argument with the form name. When the form is submitted, content in the control with be submitted with the name equal to 'elementname' argument.
g_lHeCModeStandaloneForm	control will be used as a standalone form. All necessary HTML codes for the form is output by the function. A form with name 'formname' will be created. An additional 'save' button will be displayed in the tool bar. If the 'save' button is clicked, content of the control will be sent to the page specified by 'formactionurl' with the name of 'elementname'. You can specify the target of the form by specifying the 'formtarget' arguments.
g_lHeCModeStandaloneDialog	control will be used inside a popup dialog box.
g_lHeCResizeToWindow	control should be resized automatically to match the browser window size.
g_lHeCDisableParagraph	Disable paragraph box in toolbar
g_lHeCDisableFontSize	Disable font size box in toolbar
g_lHeCDisableFontName	Disable font name box in toolbar
g_lHeCDisableHelpBtn	Disable help button box in toolbar
g_lHeCDisableCutCopyPasteBtn	Disable cut, copy and paste buttons in toolbar

g_lHeCDisableUndoRedoBtn	Disable undo and redo buttons in toolbar
g_lHeCDisableSourceBtn	Disable view source button in toolbar
g_lHeCDisableForeColor	Disable foreground color button in toolbar
g_lHeCDisableBackColor	Disable background color button in toolbar
g_lHeCDisableAlignBtn	Disable alignment buttons in toolbar
g_lHeCDisableTableBtn	Disable table button in toolbar
g_lHeCDisableImageBtn	Disable image button in toolbar
g_lHeCEnumSysFonts	Specifies this flag to enumerate all system fonts
g_lHeCBorder	Displays a black border surrounding the control
g_lHeCDetectPlainText	Specifies this flag to force QWebEditor to determine whether initial value is plaintext or HTML formatted text. If initial value is plaintext, the value will be converted (all linefeed to tags) before display.
g_lHeCDisableStatusBar	Disable status bar of the editor.
g_lHeCToTextIfFail	Convert source content to plain text if client browser does not support QWebEditor.
g_lHeCEditPage	QWebEditor is used to edit a complete HTML page instead. Context menu provides additional menu item to edit several page properties including title and some attributes in body tag.
g_lHeCDisableFormattingBtns1	If specified, bold, italic and underline toolbar buttons will be hidden
g_lHeCDisableFormattingBtns2	If specified, strike through, superscript and subscript toolbar buttons will be hidden
g_lHeCDisableFormattingBtns3	If specified, indent, outdent, bullet list and number list toolbar buttons will be hidden
g_lHeCDisableLinkBtn	If specified, hyperlink button will be hidden
g_lHeCDisableHorizontalRuleBtn	If specified, horizontal rule button will be hidden
g_lHeCDisableSymbolBtn	If specified, insert symbol button will be hidden
g_lHeCXHTMLSource	If specified, QWebEditor will use slower method to generate HTML source code but the code is more XHTML compatible. (QWebEditor will use XHTML syntax to generate HTML tag and quote attribute values. However, QWebEditor will still use deprecated tags like <b>, <i>, and so on.)
g_lHeCUseDivForIE	If specified, tag will be used instead of <p> for IE. This feature is to avoid extra spacing between paragraphs when using <p> tag. (Note: When "enter" key is pressed, IE creates a new paragraph with <p> tag but Mozilla generates a

	  tag only)
g_lHeCEnableSafeHtml	If specified, all on**** handlers and <script> tags will be removed from the HTML source returned by the editor. This feature is to avoid XSS security problem. This flag implies XHTML source output.

Return Value: None

HtmlEditGetContent

function HtmlEditGetContent(strId)

Get the content of the control. The content is what between the <body> tags in a HTML page.

StrId ID of the control.

Return Value: A string containing the content of the control.

HtmlEditGetDefContent

function HtmlEditGetDefContent(strId)

If current editing more is page mode, returns the content of a complete page. Otherwise, returns content within the <body> tag.

StrId ID of the control.

Return Value: A string containing the content of the control.

HtmlEditSetContent

function HtmlEditSetContent(strId, strContent)

This function is used to set the content of the control.

StrId ID of the control.

strContent HTML text you want to put inside the control.

Return Value: None

HtmlEditOpenEditor

function HtmlEditOpenEditor(strId, strTitle, strCharset)

Open a new dialog window that contains QWebEditor to edit the content of a block with ID attribute "strId". Please refer to "testeditor02.html" for example.

StrId ID of the control.

StrTitle Caption of popup dialog box

strCharset Character set of the popup browser window

Return Value: None

HtmlEditFocus

function HtmlEditFocus(strId)

Set the focus to the QWebEditor with ID "strID".

StrId ID of the control.

Return Value: None

HtmlEditInsertCode

function HtmlEditInsertCode(strId, strValue)

Insert HTML code to QWebEditor control's current cursor location

strId ID of the control.

strValue Code to be inserted.

Return Value: None

HtmlEditPrepareUpdate

function HtmlEditPrepareUpdate(strId)

Ask QWebEditor to refresh its toolbar buttons.

strId ID of the control.

Return Value: None

HtmlEditSetInsertPoint

function HtmlEditSetInsertPoint(strId, bBegin)

Ask QWebEditor to put the cursor to the beginning or the end of the content

strId ID of the control.

bBegin true – beginning
false – end

Return Value: None

HtmlEditExecCmd

function HtmlEditExecCmd(strId, strCmd, bUI, Param)

Send a control command string listed in
http://msdn.microsoft.com/library/default.asp?url=/workshop/author/dhtml/reference/commandids.asp for IE
Or
http://www.mozilla.org/editor/midas-spec.html for Mozilla to the editor. You should avoid calling this function directly since different browsers may implement some features differently. QWebEditor has done its best to hide those difference.

strId	ID of the control.
strCmd	Low level command specified by browsers.
bUI	Whether to display UI for some actions. For example, you can specify true when you use the "CreateLink" command. IE will display a dialog box to ask for the hyperlink (but Mozilla will not display any UI.
Param	Additional parameter required by the command. For example, font face name when applying a font.

Return Value: None

HtmlEditIsLoaded

```
function HtmlEditIsLoaded(strId)
```

Query whether the editor is completely initialized, including whether remote page or css have been loaded.

StrId	ID of the control.

Return Value: Boolean. True if the editor has been initialized completely and false otherwise.

HtmlEditIsModified

```
Function HtmlEditIsModified(strId)
```

Query whether the editor content is modified. This function is useful when a user leaving a page with the editor but the editor content is changed.

strId	ID of the control.

Return Value: Boolean. True if the editor content is modified and false otherwise.

PHP Interface

Basic Configuration

The PHP helper library provides an easier interface to incorporate the QWebEditor in your web site and also provides additional checking to display a textarea box if clients did not meet the requirements of QWebEditor. Followings are the listing of testeditor01.php.

```
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=ISO-8859-1" />
<title>Example 1</title>
<?php
        Require_once("htmledit.php");
        HtmlEditInit2();
?>
</head>
<body>
<p>This example displays the default configuration of QWebEditor in a stand-alone
form.</p>

<form name="myform" action="../php_examples/viewresult2.php" method="post">
<?php
        $html = new CQWebEditor;
        $html->SetCtrlName('myctrl');
        $html->SetElementName('myelement');
        $html->CreateControl();
?>
<p><input type="Submit" name="mysubmit" value="Submit"></p>
</form>

</body>
</html>
```

"htmledit.php" is the PHP helper library file. After include this file in your source, you should call HtmlEditInit2() that generates the initialization code mentioned in the JavaScript interface. HtmlEditInit2() has 2 optional arguments and you do not need to change them if you install QWebEditor to the default location and using English resources. Then, you can create a CQWebEditor object and call various methods to customize it. At the end, you should call CreateControl() method to generates the code for the control. If you want to include multiple controls in a form, you can use the same object and customize it. Then, you can call CreateControl() method again to create another control.

PHP Functions

HtmlEditInit2()

```
HtmlEditInit2(
  $installpath = "",
  $override = "");
```

Output the necessary HTML and JavaScript codes to initialize QWebEditor control. This function should be called once for each PHP script only in the <head> section of your HTML output, even you want to use multiple controls in one single page.

$installpath (optional)	Specifies the location where QWebEditor is installed. It must be an absolute URL or a path relative to the document root directory of your site. If omitted, the default value "/htmledit/" will be used. If you installed QWebEditor to /somewhereelse/htmledit/ directory under your document root directory, you can specify "http://www.yourwebsite.com/somewhereelse/htmledit/" or "/somewhereelse/htmledit/" for this argument. Note: you must specify the trailing slash in the path.
$override (optional)	Specifies a string of javascript statement to override settings in htmledit_cfg.js. Eg. "g_strHtmlEditLangFile = 'lang_cht.js';"

PHP Object

CQWebEditor

This object allows you to create a QWebEditor control.

Methods

SetCtrlName	Set the name of the control. A temporary control name will be assigned if not specified.
SetWidth	Set the width of the control
SetHeight	Set the height of the control
SetContent	Set initial content of the control
SetMode	Set the operation mode of the control
SetFormName	Set the form name if QWebEditor is used as a standalone form
SetElementName	Set the element name (content is submitted with this name)
SetFormActionUrl	Set the action attribute of the form
SetFormTarget	Set the target attribute of the form
EnableParagraphBox	Display or hide the toolbar paragraph dropdown list box.

	Default: Enable
EnableFontSizeBox	Display or hide the toolbar font size dropdown list box. Default: Enable
EnableFontNameBox	Display or hide the toolbar font name dropdown list box. Default: Enable
EnableCutCopyPasteBtn	Display or hide the toolbar cut, copy and paste buttons. Default: Enable
EnableUndoRedoBtn	Display or hide the toolbar undo and redo buttons. Default: Enable
EnableSourceBtn	Display or hide the toolbar view source button. Default: Enable
EnableForeColorBtn	Display or hide the toolbar foreground (text) color button
EnableBackColorBtn	Display or hide the toolbar background color button. Default: Enable
EnableAlignBtn	Display or hide the toolbar left, right, center and justify alignment buttons. Default: Enable
EnableBorder	Display or hide a black border surrounding QWebEditor
EnableEnumSysFonts	Determine whether QWebEditor should enumerates system fonts to fill in the content of the font name drop down list box. Default: Disable
SetEditorCssFile	Specifies the external CSS file used by QWebEditor control
CreateControl	Output the HTML and JavaScript codes to generate the QWebEditor control.
SetContentFromUrl	Set the initial content of the editor by the content from URL
EnableDetectPlainText	Determine whether QWebEditor should convert content to HTML code if source content is plain text formatted. Default: disable
EnableToTextIfFail	Determine whether QWebEditor should convert content to plain text if client browser does not support QWebEditor. Default: disable
EnableStatusBar	Display or hide status bar. Default: enable
EnableEditPage	Determine whether QWebEditor is used to edit a complete page or HTML code. Default: HTML code
SetOnLoadHandler	Specifies a JavaScript function that will be called after the editor is initialized completely
EnableXHtmlSource	Determine whether QWebEditor should return XHTML compliant source. This feature is implemented by walking through the whole DOM tree and therefore it slows down run time performance when HTML source is required (eg. switching between WYSIWYG and source editing modes.
EnableUseDivForIE	IE uses <p> tag by default and browsers normally render additional spacing after the <p> tag. Although users can use shift-enter to use , not all users understand how to do it.

	Therefore, by enabling this feature, is used instead of <p> tag and the additional spacing problem is avoided.
EnableSafeHtml	Determine whether QWebEditor should return safe HTML. If on, HTML source returned by editor will have all script tags and on** handlers removed. This flag set XHTML output automatically.
SetBaseHref	Set the base href of the editor
AddCustomButton	Add a custom button to tool bar
SetClassName	Instructs the editor to initialized with a style specified in htmledit_styles.js

Method Description

SetCtrlName($a_strCtrlName)

Set the name of the control. It is used to set the ID attribute of the control and it is used in the various JavaScript functions to refer to a particular control.

$a_strCtrlName	Name of the control

SetWidth($a_strWidth)

Set the width of the control.

$a_strWidth	Width of the control in CSS unit, eg. "100%", "400px", etc.

SetHeight($a_strHeight)

Set the height of the control.

$a_strHeight	Height of the control in CSS unit, eg. "100%", "400px", etc.

SetContent($a_strContent)

Set the initial content of the control.

$a_strContent	Content you want to set inside the control.

SetMode($a_mode)

Set the operation mode of the control. QWebEditor can be used as a standalone form, form element or popup window editor.

$a_mode	Should be either one of the following flags: • HeCModeFormElement • HeCModeStandaloneForm • HeCModeStandaloneDialog

SetFormName($a_value)

Set the form name if QWebEditor is used as a standalone form. You can then use client sided script to control the form.

$a_value The value you want to set as the name attribute of the form

SetElementName($a_value)

Set the element name of QWebEditor. If the form contained QWebEditor is submitted, content of QWebEditor is submitted with this name. QWebEditor actually created a hidden field with this name for the form submission purpose. However, please do not access this hidden field through client sided script since it may not hold the most updated content of the control. Normally, the hidden field is updated just before form submission.

$a_value The value you want to set as the name of the submitted value

SetFormActionUrl($a_value)

Set the action attribute of the form if QWebEditor is used as a standalone form. Content in QWebEditor will be submitted to this URL if the save button in the editor is clicked.

$a_value The value you want to set as the action attribute of the form

SetFormTarget($a_value)

Set the target attribute of the form if QWebEditor is used as a standalone form.

$a_value The value you want to set as the target attribute of the form

EnableParagraphBox($a_bEnable)

Specify to display or hide the paragraph dropdown list box.

$a_bEnable true – display, false – hide

EnableFontSizeBox($a_bEnable)

Specify to display or hide the font size dropdown list box.

$a_bEnable true – display, false – hide

EnableFontNameBox($a_bEnable)

Specify to display or hide the font name dropdown list box.

$a_bEnable true – display, false – hide

EnableCutCopyPasteBtn($a_bEnable)

Specify to display or hide the cut, copy and paste buttons. The paste function is not available due to the security model of Gecko Midas. Therefore, the three buttons are always hidden if QWebEditor is used under Gecko based browsers (eg. Netscape 7.1)

$a_bEnable true – display, false – hide

EnableUndoRedoBtn($a_bEnable)

Specify to display or hide the undo and redo buttons.

$a_bEnable true – display, false – hide

EnableSourceBtn($a_bEnable)

Specify to display or hide the view HTML source button.

$a_bEnable true – display, false – hide

EnableForeColorBtn($a_bEnable)

Specify to display or hide the foreground (text) color button.

$a_bEnable true – display, false – hide

EnableBackColorBtn($a_bEnable)

Specify to display or hide the background color button. Background color is currently not working for Gecko Midas.

$a_bEnable true – display, false – hide

EnableAlignBtn($a_bEnable)

Specify to display or hide the left, right, center and justify alignment buttons.

$a_bEnable true – display, false – hide

EnableBorder($a_bEnable)

Specify to display or hide the black border surrounding the QWebEditor control.

$a_bEnable true – display, false – hide

EnableEnumSysFonts($a_bEnable)

Specify to enumerate system fonts to fill in the values in the font name dropdown list box. This feature is working for IE6 or later browsers only. Enumerating system fonts is a slow process especially there are a lot of fonts installed in the client computers. If system fonts are not enumerated, the font names stored in the JavaScript variable "g_arrHeFonts" (defined in language resource files) will be used.

$a_bEnable true – enable enumeration, false – disable enumeration

SetEditorCssFile($a_url)

Specifies the external CSS file used by QWebEditor control

$a_url An URL to the external CSS file. It must be a complete URL (required by Mozilla based browsers). All the .style will be listed in the style drop down list box. You can disable dropdown list box by calling the DisableStyleBox() method. Note: URL of the stylesheet must have the same domain of the page displaying the QWebEditor control. Otherwise, QWebEditor is unable to enumerate the css rules in the stylesheet.

CreateControl()

Emit the HTML and JavaScript codes to generate the QWebEditor control

SetContentFromUrl($a_url)

Set the initial content of the editor by the content from URL

$a_url An URL to the web page you want to load.

EnableDetectPlainText($a_bEnable)

Determine whether QWebEditor should convert content to HTML code if source content is plain text formatted.

$a_bEnable True – convert source to HTML code if source is plain text formatted.
False (default) – no conversion.

EnableToTextIfFail ($a_bEnable)

Determine whether QWebEditor should convert content to plain text if client browser does not support QWebEditor.

$a_bEnable True – convert to plain text if client browser does not support QWebEditor.
False (default) – no conversion.

EnableStatusBar($a_bEnable)

Display or hide status bar.

$a_bEnable True (default) – display status bar
False – hide status bar

EnableEditPage($a_bEnable)

Determine whether QWebEditor is used to edit a complete page or HTML code.

$a_bEnable	True – additional feature is provided to edit page properties False (default) – no feature is provided to edit page properties.

SetOnLoadHandler ($a_strJavaScriptFunc)

Specifies a JavaScript function that will be called after the editor is initialized completely

$a_strJavaScriptFunc	A string that specify the JavaScript function

EnableXHtmlSource($a_bEnable)

Determine whether QWebEditor should return XHTML compliant source. This feature is implemented by walking through the whole DOM tree and therefore it slows down run time performance when HTML source is required (eg. switching between WYSIWYG and source editing modes.

$a_bEnable	True – enable False (default) – disable

EnableUseDivForIE($a_bEnable)

IE uses <p> tag by default and browsers normally render additional spacing after the <p> tag. Although users can use shift-enter to use
, not all users understand how to do it. Therefore, by enabling this feature, is used instead of <p> tag and the additional spacing problem is avoided.

$a_bEnable	True – enable False (default) – disable

SetBaseHref($a_strBaseHref)

Set <Base href> of the editor.

$a_strBaseHref	URL

AddCustomButton($title, $funcname, $imgname)

Add a custom button to tool bar. This function can be called multiple times to add multiple buttons to the editor.

$title	Alt text of the button

$funcname	Callback javascript function name
$imgname	Image used for toolbar button

SetClassName($a_strClassName)

Instructs the editor to initialized with a style specified in htmledit_styles.js

$a_strClassName	A string specified the style used to initialized the editor.

ASP Interface

Basic Configuration

The ASP helper library provides an easier interface to incorporate the QWebEditor in your web site and also provides additional checking to display a textarea box if clients did not meet the requirements of QWebEditor. Followings are the listing of testeditor01.asp.

```
<% Option Explicit %>
<!--- #include file="../browsersniffer.asp" --->
<!--- #include file="../htmledit.asp" --->
<html>
<head>
<meta HTTP-EQUIV="Content-Type" CONTENT="text/html; CHARSET=ISO-8859-1" />
<title>Example 1</title>
<%
    ' output HTML and JavaScript codes to initialize QWebEditor library
    HtmlEditInit2 "/htmledit/", ""
%>
</head>
<body>
<p>This example displays the default configuration of QWebEditor in a stand-alone form.
Click the save button in the editor to submit.</p>

<form name="myform" action="../asp_examples/viewresult2.asp" method="post">
<%
        dim html
        set html = new QWebEditor
        html.SetElementName "pagecontent1"
        html.SetWidth "650px"
        html.CreateControl
%><br />
<p><input type="Submit" name="mysubmit" value="Submit"></p>
</form>

</body>
</html>
```

"browsersniffer.asp" is used to detect client browser details and "htmledit.asp" is the ASP helper library file to create QWebEditor. After include these files in your source, you should call HtmlEditInit2 in the head section of your HTML output that generates the initialization code mentioned in the JavaScript interface. Then, you can create a QWebEditor object and call various methods to customize it. At the end, you should call QWebEditor.CreateControl method to generate the JavaScript code for the control. If you want to include multiple controls in a form, you can use the same object, customize it and call CreateControl method again to create another control.

Note: Windows 2003 Server (or IIS 6.0) prohibits using ".." in the include path. Therefore, QWebEditor must be installed in a subdirectory under the scripts using the editor.

ASP Functions and Subroutines

HtmlEditInit2

HtmlEditInit2 installpath, override

Output the necessary HTML and JavaScript codes to initialize QWebEditor control. This subroutine should be called once for each ASP script only in the <head> section of your HTML output, even you want to use multiple controls in one single page.

installpath	Specifies the location where QWebEditor is installed. It must be an absolute URL or a path relative to the document root directory of your site. If omitted, the default value "/htmledit/" will be used. If you installed QWebEditor to /somewhereelse/htmledit/ directory under your document root directory, you can specify "http://www.yourwebsite.com/somewhereelse/htmledit/" or "/somewhereelse/htmledit/" for this argument. Note: you must specify the trailing slash in the path.
$override	Specifies a string of javascript statement to override settings in htmledit_cfg.js. Eg. "g_strHtmlEditLangFile = 'lang_cht.js';"

ASP Object

QWebEditor

This object allows you to create a QWebEditor control.

Methods

SetCtrlName	Set the name of the control
SetWidth	Set the width of the control
SetHeight	Set the height of the control
SetContent	Set initial content of the control
SetMode	Set the operation mode of the control
SetFormName	Set the form name if QWebEditor is used as a standalone form
SetElementName	Set the element name (content is submitted with this name)
SetFormActionUrl	Set the action attribute of the form
SetFormTarget	Set the target attribute of the form
EnableParagraphBox	Display or hide the toolbar paragraph dropdown list box
EnableFontSizeBox	Display or hide the toolbar font size dropdown list box

EnableFontNameBox	Display or hide the toolbar font name dropdown list box
EnableCutCopyPasteBtn	Display or hide the toolbar cut, copy and paste buttons
EnableUndoRedoBtn	Display or hide the toolbar undo and redo buttons
EnableSourceBtn	Display or hide the toolbar view source button
EnableForeColorBtn	Display or hide the toolbar foreground (text) color button
EnableBackColorBtn	Display or hide the toolbar background color button
EnableAlignBtn	Display or hide the toolbar left, right, center and justify alignment buttons
EnableBorder	Display or hide a black border surrounding QWebEditor
EnableEnumSysFonts	Determine whether QWebEditor should enumerates system fonts to fill in the content of the font name drop down list box
CreateControl	Output the HTML and JavaScript codes to generate the QWebEditor control
SetContentFromUrl	Set the initial content of the editor by the content from URL
EnableDetectPlainText	Determine whether QWebEditor should convert content to HTML code if source content is plain text formatted. Default: disable
EnableToTextIfFail	Determine whether QWebEditor should convert content to plain text if client browser does not support QWebEditor. Default: disable
EnableStatusBar	Display or hide status bar. Default: enable
EnableEditPage	Determine whether QWebEditor is used to edit a complete page or HTML code. Default: HTML code
EnableXHtmlSource	Determine whether QWebEditor should return XHTML compliant source. This feature is implemented by walking through the whole DOM tree and therefore it slows down run time performance when HTML source is required (eg. switching between WYSIWYG and source editing modes. Default: disable
EnableUseDivForIE	IE uses <p> tag by default and browsers normally render additional spacing after the <p> tag. Although users can use shift-enter to use , not all users understand how to do it. Therefore, by enabling this feature, is used instead of <p> tag and the additional spacing problem is avoided. Default: disable
EnableSafeHtml	Determine whether QWebEditor should return safe HTML. If on, HTML source returned by editor will have all script tags and on** handlers removed. This flag set XHTML output automatically.
SetBaseHref	Set the base href of the editor
AddCustomButton	Add a custom button to tool bar
SetClassName	Instructs the editor to initialized with a style specified in htmledit_styles.js

Method Description

SetCtrlName *a_strCtrlName*

Set the name of the control. It is used to set the ID attribute of the control and it is used in the various JavaScript functions to refer to a particular control.

a_strCtrlName	Name of the control

SetWidth *a_strWidth*

Set the width of the control.

a_strWidth	Width of the control in CSS unit, eg. "100%", "400px", etc.

SetHeight *a_strHeight*

Set the height of the control.

a_strHeight	Height of the control in CSS unit, eg. "100%", "400px", etc.

SetContent *a_strContent*

Set the initial content of the control.

a_strContent	Content you want to set inside the control.

SetMode *a_mode*

Set the operation mode of the control. QWebEditor can be used as a standalone form, form element or popup window editor.

a_mode	Should be either one of the following flags: • HeCModeFormElement • HeCModeStandaloneForm • HeCModeStandaloneDialog

SetFormName *a_value*

Set the form name if QWebEditor is used as a standalone form. You can then use client sided script to control the form.

a_value	The value you want to set as the name attribute of the form

SetElementName *a_value*

Set the element name of QWebEditor. If the form contained QWebEditor is submitted, content of QWebEditor is submitted with this name. QWebEditor actually created a hidden field with this name for the form submission purpose. However, please do not access this

hidden field through client sided script since it may not hold the most updated content of the control. Normally, the hidden field is updated just before form submission.

a_value The value you want to set as the name of the submitted value

SetFormActionUrl *a_value*

Set the action attribute of the form if QWebEditor is used as a standalone form. Content in QWebEditor will be submitted to this URL if the save button in the editor is clicked.

a_value The value you want to set as the action attribute of the form

SetFormTarget *a_value*

Set the target attribute of the form if QWebEditor is used as a standalone form.

a_value The value you want to set as the target attribute of the form

EnableParagraphBox *a_bEnable*

Specify to display or hide the paragraph dropdown list box.

a_bEnable true – display, false – hide

EnableFontSizeBox *a_bEnable*

Specify to display or hide the font size dropdown list box.

a_bEnable true – display, false – hide

EnableFontNameBox *a_bEnable*

Specify to display or hide the font name dropdown list box.

a_bEnable true – display, false – hide

EnableCutCopyPasteBtn *a_bEnable*

Specify to display or hide the cut, copy and paste buttons. The paste function is not available due to the security model of Gecko Midas. Therefore, the three buttons are always hidden if QWebEditor is used under Gecko based browsers eg. Netscape 7.1

a_bEnable true – display, false – hide

EnableUndoRedoBtn *a_bEnable*

Specify to display or hide the undo and redo buttons.

a_bEnable true – display, false – hide

EnableSourceBtn *a_bEnable*

Specify to display or hide the view HTML source button.

a_bEnable true – display, false – hide

EnableForeColorBtn *a_bEnable*

Specify to display or hide the foreground text color button.

a_bEnable true – display, false – hide

EnableBackColorBtn *a_bEnable*

Specify to display or hide the background color button. Background color is currently not working for Gecko Midas.

a_bEnable true – display, false – hide

EnableAlignBtn *a_bEnable*

Specify to display or hide the left, right, center and justify alignment buttons.

a_bEnable true – display, false – hide

EnableBorder *a_bEnable*

Specify to display or hide the black border surrounding the QWebEditor control.

a_bEnable true – display, false – hide

EnableEnumSysFonts *a_bEnable*

Specify to enumerate system fonts to fill in the values in the font name dropdown list box. This feature is working for IE6 or later browsers only. Enumerating system fonts is a slow process especially there are a lot of fonts installed in the client computers. If system fonts are not enumerated, the font names stored in the JavaScript variable "g_arrHeFonts" defined in language resource files will be used.

a_bEnable true – enable enumeration, false – disable enumeration

SetEditorCssFile a_url

Specifies the external CSS file used by QWebEditor control

a_url An URL to the external CSS file. It must be a complete URL (required by Mozilla based browsers). All the .style will be listed in the style drop down list box. You can disable dropdown list box by calling the DisableStyleBox() method. Note: URL of the stylesheet must have the same domain of the page displaying

the QWebEditor control. Otherwise, QWebEditor is unable to enumerate the css rules in the stylesheet.

CreateControl

Emit the HTML and JavaScript codes to generate the QWebEditor control

SetContentFromUrl a_url

Set the initial content of the editor by the content from URL

a_url — An URL to the web page you want to load.

EnableDetectPlainText a_bEnable

Determine whether QWebEditor should convert content to HTML code if source content is plain text formatted.

a_bEnable — True – convert source to HTML code if source is plain text formatted.
False (default) – no conversion.

EnableToTextIfFail a_bEnable

Determine whether QWebEditor should convert content to plain text if client browser does not support QWebEditor.

a_bEnable — True – convert to plain text if client browser does not support QWebEditor.
False (default) – no conversion.

EnableStatusBar a_bEnable

Display or hide status bar.

a_bEnable — True (default) – display status bar
False – hide status bar

EnableEditPage a_bEnable

Determine whether QWebEditor is used to edit a complete page or HTML code.

a_bEnable — True – additional feature is provided to edit page properties
False (default) – no feature is provided to edit page properties.

SetBaseHref a_strBaseHref

Set <Base href> of the editor.

a_strBaseHref	URL

AddCustomButton title, funcname, imgname)

Add a custom button to tool bar. This function can be called multiple times to add multiple buttons to the editor.

title	Alt text of the button
funcname	Callback javascript function name
imgname	Image used for toolbar button

SetClassName a_strClassName

Instructs the editor to initialized with a style specified in htmledit_styles.js

a_strClassName	A string specified the style used to initialized the editor.

Frequently Asked Questions

Does QWebEditor behave differently under IE and Netscape/Mozilla?
Even we tried our best but there are still differences in implementation of QWebEditor under the two browsers:

- Netscape/Mozilla prohibits using Copy/Cut features through scripting by default. Therefore, CUT/COPY/PASTE buttons are disabled under Netscape/Mozilla. However, you can still use the shortcut ctrl-c (copy), ctrl-x (cut) and ctrl-v (paste) to manipulate the clipboard
- Netscape/Mozilla did not provide any method to retrieve the current insert/overwrite status. Therefore, the "Insert/Overwrite" indicator is disabled by default.
- Tab positioning and Shortcut keys (using <label>) will not work properly under Netscape/Mozilla.

Does QWebEditor behave different under Safari for Mac?
Starting from Safari version 1.3, Safari has enough support for WYSIWYG editing and we started to support since Safari 2.0. Safari is a very nice browser but it still misses some features, most notably is the incomplete editing, selection and dynamic style creation supports. Therefore, QWebEditor under Safari misses the following features:

- Stylesheet supports, including using of external stylesheet, table border guideline, etc.
- Some editing features: font size, paragraph style, ordered list and unordered list, remove link, etc. Safari does not provide these editing features.
- Toolbar action relying current context: cursor on hyperlink and click on the toolbar hyperlink button will not bring up a properties box with the URL of the hyperlink. QWebEditor has no way to determine the current context since Safari

What will happen if client browser does not support QWebEditor?
A textarea box will be displayed instead. The content can be optionally converted to plain text.

QWebEditor properties dialog boxes popup slowly. How can I speed it up?
You can instruct your web server to send the expire header for all files in QWebEditor directory (but exclude the upload directory). Therefore, browsers would not contact the server again if QWebEditor files are already stored in browser cache. For apache (with mod_expire enabled), you can create a .htaccess file in the QWebEditor source directory with the following content:

```
ExpiresActive On
ExpiresDefault "access plus 15 minutes"
```

You can achieve the same result by tweaking your IIS settings.

QWebEditor returns duplicated HTML codes for some content after submitting.
If you setup QWebEditor to generate XHTML code and user pasted malformed HTML code to the editor, QWebEditor may return duplicated HTML code for the same text under Internet Explorer. You can disable XHTML code generation to get around the problem.

Support

If you have any question, you are welcome to visit our browse the FAQ section located in our web site or contact us by email: support@qwebeditor.com.