**ASP.NET and XHTML**

This page is specific to

**Microsoft Visual Studio 2008/.NET Framework 3.5**

Other versions are also available for the following:

[Microsoft Visual Studio 2005/.NET Framework 2.0](http://msdn.microsoft.com/en-us/library/exc57y7e(VS.80).aspx)

[.NET Framework 3.0](http://msdn.microsoft.com/en-us/library/exc57y7e(VS.85).aspx)

[Microsoft Visual Studio 2010/.NET Framework 4](http://msdn.microsoft.com/en-us/library/exc57y7e(VS.100).aspx)

ASP.NET allows you to create Web pages that are conformant with XHTML standards. XHTML is a World Wide Web Consortium (W3C) standard that defines HTML as an XML document. Creating Web pages that are conformant with XHTML standards has several advantages:

* It guarantees that the elements in the pages are well formed.
* Because many browsers are increasingly moving toward supporting XHTML, creating pages that conform to XHTML standards helps ensure that your pages render consistently in all browsers.
* Using XHTML helps to make pages conform more readily to accessibility standards.
* XHTML is extensible, allowing the definition of new elements.
* An XHTML page is much easier to read programmatically for situations in which the Web page is processed by a computer instead of being read by users, and the document can be manipulated using transformations.

The W3C has identified several levels of XHTML conformance: XHTML 1.0 Transitional, XHTML 1.0 Frameset, XHTML 1.0 Strict, and XHTML 1.1. The XHTML 1.1 specification is the strictest of these levels. The XHTML 1.0 Frameset and Transitional specifications define XML-based HTML markup, but allow certain widely used constructs. Many existing Web pages can be made conformant with XHTML 1.0 Frameset or Transitional specifications, but cannot meet XHTML 1.0 Strict or XHTML 1.1 specifications without requiring substantial revision to replace functionality implemented using constructs that are not allowed in the specifications.

For more information about the XHTML standard, see the specification for the Second Edition of XHTML 1.0 on the [W3C Web site](http://go.microsoft.com/fwlink/?linkid=37125).

http://i.msdn.microsoft.com/Global/Images/clear.gif ASP.NET Features for XHTML Conformance

XHTML defines elements and attributes more strictly than HTML. By default, all markup that is produced by ASP.NET and Web server controls included with ASP.NET now conforms to the XHTML 1.0 Transitional standard. In many cases, the markup produced by ASP.NET conforms to XHTML 1.1 standards as well. Unless noted otherwise, references to the XHTML standard in this topic mean both XHTML 1.0 and XHTML 1.1.

Some of the XHTML rules that differ from HTML include the following:

* All elements either include an explicit closing tag or are self-closing (with />).
* Tag and attribute names are rendered in lowercase, and attribute values are included in double quotation marks. For example, if you use a [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control on your page, when the page is rendered, the [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx) control emits HTML that is conformant with XHTML standards. All generated elements use explicit opening and closing tags (or self-closing tags), and attribute values are included in double quotation marks.
* Formatting information is rendered using only cascading style sheet styles. To support this standard, if the page includes an XHTML **DOCTYPE** element, ASP.NET controls do not render **font** elements or attributes, such as **bgcolor**, that would not conform to XHTML standards.
* In ASP.NET, if controls generate IDs, as occurs in the [Repeater](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.repeater.aspx), [GridView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.gridview.aspx), and other controls, the format of the IDs match XHTML 1.0 Transitional guidelines.
* ASP.NET dynamically adds an **action** attribute to the **form** element. By default, **form** elements include a **name** attribute, which is allowed in the XHTML 1.0 Transitional specification. This helps maintain backward compatibility with existing applications which rely on client script that addresses **form** elements using the form name.

|  |
| --- |
| **NoteNote:** |
| The **name** attribute on the **form** element is not allowed in XHTML 1.1 guidelines. You can configure your application not to render a **name** attribute. For details, see "Controlling XHTML Rendering of ASP.NET Pages and Controls" later in this topic. |

* Because XHTML requires all elements to be enclosed in a container element, ASP.NET controls, such as **input** elements, are rendered in **div** elements. This includes the HTML markup rendered for controls, such as the [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx), [CheckBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.checkbox.aspx), and [RadioButton](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.radiobutton.aspx) controls. It also includes hidden fields, such as the element that is used to store view-state data.
* ASP.NET encodes characters, such as **&** (for example, as **&amp;**). This includes URLs that are generated to reference ECMAScript and the contents of encoded values, such as view state.
* Any **script** elements that are rendered into the page use the appropriate **type** attribute (for example, **type="type/javascript"**) and do not include a **language** attribute. This pertains to scripts that are created by the page or controls that require client script to perform a postback — such as the **HyperLink**, [LinkButton](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.linkbutton.aspx), [Calendar](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.calendar.aspx), and [TreeView](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.treeview.aspx) controls, as well as validator controls — and by the **RegisterHiddenField**, **RegisterStartupScript**, and **RegisterClientScriptBlock** methods. Script blocks that you create are not automatically amended with a **type** attribute.
* If ASP.NET renders script blocks, the content of the script blocks is rendered inside an XML (HTML) comment.

http://i.msdn.microsoft.com/Global/Images/clear.gif Controlling XHTML Rendering of ASP.NET Pages and Controls

Under some circumstances, you might want ASP.NET controls to render markup in the stricter format specified by the XHTML 1.1 specification. The default rendering includes some markup that does not conform to the XHTML 1.1 specification. For example, XHTML 1.1 standards prohibit the use of a **name** attribute in an HTML **form** element.

Conversely, you might want ASP.NET to render markup that does not conform to XHTML 1.0 Transitional specifications. This is typically true when you have existing pages that rely on tags or attributes that were supported in earlier versions of ASP.NET but do not conform to XHTML standards.

You can configure your Web site to render markup in three ways:

* Legacy (which is similar to how markup was rendered in previous versions of ASP.NET)
* Transitional (XHTML 1.0 Transitional)
* Strict (XHTML 1.0 Strict)

For more information, see [How to: Configure XHTML Rendering in ASP.NET Web Sites](http://msdn.microsoft.com/en-us/library/ms178159.aspx).

|  |
| --- |
| **NoteNote:** |
| The option to render legacy markup is provided primarily to assist you in migrating existing pages to the current version of ASP.NET, and might not be supported in future versions of ASP.NET. |

**Legacy Rendering**

When rendering is set to legacy, ASP.NET pages and controls change their rendering to the behavior of earlier versions of ASP.NET. Changes include the following:

* The **form** element is rendered with a **name** attribute.
* ASP.NET does not automatically render a **div** element inside the **form** element as a container for controls.
* Validator controls are rendered as **span** elements with custom attributes, such as **controltovalidate**.
* The **img** element does not render **alt** and **src** attributes unless you explicitly include them.
* If required to support auto-postback behavior, controls will render a **language** attribute (for example, **language="javascript"**).
* The **nowrap** attribute is included for controls that render a **div** element (such as the [Panel](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.panel.aspx) control) if the control's **Wrap** property is set to **false**.
* [ImageButton](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.imagebutton.aspx) controls render a **border** attribute.
* Any **br** elements rendered into the page are rendered as <br>. However, if you explicitly include a <br /> tag, the page renders it as-is.
* The [DataGrid](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.datagrid.aspx) and [Calendar](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.calendar.aspx) controls include a **bordercolor** attribute in the rendered **table** elements if its **BackColor** property is set.

http://i.msdn.microsoft.com/Global/Images/clear.gif Specifying the DOCTYPE Element and XHTML Namespace

Valid XHTML Web pages must contain a **DOCTYPE** declaration that identifies the page as an XHTML page and references the XHTML schema to which it conforms. The page must also include attributes on the HTML tag that reference the XHTML namespace. ASP.NET does not automatically create a **DOCTYPE** declaration when the page is rendered. Instead, you should create the **DOCTYPE** declaration with the appropriate XML namespace references.

|  |
| --- |
| **NoteNote:** |
| Visual designers, such as Visual Studio, generally include default page templates that include a **DOCTYPE** declaration. If you are using a visual designer, check that it creates new pages with the **DOCTYPE** declaration you require. For more information, see [XHTML in Visual Web Developer](http://msdn.microsoft.com/en-us/library/dac1ff35(en-us,VS.90).aspx). |

The following code example shows a **DOCTYPE** declaration that you might add to your page.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl65_ctl00_ctl02_code');" \o "Copy Code)

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html >

By not automatically generating the **DOCTYPE** declaration, ASP.NET retains greater rendering flexibility for browsers that have differing levels of compliance with the differing HTML standards that can be specified in a **DOCTYPE** declaration.

|  |
| --- |
| **NoteNote:** |
| If the **DOCTYPE** declaration is removed, XHTML compliance will not be met. The page will not be considered an XHTML page and will not reference the XHTML schema. |

Many browsers will also change their rendering based on the presence or absence of the **DOCTYPE** declaration and an XML namespace declaration. If those elements are present, browsers typically use a standards-based rendering. When the elements are not present, many browsers render using browser-specific rules that vary among browser types (which is sometimes referred to as rendering in "quirks mode"), and can therefore result in unpredictable rendering.

Similarly, you can control the MIME type of the page. By default, a page sets the MIME type to **text/html**. However, you can override the page's MIME type by setting the **ContentType** attribute in the [@ Page](http://msdn.microsoft.com/en-us/library/ydy4x04a.aspx) directive, as shown in the following code example.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl65_ctl00_ctl05_code');" \o "Copy Code)

<%@ Page Language="VB" **ContentType="application/xhtml+xml"** %>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl65_ctl00_ctl06_code');" \o "Copy Code)

<%@ Page Language="C#" **ContentType="application/xhtml+xml"** %>

**Example ASP.NET Page with Required XHTML Elements**

The following code example shows a simple ASP.NET page that is XHTML conformant.

Visual Basic

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl65_ctl00_ctl07_code');" \o "Copy Code)

<%@ Page Language="VB" AutoEventWireup="false" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html >

<script runat="server">

Sub Button1\_Click(sender As Object, e As EventArgs)

Label1.Text = "Built at " & DateTime.Now.ToString()

End Sub

Sub listFruit\_SelectedIndexChanged(sender As Object, e As EventArgs)

Label1.Text = "You selected " & listFruit.SelectedItem.Text

End Sub

</script>

<head runat="server">

<title>ASP.NET XHTML Page</title>

</head>

<body>

<form id="Form1" runat="server">

<div>

<h1>ASP.NET Sample Page for XHTML</h1>

<p>

<asp:listbox runat="server" id="listFruit" AutoPostBack="true"

onselectedindexchanged="listFruit\_SelectedIndexChanged">

<asp:listitem>Apple</asp:listitem>

<asp:listitem>Banana</asp:listitem>

<asp:listitem>Orange</asp:listitem>

</asp:listbox>

</p>

<asp:label runat="server" id="Label1" ForeColor="white"

BackColor="black" />

<br />

<asp:button runat="server" id="Button1" onclick="Button1\_Click"

Text="Click me"/>

</div>

</form>

</body>

</html>

C#

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl65_ctl00_ctl08_code');" \o "Copy Code)

<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<script runat="server">

void Button1\_Click(Object sender, EventArgs e)

{

Label1.Text = "Built at " + DateTime.Now.ToString();

}

void listFruit\_SelectedIndexChanged(Object sender, EventArgs e)

{

Label1.Text = "You selected " + listFruit.SelectedItem.Text;

}

</script>

<html >

<head runat="server">

<title>ASP.NET XHTML Page</title>

</head>

<body>

<form id="Form1" runat="server">

<div>

<h1>ASP.NET Sample Page for XHTML</h1>

<p>

<asp:listbox runat="server" id="listFruit" AutoPostBack="true"

onselectedindexchanged="listFruit\_SelectedIndexChanged">

<asp:listitem>Apple</asp:listitem>

<asp:listitem>Banana</asp:listitem>

<asp:listitem>Orange</asp:listitem>

</asp:listbox>

</p>

<asp:label runat="server" id="Label1" ForeColor="white"

BackColor="black" />

<br />

<asp:button runat="server" id="Button1" onclick="Button1\_Click"

Text="Click me"/>

</div>

</form>

</body>

</html>

Note the following:

* The **form** element does not contain an **action** attribute because the **action** attribute is added when the page is rendered.
* The formatting properties of the [Label](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.label.aspx) control will render as **style** attributes.
* Because the **script** element containing server script is not rendered to the browser, it does not require a **type** attribute.
* The page renders client script at runtime to enable the auto-postback behavior of the [ListBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) control, but the page renders the script in an XHTML-compatible manner.

http://i.msdn.microsoft.com/Global/Images/clear.gif XHTML Conformance of Static Text and HTML Elements

ASP.NET does not alter static text or non-server HTML elements that you put into a page. For example, an ASP.NET Web page might include the [TextBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.textbox.aspx) and **Button** controls as well as some static text that you add inside <p></p> tags. ASP.NET can render XHTML for the **TextBox** and **Button** controls, but it cannot correct XHTML errors that occur between the <p></p> tags. If you create static text or HTML elements, make sure that they are XHTML conformant. You can check your pages by validating them, as explained in the next section.

Unknown attributes of HTML controls are passed through to the rendered control output and will not validate as valid XHTML markup. For example, specifying the **ID** attribute for the [HtmlHead](http://msdn.microsoft.com/en-us/library/system.web.ui.htmlcontrols.htmlhead.aspx) control will result in markup that is not XHTML 1.0 Strict-compliant. To validate your markup, use a markup validator such as the [World Wide Web Consortium (W3C) Validation Markup Service](http://go.microsoft.com/fwlink/?LinkId=25315).

http://i.msdn.microsoft.com/Global/Images/clear.gif Checking XHTML Conformance of ASP.NET Web Pages

After creating your ASP.NET Web pages, you might want to check whether they will render correct XHTML. If the page contains ASP.NET Web server controls, there is no way to check the page while writing it because the controls render XHTML only when the page runs.

|  |
| --- |
| **NoteNote:** |
| Some visual designers, such as Visual Studio, can provide design-time XHTML validation of the page's markup. |

To check the validity of the XHTML for your pages, you must use a service that runs the page and checks its output. A typical strategy is to deploy your pages to a publicly available server. The server can be a test server; it does not have to be a production server. However, it must be open to the Internet. You can then use a validation service that can read your pages programmatically.

A popular service is the W3C Markup Validation Service, which is maintained by the World Wide Web Consortium. To use this validator, enter the URL of the page that you want the service to check. The validation site requests the page and produces a report of any errors that it finds. Alternatively, you can save the source for a Web page and submit it as a file to the validation service. For more information about this validation service, see the [W3C Web site](http://go.microsoft.com/fwlink/?linkid=37125).

If the page that you are checking contains dynamic content, or if users can personalize Web pages in your site, you must be sure to test pages with different content to be sure all possible content in the page is valid. In some cases, this can be difficult because the variation in possible page output is too great to be able to test effectively.

**Configuring Browser Capabilities for Markup Validation**

When a page is processed, ASP.NET examines information in the request about the current browser, and based on the browser type (user agent string), renders markup that is appropriate for that browser. For more information, see [ASP.NET Web Server Controls and Browser Capabilities](http://msdn.microsoft.com/en-us/library/x3k2ssx2.aspx).

If you submit an ASP.NET Web page to a validation service such as the W3C Markup Validation Service, ASP.NET might render a version of the page that does not conform to XHTML standards. This is because the validator service does not report itself as a browser type that ASP.NET recognizes, such as Internet Explorer or Mozilla. When ASP.NET cannot recognize the browser type, it defaults to rendering downlevel markup, which does not include XHTML-conformant elements and attributes, or features such as cascading style sheet styles.

You can configure your application to send the correct XHTML-conformant markup to the validation service by creating a browser definition for the validation service's user agent string. For example, the W3C Markup Validation Service reports a user agent that begins with "W3C\_Validator". To create a browser definition for the W3C validator, you can create a .browser file in your application's App\_Browsers folder (you can name the .browsers file anything you like) and then add the following **browsers** element.

[Copy Code](javascript:CopyCode('ctl00_MTCS_main_ctl67_ctl00_ctl03_code');" \o "Copy Code)

<browsers>

<browser id="W3C\_Validator" parentID="default">

<identification>

<userAgent match="^W3C\_Validator" />

</identification>

<capabilities>

<capability name="browser" value="W3C Validator" />

<capability name="ecmaScriptVersion" value="1.2" />

<capability name="javascript" value="true" />

<capability name="supportsCss" value="true" />

<capability name="tables" value="true" />

<capability name="tagWriter"

value="System.Web.UI.HtmlTextWriter" />

<capability name="w3cdomversion" value="1.0" />

</capabilities>

</browser>

</browsers>

For more information about creating browser definitions, see [Browser Definition File Schema (browsers Element)](http://msdn.microsoft.com/en-us/library/ms228122.aspx).

http://i.msdn.microsoft.com/Global/Images/clear.gif Exceptions to XHTML Conformance

Although ASP.NET generates XHTML–conformant markup, some controls support optional functionality that, if used, might result in non-compliant markup.

|  |
| --- |
| **NoteNote:** |
| Each control renders its own markup. Custom controls created by other parties might not be written to produce XHTML-conformant output. If you are using a custom control, check with the control vendor to determine which standards the control supports. |

**Target Attribute**

Examples of controls that might result in non compliant markup are controls that allow you to include a **target** attribute to specify their client-side behavior:

* [AdRotator](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.adrotator.aspx)
* [BulletedList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.bulletedlist.aspx)
* [HyperLink](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.hyperlink.aspx)
* [HyperLinkColumn](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.hyperlinkcolumn.aspx)
* [ImageMap](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.imagemap.aspx)
* [MenuItem](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.menuitem.aspx)
* [TreeNode](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.treenode.aspx)

Pages that include the controls with their **target** attribute set will not validate against XHTML 1.1. If it is critical for you to create pages that are 100% conformant with XHTML 1.1 standards, you should avoid using options, such as the **target** attribute, that result in non-conformant markup.

**Select Element**

A [DropDownList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.dropdownlist.aspx) or [ListBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) control can be used to create a single-item or multiple-item selection. The [DropDownList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.dropdownlist.aspx) and the [ListBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) control each render an HTML **select** element. If the [DropDownList](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.dropdownlist.aspx) or the [ListBox](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listbox.aspx) control does not contain at least one [ListItem](http://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.listitem.aspx) control, then the rendered **select** element contains no child **option** elements and will not validate against XHTML 1.1.