[[DW]]

htmlOKay plugin

The **htmlOKay** plugin enables DokuWiki administrators to set permissions on HTML access, so that designated users and groups may use HTML in their pages. There are four sets of permissions or 'access levels', enabling administrators to control the type and extent of access that any particular user or group may have to HTML techniques. This documentation contains the following pages:

- Access Levels
- Access Manager
  - Sample Copy of Access Manager[1]
- Access level violations
- Sample Page Processed at Different HTML Access Levels
  - Level "L"
  - Level "M"
  - Level "H"
  - Sample Page with Top of Page Errors Window[2]
- Installing htmlOKay[3]
- Download: github[4]
  Note: All versions of htmlOKay now support unrestricted super user access.
- Changelog[5]

Applying Access Levels

Introduction

The **htmlOKay** plugin enables the DukuWiki administrator to set permissions on embedded HTML access. Through its Embedded HTML Access Manager, it applies an extra layer of access controls on top of DukuWiki's built-in ACL. The ACL takes priority. Before users can embed HTML in a page, they must first have write access to that page.

The permissions allow for up to four levels of access and can be applied to both users and groups. **htmlOKay** requires the default setting for embedded HTML, which is that HTML is not allowed.[1)]. Only users and groups which have been granted HTML access from within the **htmlOKay** Access Manager will have the right to use HTML in their files.

A file which supports embedded HTML can alternate between standard DokuWiki mark-up and HTML mark-up. HTML is included in a DokuWiki file by placing it between the two HTML tags:

```
  =====Dokuwiki Standard Header=====
<html>
<H2>HTML Header</H2>
</html>
```

Objectives

Access levels determine the extent to which users and groups can use various features of HTML. There are two objectives:

1. to provide security
2. to maintain the integrity of the Wiki's page template

Pages which enable embedded HTML inevitably run a security risk, not unlike pages which allow external files to be included by means of the include[6] plugin. And pages with embedded HTML risk inadverently overriding the template's CSS or misaligning page structure. An unmatched <DIV>, for instance, inserted into the default template can remove all the formatting from the DokuWiki footer.

The Access Levels

There are four access levels, each with increasing degrees of restriction. These are designated in the most obvious of terms as **H**, **M**, **L**, and **U**. That is, as **H**igh, **M**edium, and **Low** degrees of restriction, and a relatively **U**nrestricted degree.

1. **H** Allows in-line formatting, including the in-line “style” attribute and onclick and mouse handlers in-line. It excludes scripting, forms, tables, iframes, ilayers, divs.
2. **M** Allows restricted use of javascript, of forms and of css. Excludes tables, iframes, ilayers, divs.
3. **H** Allows full use of css and restricted use of forms and javascript. Excludes iframes and ilayers.
4. **U** Allows full range of HTML and Javascript techniques. Users at this level are subject only to the naming conventions descibed below. This is in effect a 'super user' access level. There is a beta version[7] of **htmlOKay** which enables **U** to have completely unrestricted use of HTML through a setting in the Dokuwiki **Configuration Manager**. Simply scroll down to the **htmlOKay** configuration settings and tick off “Unrestricted Super User.” (While technically 'beta', the changes for this version were minor, and it has been operating successfully.)

Naming Conventions

All access levels are subject to the **htmlOKay** naming conventions. All CSS class names, all ID's and all javascript function names must be prefixed by **htmlO_K_**. This prefix helps to assure that there will be no naming conflicts between embedded

HTML and any other plugins and the Wiki itself. Here is an extended example which touches on all of these points:

```
 <style type="text/css">
.htmlO_K_bold { font-weight-bold }
#htmlO_K_hidden { display: none; }
</style>
<script language="Javascript">
function htmlO_K_show() {
var dom=document.getElementById('htmlO_K_hidden');
dom.style.dispay="block";
}
</script>
<span id="htmlO_K_hidden" class="htmlO_K_bold">I am Hidden</span>
<a href="javascript:htmlO_K_show();">Show</a>
```

It is also possible to set styles in the template's CSS files. The names of classes and id's referenced in the template's CSS files must follow the naming conventions described above.

Feature Tables

The following tables detail the features which are allowed and disallowed and the restrictions placed on certain allowed features for several of the access levels.

Allowed and Disallowed Features for Each Access Level

Y = YES, the feature is allowed. **N** = NO, the feature is not allowed.

Feature	H	M	L	U	Restrictions
DIV	N	N	Y	Y	
FORM	N	Y	Y	Y	**M** and **L**
JAVASCRIPT	N	Y	Y	Y	**M** and **L**
CSS	N	Y	Y	Y	**M**
TABLE	N	N	Y	Y	
IFRAME	N	N	N	Y	
ILAYER	N	N	N	Y	
JAVASCRIPT URLS	N	N	Y	Y	

Restrictions

In the table below are the restrictions placed upon certain features designated as **Y** above. If a technique is listed here, then it is not available at the specified access level. For instance, **M** can use the <FORM> element but it cannot use 'action' or 'onsubmit'. This still gives it access to all the other features of the Form element and the Javascript which supports them.

Access Level	FORM	JAVASCRIPT	CSS
M	action, onsubmit	event listeners,captureEvent,createEvent, Ajax, location=url	ID's [#id_name]
L	action	event listeners,captureEvent,createEvent, Ajax, location=url	

How Permissions are Applied

Access permissions are set by namespace. That is, only one set of permissions can apply to the files of any one namespace. Various groups and users can be included in this set of permissions and they can each be assigned different access levels. Stated slightly differently, more than one group or user can have HTML access to the same file or namespace, each with different access levels.

When applying permissions, the administrator has the option of applying them to ALL of the files in the namespace or to selected files in the namespace. Let us assume that there are 3 files in the namespace:

- namespace:start
- namespace:our_organization
- namespace:our_policies.

If the administrator selects ALL from the permissions menu, then the permissions will apply to all three files. But if namespace:our_policies is selected, then permissions for files in namespace apply only to namespace:our_policies. The other two files will not have HTML access. You cannot apply different permissions to different files in the same namepace. However, any user or group may have HTML access to the files in more than one namespace.

The inability to apply permissions on a file by file basis, as opposed to namespace basis, may be seen as a shortcoming. But the original idea behind **htmlOKay** was that the Wiki would be largely organized by projects and that each project would fall under its own namespace. The project would then be developed by a group or a single individual, who would be assigned an HTML access level.

The Display Level

Each page and/or namespace is assigned a **display** level. Normally, this is the same as the access level of the group or user who has created the page. But the Embedded HTML Access Manager will allow administrators to assign users and groups to the same namespace who have different HTML access levels for that namespace. Consequently, **htmlOKay** has a policy of applying the most restrictive permissions when a page is being viewed by an external visitor to the page, i.e. someone without HTML access to that page. If one developer has **M** access and another has **L**, the page will be displayed at **M**.

Moreover, if the user with **L** access uses features that are not supported by the **M** set of features, then those features will not work when **M** goes to add something to the page. It may not seem to make sense to apply more than one access level to a namespace, yet it can have uses when developing a project. We will look more closely at these issues when dealing with the Access Manager.

---

1) This means that the “Allow embedded HTML” option in the Configuration Manager is left unchecked.

Managing Embedded HTML

The administration tool for **htmlOKay** is the HTMLOK Access Manager, which is illustrated in this sample copy[8]. It might be useful to open the copy in a separate window to follow along; otherwise, you can return to this discussion from the sample copy by clicking on its ***administration*** breadcrumb or title or on its ***Show page*** button.

The copy doesn't, of course, have all of the functionality of the Manager; that is, you can't actually create HTML users with it. But in all other respects, it is the same. It contains two dialogs. At the top is a Selection Box of Namespaces and Files; at the bottom are two Tables, Users and Groups, where you assign the various access levels. Between them is a bar with three buttons, Save, Reset, and Scroll/Scroll Off. Let's look at each of these in turn.

Namespaces and Files

The top Selection Box consists of two rows and three columns:

Namespace	[namespace list]	**(Select Button)**
Files	[File list]	**Use Ctrl or Options key to multiple select from: html**

The namespace list follows the typographic convention of DokuWiki, which is colon-separated directory names: directory:subdirectory. The topmost, or root directory, of the namespace hierarchy is designated as _ _ROOT_ _. To select a namespace, you highlight one of the namespace options from the drop-down menu and then click **Select**. The files in the selected namespace directory will appear in the scrollable File List below. Two additional options will be presented at the top of the file list:

- No Files Selected
- ALL

In addition, the name of the selected namespace will appear in the instruction box to the right of the File List. This instruction is basic PC and Mac keyboard selection protocol, which is to hold down either the PC's CTRL key or the Mac's option key if you want to select more than one file. Again selection is made by highlighting your choices. The highlighted selections will be the ones for which the access levels will be set in the Users and Groups dialog.

Selecting 'ALL'

If you select ALL, then any file present and future which is included in the selected directory will support embedded HTML at the level assigned in the Users and Groups dialog. If you instead select one or more files from this directory, then only those files will support HTML. Any other file in that directory will be treated as a normal file.

Selecting 'No Files Selected'

The default option is 'No Files Selected'. If you assign permissions and click **Save** but fail to select a File List option, leaving “No Files Selected”, you will receive an error message. If you click **Save** and no files have been selected and no permissions have been set, the Manager will **delete** the permissions for the selected namespace, if permissions exist; otherwise the request is ignored.

Users and Groups

The purpose of this dialog is to set permissions for the namespace and/or files selected in the Namespace dialog. There are two tables, one for groups and one for users. The Users table contains the basic information about each user, including a clickable email address. Next to each user or group is a set of radio buttons labeled **H**, **M**, **L**, and **U**, used for setting its access level. Clicking on these assigns the access levels for users and groups.

Obviously, only one access level can be assigned to any one user or group. Radio buttons are mutually exclusive, so that clicking one of these will undo any other choice made previously in its set. If you make a selection for a user or group and decide cancel permission for that user or group, then you can click the Reset label **R** in the column to the right of the access levels.

Multiple Permissions

Given the way the Manager is constructed, it is possible to assign access permissions to multiple users and groups for a namespace and its files. This feature allows teams to work on a project.

It is even possible to assign different access levels to different users and groups. Because of this possibility, **htmlOKay** sets a display policy, which is the access level at which a page will be displayed when viewed by an external visitor. This is the visitor who does not have HTML access to the page or someone who may have access but is not logged in. The display level is simply the most restrictive access level that has been assigned to that page. So that if someone with an **H** and someone with an **M** set of permissions have been working on the page, it will be displayed at **H**, which permits many fewer HTML techniques than **M**. Quite simply, the display policy opts for security.

One of the offshoots of assigning different access levels to a single page is that when the developer with the lesser access edits the page, any HTML which is not supported at the lesser access level will be marked as errors and excluded from display. Nevertheless, multiple access levels can have a good use in cases where there may be, say, a lead developer in a team project with greater latitude to create more advanced pages than others in the team. When the project is concluded, the entire namespace could be elevated to the lead developer's access level, which would then become the project's display policy.

Another side effect of setting multiple access levels is that a page may be cached at the higher access level so that as long as the page remains in the cache, it will be displayed to visitors at this higher level rather than at the display level. To meet this problem it is possible to provide a **Refresh** button, to re-process the page after the developer has logged out. Hitting the **Refresh** button after the developer logs out, causes the page to be re-processed at the display level.

Code for the the **Refresh** button will be found here.

Center Button Bar

The center button bar contains three buttons: **Save**, **Reset**, and **Scroll**.

1. **Scroll**

- Users and Groups are written to a scrollable window. Clicking **Scroll** will cause the scrollbars to appear, and the button name will change to **Scroll off**. In Firefox, if the Users or Groups dialog exceeds the height of the window, scrollbars will automatically appear.

1. **Reset**

- The **Reset** button will Unselect all Users and Group options for the selected namespace. This is in contrats to the **R** label beside each user and group, which lets you unset permissions individually.

1. **Save**

- After your choices have been completed, click **Save**. All the permissions will be saved for the selected namespace, and the selected files will be highlighted in the Namespace dialog.

Deleting and Reviewing

Deleting Permissions

If more than one set of access permissions has been applied to a namespace, permissions for any one of these can be deleted simply by clicking on the **R** label beside the relevant user or group. To delete all permissions for a namespace, click the **Reset** button and then click **Save**. The permissions will be removed.

Reviewing Permissions

When you Select a namespace from the Namespace dialog, all the permissions and selected files will be filled in. To alter the permissions, you simply re-set them and then click **Save**.

Namespace Scope

A namespace consists of a directory and its subdirectories. If individual files are selected, then only those files are covered by the permissions set for the namespace. But, if **All** is selected, then the entire namespace is covered by one set of access permissions. This includes not only all of its files but will also include all of its subdirectories and their files, **unless** separate permissions are set for the subdirectories.

When **htmlOKay** searches for permissions, it of course begins with the current namespace, i.e. the directory at issue. If it doesn't find permissions for that namespace, it progressively moves up one namespace in the namespace hierarchy until it finds a set of permissions or until it reaches the top of the hierarchy–which is the directory immediately below the root directory (i.e. below data/pages). It stops when it finds a set of permissions and applies those permissions to the current namespace. If it finds no permissions, then the namespace has no HTML access.

Because it stops at the directories immediately below data/pages, the permissions set for data/pages can't be used as a set of default permissions for the entire site.

Access Level Violations

Marking Access Violations

htmlOKay deals with access violations by marking up a page in the places where the access violations occur. Examples of this markup for each of the access levels are found in the following pages, which show the same HTML but at different access levels:

- Level "L"
- Level "M"
- Level "H"

The following page has two access settings, **M** and **L:**

- Levels "M" and "L"[9]

A page with multiple access levels will always display to the visitor at the most restrictive access level, which in this case is **M**. If you click on the **Developer's toolbar**, and then click on the **Refresh** button, the page will display at **M**, if it is not already at **M**. If you then log on as developer with the password developer, the page will display as **L**. When you log out, the page will still be at **L**, but if you then hit the **Refresh** button again, it will revert to **M**. See the section on the Refresh Button later on this page.

Errors Window

In addition to the marking up of access violations, there will also be a window at the top of the containing more detailed error messages, as in the example below:

> **User Info:**
> hmtlOK_access_level: 2
> client: wsmith
> $conf['htmlok']: 1
> Scope: html
> **---End User Info---**
> ID Selectors not supported at current HTML access level: **#htmlO_K_div_1.**
> Element or Attribute not supported at current HTML access level: **TABLE**
> Element or Attribute not supported at current HTML access level: **div**
> Internal Window Elements Not Supported. External files cannot be included in wiki documents: **IFRAME.**

The **User Info** section contains four pieces of information–the current access level of this user, the name of the current user (or 'client'), the value of the $conf['htmlok'] variable, which should always be 1 (i.e. true), and the 'scope', which is the namespace under which the current page falls and from which it gets its access permissions. (See the explanation of namespace scope.) For a sample page with both the errors window and marked up access violations, see this page.[10]

If you are using the `dokuwiki` template, an action link to `Show HTML Errors` will appear at the top of the page. If not, you can install your own action link by placing the following code in your template:

```
 <?php
if($INFO['htmlOK_client'] && $INFO['hmtlOK_access_level'] > 0) {
tpl_toolsevent('htmlokaytools', array());
}
?>
```

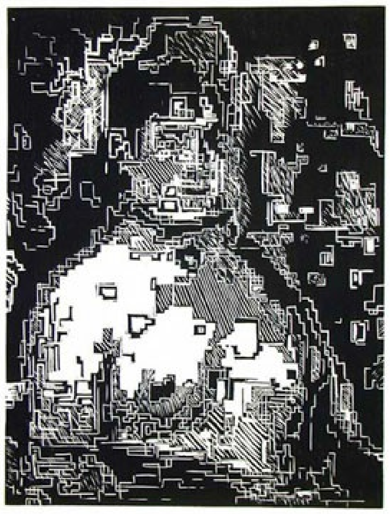
Or you can add the following button anywhere in your template that is convenient:

```
<?php if($INFO['htmlOK_client'] && $INFO['hmtlOK_access_level'] > 0) { ?>
<form name="htmlOK_errform">
<div class="no">
<input type="button" value="Show errors"
        class="button"
        onclick="htmlOK_ERRORS(0);jQuery('#htmlOKDBG_ERRORWIN').toggle();" />
</div>
</form>
<?php } ?>
```

The Clicking Page: Displayed at Access Level "H"

<script language="javascript"> var size = 100; var htmlO_K_clearint; function htmlO_K_shrink() { var dom = document.getElementById('htmlO_K_div_1'); if(size < 15) { clearInter
. . .
e.jpg'; dom.title = 'Click to Zoom in' } else { im = 'http://www.mturner.org/htmlOKay/19thcentnudelg.jpg'; dom.title = 'Click to Zoom out' } dom.src = im; } </script>
Element or Attribute not supported: **SCRIPT**

<style type= "text/css"> #htmlO_K_div_1 { background-color: yellow; border: solid 2px blue; width: 100px; height: 100px; position:relative;} </style>
Element or Attribute not supported: **STYLE**



19th Century Nude, After Manet

<table cellpading="8" width="75%"> <tr><td align="left"> The inspiration for <b>htmlOK</b> is a project I have in mind, where artists will use the Wiki as a field in which to create we
. . .
al Award</a>. While I occasionally earn a few dollars writing software and doing some sysadmin, I am essentially a self-taught hacker. </td></tr> <tr><td> </td></tr> </table>
Element or Attribute not supported: **TABLE**

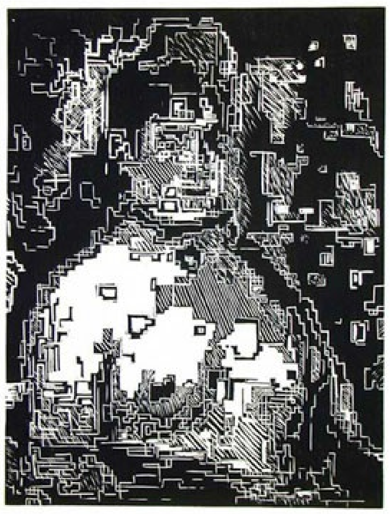
<div id="htmlO_K_div_1" onclick="htmlO_K_start_shrink();">
Element or Attribute not supported: **div**

Click The Yellow Box

<IFRAME SRC="https://dokuwiki.org"></IFRAME>
Internal Window Elements Not Supported: **IFRAME**

The Clicking Page: Displayed at Access Level "M"

ID Selectors not supported:
#htmlO_K_div_1



19th Century Nude, After Manet

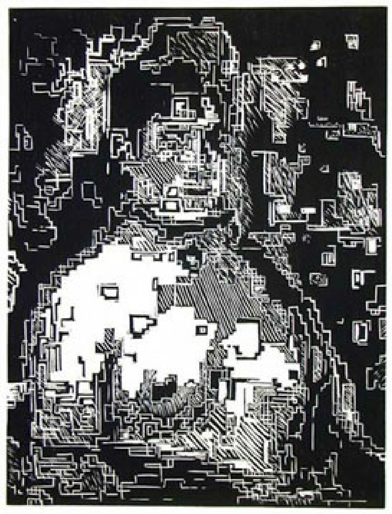
<table cellpading="8" width="75%"> <tr><td align="left"> The inspiration for <b>htmlOK</b> is a project I have in mind, where artists will use the Wiki as a field in which to create we
. . .
al Award</a>. While I occasionally earn a few dollars writing software and doing some sysadmin, I am essentially a self-taught hacker. </td></tr> <tr><td> </td></tr> </table>
Element or Attribute not supported: **TABLE**

<div id="htmlO_K_div_1" onclick="htmlO_K_start_shrink();">
Element or Attribute not supported: **div**

Click The Yellow Box

<IFRAME SRC="https://dokuwiki.org"></IFRAME>
Internal Window Elements Not Supported: **IFRAME**

The Clicking Page: Displayed at Access Level "L"


19th Century Nude, After Manet

The inspiration for **htmlOK** is a project I have in mind, where artists will use the Wiki as a field in which to create web projects. I myself am a visual and new media artist. I've been using computers in and for my work for almost 20 years and have been programming for even longer. When the first web browser came out in 1994 (Mosaic), I was bowled over, and I've been working with the Internet and the web ever since. Every year or so I seem to complete another web piece. My most recent was exhibited in February 2006 in the artport series of the Whitney Museum of American Art and was nominated for a Viper International Award. While I occasionally earn a few dollars writing software and doing some sysadmin, I am essentially a self-taught hacker.

Click The Yellow Box

<IFRAME SRC="https://dokuwiki.org"></IFRAME>
Internal Window Elements Not Supported: **IFRAME**

start.html

[1] http://www.mturner.org/htmlOKay/doku.php.htm
[2] http://www.mturner.org/htmlOKay/LEVEL-2.htm
[3] http://www.mturner.org/htmlOKay/doku.php?id=installation
[4] https://github.com/turnermm/htmlOKay/archive/master.zip
[5] http://www.mturner.org/htmlOKay/doku.php?id=notes:changelog

htmlok.html

[6] http://wiki.splitbrain.org/plugin:include
[7] http://www.mturner.org/wiki/htmlOKay_su.tgz

administration.html

[8] http://www.mturner.org/htmlOKay/doku.php.htm

html_access_violations.html

[9] http://www.mturner.org/htmlOKay/doku.php?id=html:multiple:clicking
[10] http://www.mturner.org/htmlOKay/LEVEL-2.htm