Customizing Macromedia Dreamweaver MX

Macromedia Dreamweaver MX can be customized in many ways, allowing you to work in a manner that's familiar, comfortable, and efficient for you. This article describes advanced methods for customizing Dreamweaver MX, with a focus on hand-editing configuration files.

Note: The information in this article applies only to Dreamweaver MX. Earlier versions of Dreamweaver used somewhat different configuration files and formats.

About customizing Dreamweaver

There are a variety of general approaches to customizing Dreamweaver. Several of these approaches are covered in Dreamweaver Help (Help > Using Dreamweaver); only the advanced technique of hand-editing configuration files is discussed in detail in this article.

The following are some of the general approaches to customizing that are covered in Dreamweaver Help. These approaches are not discussed in this article.

- Arrange your workspace any way you want it.
- Change settings in dialog boxes in Dreamweaver. You can set preferences in a variety of areas, including accessibility, code coloring, fonts, highlighting, and previewing in browsers, all using the Preferences panel (Edit > Preferences). You can also change keyboard shortcuts, using the Keyboard Shortcut Editor (Edit > Keyboard Shortcuts).
- Add third-party extensions to Dreamweaver using the Extension Manager (Help > Manage Extensions).

The following are some of the ways you can customize Dreamweaver by editing configuration files; for more information, see the rest of this article.

- Rearrange the objects in the Insert bar, create new tabs to reorganize the objects, or add new objects. See "Modifying the Insert bar" on page 5.
- Change the names of menu items, add new commands to menus, and remove existing commands from menus. See "About customizing Dreamweaver menus" on page 9.
- Change browser profiles or create new ones. See "Working with browser profiles" on page 20.
- Change how third-party tags (including ASP and JSP tags) appear in the Document window's Design view. See "Customizing the interpretation of third-party tags" on page 23.

Another approach to customizing Dreamweaver is to create your own extensions. For more information, see "Extending Dreamweaver: Basics" on page 23.

About customizing Dreamweaver in a multiuser environment

You can customize Dreamweaver to suit your needs even in a multiuser operating system such as Windows NT, Windows 2000, Windows XP, or Mac OS X. Dreamweaver prevents any user's customized configuration from affecting any other user's customized configuration. To accomplish this goal, the first time you run Dreamweaver in a multiuser operating system that it recognizes, the application copies various configuration files into a user configuration folder that belongs to you. When you customize Dreamweaver using dialog boxes and panels, the application modifies your user configuration files instead of modifying the master configuration files.

To customize Dreamweaver by hand-editing a configuration file in a multiuser environment, edit the appropriate user configuration file, rather than editing the master configuration files in the application folder. To make a change that affects most users, you can edit a master configuration file, but users who already have corresponding user-configuration files won't see the change. In general, if you want to make a change that affects all of the users, it's best to create an extension and install it using the Extension Manager. For more information, see "Extending Dreamweaver: Basics" on page 23.

Note: In older operating systems (Windows 98, Windows ME, and Mac OS 9.x), a single set of Dreamweaver configuration files is shared by all users, even if the operating system is configured to support multiple users.

The following are the specific locations for user configuration folders in each multiuser operating system:

- Windows NT: C:\WinNT\profiles\username\Application Data\Macromedia\Dreamweaver MX\Configuration
- Windows 2000 and Windows XP: C:\Documents and Settings\username\Application Data\Macromedia\Dreamweaver MX\Configuration (In Windows XP, this folder may be inside a hidden folder)
- Mac OS X: Hard disk/Users/username/Library/Application Support/Macromedia/Dreamweaver MX/Configuration

Note: To install extensions that all users can use in a multiuser operating system, you must be logged in as Administrator (Windows) or root (Mac OS X).

Dreamweaver copies only some of the configuration files into your user configuration folder the first time you run the application. (The files that it copies are specified in the version.xml file in the Configuration folder.) When you customize Dreamweaver from within the application (for example, when you modify one of the predesigned code snippets in the Snippets panel), Dreamweaver copies the relevant files into your user configuration folder. The version of a file in your user configuration folder always takes precedence over the version in the master configuration folder.

To hand-customize a configuration file that Dreamweaver has not copied into your user configuration folder, first copy the file from the master configuration folder to the corresponding location inside your user configuration folder. Then edit the copy in your user configuration folder.

Deleting configuration files in a multiuser environment

When you do something within Dreamweaver in a multiuser operating system that would cause a configuration file to be deleted (for example, when you delete a predesigned snippet from the Snippets panel), Dreamweaver creates a file in your user configuration folder called mm_deleted_files.xml. When a file is listed in mm_deleted_files.xml, Dreamweaver behaves as if that file doesn't exist.

Note: The mm_deleted_files.xml file isn't created until you take an action that would cause a configuration file to be deleted.

To deactivate a configuration file:

1. Quit Dreamweaver.
2. Using a text editor, edit mm_deleted_files.xml in your user configuration folder; add an `item` tag to that file, giving the path (relative to the master Configuration folder) of the configuration file to deactivate.

   Note: Do not edit mm_deleted_files.xml in Dreamweaver.
3. Save and close mm_deleted_files.xml.
4. Start Dreamweaver again.

About mm_deleted_files.xml tag syntax

The mm_deleted_files.xml file contains a structured list of items that describe configuration files which Dreamweaver is to ignore. These items are described by XML tags which you can edit in a text editor.

The following sections describe the syntax of the mm_deleted_files.xml tags. Optional attributes are marked in the attribute lists with braces (`{}`); all attributes not marked with braces are required.

<deleteditems>

Description

Container tag holding a list of items that Dreamweaver should treat as deleted.

Attributes

None.

Contents

This tag must contain one or more `item` tags.

Container

None.

Example

```
<deleteditems>
  <!-- item tags here -->
</deleteditems>
```

<item>

Description

Specifies a configuration file that Dreamweaver should ignore.

Attributes

`name`

`name` The path to the configuration file, relative to the Configuration folder. In Windows, use a backslash (\) to separate parts of the path; on the Macintosh, use a colon (:).

Contents

None (empty tag).

Container

This tag must be contained in a `deleteditems` tag.

Example

```
<item name="snippets\headers\5columnwith4links.csn" />
```

Reinstalling and uninstalling Dreamweaver in a multiuser environment

After you install Dreamweaver MX, if you later reinstall Dreamweaver MX or upgrade to a later version, Dreamweaver automatically makes backup copies of existing user configuration files, so that if you've customized those files by hand, you still have access to the changes you made.

When you uninstall Dreamweaver from a multiuser system (which you can do only if you have administrative privileges), Dreamweaver can remove each user configuration folder for you.

Changing the default file type

By default, Dreamweaver shows all the file types it recognizes in the File > Open dialog box. You can use a pop-up menu in that dialog box to limit the display to certain types of files. If most of your work involves a specific file type (such as ASP files), you can change the default display. Whatever file type is listed on the first line of the Dreamweaver Extensions.txt file becomes the default.

Note: If you want to see all file types in the File > Open dialog box, even the files Dreamweaver can't open, you must select All Files (*.*). This is different from All Documents, which shows only the files Dreamweaver can open.

To change the Dreamweaver default File > Open file type:

1. Make a backup copy of the Extensions.txt file in the Configuration folder.
2. Open Extensions.txt in Dreamweaver or in a text editor.
3. Cut the line corresponding to the new default, and paste it at the beginning of the file, to make it the first line of the file.
4. Save the file.
5. Restart Dreamweaver.

   To see the new default, select File > Open and look at the pop-up menu of file types.

To add new file types to the menu in the File > Open dialog box:

1. Make a backup copy of the Extensions.txt file in the Configuration folder.
2. Open Extensions.txt in Dreamweaver or a text editor.
3. Add a new line for each new file type. In capital letters, enter the filename extensions that the new file type can have, separated by commas; then add a colon and a brief descriptive phrase to show in the pop-up menu for file types that appears in the File > Open dialog box. For example, for JPEG files, enter the following:

   ```
   JPG,JPEG,JFIF:JPEG Image Files
   ```

4. Save the file.
5. Restart Dreamweaver.

   To see the changes, select File > Open and click the pop-up menu of file types.

Modifying the Insert bar

By default, the Insert bar is divided into several tabs. (For information about the objects in these tabs, see Dreamweaver Help.) The tabs correspond to folders in the Configuration/Objects folder within the Dreamweaver application folder.

You can move objects from one tab to another, rename tabs, and remove objects from the panel altogether. To make your changes appear in the Insert bar, you must either restart Dreamweaver or reload extensions.

The insertbar.xml file specifies the order of the tabs, and the order of objects within each tab. Dreamweaver creates the Insert bar based on the tabs and objects specified in insertbar.xml, then checks the folders corresponding to the tabs, looking for additional objects not listed in insertbar.xml. Such objects are appended to the appropriate tabs, after all the objects that are listed in insertbar.xml.

Dreamweaver ignores folders in the Objects folder that aren't listed in insertbar.xml.

Note: Remember that in a multiuser operating system, you should edit copies of configuration files in your user configuration folder rather than editing master configuration files. For more information, see "About customizing Dreamweaver in a multiuser environment" on page 2.

For each object in an Insert bar tab, there are two or three files in the corresponding folder:

- A GIF file containing an icon for the object
- An HTML file containing either the HTML to be inserted into your file or an HTML form that lets you specify data to be inserted (such as the text of a comment)
- A JavaScript file (optional) that generates the HTML to be inserted into your file

To move or copy an object from one Insert bar tab to another, or to a new location within a tab:

1. Make a backup copy of Objects/insertbar.xml.
2. Open insertbar.xml in Dreamweaver or in a text editor.
3. Find the `button` tag representing the object you want to move or copy. For example, to move the Image object from its location in the Common tab, find the `button` tag that has an `id` attribute of `"DW_Image"`.
4. Cut or copy the entire `button` tag.

5 Find the `category` tag representing the tab you want to move or copy the object to.

6 Find the location within the tab where you want the object to appear.

7 Paste the copied `button` tag.

8 Save insertbar.xml.

9 Reload extensions.

To reload extensions:

1 Control-click (Windows) or Option-click (Macintosh) the Options menu in the Insert bar's title bar.

2 Choose Reload Extensions.

To add a separator to an Insert bar tab:

1 Make a backup copy of Objects/insertbar.xml.

2 Open insertbar.xml in Dreamweaver or in a text editor.

3 Find the `button` tag representing the object you want to place a separator after.

4 After that button tag, add the following code:

```
<separator showIf=""/>
```

5 Save insertbar.xml.

6 Reload extensions.

To remove an object from the Insert bar:

1 Open insertbar.xml in Dreamweaver or in a text editor.

2 Find the `button` tag representing the object you want to remove.

3 Delete the entire `button` tag.

4 Save insertbar.xml.

5 On your disk, move the object's HTML, GIF, and JavaScript files out of the folder they're in, and into a folder that isn't listed in insertbar.xml. For example, you could create a new folder in Configuration/Objects named Unused, and move the object's files into that folder. (If you're certain you want to remove the object, you can delete those files entirely, but it's a good idea to keep backups of those files in case you need to restore the object later.)

6 Reload extensions.

To change the order of tabs in the Insert bar:

1 Make a backup copy of Objects/insertbar.xml.

2 Open insertbar.xml in Dreamweaver or in a text editor.

3 Find the `category` tag corresponding to the tab you want to move, and select that tag, including all of the tags contained in it.

4 Cut that tag.

5 Paste the tag into its new location. Be sure to paste the tag in a location that isn't inside any other `category` tag.

6 Save insertbar.xml.

7 Reload extensions.

To rename a tab on the Insert bar:

1 On your disk, find the folder corresponding to the tab you want to rename.

2 In that folder, open the _folderinfo.txt file.

3 Change the name that appears in that file and save the file.

4 Reload extensions.

To create a new tab:

1 Make a backup copy of Objects/insertbar.xml.

2 Open insertbar.xml in Dreamweaver or in a text editor.

3 Create a new `category` tag, specifying the default folder for the tab and a set of objects to appear in the tab.

For information on the syntax of the tags in insertbar.xml, see Extending Dreamweaver (Help > Extending Dreamweaver).

4 Save insertbar.xml.

5 Reload extensions.

To add a new object to the Insert bar in a specific tab:

1 Create the object. (See "Creating a simple object" on page 7.)

2 Make a backup copy of Objects/insertbar.xml.

3 Open insertbar.xml in Dreamweaver or in a text editor.

4 Find the `category` tag representing the tab you want to add the object to.

5 Inside that tag, add a new `button` tag representing the new object.

For information on the syntax of the tags in insertbar.xml, see Extending Dreamweaver (Help > Extending Dreamweaver).

Creating a simple object

You can create your own objects to add to the Insert bar. Many simple objects require no JavaScript; they contain only the HTML source code to be inserted into the document. For basic information about creating more complex objects using JavaScript, see "Extending Dreamweaver: Basics" on page 23.

After you create an object, you can package it and distribute it on the Macromedia Exchange site if you want other Dreamweaver users to be able to use it. For more information, see the Macromedia Exchange for Dreamweaver site at www.macromedia.com/exchange/dreamweaver. To package an extension, you must first download the Extension Manager installer from that site, then install the Extension Manager with the developer option.

To create a simple object:

1. In Dreamweaver, choose File > New.

   The New Document dialog box appears.

2. In the Category column on the left, choose Other.

   A list of file types appears in the center column of the dialog box.

3. Select Text from the list of file types and click Create.

   A new blank text document appears.

4. Add the code and text that you want this object to insert into your documents; for example, type the following:

```
<p>
&copy; 2002 ZII Productions, Inc.<BR>
All Rights Reserved
</p>
```

5. Save the file.

   If you want the new object to appear in one of the existing Insert bar tabs, save it in one of the subfolders of the Objects folder. For information on placing the object in a particular position in the tab or on creating a new tab, see "Modifying the Insert bar" on page 5.

6. In a graphics or image-editing application (such as Macromedia Fireworks), create an 18 x 18 pixel GIF image that will serve as the icon for your object in the Insert bar.

   If you create a larger image, Dreamweaver automatically scales it to 18 x 18 pixels. If you do not create an icon for your object, Dreamweaver displays a generic object icon for your object in the Insert bar.

7. Give your icon the same filename as your object file, but use .gif as the extension; then save the icon in the same directory as the object file.

   For example, if your object is called Copyright_ZII.htm and you saved it in the Common directory, name your icon Copyright_ZII.gif and save it in the Common directory as well.

8. Restart Dreamweaver, or reload extensions, to use your new object.

   The object appears at the bottom of the Insert menu as well as on the Insert bar. For information on reloading extensions, see "Modifying the Insert bar" on page 5.

Changing FTP mappings

The FTPExtensionMap.txt file (Windows) and the FTPExtensionMapMac.txt file (Macintosh) map filename extensions to FTP transfer modes (ASCII or BINARY).

Each line in each of the two files includes a filename extension (such as GIF) and either the word ASCII or the word BINARY, to indicate which of the two FTP transfer modes should be used when transferring a file with that extension. On the Macintosh, each line also includes a creator code (such as DmWr) and a file type (such as TEXT); when you download a file with the given filename extension, Dreamweaver assigns the specified creator and file type to the file.

If a file that you're transferring doesn't have a filename extension, Dreamweaver uses the BINARY transfer mode.

Note: Dreamweaver cannot transfer files in macbinary mode. If you need to transfer files in macbinary mode, you must use another FTP client.

For example, the following line (from the Macintosh file) indicates that files with a .html filename extension should be transferred in ASCII mode:

```
HTML    DmWr    TEXT    ASCII
```

In both files, all elements on a given line are separated by tabs. The extension and the transfer mode are in uppercase letters.

To change a default setting, edit the file in a text editor.

To add information about a new filename extension:

1. Edit the extension-map file in a text editor.
2. On a blank line, enter the filename extension (in uppercase) and press Tab.
3. On the Macintosh, add the creator code, a tab, the file type, and another tab.
4. Enter **ASCII** or **BINARY** to set an FTP transfer mode.
5. Save the file.

About customizing Dreamweaver menus

Dreamweaver creates all of its menus from the structure defined in an XML file called menus.xml, in the Configuration/Menus subfolder of the Dreamweaver application folder. Editing the menus.xml file changes the Dreamweaver menus the next time you start Dreamweaver. For basic information about XML, see Dreamweaver Help.

By editing the menus.xml file, you can add, change, and remove keyboard shortcuts for menu items, though in most cases it's easier to do that using the Keyboard Shortcut Editor. (See Dreamweaver Help.) You can also rearrange, rename, and remove menu items.

In a multiuser operating system, when you make changes within Dreamweaver that result in changes to menus.xml (such as changing keyboard shortcuts using the Keyboard Shortcut Editor), Dreamweaver creates a new menus.xml file in your user configuration folder. To customize menus.xml in a multiuser operating system, edit the copy of the file in your user configuration folder (or copy the master menus.xml file to your user configuration folder if Dreamweaver hasn't yet created a version there). For more information, see "About customizing Dreamweaver in a multiuser environment" on page 2.

If you open menus.xml in an XML editor, you may see error messages regarding the ampersands (&) in the menus.xml file. It's best to edit menus.xml in an ordinary text editor. (Don't edit it in Dreamweaver.)

Note: Always make a backup copy of the current menus.xml file, or any other Dreamweaver configuration file, before you modify it. It's easy to make mistakes in editing the menu configuration file, and there's no way to revert to a previous set of menus other than replacing the menus.xml file. In case you forget to make a backup, though, the Configuration folder contains a backup of the default menus.xml file, called menus.bak; to revert to the default menu set, replace menus.xml with a copy of menus.bak.

Modifying the Commands menu

You can add certain kinds of commands to the Commands menu, and change their names, without editing the menus.xml file. For more information about menus.xml, see "About customizing Dreamweaver menus" on page 9.

Note: The term "command" has two meanings in Dreamweaver. Strictly speaking, a command is a particular kind of extension. In some contexts, however, "command" is used interchangeably with "menu item" to mean any item that appears in a Dreamweaver menu, no matter what it does or how it's implemented.

To create new commands that are automatically placed in the Commands menu, use the History panel. Alternatively, you can use the Extension Manager to install new extensions, including commands. For more information, see Dreamweaver Help.

To reorder the items in the Commands menu, or to move items between menus, you must edit the menus.xml file. (See "Rearranging menus and menu items" on page 10.)

To rename a command you've created:

1. Choose Commands > Edit Command List.

   A dialog box appears, listing all of the commands whose names you can change. (Commands that are in the default Commands menu don't appear on this list and can't be edited using this approach.)
2. Select a command to rename.
3. Enter a new name for it.
4. Click Close.

   The command is renamed in the Commands menu.

To delete a command you've created:

1. Choose Commands > Edit Command List.

   A dialog box appears listing all of the commands you can delete. (Commands that are in the default Commands menu don't appear on this list and can't be deleted using this approach.)
2. Select a command to delete.
3. Click Delete, and then confirm that you want to delete the command.

   The command is deleted. Note that the file containing the code for the command is also deleted; deleting a command does not simply remove the menu item from the menu. Be certain that you really want to delete the command before you use this approach. If you want to remove it from the Commands menu without deleting the file, you can find the file in Configuration/Commands and move it to another folder.
4. Click Close.

Rearranging menus and menu items

By editing the menus.xml file, you can move menu items within a menu or from one menu to another, add separators to or remove them from menus, and move menus within a menu bar or even from one menu bar to another.

Note that you can move items into or out of context menus using the same procedure as for other menus.

For information, see "About menus.xml tag syntax" on page 13.

To move a menu item:

1. Quit Dreamweaver.
2. Make a backup copy of the menus.xml file.
3. Open menus.xml in a text editor such as BBEdit, HomeSite, or Wordpad. (Don't open it in Dreamweaver.)

4 Cut an entire `menuitem` tag, from the `<menuitem` at the beginning to the `/>` at the end.

5 Place the insertion point at the new location for the menu item. (Make sure it's between a `<menu>` tag and the corresponding `</menu>` tag.)

6 Paste the menu item into its new location.

To create a submenu while moving a menu item:

1 Place the insertion point inside a menu (somewhere between a `<menu>` tag and the corresponding `</menu>` tag).

2 Insert a new `<menu></menu>` pair inside the menu.

3 Add new menu items to the new submenu.

To insert a separator between two menu items:

- Enter **<separator />** between the two `menuitem` tags.

To remove an existing separator:

- Delete the corresponding `<separator />` line.

To move a menu:

1 Quit Dreamweaver.

2 Make a backup copy of the menus.xml file.

3 Open menus.xml in a text editor such as BBEdit, HomeSite, or Wordpad. (Don't open it in Dreamweaver.)

4 Cut an entire menu and its contents, from the opening `<menu>` tag to the closing `</menu>` tag.

5 Place the insertion point at the new location for the menu. (Make sure it's between a `<menubar>` tag and the corresponding `</menubar>` tag.)

6 Paste the menu into its new location.

Changing the name of a menu item or menu

You can easily change the name of any menu item or menu by editing the menus.xml file. For more information, see "About menus.xml tag syntax" on page 13.

To change the name of a menu item or menu:

1 Quit Dreamweaver.

2 Make a backup copy of the menus.xml file.

3 Open menus.xml in a text editor such as HomeSite, BBEdit, or Wordpad (don't open it in Dreamweaver).

4 If you're changing a menu item, find the appropriate `menuitem` tag, and change the value of its `name` attribute. If you're changing a menu, find the appropriate `menu` tag, and change the value of its `name` attribute. In either case, do not change the `id` attribute.

5 Save and close menus.xml; then start Dreamweaver again to see your changes.

Changing keyboard shortcuts

If the default keyboard shortcuts aren't convenient for you, you can change or remove existing shortcuts or add new ones. The easiest way to do this is to use the Keyboard Shortcut Editor. (For more information, see Dreamweaver Help.) However, you can also modify keyboard shortcuts directly in menus.xml if you prefer, though it's much easier to make mistakes entering shortcuts in menus.xml than in the Keyboard Shortcut Editor. For more information, see "About menus.xml tag syntax" on page 13.

To change a keyboard shortcut:

1. Quit Dreamweaver.
2. Make a backup copy of the menus.xml file.
3. Open menus.xml in a text editor such as BBEdit, HomeSite, or Wordpad. (Don't open it in Dreamweaver.)
4. Look at the Keyboard Shortcut Matrix (available from the Dreamweaver Support Center) and find a shortcut that's not being used, or one that you want to reassign.

   If you reassign a keyboard shortcut, change it on a printed copy of the matrix for future reference.
5. If you're reassigning a keyboard shortcut, find the menu item that the shortcut is assigned to, and remove the `key="shortcut"` attribute from that menu item.
6. Find the menu item to assign or reassign the keyboard shortcut to.
7. If the menu item already has a keyboard shortcut, find the `key` attribute on that line. If it doesn't already have a shortcut, add **key=""** anywhere between attributes inside the `menuitem` tag.
8. Between the double quotation marks of the `key` attribute, enter the new keyboard shortcut.

   Use a plus (+) sign between the keys in a key combination. For more information about modifiers, see the description of the `menuitem` tag in "<menuitem>" on page 15.

   If the keyboard shortcut is in use elsewhere and you didn't remove the other use of it, the shortcut will apply only to the first menu item that uses it in menus.xml.

   Note: You can use the same keyboard shortcut for a Windows-only menu item and for a Macintosh-only menu item.
9. Write your new shortcut in the appropriate location in the Keyboard Shortcut Matrix.

Modifying pop-up menus and context menus

Dreamweaver provides pop-up menus and context menus in many of its panels and dialog boxes. Some context menus are defined in the menus.xml file; others are defined in other XML files. You can add, remove, or modify items in those menus by hand, though in most cases it's better to write an extension to make such changes. For information on creating extensions that modify most of these XML files, see Extending Dreamweaver (Help > Extending Dreamweaver).

The following pop-up menus and context menus in Dreamweaver are specified in XML files, using the same tags as menus.xml:

- Data sources (listed in the plus (+) pop-up menu in the Bindings panel) are specified in DataSources.xml files, in subfolders of the DataSources folder.
- Server behaviors (listed in the plus (+) pop-up menu in the Server Behaviors panel) are specified in ServerBehaviors.xml files, in subfolders of the ServerBehaviors folder.

- Server formats (listed in the plus (+) pop-up menu in the Edit Format List dialog box) are specified in ServerFormats.xml files, in subfolders of the ServerFormats folder.
- Items in the formats pop-up menu for a binding in the Bindings panel are specified in Formats.xml files, in subfolders of the ServerFormats folder. You can add entries to this menu from inside Dreamweaver by using the Add Format dialog box.
- The Tag Library Editor dialog box menu items are specified in the TagLibraries/TagImporters/TagImporters.xml file.
- Menu items for parameters in the Generate Behavior dialog box, which is part of the Server Behavior Builder, are specified in Shared/Controls/String Menu/Controls.xml.
- Items for context menus associated with ColdFusion Components are specified in Components/ColdFusion/CFCs/CFCsMenus.xml.
- Items for context menus associated with ColdFusion data sources are specified in Components/ColdFusion/DataSources/DataSourcesMenus.xml.
- Items for context menus associated with JavaBeans are specified in Components/Jsp/JavaBeans/JavaBeanMenus.xml.
- Items for context menus associated with various server components are specified in XML files, in subfolders of the Components folder.

About menus.xml tag syntax

The menus.xml file contains a structured list of menu bars, menus, menu items, separators, shortcut lists, and keyboard shortcuts. These items are described by XML tags which you can edit in a text editor.

Note: Be careful when making changes to menus. Dreamweaver ignores any menu or menu item that contains an XML syntax error.

A menu bar (tagged with opening and closing `menubar` tags) is a discrete menu or set of menus—for example, there's a main menu bar, a separate Site window menu bar (which appears only in Windows, not on the Macintosh), and a menu bar for each context menu. Each menu bar contains one or more menus; a menu is contained in a `menu` tag. Each menu contains one or more menu items, each described by a `menuitem` tag and its attributes. A menu can also contain separators (described by `separator` tags) and submenus.

In addition to the keyboard shortcuts associated with menu items, Dreamweaver provides a variety of other keyboard shortcuts, including alternate shortcuts and shortcuts that are available only in certain contexts. For example, Control+Y (Windows) or Command+Y (Macintosh) is the shortcut for Redo; but Control+Shift+Z or Command+Shift+Z is an alternate shortcut for Redo. These alternates—and other shortcuts that can't be represented in the tags for menu items—are defined in shortcut lists in the menus.xml file. Each shortcut list (described by a `shortcutlist` tag) contains one or more shortcuts, each described by a `shortcut` tag.

The following sections describe the syntax of the menus.xml tags. Optional attributes are marked in the attribute lists with braces (`{}`); all attributes not marked with braces are required.

<menubar>

Description

Provides information about a menu bar in the Dreamweaver menu structure.

Attributes

`name, {app}, id, {platform}`

`name` The name of the menu bar. Although `name` is a required attribute, you can give it the value `""`.

`app` The name of the application in which the menu bar is available. Not currently used.

`id` The menu ID for the menu bar. Each menu ID in the menus.xml file should be unique.

`platform` Indicates that the menu bar should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

This tag must contain one or more `menu` tags.

Container

None.

Example

The main (Document window) menu bar uses the following `menubar` tag:

```
<menubar name="Main Window" id="DWMainWindow">
  <!-- menu tags here -->
</menubar>
```

<menu>

Description

Provides information about a menu or submenu to appear in the Dreamweaver menu structure.

Attributes

`name, {app}, id, {platform}`

`name` The name of the menu as it will appear in the menu bar. To set the menu's access key (mnemonic) in Windows, use an underscore (_) before the access letter. The underscore is automatically removed on the Macintosh.

`app` The name of the application in which the menu is available. Not currently used.

`id` The menu ID for the menu. Every ID in the file should be unique.

`platform` Indicates that the menu should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

This tag can contain one or more `menuitem` tags, and one or more `separator` tags. It can also contain other `menu` tags (to create submenus) and standard HTML comment tags.

Container

This tag must be contained in a `menubar` tag.

Example

```
<menu name="_File" id="DWMenu_File">
  <!-- menuitem, separator, menu, and comment tags here -->
</menu>
```

<menuitem>

Description

Defines a menu item for a Dreamweaver menu.

Attributes

`name`, `id`, `{app}`, `{key}`, `{platform}`, `{enabled}`, `{arguments}`, `{command}`, `{file}`, `{checked}`, `{dynamic}`

`name` The menu item name that appears in the menu. An underscore indicates that the following letter is the command's access key (mnemonic), for Windows only.

`id` Used by Dreamweaver to identify the item. This ID must be unique throughout the menu structure. If you add new menu items to menus.xml, ensure uniqueness by using your company name or another unique string as a prefix for each menu item's ID.

`app` The name of the application in which the menu item is available. Not currently used.

`key` The keyboard shortcut for the command, if any. Use the following strings to specify modifier keys:

- `Cmd` specifies the Control key (Windows) or Command key (Macintosh).
- `Alt` and `Opt` interchangeably specify the Alt key (Windows) or Option key (Macintosh).
- `Shift` specifies the Shift key on both platforms.
- `Ctrl` specifies the Control key on both platforms.
- A plus (+) sign separates modifier keys if a given shortcut uses more than one modifier. For example, `Cmd+Opt+5` in the `key` attribute means the menu item is executed when the user presses Control+Alt+5 (Windows) or Command+Option+5 (Macintosh).
- Special keys are specified by name: `F1` through `F12`, `PgDn`, `PgUp`, `Home`, `End`, `Ins`, `Del`, `Tab`, `Esc`, `BkSp`, and `Space`. Modifier keys can also be applied to special keys.

`platform` Indicates which platform the item appears on. Valid values are `"win"`, meaning Windows-only, or `"mac"`, meaning Macintosh-only. If you don't specify the `platform` attribute, the menu item appears on both platforms. If you want a menu item to behave differently on different platforms, supply two menu items with the same name (but different IDs): one with `platform="win"` and the other with `platform="mac"`.

`enabled` Provides JavaScript code (usually a JavaScript function call) that determines whether the menu item is currently enabled. If the function returns `false`, the menu item is dimmed. The default value is `"true"`, but it's best to always specify a value for clarity even if the value is `"true"`.

`arguments` Provides arguments for Dreamweaver to pass to the code in the JavaScript file that you specify in the `file` attribute. Enclose arguments in single quotation marks ('), inside the double quotation marks used to delimit an attribute's value.

`command` Specifies a JavaScript expression that's executed when the user selects this item from the menu. For complex JavaScript code, use a JavaScript file (specified in the `file` attribute) instead. You must specify either `file` or `command` for each menu item.

`file` The name of an HTML file containing JavaScript that controls the menu item. Specify a path to the file relative to the Configuration folder. (For example, the Help > Welcome menu item specifies `file="Commands/Welcome.htm"`.) Note that the `file` attribute overrides the `command`, `enabled`, and `checked` attributes. You must specify either `file` or `command` for each menu item. For information on creating a command file using the History panel, see Dreamweaver Help. For information on writing your own JavaScript commands, see Extending Dreamweaver (Help > Extending Dreamweaver).

`checked` A JavaScript expression that indicates whether the menu item has a check mark next to it in the menu; if the expression evaluates as `true`, the item is displayed with a check mark.

`dynamic` If present, indicates that a menu item is to be determined dynamically, by an HTML file; the file contains JavaScript code to set the text and state of the menu item. If you specify a tag as `dynamic`, you must also specify a `file` attribute.

`isdomrequired` Indicates whether to synchronize the Design view and the Code view before executing the code for this menu item. Valid values are `"true"` (the default) and `"false"`. If you set this attribute to `"false"`, it means that the changes to the file that this menu item makes do not use the Dreamweaver DOM. For information about the DOM, see Extending Dreamweaver (Help > Extending Dreamweaver).

Contents

None (empty tag).

Container

This tag must be contained in a `menu` tag.

Example

```
<menuitem name="_New" key="Cmd+N" enabled="true" command="dw.createDocument()"
  id="DWMenu_File_New" />
```

<separator>

Description

Indicates that a separator should be displayed at the corresponding location in the menu.

Attributes

`{app}`
`app` The name of the application in which the separator is shown. Not currently used.

Contents

None (empty tag).

Container

This tag must be contained in a `menu` tag.

Example

```
<separator />
```

<shortcutlist>

Description

Specifies a shortcut list in the menus.xml file.

Attributes

`{app}`, `id`, `{platform}`

`app` The name of the application in which the shortcut list is available. Not currently used.

`id` The ID for the shortcut list. It should be the same as the menu ID for the menu bar (or context menu) in Dreamweaver that the shortcuts are associated with. Valid values are `"DWMainWindow"`, `"DWMainSite"`, `"DWTimelineContext"`, and `"DWHTMLContext"`.

`platform` Indicates that the shortcut list should appear only on the given platform. Valid values are `"win"` and `"mac"`.

Contents

This tag can contain one or more `shortcut` tags. It can also contain one or more comment tags (which use the same syntax as HTML comment tags).

Container

None.

Example

```
<shortcutlist id="DWMainWindow">
  <!-- shortcut and comment tags here -->
</shortcutlist>
```

<shortcut>

Description

Specifies a keyboard shortcut in the menus.xml file.

Attributes

`key`, `{app}`, `{platform}`, `{file}`, `{arguments}`, `{command}`, `id`, `{name}`

`key` The key combination that activates the keyboard shortcut. For syntax details, see "<menuitem>" on page 15.

`app` The name of the application in which the shortcut is available. Not currently used.

`platform` Specifies that the shortcut works only on the indicated platform. Valid values are `"win"` and `"mac"`. If you do not specify this attribute, the shortcut works on both platforms.

`file` The path to a file containing the JavaScript code that Dreamweaver executes when you use the keyboard shortcut. The `file` attribute overrides the `command` attribute. You must specify either `file` or `command` for each shortcut.

`arguments` Provides arguments for Dreamweaver to pass to the code in the JavaScript file that you specify in the `file` attribute. Enclose arguments in single quotation marks ('), inside the double quotation marks used to delimit an attribute's value.

`command` The JavaScript code that Dreamweaver executes when you use the keyboard shortcut. Specify either `file` or `command` for each shortcut.

`id` A unique identifier for a shortcut.

`name` A name for the command executed by the keyboard shortcut, in the style of a menu item name. For example, the `name` attribute for the F12 shortcut is `"Preview in Primary Browser"`.

Contents

None (empty tag).

Container

This tag must be contained in a `shortcutlist` tag.

Example

```
<shortcut key="Cmd+Shift+Z" file="Menus/MM/Edit_Clipboard.htm"
  arguments="'redo'" id="DWShortcuts_Edit_Redo" />
```

Customizing code hints

To customize the code hints pop-up menus in Dreamweaver, edit the CodeHints.xml file. In general, you modify this file by writing an extension rather than by editing the file by hand; for more information, see Extending Dreamweaver (Help > Extending Dreamweaver).

Customizing default documents

The DocumentTypes/NewDocuments folder contains a default (blank) document of each type that you can create using Dreamweaver. When you create a new blank document by choosing File > New and selecting an item from the Basic Page, Dynamic Page, or Other categories, Dreamweaver bases the new document on the appropriate default document in this folder. To change what appears in a default document of a given type, edit the appropriate document in this folder.

Note: If you want all the pages in your site to contain common elements (such as a copyright notice) or a common layout, it's better to use templates and library items than to change the default documents. For more information on templates and library items, see Dreamweaver Help (Help › Using Dreamweaver).

Customizing page designs

Dreamweaver provides a variety of predesigned CSS style sheets, framesets, and page designs. You can create pages based on these designs by choosing File > New.

To customize the available designs, edit the files in BuiltIn/css, BuiltIn/framesets, BuiltIn/Templates, and BuiltIn/TemplatesAccessible. Note that the designs listed in the Page Designs and Page Designs (Accessible) categories are actually Dreamweaver template files; for more information on templates, see Dreamweaver Help (Help > Using Dreamweaver).

You can also create your own page designs by adding files to the subfolders of the BuiltIn folder. To make a description of the file appear in the New Document dialog box, create a Design Notes file (in the appropriate _notes folder) corresponding to the page design file.

Creating and modifying toolbars

You can rearrange or delete items in the toolbars provided with Dreamweaver (such as the Document toolbar and the Standard toolbar) by editing the Toolbars/toolbars.xml file.

To create a new toolbar, you generally need to create both XML files to specify the controls in the toolbar, and commands to perform the associated actions.

For more information about creating toolbars and editing the toolbars.xml file, see Extending Dreamweaver (Help > Extending Dreamweaver).

Creating new document types

Dreamweaver configuration files specify filename extensions, server models, and other information for each type of document that Dreamweaver recognizes. To add a new document type to Dreamweaver or to change information about a known document type, either edit the DocumentTypes/MMDocumentTypes.xml file, or create another XML file using the same format. You generally modify and create such files by writing an extension rather than by editing the files by hand; for more information, see Extending Dreamweaver (Help > Extending Dreamweaver).

Customizing the Tag Chooser

Dreamweaver configuration files provide metadata for organizing tag groupings that appear in the Tag Chooser. By editing a TagChooser.xml file (in a subfolder of the TagLibraries folder), you can regroup existing tags and group new tags. In general, you modify and create these files by writing an extension rather than by editing the files by hand; for more information, see Extending Dreamweaver (Help > Extending Dreamweaver).

Customizing the appearance of dialog boxes

The dialog box layouts for objects, commands, and behaviors are specified as HTML forms; they reside in HTML files in the Configuration folder within the Dreamweaver application directory. You edit those forms just as you would edit any form in Dreamweaver; for more information, see Dreamweaver Help.

Note: Remember that in a multiuser operating system, you should edit copies of configuration files in your user configuration folder rather than editing master configuration files; for more information, see "About customizing Dreamweaver in a multiuser environment" on page 2.

To change the appearance of a dialog box:

1. In Dreamweaver, choose Edit > Preferences, then choose the Code Rewriting category.
2. Deselect the Rename Form Items when Pasting option.

   Deselecting this option ensures that form items retain their original names when you copy and paste them.
3. Click OK to dismiss the Preferences dialog box.
4. On your disk, find the appropriate .htm file in Configuration/Objects, Configuration/Commands, or Configuration/Behaviors.
5. Make a copy of the file, somewhere other than the Configuration folder.
6. Open the copy in Dreamweaver, edit the form, and save it.
7. Quit Dreamweaver.
8. Copy the changed file back to the Configuration folder, in place of the original. (It's a good idea to first make a backup of the original, so you can restore it later if you need to.)
9. Start Dreamweaver again to see the changes.

You should change only the appearance of the dialog box, not how it works; it still must contain the same types of form elements with the same names, so that the information Dreamweaver obtains from the dialog box can still be used in the same way.

For example, the Comment object takes text input from a text area in a dialog box, then uses a simple JavaScript function to turn that text into an HTML comment and insert the comment into your document. The form that describes the dialog box is in Configuration/Objects/Invisibles/Comment.htm. You can open that file and change the size and other attributes of the text area, but if you remove the `textarea` tag entirely, or change the value of its `name` attribute, the Comment object will no longer work properly.

Changing default HTML formatting

To change general code formatting preferences, use the Code Format category of the Preferences dialog box. To change the format of specific tags and attributes, use the Tag Library Editor (Edit > Tag Libraries). For more information, see Dreamweaver Help.

You can also edit the formatting for a tag by hand-editing the .vtm file corresponding to the tag (in a subfolder of the Tag Libraries configuration folder), but it's much easier to change formatting within Dreamweaver.

If you add or remove a .vtm file, you must edit the TagLibraries.vtm file; Dreamweaver ignores any .vtm file that isn't listed in TagLibraries.vtm. (Edit this file in a text editor, not in Dreamweaver.) For more information, see Extending Dreamweaver (Help > Extending Dreamweaver).

Working with browser profiles

Browser profiles are the files Dreamweaver uses to check your documents when you run a target browser check (see Dreamweaver Help). Each profile contains information about the HTML tags and attributes that a particular browser supports. A browser profile can also contain warnings, error messages, and suggestions for tag substitutions.

Browser profiles are stored in the Configuration/BrowserProfiles folder within the Dreamweaver application folder. You can edit existing profiles or create new ones using Dreamweaver or a text editor (such as BBEdit, HomeSite, Notepad, or SimpleText). It is not necessary to quit Dreamweaver before editing or creating browser profiles.

About browser-profile formatting

Browser profiles follow a specific format. To avoid parsing errors during target browser checks, follow these rules when editing or creating profiles:

- The first line is reserved for the name of the profile. It must be followed by a single carriage return. The name on this line appears in the Target Browser Check dialog box and in the target check report. It must be unique.
- The second line is reserved for the designator `PROFILE_TYPE=BROWSER_PROFILE`. Dreamweaver uses this line to determine which documents are browser profiles. Do not change or move this line.
- Two hyphens (`--`) at the beginning of a line indicate a comment (that is, that the line will be ignored during the target check process). A comment must start at the beginning of a line—you can't put two hyphens in the middle of a line.
- You must use a space in these places: before the closing angle bracket (>) on the `!ELEMENT` line, after the opening parenthesis in a list of values for an attribute, before a closing parenthesis in a list of values, and before and after each pipe (|) in a list of values.
- You must include an exclamation point without a space before each of the following words: `ELEMENT`, `ATTLIST`, `Error`, and `msg` (`!ELEMENT`, `!ATTLIST`, `!Error`, `!msg`).

- You can include `!Error` and `!Warning` within the `!ELEMENT` or the `!ATTLIST` area.
- `!msg` messages can contain only plain text.
- HTML comments (`<!-- -->`) cannot be listed as tags in browser profiles because they interfere with parsing. Dreamweaver does not report an error for comments, because all browsers support them.

The syntax for a tag entry is as follows:

```
<!ELEMENT htmlTag NAME="tagName" >
<!ATTLIST htmlTag
unsupportedAttribute1!Error !msg="The unsupportedAttribute1
attribute of the htmlTag tag is not supported. Try using
supportedAttribute1 for a similar effect."
supportedAttribute1
supportedAttribute2( validValue1 | validValue2 | validValue3 )
unsupportedAttribute2!Error !msg="Don't ever use the
unsupportedAttribute2 attribute of the htmlTag tag!"
>
```

The elements shown in the above syntax are defined as follows:

`htmlTag` is the tag as it appears in an HTML document.

`tagName` is an explanatory name for the tag; for example, the name for the `HR` tag is "Horizontal Rule." The `NAME` attribute is optional. If specified, *`tagName`* is used in error messages; if you do not supply a name, *`htmlTag`* is used in error messages.

`unsupportedAttribute` is an attribute that is not supported. Any tags or attributes not specifically mentioned as supported attributes are assumed to be unsupported. Specify unsupported tags or attributes only when you want to create a custom error message.

`supportedAttribute` is an attribute that is supported by *`htmlTag`*. Only tags listed without an `!Error` designation are considered to be supported by the browser.

`validValue` indicates a value that is supported by the attribute.

The following example shows an entry for the `APPLET` tag that would be accurate for Navigator 3:

```
<!ELEMENT APPLET Name="Java Applet" >
<!ATTLIST APPLET
  Align ( top | middle | bottom | left | right | absmiddle |
  absbottom | baseline | texttop )
  Alt
  Archive
  Class !Warning !msg="This browser ignores the CLASS attribute for the APPLET
  tag."
  Code
  Codebase
  Height
  HSpace
  ID !Warning !msg="This browser ignores the ID attribute for the APPLET tag.
  Use NAME instead."
  Name
  Style !Warning !msg="This browser ignores the STYLE attribute for the APPLET
  tag."
  VSpace
  Width
>
```

Creating and editing a browser profile

Create a browser profile by modifying an existing profile. For example, to create a profile for a future version of Microsoft Internet Explorer, you can open the profile for the most recent version of Internet Explorer that has a profile, add any new tags or attributes introduced in the new version, and save it as a profile for the new version.

Note: Before you create a browser profile for a new version of a browser, check the Macromedia Exchange for Dreamweaver site at www.macromedia.com/exchange/dreamweaver to see if Macromedia has supplied a browser profile that you can download and install using the Extension Manager.

To create or edit a browser profile:

1. Using a text editor, open an existing profile.

   If you're creating a new profile, open the profile that most closely resembles the profile you intend to create, then save the file under a new filename.

2. If you're creating a new profile, change the name that appears on the first line of text in the file. (Two profiles cannot have the same name.)

3. Add any new tags or attributes that you know are supported by the browser, using the syntax shown in "About browser-profile formatting" on page 20.

   If you don't want to receive error messages about a particular unsupported tag, add it to the list of supported tags. If you do this, save the profile in a separate file with a new filename (such as Browsername x.x limited). Giving this alternate profile a new name preserves the original profile with only the tags that are truly supported.

4. Delete any tags or attributes that are not supported by the browser.

   This step is probably unnecessary if you are creating a profile for a new version of Netscape Navigator or Internet Explorer, because browsers rarely drop support for tags.

5. Add any custom error messages according to the syntax shown in "About browser-profile formatting" on page 20.

   The profiles that come with Dreamweaver list all supported tags for the specified browsers. To add a custom error message to a tag, add `!msg="message"` after `!Error`. For example, this information appears in the Netscape Navigator 3.0 profile (along with other attributes not shown here):

   ```
   <!ELEMENT HR name="Horizontal Rule" >
   <!ATTLIST HR
     COLOR    !Error
   >
   ```

   To add a custom error message enter **!msg=** and then your error message, in quotation marks:

   ```
   <!ELEMENT HR name="Horizontal Rule" >
   <!ATTLIST HR
     COLOR    !Error !msg="Internet Explorer 3.0 supports the COLOR tag in
     horizontal rules, but Netscape Navigator 3.0 does not."
   >
   ```

6. You can use `!Error` for all error situations, or you can use `!Warning` to indicate that a tag will be ignored but will not actually cause an error.

Extending Dreamweaver: Basics

Dreamweaver is designed to be extensible. It includes a JavaScript interpreter, so it can read and execute JavaScript code; and it provides a set of extensibility APIs (application programming interfaces); each extensibility API is a set of JavaScript or C functions that enable extension developers to add capabilities to Dreamweaver. Dreamweaver also provides a Document Object Model (DOM), which allows extensions to examine and modify a document's structure and contents.

Using the extensibility APIs, you can create a variety of different kinds of extensions to Dreamweaver, including objects, commands, toolbars, Property inspectors, behavior actions, server behaviors, data sources, server models, and data translators, among others.

You can also create new objects and simple commands without knowing anything about programming; for more information, see "Creating a simple object" on page 7 and Dreamweaver Help. But to add more advanced capabilities to Dreamweaver, you must write extensions in either JavaScript or C. For information about the APIs and the DOM, see Extending Dreamweaver (Help > Extending Dreamweaver).

After you create a Dreamweaver extension, you can package it and distribute it on the Macromedia Exchange site if you want other Dreamweaver users to be able to use it. For more information, choose Help > Creating and Submitting Extensions, or see the Macromedia Exchange for Dreamweaver site at www.macromedia.com/exchange/dreamweaver.

About JavaScript

JavaScript is an interpreted programming language. To learn more about JavaScript, read a good JavaScript book, such as *JavaScript Bible* by Danny Goodman (IDG) or *JavaScript: The Definitive Guide* by David Flanagan (O'Reilly). For information on using JavaScript to extend Dreamweaver, see Extending Dreamweaver (Help > Extending Dreamweaver).

Note: Despite the resemblance between the two names, JavaScript is not related to Java.

Editing Dreamweaver commands

All the commands in Dreamweaver menus, including those you create and save using the History panel (see Dreamweaver Help), are implemented in JavaScript. This JavaScript code usually consists mostly of calls to functions provided by the Dreamweaver extensibility API. If you know JavaScript and understand the Dreamweaver extensibility API, you can edit the JavaScript to change the operation of Dreamweaver commands.

Note: Don't attempt to change any JavaScript code unless you're certain you know what you're doing. Even if you do know what you're doing, make a backup copy of the file containing the code before you modify it.

To rename a command, move a command to a different menu, or add a keyboard shortcut to a command, see "About customizing Dreamweaver menus" on page 9.

Customizing the interpretation of third-party tags

Server-side technologies such as ASP, ColdFusion, JSP, and PHP use special non-HTML code in HTML files; servers create and serve HTML content based on that code. When Dreamweaver encounters non-HTML tags, it compares them with information in its third-party tag files, which define how Dreamweaver reads and displays non-HTML tags.

For example, ASP files contain—in addition to regular HTML—ASP code for the server to interpret. ASP code looks almost like an HTML tag, but is marked by a pair of delimiters: it begins with <% and ends with %>. The Dreamweaver Configuration/ThirdPartyTags folder contains a file named Tags.xml, which describes the format of various third-party tags, including ASP code, and defines how Dreamweaver displays that code. Because of the way ASP code is specified in Tags.xml, Dreamweaver doesn't try to interpret anything between the delimiters; instead, in the Document window's Design view, it simply displays an icon indicating ASP code.

You can define your own tag database files that define how Dreamweaver reads and displays your tags. Create a new tag database file for each set of tags, to tell Dreamweaver how to display the tags.

Note: This section explains how to define the way Dreamweaver displays a custom tag, but doesn't describe how to provide a way to edit the content or properties of a custom tag. For information on how to create a Property inspector to inspect and change the properties of a custom tag, see Extending Dreamweaver (Help › Extending Dreamweaver).

Each tag database file defines the name, type, content model, rendering scheme, and icon for one or more custom tags. You can create any number of tag database files, but all of them must reside in the Configuration/ThirdPartyTags folder to be read and processed by Dreamweaver. Tag database files have the file extension .xml.

Tip: If you are working on several different unrelated sites at once (for example, as a freelance developer), you can put all the tag specifications for a particular site in one file. Then simply include that tag database file with the custom icons and Property inspectors that you hand over to the people who will maintain the site.

You define a tag specification with an XML tag called `tagspec`. For example, the following code describes the specification for a tag named `happy`:

```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false"
  content_model="marker_model" icon="happy.gif" icon_width="18"
  icon_height="18"></tagspec>
```

You can define two different kinds of tags using `tagspec`: normal HTML-style tags and string-delimited tags. String-delimited tags start with one string and end with another string; they're like empty HTML tags (such as `img`) in that they don't surround content and don't have closing tags. The `happy` tag shown above is a normal HTML-style tag; it starts with an opening `<happy>` tag, contains data between opening and closing tags, and ends with a closing `</happy>` tag. (If the tag were a string-delimited tag, the tag specification would include the `start_string` and `end_string` attributes.) An ASP tag is a string-delimited tag; it starts with the string `<%` and ends with the string `%>`, and it has no closing tag.

The following information describes the attributes and valid values for the `tagspec` tag. Attributes marked with an asterisk (*) are ignored for string-delimited tags. Optional attributes are marked in the attribute lists with braces (`{}`); all attributes not marked with braces are required.

<tagspec>

Description

Provides information about a third-party tag.

Attributes

`tag_name`, `{tag_type}`, `{render_contents}`, `{content_model}`, `{start_string}`, `{end_string}`, `{detect_in_attribute}`, `{parse_attributes}`, `icon`, `icon_width`, `icon_height`, `{equivalent_tag}`, `{is_visual}`, `{server_model}`

`tag_name` The name of the custom tag. For string-delimited tags, `tag_name` is used only to determine whether a given Property inspector can be used for the tag. If the first line of the Property inspector contains this tag name with an asterisk on each side, then the inspector can be used for tags of this type. For example, the tag name for ASP code is `ASP`; Property inspectors that can examine ASP code should have `*ASP*` on the first line. For information on the Property inspector API, see Extending Dreamweaver (Help > Extending Dreamweaver).

`tag_type*` Determines whether the tag is empty (as with `img`), or whether it contains anything between its opening and closing tags (as with `code`). This attribute is required for normal (non-string-delimited) tags. It's ignored for string-delimited tags, since they're always empty. Valid values are `"empty"` and `"nonempty"`.

`render_contents*` Determines whether the contents of the tag should appear in the Document window's Design view, or whether the specified icon appears instead. This attribute is required for nonempty tags, and ignored for empty tags. (Empty tags have no content.) This attribute applies only to tags that appear outside of attributes; the contents of tags that appear inside the values of attributes of other tags are not rendered. Valid values are `"true"` or `"false"`.

`content_model*` Describes what kinds of content the tag can contain and where in an HTML file the tag can appear. Valid values are `"block_model"`, `"head_model"`, `"marker_model"`, and `"script_model"`:

- `block_model` Specifies that the tag can contain block-level elements such as `div` and `p`, and that the tag can appear only in the `body` section or inside other body-content tags such as `div`, `layer`, or `td`.
- `head_model` Specifies that the tag can contain text content, and that it can appear only in the `head` section.
- `marker_model` Specifies that the tag can contain any valid HTML code, and that it can appear anywhere in an HTML file. The HTML validator in Dreamweaver ignores tags that are specified as `marker_model`. However, the validator doesn't ignore the contents of such a tag; so even though the tag itself can appear anywhere, the contents of the tag may result in invalid HTML in certain places. For example, plain text can't appear (outside of a valid `head` element) in the `head` section of a document, so you can't place a `marker_model` tag that contains plain text in the `head` section. (To place a custom tag containing plain text in the `head` section, specify the tag's content model as `head_model` instead of `marker_model`.) Use `marker_model` for tags that should be displayed inline (inside a block-level element such as `p` or `div`—for example, inside a paragraph). If the tag should be displayed as a paragraph of its own, with line breaks before and after it, don't use this model.
- `script_model` Allows the tag to exist anywhere between the opening and closing HTML tags of a document. When Dreamweaver encounters a tag with this model, it ignores all of the tag's content. Used for markup (such as certain ColdFusion tags) that Dreamweaver shouldn't parse.

`start_string` Specifies a delimiter that marks the beginning of a string-delimited tag. String-delimited tags can appear anywhere in the document where a comment can appear. Dreamweaver does not parse tags or decode entities or URLs between `start_string` and `end_string`. This attribute is required if `end_string` is specified.

`end_string` Specifies a delimiter that marks the end of a string-delimited tag. This attribute is required if `start_string` is specified.

`detect_in_attribute` Indicates whether to ignore everything between `start_string` and `end_string` (or between opening and closing tags if those strings aren't defined) even when those strings appear inside attribute names or values. You should generally set this to `"true"` for string-delimited tags; the default is `"false"`. For example, ASP tags sometimes appear inside attribute values, and sometimes contain quotation marks ("); because the ASP tag specification specifies `detect_in_attribute="true"`, Dreamweaver ignores the ASP tags, including the internal quotation marks, when they appear inside attribute values.

`parse_attributes*` Indicates whether to parse the attributes of the tag. If this is set to `"true"` (the default), Dreamweaver parses the attributes; if it's set to `"false"`, Dreamweaver ignores everything until the next closing angle bracket that appears outside quotation marks. For example, this attribute should be set to `"false"` for a tag like `cfif` (as in `<cfif a is 1>`, which Dreamweaver would be unable to parse as a set of attribute name/value pairs).

`icon` Specifies the path and filename of the icon associated with the tag. This attribute is required for empty tags, and for nonempty tags whose contents are not displayed in the Document window's Design view.

`icon_width` Specifies the width of the icon in pixels.

`icon_height` Specifies the height of the icon in pixels.

`equivalent_tag` Specifies simple HTML equivalents for certain ColdFusion form-related tags. Not intended for use with other tags.

`is_visual` Indicates whether the tag has a direct visual effect on the page or not. For example, the ColdFusion tag `cfgraph` doesn't specify a value for `is_visual` (so the value defaults to `"true"`); the ColdFusion tag `cfset` is specified as having `is_visual` set to `"false"`. Visibility for server markup tags is controlled by the Invisible Elements category of the Preferences dialog box; visibility for visual server markup tags can be set independent of visibility for nonvisual server markup tags.

`server_model` If specified, indicates that the `tagspec` tag applies only on pages belonging to the specified server model. If `server_model` is not specified, the `tagspec` tag applies on all pages. For example, the delimiters for ASP and JSP tags are the same, but the `tagspec` tag for JSP specifies a `server_model` of `"JSP"`, so when Dreamweaver encounters code with the appropriate delimiters on a JSP page, it displays a JSP icon. When it encounters such code on a non-JSP page, it displays an ASP icon.

Contents

None (empty tag).

Container

None.

Example

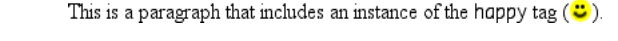
```
<tagspec tag_name="happy" tag_type="nonempty" render_contents="false"
  content_model="marker_model" icon="happy.gif" icon_width="18"
  icon_height="18"></tagspec>
```

How custom tags appear in the Design view

How custom tags appear in the Design view of the Document window depends on the values of the `tag_type` and `render_contents` attributes of the `tagspec` tag. (See "Customizing the interpretation of third-party tags" on page 23.) If the value of `tag_type` is `"empty"`, the icon specified in the `icon` attribute appears. If the value of `tag_type` is `"nonempty"` but the value of `render_contents` is `"false"`, the icon appears as it would for an empty tag. For example, an instance of the `happy` tag defined earlier might appear in the HTML like this:

```
<p>This is a paragraph that includes an instance of the <code>happy</code>
tag (<happy>Joe</happy>).</p>
```

That paragraph might appear in the Design view like this.



Note that since `render_contents` is set to `"false"` in the tag specification, the contents of the `happy` tag (the word `Joe`) are not rendered; instead the start and end tags and their contents are displayed as a single icon.

For nonempty tags that have a `render_contents` value of `"true"`, the icon does not appear in the Design view; instead, the contents between the opening and closing tags (such as the text between the tags in `<mytag>This is the contents between the opening and closing tags</mytag>`) appears. If View > Invisible Elements is enabled, the content is highlighted using the third-party tag color specified in Highlighting preferences. (Note that highlighting applies only to tags defined in tag database files.)

To change the highlighting color of third-party tags:

1. Choose Edit > Preferences and select the Highlighting category.
2. Click the Third-Party Tags color box to display the color picker.
3. Choose a color, and then click OK to close the Preferences dialog box. For information about choosing a color, see Dreamweaver Help.

Avoiding rewriting third-party tags

Dreamweaver corrects certain kinds of errors in HTML code (for details, see Dreamweaver Help). By default, Dreamweaver refrains from changing HTML in files with certain filename extensions, including .asp (ASP), .cfm (ColdFusion), .jsp (JSP), and .php (PHP). This default is set so that Dreamweaver won't accidentally modify the code contained in any such non-HTML tags. You can change the Dreamweaver default rewriting behavior so that it rewrites HTML when it opens such files, and you can add other file types to the list of types that Dreamweaver doesn't rewrite.

Note that Dreamweaver encodes certain special characters (by replacing them with numerical values) when you enter them in the Property inspector. It's usually best to let Dreamweaver perform this encoding, because the special characters will be more likely to display correctly across platforms and browsers. However, because such encoding may interfere with third-party tags, you may want to change the Dreamweaver encoding behavior when you're working with files containing third-party tags.

To allow Dreamweaver to rewrite HTML in more kinds of files:

1. Choose Edit > Preferences and select the Code Rewriting category.
2. Select either of the following options:

- Fix Invalidly Nested and Unclosed Tags
- Remove Extra Closing Tags

3. Do one of the following:

- Delete one or more extensions from the list of extensions in the Never Rewrite Code: In Files with Extensions option.
- Deselect the Never Rewrite Code: In Files with Extensions option. (Deselecting this option allows Dreamweaver to rewrite HTML in all types of files.)

To add file types that Dreamweaver should not rewrite:

1. Choose Edit > Preferences and select the Code Rewriting category.
2. Select either of the following options:

- Fix Invalidly Nested and Unclosed Tags
- Remove Extra Closing Tags

3. Make sure the Never Rewrite Code: In Files with Extensions option is selected, and add the new file extensions to the list in the text field.

   If the new file type doesn't appear in the file-types pop-up menu in the File > Open dialog box, you may want to add it to the Configuration/Extensions.txt file. For details, see "Changing the default file type" on page 4.

To turn off Dreamweaver encoding options:

1. Choose Edit > Preferences and select the Code Rewriting category.
2. Deselect either or both Special Characters options.

For information on the other Code Rewriting preferences, see Dreamweaver Help.