XHTML & JavaScript Guidelines

Version 1.0

Developer's Edition

Document Version 01.21.03

Table of Contents

Executive Summary

1 Overview

This style guide has been created for designing, maintaining and modifying the Admin Tool. The guide is designed to provide information about acceptable and appropriate design requirements. This style guide should be used as the reference for developing content on screen and designing the graphical user interfaces.

Why XHTML?

The W3C will no longer be revising HTML, except as an application of XML, i.e. XHTML. Therefore, any future enhancements and revisions will be done to the XHTML language. In addition, since XHTML is XML based, it gives us greater flexibility in developing content that is usable by any XHTML-conforming user agent.

With XHTML, we are compatible with browser versions 5.0 or above. Therefore, we will not be supporting Netscape 4.x browsers since they (a) are not fully compliant with XHTML and (b) are used by only 2% of Internet users (statistic is as of October 2002).

Guidelines & Standards

2 XHTML

This section of the document specifies the XHTML for Admin Tool. It is imperative that the XHTML produced for the Admin Tool remain consistent. This document serves as the guide that any and all individuals involved in producing or handling XHTML for the Admin Tool must follow.

This document will review the requirements for XHTML coding in particular how it will be applied.

2.1 General Overview

What is XHTML?

XHTML is a family of current and future document types and modules that reproduce, subset, and extend HTML 4. XHTML family document types are XML based, and ultimately are designed to work in conjunction with XML-based user agents. XHTML reproduces essentially all the features of HTML but is subject to the stricter syntax of XML. As a quick overview, the differences are:

1. All element and attribute names must be in lowercase.
2. All empty elements must be written using XML's special empty-element tags.
3. All end tags must be present – there are no optional end tags
4. The html element must contain a single head element, followed by a single body element or a single frameset element.
5. Every head must contain a single title element.
6. Every document must begin with an XML declaration or use a meta element.

2.2 File names

For our applications, all content element pages should be given names that help explain their functionality. Names should begin with a lower case letter and multiple words should have the first letter capitalized. The file names can be more than 8 characters long, must not contain any spaces and use either an .html or .jsp extension.

Example: EditProfile.html or EditProfile.jsp

1. Image file names will follow the same standards.
2. The root page in any directory is named `Index.html` or `Index.jsp`

2.3 XHTML Elements

2.3.1 Mandatory XHTML Elements

All of these elements must be present in every .jsp layout template document. These elements must not appear in files that are pulled into the main template since they can only appear once in a compiled .jsp page:.

```
<!DOCTYPE Doctype goes here>

<html>

<head>

        <meta>

        <title>....</title>

</head>

<body>....</body>

</html>
```

Doctype Element

There must be a DOCTYPE declaration in the document prior to the root element. The document type declaration used in our applications is XHTML 1.0 Transitional. This DTD module is identified by the PUBLIC and SYSTEM identifiers:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"

SYSTEM "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Root Element

The root element of the .jsp template must be `<html>`.

The `<html>` element must designate the XHTML namespace using the xmlns attribute. The namespace for XHTML 1.0 is: http://www.w3.org/1999/xhtml. Though not required by the XHTML 1.0 specification, you should define the primary language of your document in the root `<html>` tag. The "lang=" argument from HTML 4.0 is being phased out for XML's " `xml:lang=` ". For now, we will go with the interim solution and use both:

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

For a list of language codes, visit http://www.oasis-open.org/cover/iso639a.html

Meta Element

XHTML standards state that `<?xml version="1.0" encoding="iso-8859-1"?>` should precede the DOCTYPE element. However, some 5.0+ browser-Web server combinations are incompatible with the `<?xml...?>` tag. Therefore, that XML tag will be omitted and the `<meta>` tag will be used to declare the UTF-8 character set. (UTF-8 is backward-compatible with the ASCII character set)

```
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
```

XHTML v. HTML 4 Elements

Table 2-1 Deprecated Elements & Recommended Alternatives

Deprecated Elements	Recommended Alternative
<applet>	use object
<basefont>	use CSS style sheets
	use CSS style sheets
<dir>	use ul list and CSS style sheets
<font>	use CSS style sheets
<isindex>	use input element inside a form
<menu>	use ul list and CSS style sheets
<s>	use CSS style sheets
<strike>	use CSS style sheets
<u>	use CSS style sheets

Documents Must Be Well Formed

Essentially this means that all elements must either have closing tags or be written in a special form (as described below), and that all the elements must properly nest.

Correct: nested elements

```
<p>here is a correct emphasized <em>paragraph</em>.</p>
```

Incorrect: overlapping elements

```
<p>here is an incorrect emphasized <em>paragraph.</p></em>
```

Elements & Attributes Must Be in Lowercase

In XHTML, all tags are defined in lowercase; therefore, `<p>` and `<P>` are two different tags.

```
<body> or <td class="name">
```

All elements must be terminated by the end tag , even those that were previously not required, such as `</li>` or `</p>` or `</option>`. Elements with no corresponding end tags should use the XML "`/>`" closure.

> Note: Include a space before the trailing / and > of empty elements for compatibility with older browsers.

Example: `<br />` not `<br/>`

In XHTML 1.0, all attributes must be in lowercase, all attributes must have arguments, and all arguments must be in double quotes, even those that are numeric. Therefore, use <hr height="5" />, not <hr height=5 />.

```
<img src="<dr:image i18nImage="false" imageName="/images/spacer.gif" />" alt=""
width="135" height="21" border="0" />
```

Note: On image tags, the sizing attributes will help the browser display the page faster, while the alt tag is required by XHTML for accessibility.

Note: For internationalization, if an image is not internationalized (e.g., a blank 1 px image for spacing), then i18nImage="false" must be present within the <dr:image> tag. Otherwise, the image will not show up on the page.

Attribute minimization is not allowed. With XHTML it is not permitted to write:

```
<td nowrap>
```

Instead, all attribute-value pairs must be written in full.

```
<td nowrap="nowrap">
```

The following is a list of the minimized attributes in HTML and how they should be written in XHTML.

Table 2-2 Minimized Attributes: HTML vs. XHTML

HTML	XHTML
compact	compact="compact"
checked	checked="checked"
declare	declare="declare"
readonly	readonly="readonly"
disabled	disabled="disabled"
selected	selected="selected"
defer	defer="defer"
ismap	ismap="ismap"
nohref	nohref="nohref"
noshade	noshade="noshade"
nowrap	nowrap="nowrap"
multiple	multiple="multiple"
noresize	noresize="noresize"

Name and ID Attributes

HTML 4 defined the `name` attribute and introduced the `id` attribute. Both are designed to be used as fragment identifiers. In XML, fragment identifiers are of type `ID` and there can only be a single attribute of type `ID` per element. In XHTML 1.0, the `id` attribute is defined to be of type `ID`. In order to ensure that XHTML 1.0 documents are well-structured XML documents, XHTML 1.0 documents must use the `id` attribute when defining fragment identifiers on these elements: `a, applet, form, frame, iframe, img,` and `map`

Note: To ensure that your document will work in the older browsers, you use both name and id, with identical attribute values, like this:

```
<a name="foo" id="foo"> … </a>
```

3 JSP Custom Tags

See the JSP Standards Guide for the custom tags we're using.

4 URL Encoding

In order to ensure that documents are compatible with historical HTML user agents and XML-based user agents, ampersands used in a document that are to be treated as literal characters must be expressed themselves as an entity reference (e.g. "&"). For example, when the `href` attribute of the `a` element takes parameters, it must be expressed as:

```
<a href="<dr:action actionName="DisplayProductDetailPage"/>
&amp;productID=<bean:write name="prods" property="productID"/>
&amp;catalogID=<bean:write name="form" property="catalogID"/>">
```

5 Cascading Style Sheets

Use CSS to define a set of formatting and display properties. For example:

```
.text  { font-size: 10px; color: #000; font-family: Verdana, Sans-Serif; }
```

Then apply those properties to the XHTML tags:

```
<div class="text">
```

Do not use use comment tags to “hide” the content of the `<style>` element. XML parsers are permitted to silently remove the contents of comments. Therefore, the historical practice of "hiding" scripts and style sheets within "comments" to make the documents backward compatible is likely to not work as expected in XML-based user agents.

The three methods of incorporating style sheets, in order of preference, are:

5.1 External Style Sheets

This is the preferred method as the style sheet exists in a .css file which is separate from the XHTML document, is retrieved at display time and requires only one file to be edited for site-wide changes.

```
<link rel="stylesheet" href="style.css" />
```

5.2 Embedded Style Sheets

Use this method if defining a background image since the url encoding will not work in an external style sheet. For example:

```
<style type="text/css">

.dotsBottom{ background-image: url(<bean:write name="page"
property="baseSiteUrl"/>/images/backgrounds/products_dots_bottom.gif);
background-repeat: no-repeat; background-position: bottom; padding-top: 5px;
padding-bottom: 6px; }

</style>
```

5.3 Inline Styles

Use this method if the style is a one-off, i.e., if it is not to be used elsewhere in the application. For example:

```
<div style="display: none;">
```

5.4 Styles for the Admin Tool

Current style definitions are in the Admin/html/style.css file

6 JavaScript

Wherever possible, define all of your functions in the `<head>` of your XHTML page.

This way, all functions will be defined before any content is displayed. Otherwise, the user might perform some action while the page is still loading which would trigger an event handler and call an undefined function, leading to an error.

Also, any images for rollovers should be defined in the `<head>` tag so they can be preloaded before the page is drawn in the browser. If the images are defined in the `<body>` tag using the onLoad event handler, that event is triggered after the HTML document has been transferred from the server to the browser; therefore, the images will not be loaded until after the page is loaded.

Do not use use comment tags to “hide” the content of the `<script>` element. XML parsers are permitted to silently remove the contents of comments. Therefore, the historical practice of "hiding" scripts and style sheets within "comments" to make the documents backward compatible is likely to not work as expected in XML-based user agents.

In XHTML, the script and style elements are declared as having #PCDATA content. As a result, < and & will be treated as the start of markup, and entities such as < and & will be recognized as entity references by the XML processor to < and & respectively. Wrapping the content of the script or style element within a CDATA marked section avoids the expansion of these entities.

```
<script language=”javascript” type="text/javascript">
<![CDATA[
      ... unescaped script content ...
]]>
</script>
```

CDATA sections are recognized by the XML processor and appear as nodes in the Document Object Model, see Section 1.3 of the DOM Level 1 Recommendation [DOM].

An alternative is to use external script documents.

```
<script language="javascript" type="text/javascript" src="<bean:write
name="page" property="baseSiteUrl"/>/scripts/Standard.js"></script>
```

6.1 Variables

When declaring a variable as a constant, its value should be set on the line that it is declared on. For example:

```
var constMaxRecords = 25;
```

Functions should have short descriptive names, which helps in the understanding of their functionality. The names should take on the form of a verb/action. The function names should begin with a lower case letter and should use capital letters to separate multiple words. For example:

```
function getTask(intClaimNum)
```

6.2 JavaScript blocks (e.g. “{ … }” sections)

These should be formatted with the opening brace on its own line, separate from the encompassing statement, and the closing brace on its own line.

```
function sampleFun(intIterations)

{

        var i

        for (i = 0; i < intIterations; i ++ )

        {

                // do something

        }

}
```

Even when for, if or while statements encompass only one line, explicit “{“and “}” should be used:

```
if (boolCheck)

{

        alert('true!');

}
```

7 Document Formatting

Indent your Code!!

This is not optional. Code should be indented following the logical structure, so that all the statements in the same logical level are at the same indentation. Only deviate from this requirement to eliminate a browser's insertion of extraneous spacing.

Include comments in places where the code is unclear and/or non-portable.

1. Place comments above the appropriate tag/link.

2. Indent inline block comments to the same level as the code they describe.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" SYSTEM
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
    <meta http-equiv="content-type" content="text/html; charset=iso-8859-1" />
    <title>test document</title>
    <link rel="stylesheet" href="<bean:write name="page"
property="baseSiteUrl"/>/style.css" />
    <script language="javascript" type="text/javascript" src="<bean:write
name="page" property="baseSiteUrl"/>/scripts/Standard.js"></script>
</head>
<body>
    <table width="100%" border="0" cellpadding="0" cellspacing="5">
        <tr>
            <td>
                <img src="<dr:image imageName="/images/banner.gif" />" width="1"
height="1" alt="name" />
            </td>
            <td class="text">
                <b><dr:resource template="templateName" key="KEY_NAME" /></b>
            </td>
      </tr>
        <tr>
            <td colspan=”2” class="copy">
                <br /> <br />
                <template:get name='gutter' />
            </td>
        </tr>
    </table>
</body>
</html>
```

www.digitalriver.com

Corporate Headquarters
Digital River, Inc.
9625 West 76th Street, Suite 150
Eden Prairie, MN 55344
U.S.A.

European Office
Digital River, Inc.
9 The Pavillions
Ruscombe Business Park
Twyford, Berkshire, RG10 9NN
United Kingdom