Asbru Ltd.
www.asbrusoft.com
info@asbrusoft.com

Asbru Ltd.

Asbru Web Content Editor

Developer Guide

Easily & Inexpensively
Create, Edit & Post Your Website Content

20. October 2008 Version 7.0.19

Copyright and Proprietary Information

Notice

Asbru Ltd. reserves the right to make changes in this user guide at any time and without notice. Asbru Ltd. makes no warranties, express or implied, in this user guide. In no event shall Asbru Ltd. be liable for any indirect, special, incidental or consequential damages arising out of purchase or use of this user guide or the information contained herein.

Licenses and Trademarks

Asbru Web Content Editor and the Asbru logo are trademarks or registered trademarks of Asbru Ltd. in the United Kingdom and other countries. All other company, product, or trade names are trademarks or registered trademarks of their respective holders.

Asbru Web Content Editor includes and uses the wz_dragdrop.js library, Copyright (c) 2002-2003 Walter Zorn (www.walterzorn.com), licensed under the terms of the GNU Lesser General Public License (LGPL) (www.gnu.org/copyleft/lesser.html)

Asbru Web Content Editor includes and uses the wz_jsgraphics.js library, Copyright (c) 2002-2004 Walter Zorn (www.walterzorn.com), licensed under the terms of the GNU Lesser General Public License (LGPL) (www.gnu.org/copyleft/lesser.html)

Asbru Ltd.

Asbru Ltd. provides Internet/Web services, consultancy and solutions for businesses and individuals. Registered in England - Company Registration No. 3865324 - www.asbrusoft.com

Asbru
Web Content Editor

Easily & Inexpensively
Create, Edit & Post Your Website Content

Introduction

This document is the developer guide for the Asbru Web Content Editor. The developer guide describes how you, the website developer, install, configure and use the Asbru Web Content Editor to create, edit and post your website content.

The Asbru Web Content Editor is a web solution allowing you easy access to create and edit the content of your websites as well as any web-based applications and services. The system is full-featured but very easy to use and highly flexible.

This developer guide is divided into five main parts:

Part 1 describes how to install and configure the Asbru Web Content Editor to run on your website.

Part 2 describes how to integrate the Asbru Web Content Editor with your own and third-party web applications.

Part 3 describes how different options can change the Asbru Web Content Editor functionality and how it looks.

Part 4 describes how your web content is posted from the Asbru Web Content Editor to your website as well as the included simple file editor example web application.

Part 5 describes how to customise and extend the Asbru Web Content Editor hyperlink and media dialog windows and other functionality to your own web applications and databases.

Table of Contents

1 Installation and Configuration

Installing and configuring the Asbru Web Content Editor is easy and should take no more than a few minutes if you are familiar with web servers.

Section 1.1 describes the system requirements and what you need to do and know before you install the Asbru Web Content Editor.

Section 1.2 describes how to download and install the Asbru Web Content Editor program files.

Section 1.3 describes the initial minimal configuration of the Asbru Web Content Editor to get it running on your website.

1.1 System Requirements

The Asbru Web Content Editor is very flexible in that it runs on most major website platforms: operating systems, web browsers, web servers and programming/scripting languages. No matter which platform your website runs on it is likely to be supported.

The Asbru Web Content Editor runs on the following website platforms:

Website Platform Component	Supported Products
Operating System	Microsoft Windows Macintosh Linux Unix (+ other operating systems)
Web Server	Microsoft Internet Information Server Apache Web Server (+ any standard compliant web server)
Programming / Scripting Language	Any (for standard hyperlink and media dialog windows) ASP, ASP.NET/C#, ColdFusion, JSP/Java, PHP (for advanced hyperlink and media manager) (+ easy adaptation to other languages)
Spell Checking	Aspell (free software download from http://aspell.net/).
Web Browser (web content editors)	Microsoft Windows Internet Explorer (v4.0 or newer) Mozilla (v1.3 or newer) Mozilla Firefox (v0.7 or newer) Netscape (v7.1 or newer) Safari (v2.0.1 or newer) (+ any standard compliant web browser for standard HTML form textarea content editing)
Web Browser (web content viewers)	Any standard compliant web browser

Please note that only recent versions of the website platform software are supported by the Asbru Web Content Editor. You should always make sure to keep your website platform software updated to the latest or at least a recent version to avoid functionality and security problems.

Before installing the Asbru Web Content Editor you should make sure that:

- Your operating system, web server, programming/scripting language, database server, database drivers and web browser are installed and working correctly.

- You have access and permissions to copy files to your web server and your website directory/folder through FTP (File Transfer Protocol) or Microsoft Networking or similar.

1.2 Download and Installation

The Asbru Web Content Editor is available for download from the Asbru website (www.asbrusoft.com). The software is available in a variety of packages and formats. Please check the website for details.

The downloaded package is a compressed file archive, which you must uncompress and extract. The package includes a number of files located under a folder named "webeditor". Depending on the downloaded package the files could be:

/webeditor/ Files Example		
blank.html	empty.css	index.html
empty.html	example.css	index.asp
webeditor1.js	webeditor.css	index.aspx
webeditor2.js	cellproperties.html	index.cfm
webeditor.js	colour.html	index.jsp
webeditor_contenteditable.js	columnproperties.html	index.php
webeditor_contenteditable_mozilla.js	help2.html	media.html
webeditor_contenteditable_msie.js	help.html	rowproperties.html
webeditor_dhtml_msie.js	hyperlink.html	table.html
webeditor_textarea.js	UserGuide.pdf	tableproperties.html

Please note that some of the Asbru Web Content Editor files may be named identically to some of your existing website files in which case your existing files will be overwritten. Please make sure to backup all your existing website files before installing the Asbru Web Content Editor.

To install the Asbru Web Content Editor you must copy the "webeditor" folder and all its sub-folders and files to your website folder on your web server.

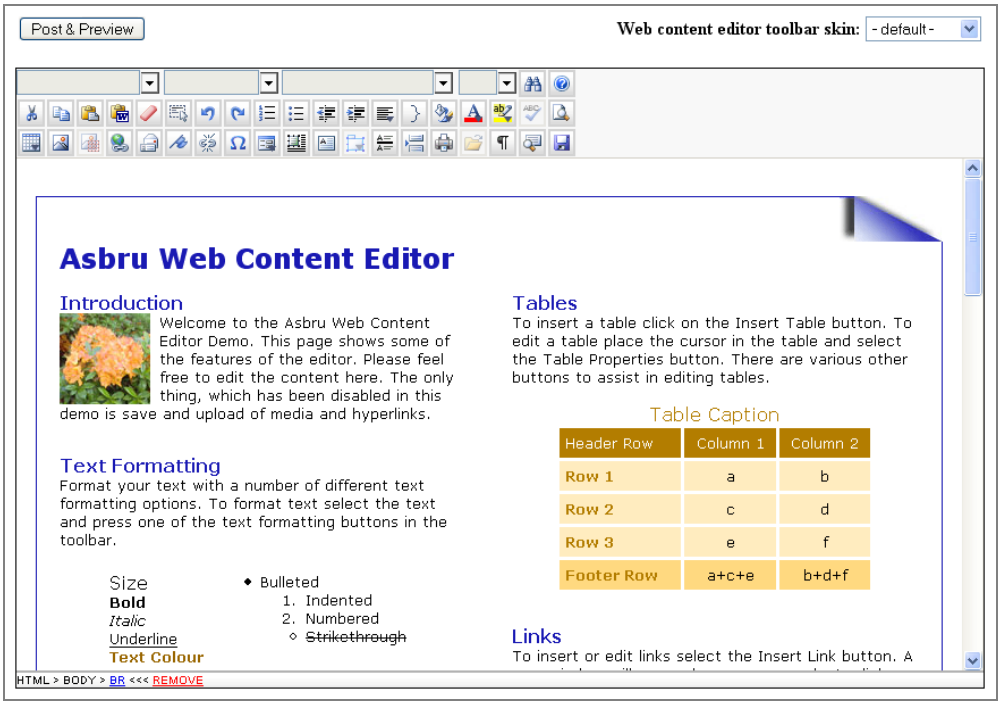
Please note that the Asbru Web Content Editor includes simple file editor examples, which you may not want to leave available/unrestricted through your website as they enable users to view and edit the content of the "simple_file_editor_example_content.html" file. Simply delete all the "simple_file_editor_example*" files from the "webeditor" folder.

The Asbru Web Content Editor also includes a number of example web pages, which you may want to delete from your production websites. All example web pages are named "index*" and all the example web page content images etc. are named "example*".

You may also want to only have one programming language version of the Asbru Web Content Editor on your production websites. All ASP specific files are named "*.asp" and "*.asp.html"; all ASP.NET specific files are named "*.ascx", "*.cs", "*.aspx" and "*aspx.html"; all ColdFusion specific files are named "*.cfm" and "*.cfm.html"; all JSP specific files are named "*.jsp" and "*.jsp.html"; and all PHP specific files are named "*.php" and "*.php.html".

1.3 Testing

After installing the Asbru Web Content Editor folder and files to your website root/home directory, access your website using your usual website domain name or IP number followed by "/webeditor/" (i.e. "http://www.asbrusoft.com/webeditor/"). If your web server and programming/scripting language is working correctly, you should now automatically get access to the default Asbru Web Content Editor example web page.



To test specific programming/scripting language versions of the Asbru Web Content Editor you can use the following example web pages:

/webeditor/index.html
/webeditor/index.asp
/webeditor/index.cs.aspx
/webeditor/index.vb.aspx
/webeditor/index.cfm
/webeditor/index.jsp
/webeditor/index.php

If these example web pages do not work then your web server and programming/scripting language is not working correctly, or the Asbru Web Content Editor has not been installed correctly.

1.4 Installed

Once you have installed the Asbru Web Content Editor as described in the previous sections, you are ready to start using and customising/integrating it.

Initially, we recommend you to use the basic web content editor features to get familiar with the basics, which are described in the separate user guide document.

Once you are familiar with the basics of the web content editor, you need to customise and/or integrate it for use with your own website or web-based application and database.

1.5 Configuration

As default the Asbru Web Content Editor uses its advanced hyperlink and media manager dialog windows for inserting images and other media files as well as for inserting hyperlinks. As default the Asbru Web Content Editor advanced dialog windows are configured to use all pages, images and other files in all folders on your website (except for the pure Javascript version of the Asbru Web Content Editor). As default spell checking and upload of images and other files are disabled.

No configuration is required for the pure Javascript version of the Asbru Web Content Editor. Spell checking and upload of images and other files are not available with the pure Javascript version of the Asbru Web Content Editor.

1.5.1 Hyperlink and media manager folders and file uploads

You can configure the Asbru Web Content Editor advanced dialog windows to only use pages, images and other files in sub-folders of your website. You can also enable upload of images and other files and configure which image and file formats users are allowed to upload. These settings are configured in the "config.asp", "config.aspx", "config.cfm", "config.jsp" and "config.php" files in the "webeditor" folder.

You must configure the following variables in the configuration file(s):

context
JSP version only. The Java application server context, which your "images_path" and "files_path" folders are located under. Set this to blank ("") if your "images_path" and "files_path" folders are located under your website root context.
root_path
The full path to your website's root folder (or your Java application server context folder, which your "images_path" and "files_path" folders are located under). Set this to blank ("") to use static HTML coded hyperlink and media options in the hyperlink and media manager web applications instead of automatically listing files in folders.
enable_upload
Set this to "yes" to give users access to manage folders and files through the Insert Hyperlink and Insert Media advanced dialog windows. Otherwise set this to "" to disable file uploads etc.

upload_path
The full path to a temporary folder for file uploads. Set this to blank (“”) to disable file upload through the hyperlink and media manager web applications. Ideally this folder should be outside your website’s root folder or have access restrictions so that temporary uploaded files cannot be accessed directly through your website.
exclude_paths
List of folder paths/names (separated by commas) which should be hidden from users in the Insert Hyperlink and Insert Media advanced dialog windows. Please note that all folder paths/names must start with “/”.
pages_path
Relative path under your “root_path” to the folder with your website pages. Please note that the path must start and end with “/”.
images_path
Relative path under your “root_path” to the folder with your website images. Please note that the path must start and end with “/”.
files_path
Relative path under your “root_path” to the folder with your website pages. Please note that the path must start and end with “/”.
page_formats
File name extensions (separated by commas) to be used for “pages”. Only files with these extensions can be uploaded to the website.
image_formats
File name extensions (separated by commas) to be used for “images”. Only files with these extensions can be uploaded to the website.
file_formats
File name extensions (separated by commas) to be used for “files”. Only files with these extensions can be uploaded to the website.

1.5.2 Spell checking

The Asbru Web Content Editor supports integrated spell checking of web content using the free third-party Aspell (http://aspell.net/) spell checking application.

To enable the spell checking functionality you must download and install the Aspell application and dictionaries on your web server. Aspell is free and can be downloaded from http://aspell.net/. Please see the Aspell documentation for details on how to install Aspell.

When Aspell has been installed on your web server you must configure the Asbru Web Content Editor and specify where Aspell is installed on your web server and which dictionaries to use. These are configured in the "config.asp", "config.aspx", "config.cfm", "config.jsp" and "config.php" files in the "webeditor" folder.

You must configure the following variables in the configuration file(s):

spellcheckCommand
The full path and file name of your installed copy of Aspell to use for spell checking. Set this to blank ("") to disable access to spell checking.
spellcheckParameters
The Aspell command line parameters to use for spell checking. As default this should be: "–a –H" Set this to blank ("") to disable access to spell checking.
spellcheckDictionary
Aspell command line parameter to use to specify which dictionary to use for spell checking. As default this should be: "-d"
spellcheckDictionaries
The dictionaries to be made available to users for spell checking. These must be specified as HTML SELECT OPTION tags. The OPTION values should be Aspell dictionary names such as "en", "en_GB" and "en_US" language/country codes or "english", "british" and "american" language names. Please see the Aspell dictionaries documentation for details.

1.6 Upgrades

The Asbru Web Content Editor is improved and extended, continuously, and new releases may be made available for download from the Asbru website (www.asbrusoft.com).

This section describes the general procedure for upgrading the Asbru Web Content Editor. However, the upgrade procedure may vary for some releases. Please make sure to read and follow any special upgrade instructions on the Asbru website.

Upgrading the Asbru Web Content Editor should usually never cause any of your website content and other data to be modified or deleted. However, before upgrading the Asbru Web Content Editor you should always make a backup copy of your existing program files, data files and database, which you can restore if anything goes wrong with the upgrade.

To download and install a new release of the Asbru Web Content Editor, simply download a package in an appropriate format, unpack it and copy it to your website folder in a similar way to your initial installation. The program files in the new release should replace your existing program files.

Please note that the new release may replace files that you have modified, so you should make sure to have a copy of your modified files and copy your modifications to the new files. Specifically, you should keep a copy of your web content editor configuration files ("/webeditor/config.*") and reconfigure the web content editor after upgrading.

2 Integration

The Asbru Web Content Editor content is included and posted from your web pages as ordinary HTML FORM data. This allows you to easily integrate the Asbru Web Content Editor with any website and web-based applications simply by replacing any existing HTML FORM TEXTAREA field with the Asbru Web Content Editor, which automatically degrades to a TEXTAREA field for web browsers without support for visual web content editing.

The following sections describe different alternative ways of integrating the Asbru Web Content Editor with your own and third-party web applications followed by sections describing the various optional additional parameters for the Asbru Web Content Editor.

2.1 Automatic HTML FORM TEXTAREA Replacement

For existing web applications with simple HTML FORM TEXTAREA input fields you can automatically replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by adding a single, simple Javascript tag to the HEAD of your web pages with the HTML FORM TEXTAREA input fields. No other changes to your web pages/applications should be required.

The Asbru Web Content Editor will automatically try to detect which programming language you are using for your web page from your HTML FORM ACTION attribute and from your web page address and use the same programming language version of the Asbru Web Content Editor.

```
<script type="text/javascript" src="/webeditor/webeditor.textareas.js"></script>
```

The HTML HEAD can include this SCRIPT tag to load the Asbru Web Content Editor and automatically replace simple TEXTAREA input fields on the web page with the Asbru Web Content Editor. When using this SCRIPT tag no other SCRIPT tags should be used to load the Asbru Web Content Editor.

If the Asbru Web Content Editor is installed to another location than the default '/webeditor/' folder, the 'src' address must be adjusted to your alternative installation folder.

Optionally, if you are using a CSS style sheet file for your website, the Asbru Web Content Editor can use it and give your web content editor users access to select the generic style names defined in your CSS style sheet file.

```
<script type="text/javascript">webeditor_stylesheet = "//webeditor/example.css";</script>
```

The HTML HEAD can include this SCRIPT tag to tell the Asbru Web Content Editor where to find your CSS style sheet file. When using this SCRIPT tag, it should be used immediately before the SCRIPT tag used to load the Asbru Web Content Editor.

Replace '/webeditor/example.css' with the folder path and file name of your own CSS style

sheet file.

The “webeditor” folder includes a “index.textareas.html” file with a simple example of how to automatically replace a simple HTML FORM TEXTAREA input field with the Asbru Web Content Editor.

2.2 Manual HTML FORM TEXTAREA Replacement Using Javascript

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with Javascript tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script>
The HTML HEAD can include this SCRIPT tag to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘src’ address must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<script type="text/javascript">WebEditor('content', 'This is my web content');</script>
The HTML BODY FORM must include this SCRIPT tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this SCRIPT tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be a Javascript string or variable with the initial content of the web content editor input field. A third parameter can be used for optional additional parameters. If used, the third parameter must be a curly bracket with the optional additional parameter names and values separated by commas. For example: WebEditor('content', 'This is my web content', {stylesheet: '/example.css', format: 'html'}); If no additional parameters are passed to the WebEditor function the default settings are used. You can also pass additional parameters to customise the web content editor functionality. Please see 3.2 Web Editor Options for details. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or

limit the size of the editable web content.
<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The “webeditor” folder includes “index.example.textarea.html” and “index.example.webeditor.html” files with a simple example of how to manually replace a simple HTML FORM TEXTAREA input field with the Asbru Web Content Editor.

2.3 Manual HTML FORM TEXTAREA Replacement Using ASP

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with ASP tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script> <!-- #include file="/webeditor/webeditor.asp" -->
The HTML HEAD can include these SCRIPT and ASP tags to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘src’ and ‘file’ addresses must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<% WebEditor "content", “This is my web content”, “” %>
The HTML BODY FORM must include this ASP tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this ASP tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be an ASP string or variable with the initial content of the web content editor input field. The third parameter is for optional additional parameters. The third parameter can be used for optional additional parameters. Please see 3.2 Web Editor Options for details. If used, the third parameter must be an ASP string or variable containing a curly bracket with the optional additional parameter names and values separated by commas.

For example: <% WebEditor "content", “This is my web content”, "{stylesheet: '/example.css', format: 'html'}" %> If no additional parameters are passed to the WebEditor function the default settings are used. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.
<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The “webeditor” folder includes various “index*.asp” files with examples of how to use the Asbru Web Content Editor with ASP.

2.4 Manual HTML FORM TEXTAREA Replacement Using ASP.NET/VB

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with ASP.NET/VB tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script> <!-- #include file="/webeditor/webeditor.vb.aspx" -->
The HTML HEAD can include these SCRIPT and ASP.NET/VB tags to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘src’ and ‘file’ addresses must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<% WebEditor("content", “This is my web content”, "") %>
The HTML BODY FORM must include this ASP.NET/VB tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this

ASP.NET/VB tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be an ASP.NET/VB string or variable with the initial content of the web content editor input field. The third parameter is for optional additional parameters. The third parameter can be used for optional additional parameters. Please see 3.2 Web Editor Options for details. If used, the third parameter must be an ASP.NET/VB string or variable containing a curly bracket with the optional additional parameter names and values separated by commas. For example: <% WebEditor("content", “This is my web content”, "{stylesheet: '/example.css', format: 'html'}") %> If no additional parameters are passed to the WebEditor function the default settings are used. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.
<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The “webeditor” folder includes various “index*.vb.aspx” files with examples of how to use the Asbru Web Content Editor with ASP.NET/VB.

2.5 Manual HTML FORM TEXTAREA Replacement Using ASP.NET/C#

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with ASP.NET/C# tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script> <!-- #include file="/webeditor/webeditor.cs.aspx" -->
The HTML HEAD can include these SCRIPT and ASP.NET/C# tags to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘src’ and ‘file’ addresses must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be

displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<% WebEditor("content", “This is my web content”, ""); %>
The HTML BODY FORM must include this ASP.NET/C# tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this ASP.NET/C# tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be an ASP.NET/C# string or variable with the initial content of the web content editor input field. The third parameter is for optional additional parameters. The third parameter can be used for optional additional parameters. Please see 3.2 Web Editor Options for details. If used, the third parameter must be an ASP.NET/C# string or variable containing a curly bracket with the optional additional parameter names and values separated by commas. For example: <% WebEditor("content", “This is my web content”, "{stylesheet: '/example.css', format: 'html'}"); %> If no additional parameters are passed to the WebEditor function the default settings are used. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.
<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The “webeditor” folder includes various “index*.cs.aspx” files with examples of how to use the Asbru Web Content Editor with ASP.NET/C#.

2.6 Manual HTML FORM TEXTAREA Replacement Using ColdFusion

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with ColdFusion tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script> <cfinclude template="/webeditor/webeditor.cfm" >
The HTML HEAD can include these SCRIPT and ColdFusion tags to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’

folder, the 'src' and 'file' addresses must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<cfoutput>#WebEditor(name="content", content=" This is my web content ")#</cfoutput>
The HTML BODY FORM must include this ColdFusion tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this ColdFusion tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be a ColdFusion string or variable with the initial content of the web content editor input field. The third parameter is for optional additional parameters. The third parameter can be used for optional additional parameters. Please see 3.2 Web Editor Options for details. If used, the third parameter must be a ColdFusion string or variable containing a curly bracket with the optional additional parameter names and values separated by commas. For example: <cfoutput>#WebEditor(name="content", content=" This is my web content ", options="{stylesheet: '/example.css', format: 'html'}")#</cfoutput> If no additional parameters are passed to the WebEditor function the default settings are used. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.
<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The "webeditor" folder includes various "index*.cfm" files with examples of how to use the Asbru Web Content Editor with ColdFusion.

2.7 Manual HTML FORM TEXTAREA Replacement Using JSP

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with JSP tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script> <%@ include file="/webeditor/webeditor.jsp" %>
The HTML HEAD can include these SCRIPT and JSP tags to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default '/webeditor/' folder, the 'src' and 'file' addresses must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<%= new WebEditor("content", "This is my web content", "") %>
The HTML BODY FORM must include this JSP tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this JSP tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be a JSP string or variable with the initial content of the web content editor input field. The third parameter is for optional additional parameters. The third parameter can be used for optional additional parameters. Please see 3.2 Web Editor Options for details. If used, the third parameter must be a JSP string or variable containing a curly bracket with the optional additional parameter names and values separated by commas. For example: <%= new WebEditor("content", "This is my web content", "{stylesheet: '/example.css', format: 'html'}") %> If no additional parameters are passed to the WebEditor function the default settings are used. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.
<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The “webeditor” folder includes various “index*.jsp” files with examples of how to use the Asbru Web Content Editor with JSP.

2.8 Manual HTML FORM TEXTAREA Replacement Using PHP

For existing web applications with simple HTML FORM TEXTAREA input fields you can manually replace the HTML FORM TEXTAREA input fields with the Asbru Web Content Editor simply by **replacing** your existing TEXTAREA tags with PHP tags.

<script type="text/javascript" src="/webeditor/webeditor.js"></script> <?php include "/webeditor/webeditor.php"; ?>
The HTML HEAD can include these SCRIPT and PHP tags to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘src’ and ‘file’ addresses must be adjusted to your alternative installation folder.
<script type="text/javascript">WebEditorToolbar();</script>
The HTML BODY must include this SCRIPT tag to display the Asbru Web Content Editor Toolbar. The toolbar is displayed where this SCRIPT tag is used. If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can also be customized by passing parameters defining which options to display and in which order. You can even replace the default functionality or add you own functionality. Please see 5.4 Custom Toolbar for details.
<?php WebEditor("content", “This is my web content”, ""); ?>
The HTML BODY FORM must include this PHP tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this PHP tag is used. The first parameter must be the name of the input field – this is the name of the HTML FORM data posted to the web server. The second parameter must be a PHP string or variable with the initial content of the web content editor input field. The third parameter is for optional additional parameters. The third parameter can be used for optional additional parameters. Please see 3.2 Web Editor Options for details. If used, the third parameter must be a PHP string or variable containing a curly bracket with the optional additional parameter names and values separated by commas. For example: <?php WebEditor("content", “This is my web content”, "{stylesheet: '/example.css', format: 'html'}"); ?> If no additional parameters are passed to the WebEditor function the default settings are used. As default the editable web content is automatically sized to be 100% wide and 100% high to fill the available display area. Surround this tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.

<script type="text/javascript">WebEditorDOMInspector();</script>
Optionally, the HTML BODY may include this SCRIPT tag to display the Asbru Web Content Editor DOM Inspector. The DOM Inspector is displayed where this SCRIPT tag is used.

The “webeditor” folder includes various “index*.php” files with examples of how to use the Asbru Web Content Editor with PHP.

2.9 ASP.NET/VB User Control

As an alternative to using direct HTML and Javascript code and the programming language functions described above to use the Asbru Web Content Editor on a web page, you may prefer to use an ASP.NET user control.

<%@ Register tagprefix="Asbru" tagname="WebEditor" src="/webeditor/webeditor.vb.ascx" %>
The web page must register the Asbru:WebEditor ASP.NET control before it can be used. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘src’ location must be adjusted to your alternative installation folder.
<script type="text/javascript" src="/webeditor/webeditor.js"></script>
The HTML HEAD can include this SCRIPT tag to load the Asbru Web Content Editor. If the Asbru Web Content Editor is installed to another location than the default ‘/webeditor/’ folder, the ‘file’ address must be adjusted to your alternative installation folder.
<form method="post" onsubmit="WebEditorSubmit()" action="index.postback.vb.aspx" runat="server">
The HTML BODY FORM should call the “WebEditorSubmit” Javascript function on submit to save the web content editor content for postbacks.
<Asbru:WebEditor ID="webeditor" Name="content" Content=” This is my web content” Options="{ stylesheet: '/example.css', format: 'html' }" runat="server" />
The HTML BODY FORM must include this ASP.NET control tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this ASP.NET control tag is included. The editable web content is automatically sized to fill the available display area. Surround this SCRIPT tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content. The parameters passed to the Asbru:WebEditor control are:

- ID – Unique id for each Asbru Web Content Editor control tag to be used by your ASP.NET program code to identify the Asbru Web Content Editor control tag.
- Name – The HTML FORM data name.
- Content – The web content to be edited.
- Options – Optional additional parameters.

2.10 ASP.NET/C# User Control

As an alternative to using direct HTML and Javascript code and the programming language functions described above to use the Asbru Web Content Editor on a web page, you may prefer to use an ASP.NET user control.

```
<%@ Register tagprefix="Asbru" tagname="WebEditor" src="/webeditor/webeditor.cs.ascx" %>
```

The web page must register the Asbru:WebEditor ASP.NET control before it can be used.

If the Asbru Web Content Editor is installed to another location than the default '/webeditor/' folder, the 'src' location must be adjusted to your alternative installation folder.

```
<script type="text/javascript" src="/webeditor/webeditor.js"></script>
```

The HTML HEAD can include this SCRIPT tag to load the Asbru Web Content Editor.

If the Asbru Web Content Editor is installed to another location than the default '/webeditor/' folder, the 'file' address must be adjusted to your alternative installation folder.

```
<form method="post" onsubmit="WebEditorSubmit()" action="index.postback.cs.aspx" runat="server">
```

The HTML BODY FORM should call the "WebEditorSubmit" Javascript function on submit to save the web content editor content for postbacks.

```
<Asbru:WebEditor ID="webeditor" Name="content" Content=" This is my web content" Options="{ stylesheet: '/example.css', format: 'html' }" runat="server" />
```

The HTML BODY FORM must include this ASP.NET control tag to display the editable web content for the Asbru Web Content Editor. The editable web content is displayed where this ASP.NET control tag is included.

The editable web content is automatically sized to fill the available display area. Surround this SCRIPT tag with TABLE/TR/TD tags or similar to fix or limit the size of the editable web content.

The parameters passed to the Asbru:WebEditor control are:

- ID – Unique id for each Asbru Web Content Editor control tag to be used by your ASP.NET program code to identify the Asbru Web Content Editor control tag.
- Name – The HTML FORM data name.
- Content – The web content to be edited.
- Options – Optional additional parameters.

2.11 Technical Implementation Troubleshooting Notes

Please note the following web browser limitations in relation to using the Asbru Web Content Editor.

2.11.1 Hiding the web content editor in Safari

The current Safari web browser may not support using the Asbru Web Content Editor inside a … tag or similar at all - causing the web content editor input field to stop working after being hidden. Technically, the Safari web browser may “loose” IFRAMEs such as the Asbru Web Content Editor input field when they are hidden in this way. Alternative methods of hiding the Asbru Web Content Editor may be required. Please see the implementation of “tabs” in the Asbru Website Manager and the Asbru Web Content Management system for an example on how to hide the Asbru Web Content Editor in Safari.

2.11.2 HTML FORM tags in Safari

The current Safari web browser may not support using the Asbru Web Content Editor if the HTML code is invalid/malformed with the FORM tag located inside a TABLE tag between for example two TR tags. The FORM tag must be outside the TABLE tag or it must be inside a TH or TD tag.

3 Web Content Editor Options

As default the Asbru Web Content Editor is ready to use with pre-defined options. Many options can be changed to change the functionality and the look of the web content editor to your requirements.

3.1 Web Editor Skins

The Asbru Web Content Editor includes a number of pre-defined skins and supports custom skins, which determine how the web content editor toolbar etc. looks. As default the "default" skin is used. Use the "WebEditorSkin" Javascript function to change the skin. For example:

<script type="text/javascript"> WebEditorSkin('classic');</script>

The included pre-defined skins are:

default
classic
blue
blue2007
green
green2007
grey2007
olive
olive2007
red
red2007
white
white2007
yellow
yellow2007

3.2 Web Editor Options

A number of optional parameters for the "WebEditor" function can be used to modify the behaviour of the Asbru Web Content Editor. These optional parameters must be passed to the "WebEditor" function as its third parameter. Any number and combination of the optional parameters can be used.

The "webeditor" folder includes various "index*" files with examples of how to use the Asbru Web Content Editor with the various optional parameters.

Web Content CSS Style Sheet
A CSS style sheet file can be used for the edited content. The style sheet formatting will be applied to the content in the web content editor input field and generic classes/styles defined in the style sheet will be added to the web content editor toolbar's "formatclass" drop-down

list. The style sheet can be specified with a ‘stylesheet’ parameter. For example:

{ stylesheet: ‘/example.css’ }

Please note that because the given style sheet may be accessed by the web content editor from another location than where the edited content and the style sheet are located the ‘stylesheet’ parameter address should be the full, local path starting with a “/” - for example “/example.css” or “/css/default.css”. Due to general web browser security restrictions it may not be possible to load a style sheet from another website domain address than used for the web page with the web content editor.

Web Content HTML/XHTML Code Output Format

As default the web browser determines how the edited content’s HTML code is formatted. Different web browsers may output differently formatted HTML code. The default output formatting can be overridden with a ‘format’ parameters. For example:

{ format: ‘’ }

{ format: ‘html’ }

{ format: ‘xhtml’ }

Use format: ‘’ for default web browser formatted HTML code.

Use format:’ html’ for Asbru Web Content Editor formatted HTML code.

Use format: ‘xhtml’ for Asbru Web Content Editor formatted XHTML code.

Web Content Encoding

As default the Asbru Web Content Editor uses UTF-8 encoding to preview XHTML formatted content etc. The encoding can be specified with a ‘encoding’ parameter. For example:

{ encoding: ‘UTF-8’ }

Please note that this does not change the encoding of your web content - this only declares which encoding you are using.

Web Content Editor Output On Enter, Shift+Enter, Ctrl+Enter And Alt+Enter

As default the web browser determines what happens when the Enter key is pressed. Different web browsers may output different HTML codes. For example one web browser may output a BR tag on Enter while another web browser may output a P tag on Enter. The default output on Enter can be overridden with ‘onEnter’, ‘onShiftEnter’, ‘onCtrlEnter’ and ‘onAltEnter’ parameters. For example:

{ onEnter: '<p>', onShiftEnter: '<br>', onCtrlEnter: ' ', onAltEnter: '<hr>' }

{ onEnter: '<br>', onShiftEnter: '<p>', onCtrlEnter: ' ', onAltEnter: '<hr>' }

Web Content Editor Input Field Width And Height

As default the Asbru Web Content Editor will use 100% width and height for the web content editor input field. The width and height can be specified with 'width' and 'height' parameters. The minimum width and minimum height can be specified with 'minWidth' and 'minHeight' parameters. For example:

{ width: '800', height: '600' }

{ width: '100%', height: '100%', minWidth: '800', minHeight: '600' }

Web Content Editor Input Field Text Direction

As default the web browser determines if the content will be edited left-to-right for Western European languages or right-to-left for Arabic and Hebrew languages. The text direction can be specified with a 'direction' parameter. For example:

{ direction: '' }

{ direction: 'ltr' }

{ direction: 'rtl' }

Use direction: '' for default web browser handling of text editing direction.

Use direction: 'ltr' for left-to-right text editing.

Use direction: 'rtl' for right-to-left text editing direction.

Please note that the actual left-to-right / right-to-left editing is handled by the web browser and may require installation of additional operating system components.

Insert Hyperlink And Insert Media Dialog Windows

As default the Asbru Web Content Editor will use the advanced hyperlink and media manager dialog windows when used with ASP, ASP.NET, ColdFusion, JSP and PHP. For the pure Javascript version of the Asbru Web Content Editor the simple hyperlink and media dialog windows will be used as default. The dialog windows to be used can be specified with a 'manager' parameter. For example:

{ manager: '' }

{ manager: 'basic' }

{ manager: 'manager' }

Use manager: '' for the simple hyperlink and media dialog windows ("/webeditor/hyperlink.html" and "/webeditor/media.html").

Use manager: 'basic' for the basic hyperlink and media dialog windows

(“/webeditor/hyperlinkbasic.*” and “/webeditor/mediabasic.*”). Use manager: ‘manager’ for the advanced hyperlink and media manager dialog windows (“/webeditor/hyperlinkmanager.*” and “/webeditor/mediamanager.*”). Use manager: ‘MyOwnManager’ for your own custom programmed hyperlink and media manager dialog windows (“/webeditor/hyperlinkMyOwnManager.*” and “/webeditor/mediaMyOwnManager.*”).
Web Content Base Href
As default the Asbru Web Content Editor automatically detects the location of its web page and uses this as the BASE HREF for links and images etc. in the edited content. The base href to be used for the edited content in the web content editor can be specified with a ‘baseHref’ parameter. For example: { baseHref: ‘/’ } { baseHref: ‘http://www.yourwebsite.com/’ }
Web Content Editor Programming Language
As default the Asbru Web Content Editor automatically detects the programming language used for its web page / HTML FORM. If for some reason this should not work or if you need to use another programming language version, the webeditor programming language can be specified with a ‘language’ parameter. For example: { language: ‘’ } { language: ‘asp’ } { language: ‘aspx’ } { language: ‘cfm’ } { language: ‘jsp’ } { language: ‘php’ } { language: ‘xxx’ } Use language: ‘’ for the pure Javascript version of the Asbru Web Content Editor. Use language: ‘asp’ for the ASP version of the Asbru Web Content Editor. Use language: ‘aspx’ for the ASP.NET version of the Asbru Web Content Editor. Use language: ‘cfm’ for the ColdFusion version of the Asbru Web Content Editor. Use language: ‘jsp’ for the JSP version of the Asbru Web Content Editor.

Use language: ‘php’ for the PHP version of the Asbru Web Content Editor.

Use language: ‘xxx’ for your own custom programmed version of the Asbru Web Content Editor (“/webeditor/*.xxx”).

Web Content Editor Installation Folder

As default the Asbru Web Content Editor automatically detects its own installation location. If for some reason this should not work the webeditor folder path can be specified with a ‘rootpath’ parameter. Usually, this should not be required. For example:

{ rootpath: ‘/webeditor/’ }

Web Content Editor Toolbar

As default the Asbru Web Content Editor uses toolbars on the same web page / in the same frame/iframe as the input fields. A toolbar located in another frame/iframe than the input field can be specified with a ‘toolbar’ parameter. For example:

{ toolbar: ‘MyToolbarFrame’ }

{ toolbar: document.getElementById('MyToolbarFrame') }

{ toolbar: parent.document.getElementById('MyToolbarFrame') }

The ‘toolbar’ parameter value can be a string/variable with the toolbar frame’s “id”, or the ‘container’ parameter value can be the toolbar frame’s Javascript DOM element.

Please note that no new toolbar is created by this parameter. This only links the web content editor input field with the toolbar.

Web Content Editor Container And Dynamic Creation

As default the Asbru Web Content Editor is inserted on the web page where the “WebEditor” function is called. A HTML container (such as a DIV tag or a TD tag) inside which the web content editor input field should be created can be specified with a ‘container’ parameter. For example:

{ container: ‘MyWebEditorInputField’ }

{ container: document.getElementById(‘MyWebEditorInputField’) }

The ‘container’ parameter value can be a string/variable with the target container’s “id”, or the ‘container’ parameter value can be the container’s Javascript DOM element.

The web content editor input field completely replaces the container’s existing content.

Using the ‘container’ parameter Asbru Web Content Editor input fields can be created dynamically. For example, in web applications where users can add multiple input fields or input fields for various content need to be created dynamically on request.

3.3 Web Editor Toolbar Options

As default the Asbru Web Content Editor uses a compact toolbar with all the various web content editor toolbar functions in the toolbar and in its drop-down menus. However, different pre-defined toolbars are available and the toolbar is fully customisable. Toolbar drop-down list options can be changed; toolbar buttons etc. can be rearranged and removed and your own custom toolbar buttons and drop-down lists can be added for your own custom Javascript functions.

3.3.1 Web Editor Toolbar And Input Field Association

As default all Asbru Web Content Editor toolbars are linked to all Asbru Web Content Editor input fields on the same web page – all always reflecting the currently focused input field. A toolbar can be linked to a specific input field by specifying the input field id/name as the WebEditorToolbar function's first parameter. For example:

WebEditorToolbar("content")

3.3.2 Web Editor Toolbar Container And Dynamic Creation

A number of optional parameters for the "WebEditorToolbar" function can be used to modify the behaviour of the Asbru Web Content Editor. These optional parameters must be passed to the "WebEditorToolbar" function as its second parameter. Any number and combination of the optional parameters can be used.

Web Content Editor Toolbar Container
As default the toolbar is inserted on the web page where the "WebEditorToolbar" function is called. A HTML container (such as a DIV tag or a TD tag) inside which the web content editor toolbar should be created can be specified with a 'container' parameter. For example: { container: 'MyWebEditorToolbar' } { container: document.getElementById('MyWebEditorToolbar') } The 'container' parameter value can be a string/variable with the target container's "id", or the 'container' parameter value can be the container's Javascript DOM element. The web content editor toolbar completely replaces the container's existing content. Using the 'container' parameter Asbru Web Content Editor toolbars can be created dynamically. For example, in web applications where users can add multiple input fields or input fields for various content need to be created dynamically on request.

3.3.3 Web Editor Toolbar Drop-Down List Options

A number of optional parameters for the "WebEditorToolbar" function can be used to modify the behaviour of the Asbru Web Content Editor. Any number and combination of the optional parameters can be used.

Style Formatting Options
The "formatclass" drop-down list includes the generic classes/styles defined in the style sheet specified for the web content editor input field. Please see 3.2 Web Editor Options for details.
Block Formatting Options

As default the "formatblock" drop-down list includes all the various block formatting options, which can be applied to text selections. The "formatblock" drop-down options can be specified with a 'formatblock' parameter. For example:
 	
{ formatblock: {'Normal':'<p>', 'Paragraph':'<p>', 'Formatted':'<pre>', 'Heading 1':'<h1>', 'Heading 2':'<h2>', 'Heading 3':'<h3>', 'Heading 4':'<h4>', 'Heading 5':'<h5>', 'Heading 6':'<h6>', 'Numbered List':' ', 'Bulleted List':' ', 'Directory List':'<dir>', 'Menu List':'<menu>', 'Definition Term':'<dt>', 'Definition':'<dd>', 'Address':'<address>'}
 	
Please note that the supported options are determined by the web browser. Unwanted options can be removed from the drop-down list, but additional options cannot simply be added.

Font Name Options
As default the "fontname" drop-down list includes a number of common web page font options, which can be applied to text selections. The "fontname" drop-down options can be specified with a 'fontname' parameter. For example: { fontname: [' Times New Roman, Times, serif', 'Arial, Helvetica, sans-serif', ' Courier New, Courier, monospace'] } { fontname: {'Times New Roman': ' Times New Roman, Times, serif', 'Arial/Helvetica': ' Arial, Helvetica, sans-serif', 'Courier': 'Courier New, Courier, monospace'} } Use a simple list of font names in square brackets to display the same font names in the "fontname" drop-down list as actually applied to the content. Use a name/value list of font names and values in curly brackets to display different font names in the "fontname" drop-down list than actually applied to the content.
Font Size Options
As default the "fontsize" drop-down list includes a number of common web page font size options, which can be applied to text selections. The "fontsize" drop-down options can be specified with a 'fontsize' parameter. For example: { fontsize: {8:1, 10:2, 12:3, 14:4, 18:5, 24:6, 36:7} }

3.3.4 Web Editor Toolbar Buttons And Drop-Down Lists

As default the Asbru Web Content Editor uses a compact toolbar with all the various web content editor toolbar functions in the toolbar or in its drop-down menus. However, different pre-defined toolbars are available and the toolbar is fully customisable.

Pre-Defined Toolbar Options
As default the Asbru Web Content Editor uses a compact toolbar with all the various web content editor toolbar functions in the toolbar and in its drop-down menus. Use of various pre-defined toolbars can be specified with a 'toolbar' parameter. For example: { toolbar: '-COMPACT-' } { toolbar: '-COMPACT2-' } { toolbar: '-MINIMAL-' } { toolbar: '-FULL -' } { toolbar: '-ALL -' } Use toolbar: '-COMPACT-' for a compact toolbar on three lines with some toolbar buttons in drop-down menus. (This is the default toolbar). Use toolbar: '-COMPACT2-' for a compact toolbar on two lines with some toolbar buttons in drop-down menus. Use toolbar: '-MINIMAL-' for a minimal toolbar without text formatting toolbar buttons etc. Use toolbar: '-FULL-' for a full toolbar with all toolbar buttons in the actual toolbar. Use toolbar: '-ALL-' for an expanded toolbar with both drop-down menus and all toolbar buttons in the actual toolbar. Usually, this should only be used for testing.
Customized Toolbar Options
As default the Asbru Web Content Editor uses a compact toolbar with all the various web content editor toolbar functions in the toolbar and in its drop-down menus. However, the toolbar is fully customisable. Toolbar buttons etc. can be rearranged and removed and your custom toolbar buttons etc. for your own custom Javascript functions can be added. Customized toolbars can be specified with 'toolbar1', 'toolbar2', 'toolbar3', 'toolbar4', 'toolbar5', 'toolbar6', 'toolbar7', 'toolbar8' and 'toolbar9' parameters. For example: { toolbar1: "formatclass formatblock fontname fontsize bold italic underline forecolor backcolor superscript subscript strikethrough removeformat help", toolbar2: "cut copy paste clean delete selectall undo redo specialcharacter insertmedia imagemap iframe createlink mailto anchor unlink inserthorizontalrule insertorderedlist insertunorderedlist outdent indent justifyleft justifycenter justifyright justifyfull nobr ltr", toolbar3: "createtable tableproperties insertcaption insertrowhead insertrowfoot rowproperties insertrowabove insertrowbelow deleterow splitcellrows columnproperties insertcolumnleft insertcolumnright deletecolumn splitcellcolumns cellproperties insertcellleft insertcellright deletecell splitcell mergecells

import find printbreak print preview rtl", toolbar4: "form submitbutton resetbutton backbutton imagebutton file button text password hidden textarea checkbox radio select position forwards backwards front back abovetext belowtext box spellcheck viewdetails viewsource save" }

Customized Toolbar Button And Drop-Down List Names	
The following toolbar button and drop-down list names can be used to specify customized toolbars for the Asbru Web Content Editor's default functionality.	
Text Formatting and Style	
formatclass	Text Style drop-down list
formatblock	Text Format drop-down list
fontname	Font Name drop-down list
fontsize	Font Size drop-down list
textformat	Text Formatting drop-down menu
bold	Bold button
italic	Italic button
underline	Underline button
forecolor	Text Colour button
backcolor	Background Colour button
strikethrough	Strikethrough button
superscript	Superscript button
subscript	Subscript button
removeformat	Remove Text Formatting button
Editing	
cut	Cut button
copy	Copy button
paste	Paste button

clean	Clean HTML Code button
delete	Delete button
selectall	Select All button
redo	Redo button
undo	Undo button
Web Content	
specialcharacter	Insert Special Character/Code button
insertmedia	Insert Media (Image) button
imagemap	Define Imagemap button
createlink	Insert Link button
iframe	Insert Frame button
mailto	Insert Mail To Link button
anchor	Insert Anchor/Bookmark button
unlink	Unlink button
inserthorizontalrule	Insert Horizontal Rule button
Indentation and Alignment	
outdent	Decrease Indent button
indent	Increase Indent button
justify	Justify drop-down menu
justifyleft	Justify Left button
justifycenter	Justify Center button
justifyright	Justify Right button
justifyfull	Justify Full button
nobr	Do Not Break button
insertorderedlist	Ordered List button

insertunorderedlist	Unordered List button
Table Editing	
table	Table drop-down menu
createtable	Insert Table button
tableproperties	Table Properties button
insertcaption	Insert Caption button
insertrowhead	Insert Header Row button
insertrowfoot	Insert Footer Row button
rowproperties	Row Properties button
insertrowabove	Insert Row Above button
insertrowbelow	Insert Row Below button
deleterow	Delete Row button
splitcellrows	Split Cell Rows button
columnproperties	Column Properties button
insertcolumnleft	Insert Column Left button
insertcolumnright	Insert Column Right button
deletecolumn	Delete Column button
splitcellcolumns	Split Cell Columns button
cellproperties	Cell Properties button
insertcellleft	Insert Cell Left button
insertcellright	Insert Cell Right button
deletecell	Delete Cell button
splitcell	Split Cell button
mergecells	Merge Cells button
Form Editing	

formmenu	Form drop-down menu
form	Insert Form button
submitbutton	Insert Submit Button button
resetbutton	Insert Reset Button button
backbutton	Insert Back Button button
imagebutton	Insert Image Button button
file	Insert File Selector button
button	Insert Button button
text	Insert Text Input button
password	Insert Password Input button
hidden	Insert Hidden Input button
textarea	Insert Text Area Input button
checkbox	Insert Checkbox button
radio	Insert Radio Button button
select	Insert Select List button
Positioning	
positionmenu	Absolute Positioning drop-down menu
position	Absolute Positioning button
forwards	Bring Forwards button
backwards	Send Backwards button
front	Bring To Front button
back	Send To Back button
abovetext	Bring Above Text button
belowtext	Send Below Text button
box	Insert Box button

Special	
help	Help button
import	Import File button
find	Find button
printbreak	Insert Print Page Break button
print	Print button
preview	Preview button
save	Save / Submit button
viewdetails	Show Hidden Details button
viewsource	Show HTML Source button
spellcheck	Check Spelling button
ltr	Left-To-Right Editing button
rtl	Right-To-Left Editing button
BlockDirLTR	Left-To-Right Editing button
BlockDirRTL	Right-To-Left Editing button

3.4 Web Editor DOM Inspector Options

3.4.1 Web Editor DOM Inspector And Input Field Association

As default all Asbru Web Content Editor DOM Inspectors are linked to all Asbru Web Content Editor input fields on the same web page – all always reflecting the currently focused input field. A DOM inspector can be linked to a specific input field by specifying the input field id/name as the WebEditorDOMInspector function's first parameter. For example:

WebEditorDOMInspector ("content")

3.4.2 Web Editor DOM Inspector Container And Dynamic Creation

As default the Asbru Web Content Editor DOM Inspector is inserted on the web page where the "WebEditorDOMInspector" function is called. A HTML container (such as a DIV tag or a TD tag) inside which the web content editor DOM inspector should be created can be specified as the WebEditorDOMInspector function's second parameter. For example:

WebEditorDOMInspector ("content", "MyWebEditorDOMInspector")

The ‘container’ parameter value can be a string/variable with the target container’s “id”, or the ‘container’ parameter value can be the container’s Javascript DOM element.

The web content editor DOM inspector completely replaces the container’s existing content.

Using the ‘container’ parameter Asbru Web Content Editor DOM inspectors can be created dynamically. For example, in web applications where users can add multiple input fields or input fields for various content need to be created dynamically on request.

3.5 Web Content Editor Example Web Pages

3.5.1 Editor with advanced dialog windows

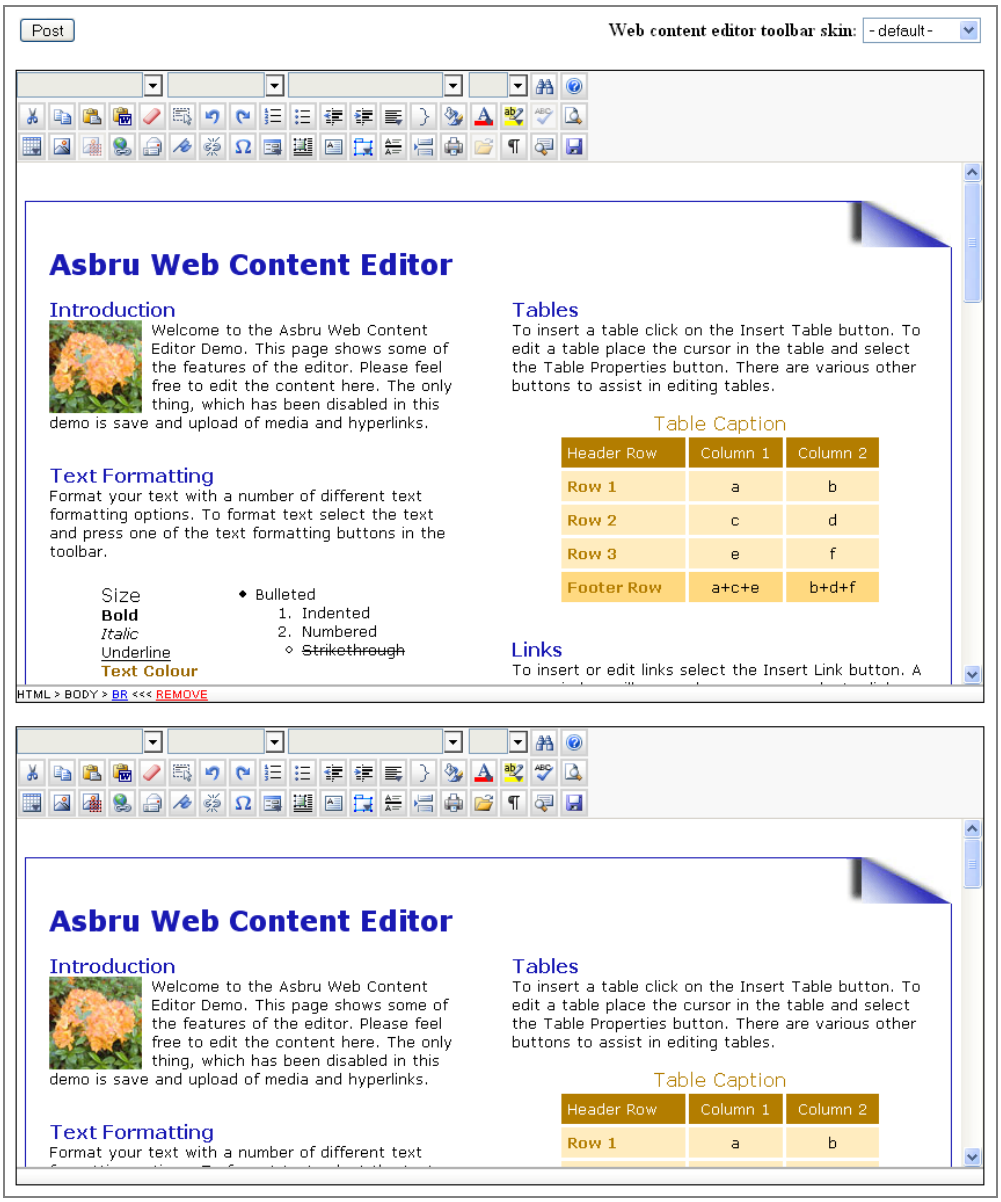
The Asbru Web Content Editor can use advanced Insert Hyperlink and Insert Media dialog windows.

See “index.manager.html”, “index.manager.asp”, “index.manager.aspx”, “index.manager.cfm”, “index.manager.jsp” and “index.manager.php” in the “webeditor” folder and 5.3 Hyperlink And Media Dialog Windows for details.

3.5.2 Multiple editors on the same webpage with separate toolbars

Multiple Asbru Web Content Editors can be located on the same webpage. Each editor may have a separate toolbar.

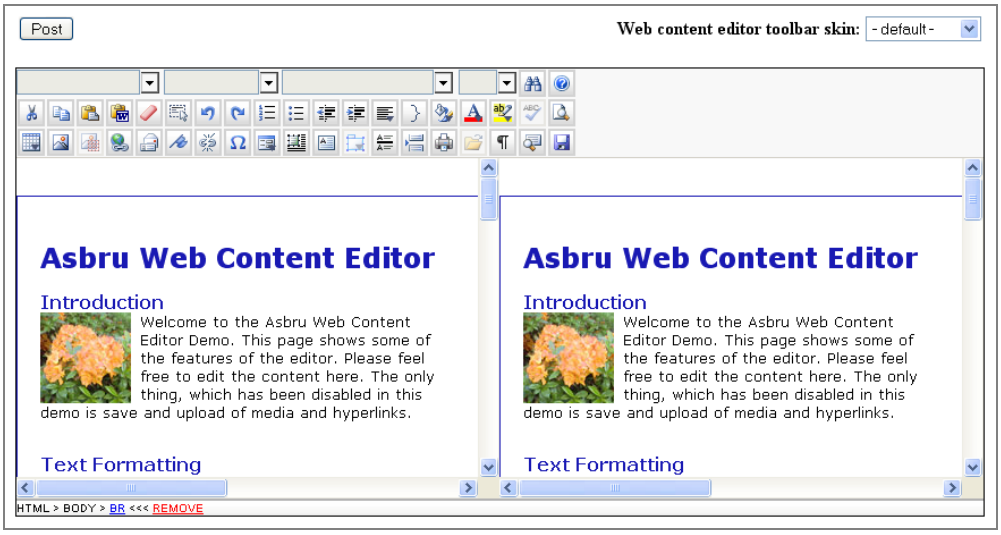
Please see “index.multi.html” in the “webeditor” folder for details.



3.5.3 Multiple editors on the same webpage with shared toolbar

Multiple Asbru Web Content Editors can be located on the same webpage. All editors may share a single toolbar.

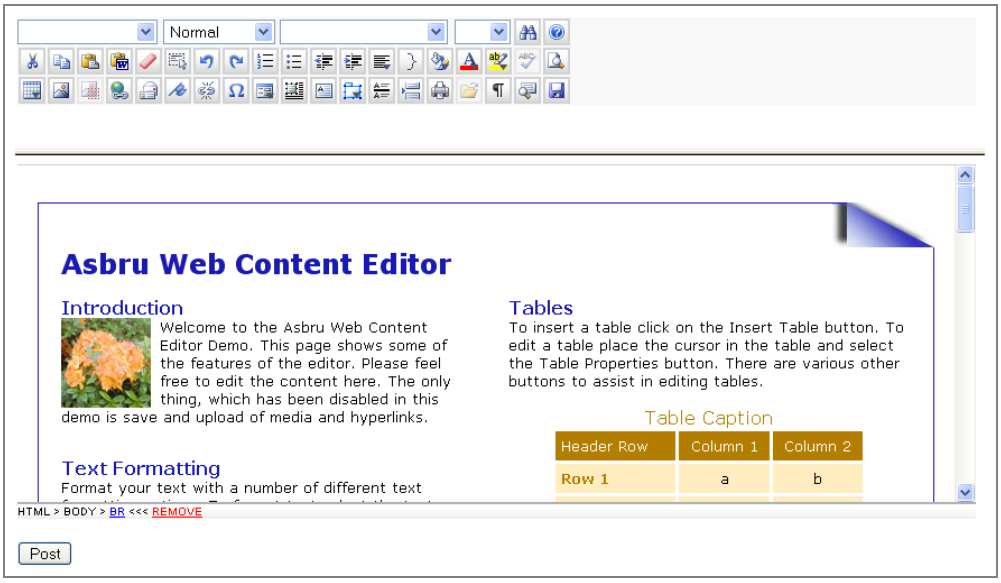
Please see “index.multi2.html” in the “webeditor” folder for details.



3.5.4 Toolbar and editable web content in HTML FRAMESET FRAMEs

The Asbru Web Content Editor toolbar and the editable web content may be located in separate HTML FRAMEs within a HTML FRAMESET.

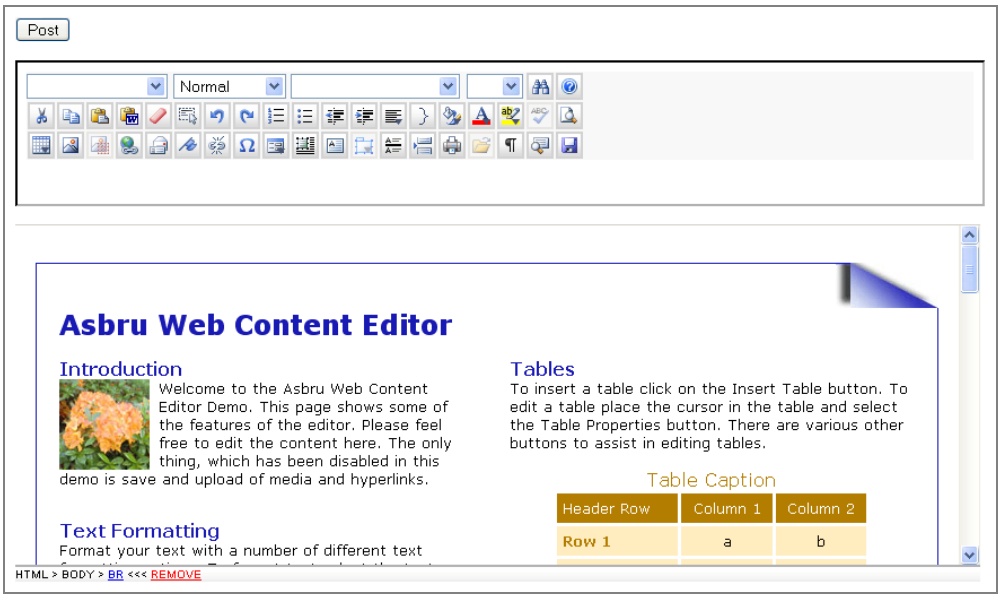
Please see “index.frameset.html” in the “webeditor” folder for details.



3.5.5 Toolbar in HTML IFRAME

The Asbru Web Content Editor toolbar may be located in a HTML IFRAME with the editable content located in the main web page.

Please see “index.iframe.html” in the “webeditor” folder for details.



3.5.6 Right to left editing

The Asbru Web Content Editor supports right to left editing for Arabic and Hebrew languages (as well as the default left to right editing).

Pease see **Error! Reference source not found. Error! Reference source not found.** – Direction for details on how to specify a parameter to make the web content editor use right to left editing instead of the default left to right editing.

You may also want to add toolbar buttons for switching between left to right editing and right to left editing. As default these two toolbar buttons are not displayed. To display these two toolbar buttons you need to use a custom toolbar with two toolbar buttons named "BlockDirLTR" and "BlockDirRTL". Please see 5.4 Custom Toolbar for details.

4 Handling posted web content

The Asbru Web Content Editor content is included and posted from your webpage as standard HTML FORM data. This makes it very simple to handle the posted web content. It also allows you to easily integrate the Asbru Web Content Editor with any website and web-based applications and databases simply by replacing any existing HTML FORM TEXTAREA field with the Asbru Web Content Editor, which automatically degrades to a TEXTAREA field for web browsers without support for visual web content editing.

A simple file editor example web application is included as a simple example of how to use the Asbru Web Content Editor to update content on your website.

4.1 Standard HTML FORM Data

The “webeditor” folder includes “preview.asp”, “preview.aspx”, “preview.cfm”, “preview.jsp” and “preview.php” files with simple examples of how to extract and display posted web content from the Asbru Web Content Editor. Integrating the Asbru Web Content Editor only requires accessing automatically handled HTML FORM data variables using the built-in features of your preferred scripting language such as:

- ASP: Request.Form("content")
- ASP.NET/C#: Request.Form.Get("content")
- ColdFusion: Form.content
- JSP: request.getParameter("content")
- PHP: stripslashes($HTTP_POST_VARS["content"])

On the web server you have to handle the posted web content just like any other posted HTML FORM data to store it in files, databases, web content management systems and any other web-based applications and services. How this is done is up to you and your web-based application and database.

Alternatively, you may also use the Asbru Web Content Editor Javascript programming API to get and handle the content instead of simply posting the content to your web server as ordinary HTML FORM data. Please see 5.7 Javascript Programming API for details.

4.2 Simple File Editor Example

Even if you only have a static website consisting of pure HTML web pages without any additional web applications, you can easily use the Asbru Web Content Editor to update parts of your website.

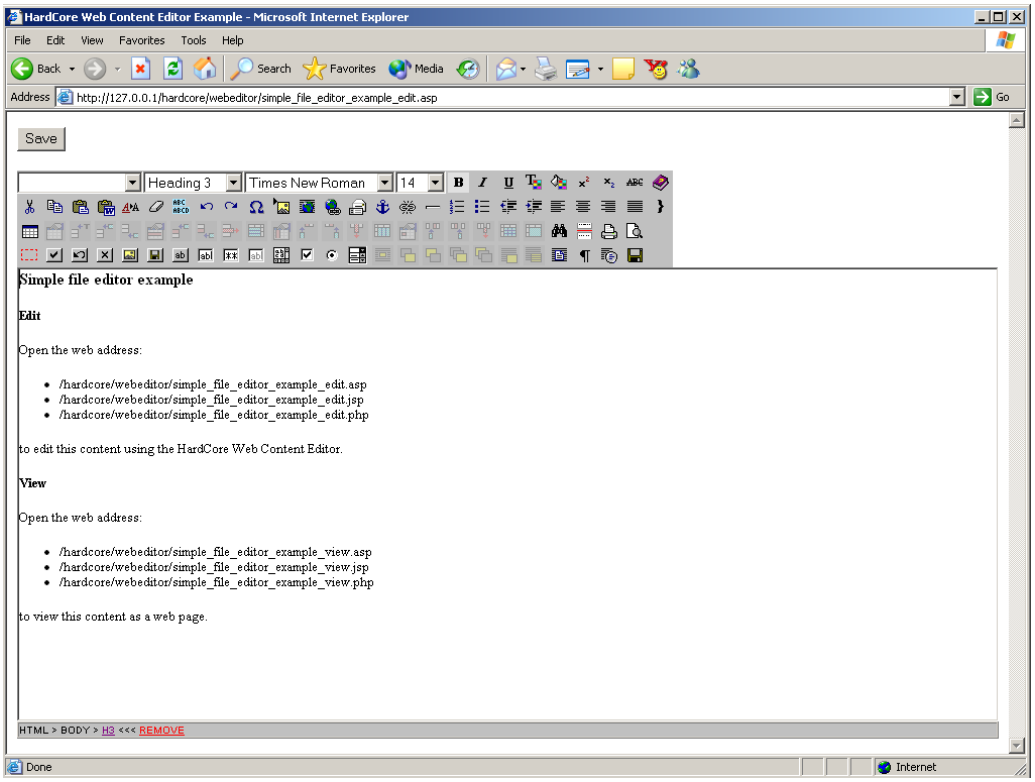
The “webeditor” folders contains a number of “simple_file_editor_example” files:

- simple_file_editor_example.html
  HTML file which is updated by the simple file editor example web application.

- simple_file_editor_example.asp/.aspx/.cfm/.jsp/.php
  Simple ASP/ASP.NET/ColdFusion/JSP/PHP program file which updates the content of the "simple_file_editor_example.html" file in the Asbru Web Content Editor.

Simply copy and modify the simple_file_editor_example program file to make your own simple file editors to update and display HTML file content on your website.

Please note that you should restrict access to your copies of the "simple_file_editor_example.asp/.aspx/.cfm/.jsp/.php" file (i.e. using your web server's access control features or by keeping the file names secret), or any Internet user with access to your website and knowledge of your "editor" file names may be able to update that content on your website.



5 Customization and extension

The Asbru Web Content Editor includes several features that enable you to customise and extend it:

- Custom skins
- Internationalisation with additional language translations
- Customisable hyperlink and media dialog windows
- Customisable and extendable toolbar
- Javascript programming API for setting, modifying and getting editable web content

5.1 Asbru Web Content Editor Custom Skins

The Asbru Web Content Editor supports "skins" to customize the look of the web content editor toolbar etc. A number of pre-defined skins are included. Each skin is a sub-folder under the "/webeditor/" folder and contains a "webeditor.css" file as well as web content editor toolbar button image files.

To add your own custom skin simply copy one of the included skins and modify the "webeditor.css" file and the image files.

5.2 Internationalisation With Additional Language Translations

The Asbru Web Content Editor supports internationalisation with translations of all text to other languages than the default (English) and automatic detection of each user's language preferences. The Asbru Web Content Editor also supports right to left editing for Arabic and Hebrew languages.

5.2.1 Asbru Web Content Editor texts

All texts in the Asbru Web Content Editor are located in the "/webeditor/properties.js" file. This file contains the default texts used if no specific language is selected and available.

Additional language files can be created with translations of all texts to other languages. As default English, French, Spanish, German, Czech, Slovak and Danish language translation files are included. For example, the "/webeditor/properties_da.js" file contains all texts in the Asbru Web Content Editor translated to Danish.

To add support for other languages, simply copy the "/webeditor/properties.js" file and translate its contents (and add the language to the "/webeditor/webeditor.properties.js" as described below). The copied file must be named "/webeditor/properties_xx.js" where "xx" is the ISO 639 language code such as:

- ar – Arabic
- de – German

- en – English
- es – Spanish
- fr – French
- it - Italian
- ja – Japanese
- zh - Chinese

Additional language files for language variations can be created with translations for individual countries. To add support for language variations, simply copy the "/webeditor/properties.js" file or another language file and translate its contents. The copied file must be named "/webeditor/properties_xx_YY.js" where "xx" is the ISO 639 language code as described above and where "YY" is the ISO 3166 country code such as:

- AU – Australia
- CA – Canada
- GB – United Kingdom
- US – United States

To change the default language, simply replace the default English "/webeditor/properties.js" file with a copy of another language file.

To add support for new language files, the "/webeditor/webeditor.properties.js" file must be modified. Add the language/country code to the following line at the top of the file:

```
var webeditor_languages = "|ar|da|de|en|es|fr|it|ja|zh|cs|cz|sk|";
```

To add support for a new language file named "/webeditor/properties_xx.js" modify the line to:

```
var webeditor_languages = "|ar|da|de|en|es|fr|it|ja|zh|cs|cz|sk|xx";
```

To add support for a new language file named "/webeditor/properties_xx_YY.js" modify the line to:

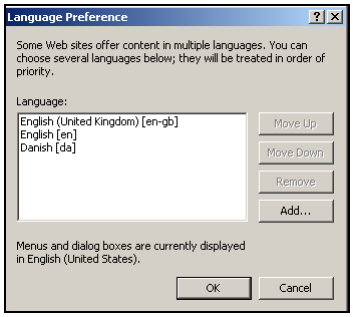
```
var webeditor_languages = "|ar|da|de|en|es|fr|it|ja|zh|cs|cz|sk| xx_YY|";
```

The language/country codes must be separated and enclosed by | characters.

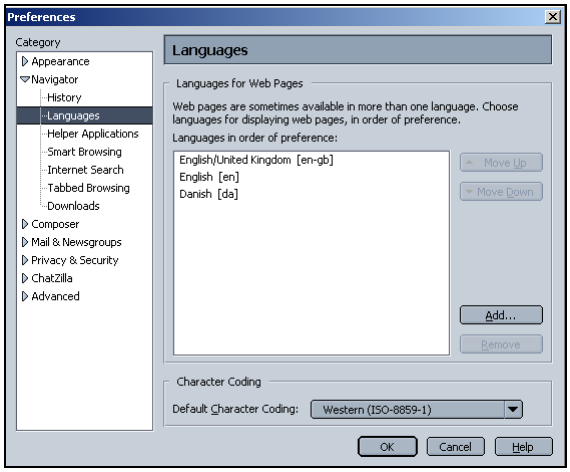
5.2.2 User Language Preferences

Each individual user can select the language used by the Asbru Web Content Editor through his/her web browser's language preferences settings.

In Microsoft Internet Explorer language preferences can be selected through the Tools – Internet Options – General – Languages menu/window. Please see the Microsoft Internet Explorer documentation for details.



In Mozilla language preferences can be selected through the Edit – Preferences – Navigator – Languages menu/window. Please see the Mozilla documentation for details.



The Asbru Web Content Editor will detect the selected web browser language preferences and use one of the selected languages if available. If none of the selected languages are available the default language will be used.

5.3 Hyperlink And Media Dialog Windows

As described earlier in this document, the Asbru Web Content Editor includes both simple standard hyperlink and media dialog windows as well as advanced hyperlink and media manager dialog windows. You probably want to customise these and integrate them with your website and web-based application and database.

You can do this by modifying the default standard hyperlink and media dialog windows (“hyperlink.html” and “media.html”). You can also make your own basic hyperlink and media “manager” scripts (“hyperlinkbasic.asp”, “hyperlinkbasic.aspx”, “hyperlinkbasic.cfm”, “hyperlinkbasic.jsp”, “hyperlinkbasic.php”, “mediabasic.asp”, “mediabasic.aspx”, “mediabasic.cfm”, “mediabasic.jsp”, “mediabasic.php”) and advanced hyperlink and media “manager” scripts (“hyperlinkmanager.asp”, “hyperlinkmanager.aspx”, “hyperlinkmanager.cfm”, “hyperlinkmanager.jsp”, “hyperlinkmanager.php”, “mediamanager.asp”, “mediamanager.aspx”, “mediamanager.cfm”, “mediamanager.jsp”, “mediamanager.php”) and configure the Asbru Web Content Editor to use them instead of the default standard hyperlink and media dialog window scripts by using the ‘scripting language’

and ‘basic’ or ‘manager’ parameter as described in the first part of this document. Examples of advanced hyperlink and media manager dialog windows are included in ASP, ASP.NET, ColdFusion, JSP and PHP. You can relatively easily modify these to other scripting languages such as Perl.

5.3.1 Quicklinks

You may simply want to use default hyperlink and media dialog window scripts but add your own hyperlinks and images to the “Quicklinks”. This is easily done by locating the

<option value="http://www.asbrusoft.com/">www.asbrusoft.com

and

<option value="http:// www.asbrusoft.com /logo.gif">Asbru logo

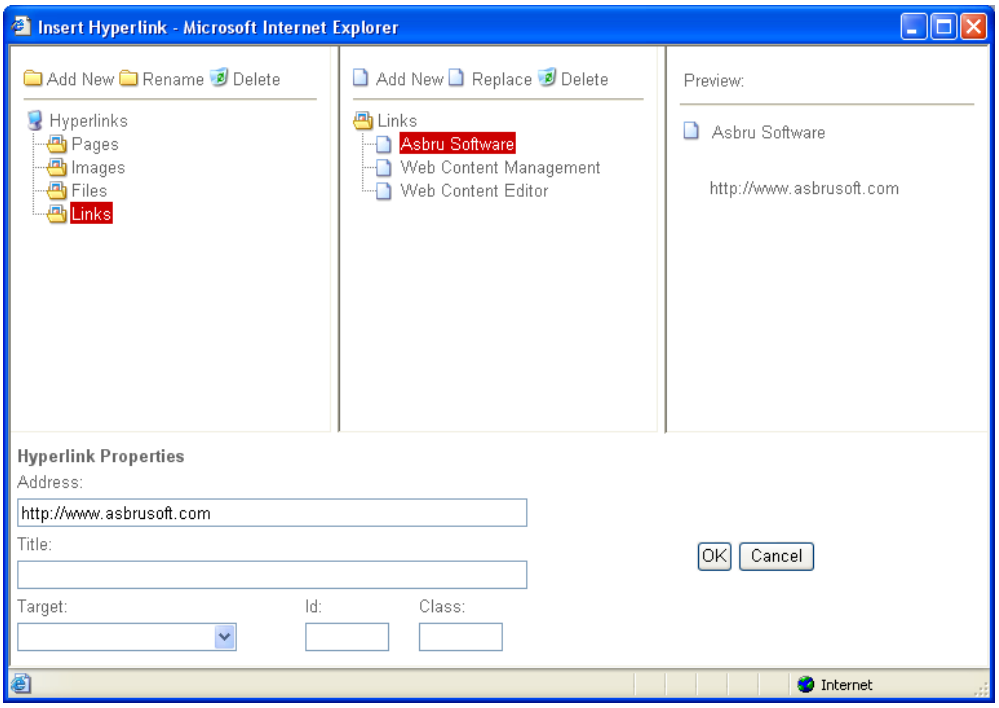
program lines in the “hyperlink.html” and “media.html” files and replace them with your own options depending on your websites and web-based applications and databases:

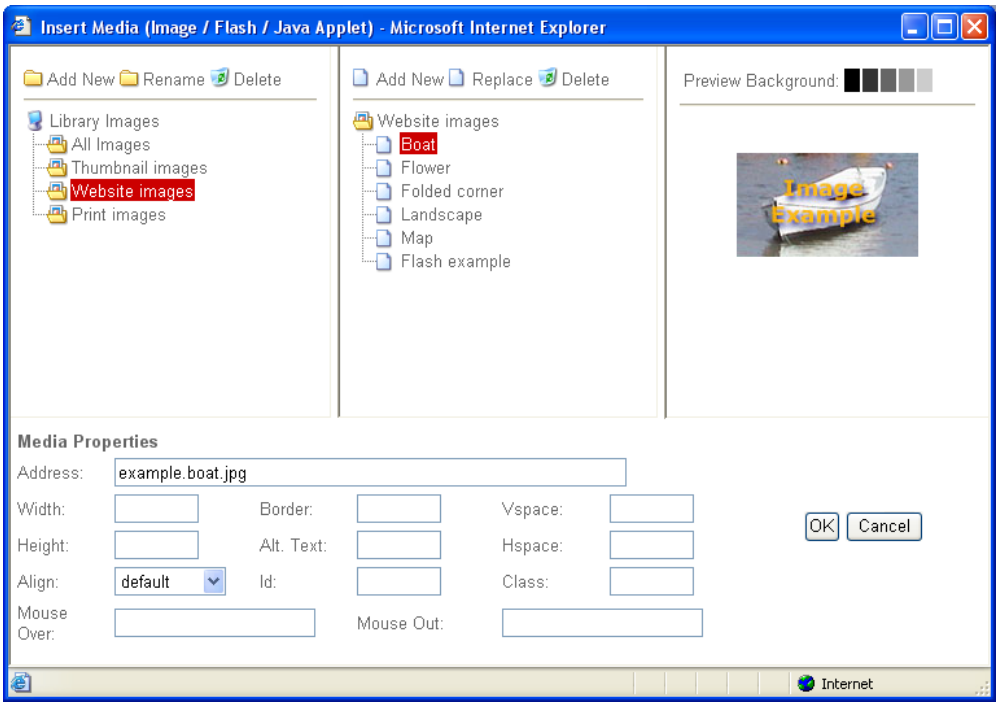
- Simply add static options as direct HTML OPTION tags.
- Write a small script to list the images in your own website image library folder and generate HTML OPTION tags for each of your images.
- Write a small script to extract hyperlink and media lists from your web-based applications and generate HTML OPTION tags for each of them.
- Write a small script to extract hyperlink and media lists from your databases and generate HTML OPTION tags for each of them.

Exactly how to do this is out of the scope of this document. Please see one of the many available books and websites about ASP, ASP.NET, ColdFusion, JSP, PHP and database programming.

5.3.2 Hyperlink and media managers

You may also want to make your own complete hyperlink and media managers for handling large amounts of options and dynamically creating, modifying and deleting options. Examples of advanced hyperlink and media manager dialog windows are included in ASP, ASP.NET, ColdFusion, JSP and PHP. You can easily modify these to other scripting languages such as Perl.





Examples of advanced hyperlink and media manager dialog windows are included in ASP, ASP.NET, ColdFusion, JSP and PHP. You can easily modify these to other scripting languages such as Perl.

For the advanced hyperlink and media manager dialog windows to be useful, they must be integrated with your web applications. I.e. to list, create and delete content categories and content items in your web content management system. How this is done depends on your web applications. For an example of how the Asbru Web Content Editor is integrated with a web content management system, please see our Asbru Web Content Management system: http://wcm.asbrusoft.com/.

5.3.3 Additional hyperlink and media dialog window attributes

The default "hyperlink.html" and "media.html" standard dialog windows include the typical attributes used for hyperlinks and images. However, the Asbru Web Content Editor also supports some additional attributes, which you may want to use.

5.3.3.1 Hyperlink

A couple of optional additional form data can be used in relation to advanced scripting and style sheets:

- HTML ID attribute – for unique identification of each individual hyperlink or image in your web content.
- HTML CLASS attribute – for classification of different classes of hyperlinks and images.

To use these you must add HTML FORM INPUT fields for them and modify the "linkit" Javascript function (in "hyperlink.html") to pass them to the Asbru Web Content Editor. The "linkit" Javascript function includes the program lines:

```
var htmlid = '';
var htmlclass = '';
```

These should be changed to get their value from your additional HTML FORM INPUT fields instead of simply being set to empty strings.

5.3.3.2 Media

A couple of optional additional form data can be used in relation to advanced scripting and style sheets and a number of additional image attributes can be defined:

- HTML ID attribute – for unique identification of each individual hyperlink or image in your web content.
- HTML CLASS attribute – for classification of different classes of hyperlinks and images.
- HTML IMG VSPACE attribute – for setting vertical spacing for images.
- HTML IMG HSPACE attribute – for setting horizontal spacing for images.
- HTML IMG ALIGN attribute – for setting alignment for images.

To use these you must add HTML FORM INPUT fields for them and modify the “linkit” Javascript function (in “media.html”) to pass them to the Asbru Web Content Editor. The “linkit” Javascript function includes the program lines:

```
var vspace = '';
var hspace = '';
var align = '';
var htmlclass = '';
var htmlid = '';
```

These should be changed to get their value from your additional HTML FORM INPUT fields instead of simply being set to empty strings.

5.4 Custom Toolbar Buttons And Functionality

If no parameters are passed to the WebEditorToolbar function the default toolbar will be displayed. However, the toolbar can be customized by passing parameters defining which options to display and in which order.

5.4.1 Custom functionality

In addition to the built-in functionality, you can also add your own functionality or replace the built-in functionality.

5.4.1.1 Toolbar buttons

To replace the built-in functionality, simply define your own Javascript functions named “webeditor_custom_xxx”. Replace “xxx” with one of the toolbar option names as listed above.

Example – To replace the built-in “help” functionality use:

```
<script>
function webeditor_custom_help() {
  alert('Please see the user guide');
}
WebEditorToolbar();
</script>
```

To add your own functionality, simply define your own Javascript functions named “webeditor_custom_xxx”, add “xxx” as an option to your toolbar, and add your own “xxx.gif” button image file to the “webeditor” folder. Replace “xxx” with a toolbar option name of your own choice.

Example – To add your own “helloworld” functionality and display it as your only toolbar option use:

```
<script>
function webeditor_custom_helloworld() {
  alert('Hello World');
}
```

```
WebEditorToolbar(null, { toolbar1: "helloworld" } );
</script>
```

5.4.1.2 Custom toolbar list selectors and other toolbar items

Other custom toolbar items than "buttons" such as select lists etc. can be added through your own custom Javascript functions.

If you add a custom toolbar item named "xxx" and define your own Javascript function named "webeditor_custom_toolbar_xxx" then the HTML code returned by that Javascript function will be added to the toolbar instead of a regular custom toolbar button.

For example, the following Javascript function will add a toolbar selector list for a custom toolbar item named "helloworld":

```
<script>
function webeditor_custom_toolbar_helloworld() {
    var html = '';
    html += '<td colspan="5">';
    html += '<select unselectable="on" class="webeditor_select" id="helloworld"
title="Hello World" style="width: 125px;">';
    html += '  <option value=""> <\/option>';
    html += '  <option value="hello">Hello<\/option>';
    html += '  <option value="world">World<\/option>';
    html += '<\/select>';
    html += '<\/td>';
    return html;
}

function webeditor_custom_helloworld() {
    alert('Hello ' + document.getElementById("helloworld").value);
}

WebEditorToolbar(null, { toolbar1: "helloworld" } );
</script>
```

The output from the webeditor_custom_toolbar_xxx Javascript function is not limited to select lists like the above - any type of HTML code can be returned and added to the toolbar.

5.4.1.3 Web content editor focus

Two special functions for "focusing" and "blurring" the web content editor input field(s) are also supported:

```
<script>
function webeditor_custom_onfocus(name) {
  alert('Focused '+name);
}
function webeditor_custom_onblur(name) {
  alert('Blurred '+name);
}
 </script>
```

These two functions are called when the web content editor input field is “focused” and “blurred”.

The functions can be used to only display the toolbar and the DOM inspector when the web content editor input field is focused and otherwise hide the toolbar and DOM inspector when the web content editor input field is not focused.

Please see “index.onfocus-onblur.html” in the “webeditor” folder for details

5.4.1.4 Custom web content editor event handling

A special function for detecting and handling web content editor events is supported:

```
<script>
function webeditor_custom_event(event) {
  if (event.type == XXX) {
    /* handle event here */
    return true;
  } else {
    /* let web content editor handle event */
    return false;
  }
}
</script>
```

This function is called on all Javascript events handled by the web content editor input field. The function parameter is the Javascript Event object to be handled.

5.5 Custom Encoding/Decoding And Reformatting Of HTML Content

For various reasons you may need/want to custom encode your content while it is being edited in the web content editor or modify the content when/before it is being submitted to the web server:

- Web browsers and the Asbru Web Content Editor may sometimes modify link and image URLs – for example when they are copied and pasted. The Asbru Web Content Editor tries to repair link and image URLs that may have been modified by the web browser. However, it may not always succeed in doing so depending on your specific edited content and web application. To avoid this you may want to use your own custom URL pattern matching and replacement settings.

- Web browsers and the Asbru Web Content Editor may generate HTML code, which is not compatible with your web applications, or which you do not allow users to submit to the web server. To avoid this you may want to “strip” certain HTML tags and attributes from your HTML content and/or to “custom decode” (to clean up or otherwise modify) your HTML content before it is submitted to the web server.

- Your content may include ASP, ASP.NET, ColdFusion, JSP, PHP or similar tags, which web browsers may not handle correctly and which web browsers may modify. To avoid this you may want to “custom encode” your HTML content into another format while it is being edited to avoid it being incorrectly modified by the web browsers (and “custom

decode" the content again before it is submitted to the web server).

- Your web content management system or other web applications may use non-standard HTML-like custom tags which web browsers may not handle correctly and which web browsers may modify. To avoid this you may want to "custom encode" your HTML content into another format while it is being edited to avoid it being incorrectly modified by the web browsers (and "custom decode" the content again before it is submitted to the web server).

5.5.1 HTML content URL attributes replacement patterns

You can modify certain link and image URLs in your HTML content by adding "webeditor.shortenLocalURLsRegExp" Javascript statements on your web pages with the Asbru Web Content Editor. The "webeditor.shortenLocalURLsRegExp" Javascript statements must be placed on your web pages after that the Asbru Web Content Editor program files have been loaded and before the Asbru Web Content Editor functions are called.

For example:

```
webeditor.shortenLocalURLsRegExp["^http://.*/images/"] = "images/";
webeditor.shortenLocalURLsRegExp["^/images/"] = "images/";
webeditor.shortenLocalURLsRegExp["^/../../../../../images/"] = "images/";
```

- webeditor.shortenLocalURLsRegExp["^http://.*/images/"] = "images/";
  will change absolute URLs for your "images" folder (for example "http://www.youerwebsite.com/images/home.gif") to a relative URL for your "images" folder (for example "images/home.gif".

- webeditor.shortenLocalURLsRegExp["^/images/"] = "images/";
  will remove the leading "/" path from URLs for your "images" folder (for example "/images/home.gif") to a relative URL for your "images" folder (for example "images/home.gif".

- webeditor.shortenLocalURLsRegExp["^/../../../../../images/"] = "images/";
  will remove the leading "/../../../../../" path from URLs for your "images" folder (for example "/../../../../../images/home.gif") to a relative URL for your "images" folder (for example "images/home.gif".

This functionality is intended to "shorten"/repair URLs, which may have been incorrectly modified by your web browser and the Asbru Web Content Editor. However, this functionality can also be used to modify your URLs in other ways – for example changing URLs from one folder to another. The "webeditor.shortenLocalURLsRegExp" keys and values are standard Javascript regular expression search patterns and replacement strings. Please see general Javascript document for details.

5.5.2 HTML content tags and attributes stripping

You can strip (remove) certain HTML tags and attributes from your HTML content by adding a "webeditor.strip" Javascript statement on your web pages with the Asbru Web Content Editor. The "webeditor.strip" Javascript statement must be placed on your web pages after that the Asbru Web Content Editor program files have been loaded and before the Asbru Web Content Editor functions are called.

For example:

```
webeditor.strip = {
  "TBODY":false,
  "SCRIPT":true,
  "A.target":false,
  ".onclick":false
};
```

- "TBODY":false
  will remove all “TBODY” tags from your HTML content, but everything inside the TBODY tag will be kept – only the “TBODY” tag itself will be removed.
- “SCRIPT”:true
  will remove all “SCRIPT” tags and everything inside the “SCRIPT” tags from your HTML content.
- “A.target”:false
  will remove all “target” attributes from “A” tags in your HTML content.
- “.onclick”:false
  will remove all “onclick” attributes from all tags in your HTML content.

Please note that the HTML content will only be stripped if the web content editor output format is set to “html” or “xhtml” – the HTML content will not be stripped if the output format is the web browser’s default output. Please see 3.2 Web Editor Options for details.

5.5.3 HTML content custom encoding/decoding

If your web pages with the Asbru Web Content Editor include support for Javascript functions named “webeditor_custom_encode” and “webeditor_custom_initialize”, these functions will be called when/before your HTML content is inserted into the web content editor. Your HTML content is passed as a parameter to the Javascript functions, which may modify the HTML content before it is returned (and inserted into the web content editor).

If your web pages with the Asbru Web Content Editor include a Javascript function named “webeditor_custom_decode” and/or “webeditor_custom_initialize”, that function will be called when/before your HTML content is submitted to your web server. Your HTML content is passed as a parameter to the Javascript function, which may modify the HTML content before it is returned (and submitted to the web server).

For example to custom encode/decode your HTML content, the following two functions could be used:

```
<script>
function webeditor_custom_encode(content) {
    // Place your own Javascript code here to modify the content
    // before the content is inserted into the web content editor
    return content;
}
function webeditor_custom_decode(content) {
    // Place your own Javascript code here to modify the content
    // before the content is submitted to the web server
    return content;
}
</script>
```

Exactly, what needs to be encoded/decoded and how to encode/decode it, depends on your web application, so you need to customise these functions for your web application.

5.5.4 HTML content for use with Macromedia Flash

You may want to use the Asbru Web Content editor to edit content to be used in Macromedia Flash applications. Macromedia Flash applications may handle HTML content but may not handle all HTML content. For example Macromedia Flash applications may use absolute point/pixel font sizes while standard HTML uses pre-defined, relative font sizes.

For example to custom encode/decode Macromedia Flash HTML content, the following two functions could be used:

```
<script>
function webeditor_custom_encode(content) {
    RegExp.global = true;
    RegExp.multiline = true;
    content = content.replace(/<font([^>]*)size="8"([^>]*)>/gi, "<font$1size=\"1\"$2>");
    content = content.replace(/<font([^>]*)size="10"([^>]*)>/gi, "<font$1size=\"2\"$2>");
    content = content.replace(/<font([^>]*)size="12"([^>]*)>/gi, "<font$1size=\"3\"$2>");
    content = content.replace(/<font([^>]*)size="14"([^>]*)>/gi, "<font$1size=\"4\"$2>");
    content = content.replace(/<font([^>]*)size="18"([^>]*)>/gi, "<font$1size=\"5\"$2>");
    content = content.replace(/<font([^>]*)size="24"([^>]*)>/gi, "<font$1size=\"6\"$2>");
    content = content.replace(/<font([^>]*)size="36"([^>]*)>/gi, "<font$1size=\"7\"$2>");
    return content;
}
function webeditor_custom_decode(content) {
    RegExp.global = true;
    RegExp.multiline = true;
    content = content.replace(/<font([^>]*)size="1"([^>]*)>/gi, "<font$1size=\"8\"$2>");
    content = content.replace(/<font([^>]*)size="2"([^>]*)>/gi, "<font$1size=\"10\"$2>");
    content = content.replace(/<font([^>]*)size="3"([^>]*)>/gi, "<font$1size=\"12\"$2>");
    content = content.replace(/<font([^>]*)size="4"([^>]*)>/gi, "<font$1size=\"14\"$2>");
    content = content.replace(/<font([^>]*)size="5"([^>]*)>/gi, "<font$1size=\"18\"$2>");
    content = content.replace(/<font([^>]*)size="6"([^>]*)>/gi, "<font$1size=\"24\"$2>");
    content = content.replace(/<font([^>]*)size="7"([^>]*)>/gi, "<font$1size=\"36\"$2>");
    return content;
}
</script>
```

Exactly, what needs to be encoded/decoded and how to encode/decode it, depends on your web application, so you need to customise these functions for your web application.

5.5.5 ASP, ASP.NET, ColdFusion, JSP, PHP and similar tags

To use the Asbru Web Content Editor to edit content with ASP, ASP.NET, ColdFusion, JSP, PHP and similar tags, you may need to encode/decode your content.

To edit and preserve content containing such HTML-like tags, the tags need to be encoded into a HTML-compliant editable format before editing the content, and the tags need to be decoded into the original format again after editing the content.

To make such encoding and decoding easy to add for your web applications, the Asbru Web Content Editor supports two custom Javascript functions: “webeditor_custom_encode” and “webeditor_custom_decode”. If functions with these names are included on your web pages where the Asbru Web Content Editor is used, the Asbru Web Content Editor will call the functions to encode and decode the content, automatically.

For example to custom encode/decode ASP, ASP.NET, ColdFusion, JSP and PHP tags, the following two functions could be used:

```
<script>
function webeditor_custom_encode(content) {
    RegExp.global = true;
    RegExp.multiline = true;
    content = content.replace(/<(%.*%)>/gi, "&lt;$1&gt;");
    content = content.replace(/<(\?php.*\?)>/gi, "&lt;$1&gt;");
    content = content.replace(/<(cf[^>]*)>/gi, "&lt;$1&gt;");
    content = content.replace(/<(\/cf[^>]*)>/gi, "&lt;$1&gt;");
    return content;
}

function webeditor_custom_decode(content) {
    RegExp.global = true;
    RegExp.multiline = true;
    content = content.replace(/&lt;(%[^&]*%)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(%.*%)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(\?php[^&]*\?)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(\?php.*\?)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(cf[^&]*)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(cf.*)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(\/cf[^&]*)&gt;/gi, "<$1>");
    content = content.replace(/&lt;(\/cf.*)&gt;/gi, "<$1>");
    return content;
}
</script>
```

5.5.6 Non-standard HTML-like custom tags

To use the Asbru Web Content Editor with some web content management systems and other web applications, you may need to encode/decode your content.

Some web content management systems, template languages and scripting languages etc. use HTML-like tags (i.e. <if condition=”true”>Content A<else>Content B</if>), which web browsers may not be able to edit and preserve unmodified.

To edit and preserve content containing such HTML-like tags, the tags need to be encoded into a HTML-compliant editable format before editing the content, and the tags need to be decoded into the original format again after editing the content.

For example:

<if condition=”true”>Content A<else>Content B</if>

may be encoded into:

[if condition=”true”]Content A[else]Content B[/if]

or something completely different.

To make such encoding and decoding easy to add for your web applications, the Asbru Web Content Editor supports four custom Javascript functions: “webeditor_custom_encode” / “webeditor_custom_initialize” and “webeditor_custom_decode” / “webeditor_custom_finalize”. If functions with these names are included on your web pages

where the Asbru Web Content Editor is used, the Asbru Web Content Editor will call the functions to encode and decode the content, automatically.

For example to encode/decode the example above, the following two functions could be used:

```
<script>
function webeditor_custom_encode(content) {
    RegExp.global = true;
    RegExp.multiline = true;
    content = content.replace(/<if([^>]*)>/gi, "[if$1]");
    content = content.replace(/<else>/gi, "[else]");
    content = content.replace(/<\/if>/gi, "[/if]");
    return content;
}
function webeditor_custom_decode(content) {
    RegExp.global = true;
    RegExp.multiline = true;
    content = content.replace(/\[if([^\]]*)]/gi, "<if$1>");
    content = content.replace(/\[else]/gi, "<else>");
    content = content.replace(/\[\/if]/gi, "</if>");
    return content;
}
</script>
```

Exactly, what needs to be encoded/decoded and how to encode/decode it, depends on your web application and the HTML-like format and tags it uses, so you need to customise these functions for your web application.

5.6 CSS Style Sheet Style Names

As default the extracted CSS style sheet names are displayed for the available styles.

You may want to replace the possibly abbreviated CSS style sheet names with more user friendly names. This can be done by adding a “webeditor_custom_formatclass_option” custom Javascript function to your web pages where the Asbru Web Content Editor is used.

This Javascript function is called for each CSS style sheet name with the name as parameter, and the function must return the (modified) name to be displayed instead of the CSS style sheet name or an empty string to ignore the CSS style sheet name option. For example:

```
function webeditor_custom_formatclass_option(name) {
    switch (name) {
    case "head1":
        return "Heading 1";
        break;
    case "head2":
        return "Heading 2";
        break;
    case "ignore":
        return "";
        break;
    default:
        return name;
        break;
    }
}
```

5.7 Javascript Programming API

Instead of simply using the Asbru Web Content Editor as a HTML FORM TEXTAREA replacement and POST the editable web content to your web server, you may use the Asbru Web Content Editor Javascript programming API for more advanced integration with your web applications.

The following built-in Javascript functions are available for getting and setting your Asbru Web Content Editor editable web content:

5.7.1 WebEditorActivate() / WebEditorEnable()

Use "WebEditorActivate()" or "WebEditorEnable()" to make the web content editor input field editable.

If you have multiple editable web content areas you may pass the name of an editable web content area (or its IFRAME Javascript DOM node) to activate/enable that editable web content area (i.e. use "WebEditorActivate('content')" to activate/enable the web content editor input field named "content").

5.7.2 WebEditorDeactivate() / WebEditorDisable()

Use "WebEditorDeactivate()" or "WebEditorDisable()" to make the web content editor input field uneditable.

If you have multiple editable web content areas you may pass the name of an editable web content area (or its IFRAME Javascript DOM node) to deactivate/disable that editable web content area (i.e. use "WebEditorDeactivate('content')" to deactivate/disable the web content editor input field named "content").

5.7.3 WebEditorFocus()

Use "WebEditorFocus()" to focus the editable web content area.

If you have multiple editable web content areas you may pass the name of an editable web content area to focus that editable web content area (i.e. use "WebEditorFocus('content')" to focus the web content editor input field named "content").

5.7.4 WebEditorToolbarRefresh()

Use "WebEditorToolbarRefresh()" to refresh the toolbar for the editable web content area.

If you have multiple editable web content areas you may pass the name of an editable web content area to refresh the toolbar for that editable web content area (i.e. use "WebEditorToolbarRefresh('content')" to focus the web content editor input field named "content").

5.7.5 WebEditorGetContent()

Use "WebEditorGetContent()" to get the full content from the currently focused editable web content area.

If you have multiple editable web content areas you may pass the name of an editable web content area to get the full content from that editable web content area (i.e. use “WebEditorGetContent(‘content’)” to get the full content from the editable web content named “content”).

Please note that this function switches the web content editor input field to WYSIWYG mode if it is in HTML mode ("cleaning" unclosed tags and other invalid HTML code in the edited content). Alternatively, use the "WebEditorGetContentEdited()" function.

5.7.6 WebEditorGetContentEdited()

Use “WebEditorGetContentEdited()” to get the full content from the currently focused editable web content area.

If you have multiple editable web content areas you may pass the name of an editable web content area to get the full content from that editable web content area (i.e. use “WebEditorGetContentEdited(‘content’)” to get the full content from the editable web content named “content”).

Please note that this function get the "raw" HTML code from the web content editor input field without automatically switching to WYSIWYG mode if it is in HTML mode (not "cleaning" unclosed tags and other invalid HTML code in the edited content). Alternatively, use the "WebEditorGetContent()" function.

5.7.7 WebEditorGetContentSelection()

Use “WebEditorGetContentSelection()” to get the current content selection from the currently focused editable web content area.

If you have multiple editable web content areas you may pass the name of an editable web content area to get the content selection from that editable web content area (i.e. use “WebEditorGetContentSelection(‘content’)” to get the content selection from the editable web content named “content”).

5.7.8 WebEditorGetContentSelectionContainer()

Use “WebEditorGetContentSelectionContainer()” to get the Javascript DOM parent node for the current content selection from the currently focused editable web content area.

5.7.9 WebEditorSetContent()

Use “WebEditorSetContent(‘<h1>Hello World</h1>’)” to replace the full content of the currently focused editable web content.

If you have multiple editable web content areas you may pass the name of an editable web content area to replace the full content of that editable web content area (i.e. use “WebEditorSetContent(‘<h1>Hello World</h1>’,’content’)” to replace the full content of the editable web content named “content”).

5.7.10 WebEditorPasteContent()

Use "WebEditorPasteContent('<h1>Hello World</h1>')" to insert content into the currently focused editable web content area at the current position of the caret/cursor.

If you have multiple editable web content areas you may pass the name of an editable web content area to insert content into that editable web content area (i.e. use "WebEditorPasteContent('<h1>Hello World</h1>','content')" to insert content into the editable web content named "content").

5.7.11 WebEditorPreview(id)

Use "WebEditorPreview()" to preview the content from the currently focused editable web content area in a new web browser window.

If you have multiple editable web content areas you may pass the name of an editable web content area to preview the content from that editable web content area (i.e. use "WebEditorPreview('content')" to preview the content from the editable web content named "content").

5.7.12 WebEditorSubmit()

Use "WebEditorSubmit()" to prepare the web content area to be submitted to the web server by moving the edited content from the web content area to a hidden HTML FORM input field. As default the web content editor attaches as a HTML FORM SUBMIT handler and this function is called automatically when the form is submitted. However, if you override the HTML FORM SUBMIT handler you must call this function manually before the form is submitted.

5.7.13 WebEditorStylesheet()

Use "WebEditorStylesheet(URL)" to load another CSS style sheet for the web content editor. Set the style sheet URL to '' to unload any loaded CSS style sheet.

5.7.14 WebEditorCleanContent()

Use "WebEditorCleanContent(all_html, all_xml, all_namespace, all_lang, all_class, all_style, empty_span, all_span, empty_font, all_font, all_del_ins, empty_p_div)" to clean the content for unwanted HTML tags and attributes.

Each parameter should be true or false to clean or ignore the different types of content:

- All HTML – Delete all HTML tags from content.
- All XML – Delete all XML tags from content.
- All namespace – Delete all XML namespace tags from content.
- All LANG – Delete all LANG attributes from content.
- All CLASS – Delete all CLASS attributes from content.
- All STYLE – Delete all STYLE attributes from content.
- Empty SPAN – Delete empty and double SPAN tags.
- All SPAN – Delete all SPAN tags.
- Empty FONT – Delete empty and double FONT tags.
- All FONT – Delete all FONT tags.

- All DEL and INS – Delete all DEL and INS tags.
- Empty P and DIV – Delete all empty P and DIV tags.

5.7.15 WebEditorCleanContentString()

Use “content = WebEditorCleanContentString(content, all_html, all_xml, all_namespace, all_lang, all_class, all_style, empty_span, all_span, empty_font, all_font, all_del_ins, empty_p_div)” for to clean the content string variable for unwanted HTML tags and attributes.

For example, use this from your own “webeditor_custom_decode” Javascript function to automatically clean the content when/before it is submitted to the web server. (Please see 5.5 Custom Encoding/Decoding And Reformatting Of HTML Content for details).

Each parameter should be true or false to clean or ignore the different types of content:

- All HTML – Delete all HTML tags from content.
- All XML – Delete all XML tags from content.
- All namespace – Delete all XML namespace tags from content.
- All LANG – Delete all LANG attributes from content.
- All CLASS – Delete all CLASS attributes from content.
- All STYLE – Delete all STYLE attributes from content.
- Empty SPAN – Delete empty and double SPAN tags.
- All SPAN – Delete all SPAN tags.
- Empty FONT – Delete empty and double FONT tags.
- All FONT – Delete all FONT tags.
- All DEL and INS – Delete all DEL and INS tags.
- Empty P and DIV – Delete all empty P and DIV tags.

5.7.16 WebEditorSkin()

Use “WebEditorSkin(foldername)” to load another “skin" in the form of a CSS style sheet and toolbar button images etc. for the web content editor. The parameter must be the name of a sub-folder under the “/webeditor/” folder. The folder must contain a “webeditor.css” file and toolbar button image files.