

XHTML Tests

TABLE OF CONTENTS

1	INTRODUCTION.....	3
2	REFERENCES.....	3
3	OBJECTIVES	4
3.1	XHTML BASIC/MP FUNCTIONALITY	4
4	MODULES	5
4.1	DOCUMENT CONFORMANCE.....	5
4.2	UA CONFORMANCE MODULE	7
4.3	STRUCTURE MODULE.....	19
4.4	TEXT MODULE	23
4.5	HYPertext MODULE.....	35
4.6	LIST MODULE	40
4.7	BASIC FORMS MODULE.....	43
4.8	BASIC TABLES.....	62
4.9	IMAGE MODULE	73
4.10	LINK MODULE.....	78
4.11	BASE MODULE	79
4.12	COMMON ATTRIBUTES MODULE.....	82
4.13	XHTML MOBILE PROFILE.....	89

1 INTRODUCTION

This document specifies the test requirements and procedures for testing browsers that support **XHTML Basic**, an XML-compliant version of HTML and is a subset of the full XHTML specification, as well as **XHTML Mobile Profile**, a version of XHTML that includes all of XHTML Basic as well as a limited number of extensions

2 REFERENCES

- [XHTML Basic](http://www.w3.org/TR/xhtml1-basic/), W3C Recommendation 19 December 2000 [XHTMLBASIC]
- [XHTML Mobile Profile](http://www.wapforum.com/what/technical.htm), WAP-277-XHTMLMP-20011029-a [XHTMLMP]
- [Modularization of XHTML](http://www.w3.org/TR/xhtml1-modularization/), W3C Recommendation 10 April 2001 [XHTMLMOD]
- [XHTML 1.0: The Extensible HyperText Markup Language](http://www.w3c.org/TR/xhtml1), W3C Rec 26 January 2000 [XHTML10]
- [HTML 4.01 Specification](http://www.w3c.org/TR/html4), W3C Recommendation 24 December 1999 [HTML401]
- [Key Words for Use in RFCs to Indicate Requirement Levels](http://www.ietf.org/rfc/rfc2119.txt), March 1997 [RFCKeyWords]
- [The text/html Media Type](http://www.ietf.org/rfc/rfc2854.txt), June 2000 [RFC2854]
- [The application/xhtml+xml Media Type](http://www.rfc-editor.org/rfc/rfc3236.txt), Jan 2002 [RFC3236]
- [Returning Values from Forms: multipart/form-data](ftp://ftp.isi.edu/in-notes/rfc2388.txt), August 1998 [RFC2388]

3 OBJECTIVES

3.1 XHTML Basic/MP Functionality

Testing will verify that the browser functions as described in the XHTML Basic Specification [XHTMLBASIC], including the normative documents given as references in that specification and the XHTML Mobile Profile Specification [XHTMLMP]. Complete references to sections of each document used to create the test assertions are included with the description of each individual test, below. In the cases where a specification does not provide clear direction on a particular User Agent expectation (such as is the case with some of the negative tests), a behavior is inferred. These cases are marked "*inferred*" in the References section for that test.

4 MODULES

4.1 Document Conformance

4.1.1 Testing Objective

The Document Conformance Module is used to verify whether a user agent generates appropriate errors when attempting to load documents in which the DOCTYPE is missing or incorrect.

Note: *The assertions in this module apply to VALIDATING UAs only.* If a non-validating user agent is used, the tester must verify that the UA loads all test pages without crashing the browser or generating an error.

4.1.2 Test Content

The tests for this module begin with the number 45.1.x. The `/xbasic/doc_conformance/default.html` page serves as a menu for the local versions of the tests, and contains links to the starting pages for this module. The directory within Test Suite which contains these tests is `/xhtml_DOC_conformance`. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
45.01.01	Missing DOCTYPE	missing_doctype.html

Assertion: When a **DOCTYPE** declaration is not present in the document, the user agent MUST report an error.

References:
[XHTMLBasic] Section 2.1

Actions:

1. Verify that a document titled "Missing DOCTYPE" is loaded.
2. Activate the "GO" link to load a page with a missing DOCTYPE.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the page loads without error and does not crash the browser.

45.01.02	DOCTYPE precede	doctype_position.html
----------	-----------------	-----------------------

Assertion: When a **DOCTYPE** declaration does not precede the root element, the user agent MUST report an error.

References:
[XHTMLBasic] Section 2.1

Actions:

1. Verify that a document titled "DOCTYPE Precede" is loaded.
2. Activate the "GO" link to load a page in which the DOCTYPE is in the wrong place.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the page loads without error and does not crash the browser.

45.01.03	Wrong Public Identifier	wrong_pubid.html
----------	-------------------------	------------------

Assertion: When there is an error in the public identifier of the **DOCTYPE** declaration, the user agent MUST report an error.

References:

[XHTMLBasic] Section 2.1

Actions:

5. Verify that a document titled "Wrong Public Identifier" is loaded.
6. Activate the "GO" link to load a page with an incorrect identifier.
7. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
8. For non-validating UAs, verify that the page loads without error and does not crash the browser.

45.01.04 **Wrong DTD Reference** **wrong_dtdref.xhtml**

Assertion: When there is an error in the DTD reference of the **DOCTYPE** declaration, the user agent MUST report an error.

References:

[XHTMLBasic] Section 2.1

Actions:

1. Verify that a document titled "Wrong DTD Ref" is loaded.
2. Activate the "GO" link to load a page with an invalid DTD reference.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the page loads without error and does not crash the browser.

45.01.05 **DoCTYPE Capitals** **doctype_caps.xhtml**

Assertion: When the **DOCTYPE** keyword is not in all caps, the user agent MUST report an error.

References:

[XHTMLBasic] Section 2.1

Actions:

1. Verify that a document titled "DoCTYPE Capitals" is loaded.
2. Activate the "GO" link to load a page which contains "DoCTYPE".
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the page loads without error and does not crash the browser.

4.2 UA Conformance Module

4.2.1 Testing Objective

The UA Conformance Module provides tests for basic element, attribute, character entity, and whitespace behaviors. Some tests, which require validation of the content based on a DTD, may only be applicable to validating user agents and are indicated as such below.

4.2.2 Test Content

Each test in this module begins with the number 45.2.x. The xbasic/UA_conformance/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages of all tests for this module. The directory within Test Suite which contains these tests is /xhtml_UA_conformance. The subdirectories /DTD, /elements, /attributes, /char_entities, /whitespace, and /content_type contain the associated tests in both the local version and Test Suite version. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.02.01	Invalid Position	DTD/invalid_child.xhtml
----------	------------------	-------------------------

Assertion: When a document is not consistent with the definitions in the associated DTD, the user agent MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. This particular test checks for only one aspect of DTD validation. A non-validating user agent MAY attempt to render the document, but MUST NOT crash the browser.

References:

[XHTMLBasic]	Section 2.2
[XHTML10]	Section 3.2.1

Actions:

1. Verify that the page "Invalid Position" is loaded.
2. Select the "GO" link to load a page in which elements appear in incorrect positions within the document.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.02	Required Child Element	DTD/missing_req_child.xhtml
----------	------------------------	-----------------------------

Assertion: When a document is not consistent with the definitions in the associated DTD, the user agent MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. This particular test checks for only one aspect of DTD validation. A non-validating user agent MAY attempt to render the document, but MUST NOT crash the browser.

References:

[XHTMLBasic]	Section 2.2
[XHTML10]	Section 3.2.1

Actions:

1. Verify that the page "Required Child element" is loaded.

2. Select the "GO" link to load a page in which a child element is missing.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.03 Required Attribute DTD/missing_req_attr.xhtml

Assertion: When a document is not consistent with the definitions in the associated DTD, the user agent MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. This particular test checks for only one aspect of DTD validation. A non-validating user agent MAY attempt to render the document, but MUST NOT crash the browser.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1

Actions:

1. Verify that the page "Required Attribute" is loaded.
2. Select the "GO" link to load a page in which a required attribute is missing.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.04 Unrecognized Element elements/unrec_elem.xhtml

Assertion: When an element is encountered and the user agent does not recognise it, it MUST render the element's content.

Note: This assertion applies to NON-VALIDATING UAs only. Validating UAs will reject the document and give an error.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.4

Actions:

1. For non-validating UAs, verify that a page with the title "Unrec. Element" loads, that the text in the paragraph is displayed normally, and that the tags, <foo> and </foo> are not displayed.
2. For validating UAs, verify that an error is generated.

45.02.05 Empty Element elements/empty_elem.xhtml

Assertion: When an element is declared EMPTY in the XHTML Basic DTD and is specified using the short hand notation **<element/>** or as **<element></element>**, the user agent MUST treat it as well formed, and MUST render the element correctly without displaying the closing tag.

Note: This mandatory requirement must be implemented even for NON-VALIDATING UAs.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.6

Actions:

1. Verify that a page with the title, "Empty Element" loads.
2. Select the "Go" link to load a page which uses conventional and shorthand markup for an empty element.
3. Verify that two images of an hourglass are displayed.

45.02.06 Misplaced Slash **elements/misplaced_slash.xhtml**

Assertion: The browser MUST NOT crash when encountering an empty element with its closing slash placed within the element, rather than at its end.

Note: This assertion applies to NON-VALIDATING UAs only. (Validating UAs will report "Bad Content".)

References: (*inferred*)

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1

Actions:

1. Verify that a page with the title, "Misplaced Slash" loads.
2. Select the "Go" link to load a page which contains an empty element with a misplaced closing slash.
3. Verify that the browser does not crash.

45.02.07 Space After Slash **elements/space_after_slash.xhtml**

Assertion: The browser MUST NOT crash when encountering an empty element with whitespace between its closing slash and the end of the tag (e.g. "/ >"). The browser SHOULD ignore the whitespace and render the content normally.

Note: This assertion applies to NON-VALIDATING UAs only. (Validating UAs will report "Bad Content".)

References: (*inferred*)

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1

Actions:

1. Verify that a page with the title, "Space After Slash" loads.
2. Select the "Go" link to load a page, which contains an element with a space between the closing slash and the end of the tag.
3. Verify that the browser does not crash, and that the image of a timepiece loads correctly.

45.02.08 Missing End Tag **elements/missing_endtag.xhtml**

Assertion: When the start tag(<element>) of an element which is not declared as EMPTY in the XHTML Basic DTD is encountered and its end tag(</element>) is not present, the user agent MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. Non-validating UAs MAY attempt to insert the missing end tags and render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.3

Actions:

1. Verify that a page with the title, "Missing End Tag" loads.
2. Select the "Go" link to load a page which is missing end tags.
3. For validating UAs, verify that the user agent reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.09 **Extra Closing Tags** `elements/extra_endtag.xhtml`

Assertion: When there are extra end or closing tags for an element, the UA MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. Non-validating UAs MAY attempt to ignore the extra end tags and render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.3, 4.6

Actions:

1. Verify that a page with the title, "Extra End Tags" loads.
2. Select the "Go" link to load a page which has several extra end tags.
3. For validating UAs, verify that the user agent reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.10 **Non Lower Case** `elements/non_lowercase.xhtml`

Assertion: When the element names are specified in non-lowercase, the user agent MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. Non-validating agents MAY attempt to render the content as if it were in lowercase, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.2

Actions:

1. Verify that a page with the title, "Non lowercase" loads.
2. Select the "Go" link to load a page which uses upper case element names.
3. For validating UAs, verify that the user agent reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.11 Invalid Nesting `elements/invalid_nesting.xhtml`

Assertion: When a document has two or more elements, delimited by their start and end tags, not nested properly within one another, the user agent **MUST** report an error.

Note: This assertion is applicable for **VALIDATING** UAs only. Non-validating UAs **MAY** ignore improper nesting or attempt to repair it, but **MUST NOT** crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.1

Actions:

1. Verify that a page with the title, "Invalid Nesting" loads.
2. Select the "Go" link to load a page which uses improperly nested elements.
3. Verify that the user agent reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.12 Unrecognized Attribute `attributes/unrec_attr.xhtml`

Assertion: When an attribute is encountered and the user agent does not recognize it, it **MUST** ignore the attribute and its value.

Note: This assertion is applicable to **NON-VALIDATING** UAs only. Validating UAs will reject the document with an error.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.5

Actions:

1. For non-validating UAs, verify that a page with the title "Unrec. Attribute" loads, the paragraph is displayed normally, and the invalid attribute "city=Boston" is ignored.
2. For validating UAs, verify that the UA reports an error.

45.02.13 Non Lower Case `attributes/non_lowercase.xhtml`

Assertion: When the attribute names are specified in non-lower-case, the user agent **MUST** report an error.

Note: This assertion is applicable to **VALIDATING** UAs only. Non-validating agents **MAY** attempt to render the content as if it were in lowercase, but **MUST NOT** crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.2

Actions:

1. Verify that the page "Non lowercase" is loaded.
2. Select the "GO" link to load a page with non-lowercase attributes.

3. For validating UAs, verify that the user agent generates an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.14 Invalid Value `attributes/invalid_attr_value.xhtml`

Assertion: When an attribute is specified with a value and the user agent does not recognize the value, it SHOULD use the default attribute value.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.6

Action:

1. Verify that the page "Invalid Attribute value" is loaded and a single line text input field is rendered, even though the attribute "type" is set to the invalid type "foo".

45.02.15 Char Entity Value `attributes/centity_attr.xhtml`

Assertion: When the attribute value contains character entities, the user agent MUST replace them with the correct character from respective document character set.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.2, 5.3.2

Action:

1. Verify that the page entitled "Char Entity - Attr. Value" is loaded. Verify that the alt text, which contains a character entity, &, is displayed correctly.

45.02.16 Whitespace Value `attributes/wspace_attr.xhtml`

Assertion: When an attribute is specified with a value having a sequence of whitespace characters (including line breaks), the user agent MAY reduce them to a single inter-word space. User agent MAY also strip any leading and trailing whitespace from attribute values.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 4.7

Actions:

1. Verify that the page "Whitespace - Attr Value" is loaded.
2. Verify that the "alt" text for the image is displayed.
3. The text "an image" may be displayed with only one character space in between the two words, even though there is multiple whitespace between the two words.

45.02.17 Nested Quotes 1 `attributes/nested_quotes1.xhtml`

Assertion: When an attribute value is enclosed in double quotation marks (") and the value contains single quotation marks ('), the user agent MUST treat that as well-formed.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.2

Actions:

1. Verify that the page "Nested Quotes 1" is loaded.
2. Verify that the word 'ok' in the alt text is surrounded by single quotes.

45.02.18 Nested Quotes 2 **attributes/nested_quotes2.xhtml**

Assertion: When an attribute value is enclosed in single quotation marks (') and the value contains double-quotation marks ("), the user agent MUST consider it as well-formed.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.2

Actions:

1. Verify that the page "Nested Quotes 2" is loaded.
2. Verify that the word "ok" in the alt text is surrounded by double quotes.

45.02.19 Nested Quotes 3 **attributes/nested_quotes3.xhtml**

Assertion: When an attribute value contains improperly nested quotes, the page containing the content MUST NOT crash the browser.

Note: This assertion applies to NON-VALIDATING UAs only. (Validating UAs will report "Bad Content".)

References: (*inferred*)

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.2

Action:

1. Verify that the page "Nested Quotes 3" loads without crashing the browser.

45.02.20 No Value **attributes/no_value.xhtml**

Assertion: When an attribute name is specified without a value (for e.g. checked=, multiple=), the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY simply ignore the content and render the rest of the document, or MAY attempt to repair it (e.g. change "checked" to "checked=checked"), but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.5

Actions:

1. Verify that the page "No Value" is loaded.
2. Select the "GO" link to load a page that contains the attribute "checked".
3. For validating UAs, verify that the user agent reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.21 No Quotes `attributes/no_quotes.xhtml`

Assertion: When an attribute is specified and the value is not enclosed within quotation marks, the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY simply attempt to render using the attribute value as if it were quoted, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.4

Actions:

1. Verify that the page "No Quotes" is loaded.
2. Select the "GO" link to load a page that contains the construct "".
3. For validating UAs, verify that the user agent reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.02.22 Solo Quotes `attributes/solo_quotes.xhtml`

Assertion: When an attribute contains only one quote (missing an opening or closing quote) the UA will MUST NOT crash.

Note: This assertion applies to NON-VALIDATING UAs only. (Validating UA's will report "Bad Content".)

References: (*inferred*)

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.1, 4.4

Action:

1. Verify that the page "Solo Quotes" is loads without crashing the browser.

45.02.23 Named Char Entity `char_entities/char_entity1.xhtml`

Assertion: When a named character entity is encountered (e.g. &), the user agent MUST replace it with the associated character from the document character set and then display it.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.3, Section 24.2.1

Actions:

1. Verify that the page "Named Char Entity" is loaded.
2. Verify that each character is displayed as described, and that there is no extraneous whitespace between the character and the enclosing square brackets.
3. If the user agent does not have a font for an entity, verify that an appropriate symbol is displayed instead for that entity.
4. Select the "Next" links to move through all pages of characters.

45.02.24 Decimal Char Entity `char_entities/char_entity2.xhtml`

Assertion: When a decimal numeric character entity is encountered, the user agent MUST replace it with the associated character from the document character set and then display it.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.3

Actions:

1. Verify that the page "Decimal Char Entity" is loaded.
2. This page has a sample of entities from the Latin 1, Greek, and extended sets.
3. Verify that the entities are displayed as they are described.
4. If the user agent does not have a font for an entity, verify that an appropriate symbol is displayed instead for that entity.

45.02.25 Hexadecimal Char Entity `char_entities/char_entity3.xhtml`

Assertion: When a hexadecimal numeric character entity is encountered, the user agent MUST replace it with the associated character from the document character set and then display it.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.3

Actions:

1. Verify that the page "Hexadecimal Char Entity" is loaded.
2. This page has a sample of entities from the Latin 1, Greek, and extended sets.
3. Verify that the entities are displayed as they are described. If the user agent does not have a font for an entity, verify that an appropriate symbol is displayed instead for that entity.

45.02.26 Unrecognized Entity `char_entities/char_entity4.xhtml`

Assertion: When the user agent encounters an entity reference that is not recognized, then the user agent MUST render the entity reference as the characters (starting with the ampersand and ending with the semi-colon).

References:

[XHTMLBasic] Section 2.2
[XHTMLMOD] Section 3.5.7

Actions:

1. Verify that the page "Unrec. Char Entity" is loaded.
2. Verify that the unknown entities are displayed as "&cat;" and "&mouse;".

45.02.27 Unrenderable Entity `char_entities/char_entity5.xhtml`

Assertion: When the user agent encounters character entity references that are recognized but not renderable, then the user agent MUST render the document in such a way that it is obvious that the normal rendering has not taken place.

References:

[XHTMLBasic] Section 2.2

[XHTMLMOD] Section 3.5.8

Actions:

1. Verify that the page "Unrenderable Entities" is loaded.
2. Verify that the two non-rendering entities, `̉` and `̊` are displayed without normal rendering, (i.e. they are replaced with boxes, question marks, or with some other symbol).

45.02.28 XML Comment `whitespace/ws_comment.xhtml`

Assertion: When an XML comment is included in a document, the user agent MUST NOT render its contents.

References:

[XHTMLBasic] Section 2.2

[XHTML10] Section 3.2.9

Actions:

1. Verify that the page "XML Comment" is loaded.
2. Verify that the line, "Line 2", which is commented out using XML comments, is not rendered.

45.02.29 Surrounding Space `whitespace/ws_no_pre1.xhtml`

Assertion: When the user agent encounters white space surrounding block elements and the `xml:space` attribute is not set to 'preserve', the user agent MUST remove the whitespace.

References:

[XHTMLBasic] Section 2.2

[XHTML10] Section 3.2.9

Actions:

1. Verify that the page "Surrounding Space" is loaded.
2. Verify that the extraneous space before and after each paragraph is eliminated, and the two paragraphs are separated by normal paragraph spacing.

45.02.30 Lead/trail Space `whitespace/ws_no_pre2.xhtml`

Assertion: When leading and trailing white spaces are encountered within the contents of a block element and the `xml:space` attribute for that element is not set to 'preserve', the user agent MUST remove the leading and trailing whitespace.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.9

Actions:

1. Verify that the page "Leading/Trailing Space" is loaded.
2. Verify that the extraneous leading and trailing space within each paragraph is eliminated, and the two paragraphs are separated by normal paragraph spacing.

45.02.31 **Space Within** **whitespace/ws_no_pre3.xhtml**

Assertion: When a sequence of white space characters is encountered in the contents of an element and the xml:space attribute is not set to 'preserve' for that element, the user agent MUST reduce the sequence to a single space character.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.9

Actions:

1. Verify that the page "Space within" is loaded.
2. Verify that the paragraph, which contains large sequences of whitespace characters between each word, is rendered with only one space between each word.

45.02.32 **Line Feeds** **whitespace/ws_lf.xhtml**

Assertion: When the user agent encounters a line feed character within a block element and the xml:space attribute is not set to 'preserve', the user agent MUST convert the line feed into a single space.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.9

Actions:

1. Verify that the page "Line Feeds" is loaded.
2. Verify that the word "L i n e 1", which is composed of letters and line breaks, is rendered on one line, with a space between each letter.

45.02.33 **Char Entity Space** **whitespace/ws_centity.xhtml**

Assertion: When the Space(), Tab(), Carriage return(), Line feed(
), Form feed() or the Zero-width space(​) character entities are encountered, the user agent MUST render them as white space characters.

References:

[XHTMLBasic] Section 2.2
[XHTML10] Section 3.2.9

Action:

1. Verify that the page titled "Char Entity Space" is loaded.

2. Verify that the sentence "Welcome to the world of mobile phones.", in which each word is separated with one of the valid whitespace character entities listed above, displays a single space between each word.

45.02.34 TEXT/HTML `content_type/content_html.xhtml`

Assertion: The User Agent MUST accept documents with the content type of "text/html" and render them normally.

References:

[RFC2854] Section 2

Actions:

1. Verify that the page titled "Content-type:text/html" is loaded.
2. Select the GO link to load a page with that content type.
3. Verify that that page loads without errors and that the content header is not displayed at the top of the page.

45.02.35 XHTML+XML `content_type/content_xhtml.xhtml`

Assertion: The User Agent MUST accept documents with the content type of "application/xhtml+xml" and render them normally.

References:

[RFC3236] Section 2

Actions:

1. Verify that the page titled "Content-type:application/xhtml+xml" is loaded.
2. Select the GO link to load a page with that content type.
3. Verify that that page loads without errors and that the content header is not displayed at the top of the page.

45.02.36 Extended Named `char_entities/extended_named.xhtml`

Assertion: When a named extended character entity is encountered (e.g. ♠), the user agent MAY replace it with the associated character from the document character set and then display it, but MUST NOT crash.

References:

[XHTMLBasic] Section 2.2
[HTML401] Section 3.2.3, Section 24.3.1

Actions:

1. Verify that the page "Extended Named" is loaded.
2. Select the "Go" link. Verify that the page "Extended Characters XHTML" loads without crashing the browser.
3. The UA may attempt to render the character entities, and should replace entities it can not display with a meaningful symbol.

4.3 Structure Module

4.3.1 Testing Objective

The Structure Module provides tests for structural elements and attributes which make up a simple XHTML page, such as the <head>, <body>, and <title> elements.

4.3.2 Test Content

Each test in this module begins with the number 45.3.x. The xbasic/structure_module/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages of all tests for this module. The directory within Test Suite which contains these tests is /xhtml_structure. Starting pages are described below. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.03.01	Head, Body, & Title	complete.xhtml
----------	---------------------	----------------

Assertion: When the *title* element is encountered, the user agent MUST make the contents visible to the user as a document title.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.1
[HTML401]	Section 7.4.2

Action:

1. Verify that a page with the title "OMA" is loaded. Post the Results from this page.

45.03.02	No Head	no_head.xhtml
----------	---------	---------------

Assertion: When the *head* element is not included in the document, the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.1

Actions:

1. Verify that a page called "No head" is loaded.
2. Activate the "GO" link to load a page without a head element.
3. For validating UAs, verify that the User Agent reports an error and allows you to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.03.03	No Body	no_body.xhtml
----------	---------	---------------

Assertion: When the *body* element is not included in the document, the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY attempt to render the document, but MUST NOT crash when loading the test page. Note that in the case of

non-validating UAs, the "Back" link on the second page might not be rendered, and the tester may have to use the built-in "back" feature of the browser to return to the starting page to post a result.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.1

Actions:

1. Verify that a page called "No body" is loaded.
2. Activate the "GO" link to load a page without a body element.
3. For validating UAs, verify that the User Agent reports an error and allows you to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.03.04 **No Title** **no_title.xhtml**

Assertion: When the *head* element is encountered and the *title* element is not included in the contents of the *head* element, the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY attempt to render the document without a title, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.1
[HTML401] Section 7.4.2

Actions:

1. Verify that a page called "No title" is loaded.
2. Activate the "GO" link to load a page without a title element.
3. For validating UAs, verify that the User Agent reports an error and allows you to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.03.05 **No Root** **no_root.xhtml**

Assertion: When the root element for the document is missing, the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.1
[XML] Section 2.1

Actions:

1. Verify that a page called "No root" is loaded. Activate the "GO" link to load a page without a root element.
2. For validating UAs, verify that the User Agent reports an error and allows you to return to the starting page.
3. For non-validating UAs, verify that the browser does not crash.

45.03.06 Invalid Root Element `invalid_root.xhtml`

Assertion: When the root element of the document is not `<html>`, the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY ignore the suspect element and attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.1

Actions:

1. Verify that a page called "Invalid root" is loaded.
2. Activate the "GO" link to load a page with `<foo>` as the root element.
3. For validating UAs, verify that the User Agent reports an error and allows you to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.03.07 Invalid Namespace `wrong_html_ns.xhtml`

Assertion: When the `html` element is encountered and the `xmlns` attribute is set to a value other than "http://www.w3.org/1999/xhtml", the user agent MUST report an error.

Note: This assertion applies to VALIDATING UAs only. Non-validating agents MAY ignore this and attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.1

Actions:

1. Verify that the page "Wrong Namespace" is loaded.
2. Activate the "GO" link to load a page with an invalid namespace link.
3. For validating UAs, verify that the UA reports an error and allows the user to return to the start page.
4. For non-validating UAs, verify that the browser does not crash.

45.03.08 Case Sensitive Namespace `case_sense_ns.xhtml`

Assertion: When a page contains two ids which differ only by case, the UA must not crash.

Note: While a page may not contain two identical ids with differing cases, the UA behavior for this situation is undefined - some UAs may report an error, while others may attempt to render the page.

References:

[HTML401] Section 7.5.2, 12.2.1

Actions:

1. Verify that the page "Case Sensitive Namespace" is loaded.
2. Activate the "GO Case Sens 2" link to load a page with two ids, "BOTTOM" and "bottom".
3. Activate the "Go to BOTTOM" link to see if the UA will go to the correct internal document reference (this behavior is preferable, but not required).

4. Verify that the browser does not crash. Select the "Back" link.
5. Activate the "GO Case Sens 3" link to load a page with two links to the same anchor, but with different capitalizations of the word "mixed".
6. Activate the two links to see if the UA will go to the correct internal document reference (this behavior is preferable, but not required).
7. Verify that the browser does not crash. Select the "Back" link and post your results.

45.03.09 Non-Unique IDs `non_unique_ns.xhtml`

Assertion: When a page contains two ids with the same value, the UA must not crash.

Note: While a page may not contain two identical ids, the UA behavior for this situation is undefined - some UAs may report an error, while others may attempt to render the page.

References:

[XHTMLMOD] Section 4.3
[HTML401] Section 7.5.2, 12.2.1

Actions:

1. Verify that the page "Non-Unique IDs" is loaded.
2. Activate the "GO" link to load a page with two "a1" ids.
3. Activate the "Go to ref a1" link and verify that the browser does not crash.

45.03.10 Profile `profile.xhtml`

Assertion: When a page's <head> element contains a "profile" attribute with an invalid URI, the UA must not crash.

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.1
[HTML401] Section 7.4.1

Action:

1. Verify that the page "Profile" loads without crashing the browser and that the content is rendered normally.

4.4 Text Module

4.4.1 Testing Objective

The Text Module contains tests for validating functionality for all text related elements of XHTML Basic, such as <blockquote>, <div>, <p>, , etc. Note that since Mobile User Agents often have limited font support, text rendering functionality may vary greatly from one user agent to another. Therefore, most of the assertions in this section are optional.

4.4.2 Test Content

The tests for this module start with the number 45.4.x. The `xbasic/text_module/default.xhtml` page serves as a menu for the local versions of the tests, and contains links to the starting pages of all tests for this module. The directory within Test Suite which contains these tests is `/xhtml_text`. Starting pages are described below. Tests for the line break tests reside in a separate subdirectory, called `/line_breaks`. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
45.04.01	ABBR	abbr_element.xhtml

Assertion 1: When the ***abbr*** inline element is used, the abbreviated text enclosed within the ***abbr*** element MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed abbreviated text.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Action:

1. Verify that the page "ABBR Element" loads without error.

45.04.02	ACRONYM	acronym_element.xhtml
----------	----------------	------------------------------

Assertion 1: When the ***acronym*** inline element is used, the acronym enclosed within the ***acronym*** element MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed acronym text.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Actions:

1. Verify that the page "ACRONYM Element" loads without error.

45.04.03 ADDRESS

address_element.xhtml

Assertion 1: When the **address** inline element is used, the address text enclosed within the **address** element MAY be set apart from the surrounding content. The text MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed address text.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 7.5.6

Actions:

1. Verify that the page "ADDRESS element" is loaded.
2. Verify that the address text is rendered as a separate block and on a new line. The text may be indented or italicized.

45.04.04 BLOCKQUOTE

bquote_element.xhtml

Assertion 1: When a **blockquote** element is used, the content enclosed within the **blockquote** element MUST be rendered so that it is set apart from the surrounding content, and MAY be indented.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.2

Actions:

1. Verify that the page titled "BLOCKQUOTE element" is loaded.
2. Verify that the second paragraph is rendered as a separate block on a new line. The text may be indented.

45.04.05 CITE

cite_element.xhtml

Assertion 1: When the **cite** inline element is encountered, the citation text enclosed within the **cite** element MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed citation text.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2

[HTML401] Section 9.2.1

Actions:

1. Verify that the page "Cite Element" loads without error.

45.04.06 **CODE**

code_element.xhtml

Assertion 1: When the **code** inline element is encountered, the software code enclosed within the **code** element MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed software code.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.2
[HTML401] Section 9.2.1

Actions:

1. Verify that the page "Code Element" loads without error.

45.04.07 **DFN**

dfn_element.xhtml

Assertion 1: When the **dfn** inline element is encountered, the defining text enclosed within the **dfn** element MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed defining text.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.2
[HTML401] Section 9.2.1

Actions:

1. Verify that the page "DFN Element" loads without error.

45.04.08 **DIV**

div_element.xhtml

Assertion 1: When the **div** element is encountered, the content enclosed within the element MUST be rendered so that it is set apart from the surrounding content.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 7.5.4

Actions:

1. Verify that the page "DIV element" is loaded.
2. Verify that the text "Block1" and "Block2" are rendered as two separate blocks or two separate lines.

45.04.09 **EM**

em_element.xhtml

Assertion 1: When the **em** inline element is encountered the text enclosed within the **em** element MAY be displayed in an alternative typeface to give emphasis to the text.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Actions:

1. Verify that the page "EM element" is loaded.
2. Verify that the text "Boston" in "Boston is a beautiful city" is rendered in an alternative typeface (most UAs will render this as italicized).

45.04.10 **Headings**

heading_elements.xhtml

Assertion 1: There are six levels of headings in HTML with [H1](#) as the most important and [H6](#) as the least. Visual browsers usually render more important headings in larger fonts than less important ones

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Actions:

1. Verify that the page "Heading elements" is loaded. The six headings SHOULD be rendered in some way that distinguishes the top level headings from the lower level headings e.g. using progressively decreasing font size.

45.04.11 KBD

kbd_element.xhtml

Assertion 1: When the ***kbd*** inline element is encountered, the content enclosed within the ***kbd*** element MAY be displayed in an alternative typeface to give a change of emphasis to the enclosed text.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Actions:

1. Verify that the page "KDB" element is loaded without error.

45.04.12 P Element

p_element.xhtml

Assertion 1: When the ***p*** element is encountered, the implementation MUST place paragraph breaks before and after ***p*** element.

Assertion 2: When the ***p*** element is encountered, and the ***align*** attribute is not set, the implementation MUST left align the paragraph.

Assertion 3: When the ***p*** element is encountered, the implementation MUST line wrap the paragraph.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.5, 15.1.2

Actions:

1. Verify that the page "P element" is loaded.
2. Verify that the two paragraphs are left aligned and word wrapped, and that each is preceded and followed by a paragraph break. (Most UAs render the paragraph break as a blank line.)

45.04.13 PRE Element

pre_element.xhtml

Assertion 1: When the ***pre*** element is encountered, the enclosed text is 'preformatted', i.e. it MAY leave the white space intact.

Assertion 2: When the ***pre*** element is encountered, the user agent MAY disable automatic word wrapping.

Assertion 3: When the ***pre*** element is encountered, the user agent MUST NOT disable four-way scrolling, in order to allow the user to view all content.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.4

Actions:

1. Verify that the page "PRE element" is loaded.
2. The text given may preserve multiple whitespace characters between each word and may not wrap to a second line.
3. Verify that four-way scrolling (i.e. the ability to scroll horizontally and vertically) is not disabled.

45.04.14 PRE Font**pre_fwfont.xhtml**

Assertion 1: When the *pre* element is encountered, the enclosed text MAY be rendered in a fixed width font.

Note: This assertion is only applicable to user agents that support fixed width fonts.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.4

Actions:

1. Verify that the page "PRE Font" is loaded.
2. The text given may be rendered in a fixed width font. Example: These are fixed width chars.

45.04.15 PRE Exclude**pre_exclude1.xhtml**

Assertion: When the *img* element is encountered within the *pre* element, the user agent MUST report an error.

Note: This assertion is only applicable to VALIDATING UAs. Non-validating agents MAY attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.4

Actions:

1. Verify that the page "PRE exclude 1" is loaded.
2. Select the "GO" link.
3. For validating UAs, verify that the UA reports an error and allows user to return to start page.
4. For non-validating UAs, verify that the browser does not crash.

45.04.16 Q Element

q_element.xhtml

Assertion 1: When a **q** element is encountered, the user agent **MUST** render the content with delimiting quotation marks. The quoted text **MUST** appear without paragraph breaks.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.2

Action:

1. Verify that the page "Q Element" is loaded and that the text "The buck stops here" in the second paragraph is surrounded by quotes.
2. Verify also that this phrase is **NOT** rendered as a separate paragraph, e.g. there should be no paragraph break before or after the quotes.

45.04.17 Nested Quotes

nested_q.xhtml

Assertion 1: When a **q** element is nested in another **q** element, the user agent **MUST** ensure that the quotation marks are rendered with different quotation styles, in a language-sensitive manner.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.2

Actions:

1. Verify that the page "Nested Q" is loaded.
2. Verify that the phrase "Yesterday, my daughter greeted me, 'Happy Birthday, Daddy' and I became very happy" is rendered with two different styles of quotes, as shown. (The outer quotes may be double and the inner single, or vice versa.)

45.04.18 Nested 3 Levels

nested_3_levels.xhtml

Assertion: When several levels of **q** elements are nested, the browser **MUST NOT** crash.

References: (*inferred*)

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.2

Action:

1. Verify that the page "Nested 3 Levels" loads without errors and does not crash the browser.

45.04.19 Mixed Quotes

mixed_quotes.xhtml

Assertion: When a **q** element is nested in another **q** element, and the content within each element contains character entities which correspond to various quote styles, the browser MUST NOT crash. In addition, the inclusion of these character entities MUST NOT affect the normal rendering of the quotes generated by the **q** elements.

References: (*inferred*)

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.2, 3.2.2, 5.3.2

Actions:

1. Verify that the page "Mixed Quotes" loads without crashing the browser.
2. Verify that the first occurrence of the text OUTER Q is preceded with a quote and is NOT followed by a quote (e.g. "OUTER Q), and that the second occurrence of the text OUTER Q is followed by the same style of quote and is NOT preceded with a quote (e.g. OUTER Q"). Note that these quotes are generated by the **q** element.
3. Between the two OUTER Qs, there will be additional text and quotes.
4. Verify that the text INNER Q is surrounded by a different style of quote than OUTER Q (e.g. 'INNER Q'). Note that these quotes are generated by the **q** element.
5. Between the OUTER Qs and the INNER QUOTE, there will be additional text and character entity quotes.
6. Verify quote patterns correspond to the descriptions given, e.g. *two quotes*: " ", *single apos*: ' ', *irregularly nested*: ' ' ' ', etc. Note that all of these quotes are created by character entities.

45.04.20 SAMP

samp_element.xhtml

Assertion 1: When the **samp** inline element is encountered, the text (denoting computer output from program or script) enclosed within the **samp** element MAY be displayed in an alternative typeface to give a change of emphasis.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Action:

1. Verify that the page "SAMP element" loads without error.

45.04.21 SPAN

span_element.xhtml

Assertion: When the **span** element is encountered, the user agent MUST render the text enclosed within the element inline with the surrounding content.

References:

[XHTMLBasic]	Section 3
--------------	-----------

[XHTMLMOD] Section 5.2.2
[HTML401] Section 7.5.4

Action:

1. Verify that the page "SPAN element" is loaded, and that the word "Boston" in the second paragraph appears as normal inline text (i.e. it is NOT rendered separately, with a paragraph break new line, indentation, or other types of blocking.)
2. If the UA supports CSS, the word may be italicized, due to the in-line "font-style" property being used in this test.

45.04.22 **STRONG** **strong_element.xhtml**

Assertion 1: When the ***strong*** inline element is encountered, the text enclosed within the ***strong*** element is displayed in an alternative typeface to give strong emphasis to the text.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.2
[HTML401] Section 9.2.1

Actions:

1. Verify that the page "STRONG element" is rendered and that the word "Boston" appears in and emphasized text style, such as bold.

45.04.23 **STRONG and EM** **strong_em.xhtml**

Assertion: When the ***em*** and the ***strong*** inline element is encountered together in a document, the text enclosed within the ***strong*** element MAY be more emphasized than the text enclosed by the ***em*** element.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.2
[HTML401] Section 9.2.1

Action:

1. Verify that the page "STRONG and EM". The text "Boston" MAY be more emphasized than "is a historical city". For example, "Boston" might be bold and the rest of the text might be italicized.

45.04.24 **VAR** **var_element.xhtml**

Assertion 1: When the ***var*** inline element is encountered, the elements of the programming code samples enclosed within the ***var*** element MAY be displayed in an alternative typeface to give a change of emphasis.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.2.1

Actions:

1. Verify that the page "Var element" loads without error.

45.04.25 BR element**line_breaks/br_element.xhtml**

Assertion: When the *br* element is encountered, the current text line is ended, and text output is resumed on a new text line.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.2

Actions:

1. Verify that the page "BR element" is loaded, and that the numbers 1, 2, and 3 appear on three separate lines.
2. Verify that there are 2 blank lines between 3 and 4.

45.04.26 BR with content**line_breaks/br_error1.xhtml**

Assertion: When the *br* element is encountered and some content is included within the start and the end tags of the element, the user agent MUST report an error.

Note: This assertion applies only to VALIDATING UAs. Non-validating agents MAY ignore the content and attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.2

Action:

1. Verify that the page "BR with content" is loaded.
2. Select the "GO" link.
3. For validating UAs, verify that the UA reports an error and allows the user to return to the starting page.
4. For non-validating UAs, verify that the browser does not crash.

45.04.27 NBSP**line_breaks/nbsp.xhtml**

Assertion: When a non-breaking space entity (or) is encountered between two words, the user agent SHOULD NOT cause a line break.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.2

Actions:

1. Verify that the page "NBSP" is loaded, which contains three paragraphs. The first paragraph has between each word, the second has between each word, and the third has between each word.
2. None of the three paragraphs SHOULD wrap to a new line.

Note: E.g. browsers that don't support horizontal scrolling might wrap at the end of the screen.

45.04.28 **Plain Hyphen** **line_breaks/plain_hyphen.xhtml**

Assertion: When a plain-hyphen ("-") or a plain-hyphen character entity (- or -) is encountered the user agent MUST interpret it and display a "-"

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.2

Actions:

1. Verify that the page "Plain Hyphen" is loaded and that there is a hyphen between each work in the test paragraph.
2. Verify that the paragraph follows normal wrapping rules.

45.04.29 **Soft Hyphen** **line_breaks/soft_hyphen.xhtml**

Assertion 1: When a soft-hyphen character entity(­,­ or &#AD;) is encountered, the user agent MUST output a hyphen in its place if it is at the end of a line.

Assertion 2: When a soft-hyphen character entity(­,­ or &#AD;) is encountered, the user agent MUST NOT output a hyphen in its place if it is not at the end of a line.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 9.3.3

Actions:

1. Verify that the page "Soft Hyphen" is loaded, which contains three paragraphs. The first paragraph has ­ between each word, the second has ­ between each word, and the third has ­ between each word.
2. Verify that a hyphen is rendered at the end of each line for each paragraph, i.e. at the point at which the line wraps.
3. Verify that no hyphen appears between each word at any other point in the paragraph, that the hyphens between each word are ignored, and that no other characters, such as space, replace these hyphens. For example, typical output for a paragraph might look like this:

```
para1para1para1para1-
para1para1para1para1-
para1para1para1para1-
```

para1para1para1

4.5 Hypertext Module

4.5.1 Testing Objective

The Hypertext Module includes tests which verify the rendering and functionality of the <a> element and its attributes.

4.5.2 Test Content

The tests for this module begin with the number 45.5.x. The xbasic/hypertext/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages of all tests for this module. The directory within Test Suite which contains these tests is /xhtml_hypertext. Starting pages are described below. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.05.01	A Element	a_element.xhtml
----------	-----------	-----------------

Assertion 1: When the **a** element is encountered, the user agent MUST render the contents enclosed within the element inline with the surrounding content list.

Assertion 2: When the **a** element is encountered and the **href** attribute is set, the user agent MUST render the link in such a way as to make it visually distinct to the user.

Assertion 3: When the **a** element is encountered, the **href** attribute is set, and the link is activated, the user agent MUST load the resource specified by the attribute.

Assertion 4: When the **a** element is encountered, the **href** attribute is set, the link is activated and the anchor relating to the **href** attribute does not exist, the user agent MUST alert the user.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 12.2

Actions:

1. Verify that the page "A Element" is loaded.
2. Verify that the links "Valid link" and "Invalid link" are displayed inline with the surrounding content.
3. Verify that the links are visually distinct. Activate and verify that the "Valid link" loads a page titled "Valid works". Verify that when the "Invalid link" is activated, user agent alerts the user with an error such as "File not found".

45.05.02	Rel and Rev	rel_and_rev.xhtml
----------	-------------	-------------------

Assertion : The browser MUST NOT crash when the **rel** or **rev** attributes are encountered within the **a** element.

Note: These attributes are used to provide hints about the content in the linked page, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 12.2

Actions:

1. Verify that the page "Rel and "Rev" loads without error.
2. Select the "Table of Contents" link and verify that the linked page loads without error.

45.05.03 **Hreflang** **hreflang.xhtml**

Assertion: The browser MUST NOT crash when the *hreflang* attribute is encountered within the **a** element.

Note: This attribute is used to provide a hint about the language of the content in the linked page, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 12.2

Actions:

1. Verify that the page "Hreflang" loads without error.
2. Select the "Content" link and verify that the linked page loads without error.

45.05.04 **Type** **type_attrib.xhtml**

Assertion: The browser MUST NOT crash when the *type* attribute is encountered within the **a** element.

Note: This attribute is used to provide a hint about the content-type (MIME type) of the content in the linked page, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 12.2

Actions:

1. Verify that the page "Type Attrib" loads without error.
2. Select each link to load the described content. (Note that the "GIF Image" link only loads an image, not a full page, so you will have to use the "back" button on the browser to get back to the start page.)

45.05.05 **Charset** **charset.xhtml**

Assertion: The browser MUST NOT crash when the *charset* attribute is encountered within the **a** element.

Note: This attribute is used to provide a hint about the character set used in the linked page, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
--------------	-----------

[XHTMLMOD] Section 5.2.3
[HTML401] Section 12.2

Actions:

1. Verify that the page "Charset" loads without error.
2. Select the "Latin 1" link and verify that the linked page loads without error.

45.05.06 **Links Within** `links_within.xhtml`

Assertion 1: When the **a** element's **href** attribute is set to a fragment identifier matching another element's **id** attribute (e.g. `<a href="#<id>">`), the user agent MUST link to the element with the matching id, when the link source anchor is activated.

Assertion 2: When the **a** element is encountered and the **href** attribute is set using *only* a fragment identifier as its value and the link is activated, the user agent MUST navigate to the resource in the *current document* that matches the specified name.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.3
[HTML401] Section 12.2, 12.2.3

Actions:

1. Verify that the page "Links within" is loaded.
2. Activate the "Go to Bottom" link and verify that the user agent moves to the line near the bottom of the document, "This is Bottom".
3. Activate the "Go to Top" link and verify that the user agent moves to the top of the document.

45.05.07 **Relative Fragment URL** `rel_fragment.xhtml`

Assertion: When the **a** element is encountered, and the **href** attribute is set using a relative URL followed by a fragment identifier and the link is activated, the user agent MUST visit the resource specified by the destination anchor.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.3
[HTML401] Section 12.2

Actions:

1. Verify that the page "Rel Fragmt URL" is loaded.
2. Select the "Go Bottom" link to visit a new page.
3. Verify that the new page loads and that the user agent positions at the bottom of that page.

45.05.08 **Absolute Fragment URL** `abs_fragment.xhtml`

Assertion: When the **a** element is encountered, and the **href** attribute is set using an absolute URL followed by a fragment identifier and the link is activated, the user agent MUST visit the resource specified by the destination anchor.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.3

[HTML401] Section 12.2

Actions:

1. Verify that the page "Abs Fragmt URL" is loaded.
2. Select the "Go Bottom" link to visit a new page. Verify that the new page loads and that the user agent positions at the bottom of that page.

45.05.09 **Invalid Fragment** `invalid_frag.xhtml`

Assertion: When an unknown fragment identifier reference within the **a** element is encountered, and the link is activated, the user agent SHOULD alert the user, but MUST NOT crash.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.3
[HTML401] Section 12.2.4

Actions:

1. Verify that the page "Invalid Fragment" is loaded.
2. Select the "Go" link to attempt to visit a new page with an unknown identifier, "middle". The user agent should generate an error. Verify that the browser does not crash.

45.05.10 **Empty Anchor** `empty_anchor.xhtml`

Assertion: When a page contains an anchor element with no content, the user agent MUST NOT crash. The UA SHOULD render the page.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.3
[HTML401] Section 12.2

Actions:

1. Verify that the page "Empty Anchor" is loaded.
2. Select the "Go" link to visit a new page with an anchor, "middle", that has no content between its open and end tags. The new page should load. Verify that the browser does not crash.

45.05.11 **Illegal Nesting** `nested_links.xhtml`

Assertion: When an **a** element is nested within another **a** element, and the link is activated, the user agent MUST NOT crash. The UA MAY attempt to load the new page.

Note: This assertion applies to NON-VALIDATING UAs only. (Validating UAs will generate a "bad content" error.)

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.3
[HTML401] Section 12.2.2

Actions:

1. Verify that the page "Illegal Nesting" is loaded.
2. Select the "Go" link to visit a new page which contains embedded links. The link, "Inner", is embedded within the link "OUTER". Since the behaviors of this is undefined, the UA may attempt to render them as one or more links.
3. Activate the link(s). The new page(s) may load(s). Verify that the browser does not crash.
4. Return to the starting page to post your results.

4.6 List Module

4.6.1 Testing Objective

The List Module provides tests for list elements and their attributes including ``, ``, and others.

4.6.2 Test Content

The tests for this module begin with the number 45.6.x. The `xbasic/List/default.xhtml` page serves as a menu for the local versions of the tests, and contains links to the starting pages of all tests for this module. The directory within Test Suite which contains these tests is `/xhtml_list`. Starting pages are described below. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.06.01	Un-ordered List	un_ordered.xhtml
----------	------------------------	-------------------------

Assertion 1: When a *li* element is encountered in the content of an *ul* element, the user agent MUST display an unnumbered bulleted list item.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.4
[HTML401]	Section 10.2

Actions:

1. Verify that the page "UL/LI" is loaded.
2. Verify that the names of fruits are displayed as an unnumbered bulleted list.

45.06.02	Ordered List	ordered_list.xhtml
----------	---------------------	---------------------------

Assertion 1: When a *li* element is encountered within the content of an *ol* element, the user agent MUST display an automatically numbered bulleted list item.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.4
[HTML401]	Section 10.2

Actions:

1. Verify that the page "OL/LI" is loaded.
2. Verify that the names of fruits are displayed as a numbered list.

45.06.03	Nested UL	nested_ul.xhtml
----------	------------------	------------------------

Assertion: When a *ul* element is encountered nested within the content of a *li* element which is enclosed in an *ul* element, the user agent MUST display an unnumbered list item for each *li* element within the given subordinate *ul* element. Each subordinate unnumbered lists indentation MUST be greater than it's parent list.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.4
[HTML401] Section 10.2

Actions:

1. Verify that the page "Nested UL" is loaded.
2. Verify that there is an unnumbered bulleted list with the items "Automobiles", "Buses", "Planes", and "Trains". Verify that each nested list is indented further than its parent list.

45.06.04 **Nested OL**

nested_ol.xhtml

Assertion: When a *ol* element is encountered nested within the content of a *li* element which is enclosed in an *ol* element, the user agent MUST display numbered list items for each *li* element within the given subordinate *ol* element. Each subordinate unnumbered lists indentation MUST be greater than it's parent list.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.4
[HTML401] Section 10.2

Actions:

1. Verify that the page "Nested OL" is loaded.
2. Verify that there is a numbered list with the items "Flowers", "Roots", and "Fruits".
3. Verify that the fruit names are also numbered and are indented farther than its parent list.

45.06.05 **Definition List**

defn_list.xhtml

Assertion 1: When a *dl* element is encountered, the user agent MUST display a list of "defined terms" for each *dt* element and "definitions" for each *dd* element, and display them in a visually distinct block.

Assertion 2: When a *dd* element is encountered within a *dl* element, the user agent MUST display its contents in a visually distinct block.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.2.4
[HTML401] Section 10.3

Action:

1. Verify that the page "Defn List" is loaded. Verify that each Term/Definition pair is displayed as a visually distinct block. For example, each pair might be on a separate line with a little bit of space above and below the pair, as is common with block elements.
2. Verify that the definition items are visually distinct from the term items. For example, many user agents indent the definition.

4.7 Basic Forms Module

4.7.1 Testing Objective

This module verifies the rendering and functionality of elements and attributes related to forms, such as <form> and <input>.

4.7.2 Test Content

Tests for this module begin with the number 45.7.x. The xbasic/forms/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages for these tests. The directory within Test Suite which contains these tests is /xhtml_forms. Starting pages are described below. Each test is described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
45.07.01	Defaults	/form_elem/form_method_default.xhtml

Assertion 1: When the *form* element is encountered and the *method* attribute is not set and the enclosing form is submitted, the user agent MUST use the default method "GET" to transmit responses to the server.

Assertion 2: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is not set, the user agent MUST display a single line text input field.

Assertion 3: When the *form* element is encountered and the *action* attribute is set as a URL and the enclosing form is submitted, the user agent MUST submit form responses to that URL.

Assertion 4: When the *form* element is encountered and it contains markup, the user agent MUST render that markup in addition to the form controls.

Assertion 5: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "submit", the user agent MUST display a selectable button. When this button is selected, the user agent MUST submit all of the name-value pairs in the enclosing *form* element to the server.

Assertion 6: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "reset", the user agent MUST display a button control. When this button is pressed, the user agent MUST reset all of the elements in the enclosing *form* element to their original state (reset them to the values specified by the *value* attributes of each, if present or to blank values, if not).

Assertion 7: Text that follows the <form> element will be rendered on a new line.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.4, 17.4.1

Actions:

1. Verify that the page "Form Defaults" is loaded.
2. Verify that an emphasized "First Name:" label, created by embedding markup in the *form* element, and a text input field, created because no type was specified, are rendered.
3. Verify that the "Submit" and "Clear" buttons are displayed on separate lines.
4. Change the text in the "First Name" field.

5. Select the "Clear" button and verify that the field is reset to its initial value, "Joe".
6. Enter more text into the field and Submit the form.
7. Verify that the form is submitted to the correct URL by making sure the "Form Results" page is returned.
8. Verify that the page reports "Get Method works" and that the name entered into the form field is displayed.

45.07.02 Get Method `/form_elem/form_method_get.xhtml`

Assertion 1: When the *form* element is encountered and the *method* attribute is set as "GET" and the enclosing form is submitted, the user agent MUST use the method specified as the value to transmit responses to the server.

Assertion 2: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "text", the user agent MUST display a single line text input field.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.4.1

Actions:

1. Verify that the page "Form-Get" is loaded. Verify that a single line text input field is displayed.
2. Submit the form and verify that the response page reports "Get Method works".

45.07.03 Post Method `/form_elem/form_method_post.xhtml`

Assertion 1: When the *form* element is encountered and the *method* attribute is set as "POST" and the enclosing form is submitted, the user agent MUST use the method specified as the value to transmit responses to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3

Action:

1. Verify that the page "Form-Post" is loaded.
2. Submit the form. Verify that the response reports "Post Method works".

45.07.04 Invalid Action `/form_elem/invalid_action.xhtml`

Assertion 1: When the *form* element is encountered and the *action* attribute is set to an invalid URI and the enclosing form is submitted, the user agent MUST alert the user.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 12.2.4

Actions:

1. Verify that the page "Invalid Action" is loaded.
2. Submit the form. Verify that the UA alerts the user, such as with a "File Not Found" error.

45.07.05 Invalid Method `/form_elem/invalid_method.xhtml`

Assertion: When the *form* element is encountered and the *method* attribute is set to an invalid value and the enclosing form is submitted, the user agent MAY use the default method to transmit responses to the server, but MUST NOT crash the browser.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.13.1

Actions:

1. Verify that the page "Invalid Method" is loaded.
2. Submit the form and verify that the browser does not crash. The data may be submitted using the "get" method. If the response page loads, verify that the response reports "Get Method works".

45.07.06 Enctype Unset `/form_elem/enctype_unset.xhtml`

Assertion: When the *form* element is encountered and the *enctype* attribute is not set, the user agent MUST use the default encoding mechanism "application/x-www-form-urlencoded" to transmit the form responses to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3

Actions:

1. Verify that the page "Enctype Unset" is loaded.
2. Select the link to start the test.
3. Enter all the characters requested, in the order requested, into the text box and submit. The characters will be sent to the server using the Get method and default encoding.
4. Verify on the response page that the string of characters under "decoded:" matches what you typed. If they do not, there is a problem with the way the characters were encoded. The string under "encoded" is the actual encoding of the text you typed, and is useful for debugging if the test fails.
5. Post from the result page, or you may use the "Post Results" link on the start page if the results page doesn't load.

45.07.07 Enctype Set `/form_elem/enctype_set.xhtml`

Assertion: When the *form* element is encountered and the *enctype* attribute is set, the user agent MUST use the encoding mechanism specified by the value to transmit the form responses to the server.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.3

Action:

1. Verify that the page "Enctype Set" is loaded.
2. This test explicitly sets the enctype to "application/x-www-form-urlencoded", which is the only encoding we support.
3. Select the link to start the test.
4. Enter all the characters requested, in the order requested, into the text box and submit. The characters will be sent to the server using the Get method and this encoding.
5. Verify on the response page that the string of characters under "decoded:" matches what you typed. If they do not, there is a problem with the way the characters were encoded. The string under "encoded" is the actual encoding of the text you typed, and is useful for debugging if the test fails.
6. Post from the result page, or you may use the "Post Results" link on the start page if the results page doesn't load.

45.07.08 **Enctype Invalid** /form_elem/enctype_invalid.xhtml

Assertion: When the *form* element is encountered and the *enctype* attribute is set to an invalid value, the user agent MAY use the default encoding mechanism to transmit the form responses to the server, but must not crash the browser.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.3, 17.13.4

Action:

1. Verify that the page "Enctype Invalid" is loaded. This test sets the enctype to "foo", which is an invalid value.
2. Select the link to start the test.
3. Enter all the characters requested, in the order requested, into the text box and submit.
4. Verify that the browser does not crash. The characters may be sent to the server using the Get method and the default encoding, which is "application/x-www-form-urlencoded".
5. If the response page loads, verify that the string of characters under "decoded:" matches what you typed. If they do not, there is a problem with the way the characters were encoded. The string under "encoded" is the actual encoding of the text you typed, and is useful for debugging if the test fails.
6. Post from the result page, or you may use the "Post Results" link on the start page if the results page doesn't load.

45.07.09 **Input Size** /text_input/input_text_size.xhtml

Assertion 1: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "text" and the *value* attribute is specified, the user agent MUST display the value of the *value* attribute as the initial content of the text input field.

Assertion 2: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password" and the *value* attribute is set, the user agent MUST initially display the value of the attribute in the form of asterisks (*).

Assertion 3: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password", the user agent MUST display a single line text input field. When typing characters into this field, the user agent MUST display the characters using a mechanism that hides their true value (e.g. outputting a * for each character).

Assertion 4: When the *input* element is encountered within the contents of a *form* element and the *type* attribute of the *input* element is set to "text" and the *value* attribute is specified, and the user resets the form, the user agent MUST reset the content of the text input to the value of the *value* attribute.

Assertion 5: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password" and the *value* attribute is set, some other text is entered as input and the user resets the form, the user agent MUST reset the content of the password field to the value of the *value* attribute.

Assertion 6: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "text" and the *size* attribute is set as a positive integer, the user agent MUST permit more characters than defined by *size* attribute to be provided as input.

Assertion 7: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password" and the *size* attribute is set as a positive integer, the user agent MUST permit more characters than defined by *size* attribute to be provided as input.

Assertion 8: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "hidden", the user agent MUST NOT render any visual indication of the element.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.4, 17.4.1

Action:

1. Verify that the page "input-size" is loaded, where there are two rectangular single line input boxes displayed.
2. Verify that the "First Name" field contains the value "Joe" and the password field contains four asterisks.
3. Modify the two fields and verify that the "First Name" field displays the characters typed while the "Password" field replaces the letters typed with asterisks.
4. Select the "Clear" button and verify that the fields are reset to their initial values.
5. Enter more text into both fields and verify that each field allows as many characters to be entered as you want.

6. Verify that there is no visual indication of the hidden field, called "last", or its value, "Turner".
7. Submit the form.
8. Verify that the "First Name" and "Password" values reported are the same as what you entered, and that the value in the hidden field is "Turner".

45.07.10 Input Maxlength-Size /text_input/input_text_maxlength.xhtml

Assertion 1: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "text" and the *maxlength* attribute is set as a positive integer which is greater than that set for the *size* attribute, the user agent MUST permit more characters than defined by *size* attribute to be provided as input.

Assertion 2: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password" and the *maxlength* attribute is set as a positive integer which is greater than that set for the *size* attribute, the user agent MUST permit more characters than defined by *size* attribute to be provided as input.

Assertion 3: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "text" and the *name* attribute is set and some input is entered by the user and the enclosing form is submitted, the user agent MUST associate the text entered by the user as the value, along with the value of the *name* attribute, and submit the pair to the server.

Assertion 4: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password" and the *name* attribute is set and some input is entered by the user and the form is submitted, the user agent MUST associate the text entered by the user as the value, along with the value of the *name* attribute and submit the pair to the server.

Assertion 5: When the *input* element is encountered within the contents of a *form* element, the *type* attribute is set to "hidden", the *name* and *value* attributes are specified and the enclosing form is submitted, the user agent MUST associate the value specified by the *name* attribute with the value of the *value* attribute and submit the pair to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.2, 17.4, 17.4.1

Actions:

1. Verify that the page "MaxLength-Size" is loaded.
2. Type 4 lowercase m's into the first field.
3. Attempt to type more than 8 m's into the second and third fields.
4. Verify that the first field accepts the 4 characters, but the second and third fields only accept 8 characters.
5. Submit the form.

Verify that the results page reports that "First Name", "Last Name", and "Password" are equal to the values you entered, and that "Hidden" is equal to "pumpkin pie".

45.07.11 Input No Name /text_input/input_text_no_name.xhtml

Assertion 1: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "text" and the *maxlength* attribute is specified as a positive integer, the user agent MUST only allow that many number of characters to be provided as the input.

Assertion 2: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "password" and the *maxlength* attribute is specified as a positive integer, the user agent MUST only allow that many number of characters to be entered as the text input.

Assertion 3: When the *input* element is encountered within the contents of a *form* element, the *type* attribute is set to "text", the *name* attribute is not set and some input is entered by the user and the form is submitted, the user agent MUST ignore this text field and the value MUST NOT be submitted to the server.

Assertion 4: When the *input* element is encountered within the contents of a *form* element, the *type* attribute is set to "password", the *name* attribute is not set and some input is entered by the user and the form is submitted, the user agent MUST ignore this password field and the text MUST NOT be submitted to the server.

Assertion 5: When the *input* element is encountered within the contents of a *form* element, the *type* attribute is set to "hidden", the *name* attribute is not set and a value has been set for the element, the user agent MUST ignore this field and the value MUST NOT be submitted to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.2, 17.4, 17.4.1

Actions:

1. Verify that the page "Input No Name" is loaded.
2. Enter text into each field.
3. Verify that the fields MUST only accept up to 6 characters each (but may display as many as they can).
4. Submit the form and verify that the two fields and the hidden field do not contain any values.

45.07.12 **Illegal Names**

/text_input/input_illegal_names.xhtml

Assertion 1: *Name* attribute values MUST begin with a letter ([A-Za-z]) and MAY be followed by any number of letters, digits ([0-9]), hyphens ("-"), underscores ("_"), colons (":"), and periods (".").

Assertion 2: Documents that contain illegal *name* attribute values MUST NOT crash the browser.

Note: The behavior of UAs is not clearly defined for documents that do not conform to this assertion.

References:

[HTML401]	Section 6.2
-----------	-------------

Actions:

1. Verify that the page titled "Illegal Names" loads without crashing the browser. The "Employee", "Company", and "Employee ID" fields all contain illegal names: the first begins with a number; the second contains the illegal characters ";", "!", and "~"; and the third begins with "-". The "Password" and hidden fields contain complex, but legal names.

2. Submit the form and verify that the browser does not crash.

45.07.13 Illegal Size `/text_input/input_illegal_size.xhtml`

Assertion: If the *size* attribute contains a negative value, the attribute value SHOULD be ignored.

Note: *Size* is defined as a number of pixels. Although it is implied that a pixel cannot be a negative value, the behavior for UA's which encounter a negative *size* value is undefined. Presumably UAs would attempt to render the page and allow the user to enter text into the fields, perhaps using the default size for the field.

References: (*inferred*)
[HTML401] Section 17.4, 6.6

Actions:

1. Verify that the page "Illegal Size" loads without crashing the browser.
2. Verify that there are two fields, each of which allow the user to enter text.
3. Submit the form.
4. Verify that the response page loads without crashing and that the text entered is displayed along with the contents of the hidden field.

45.07.14 Illegal Maxlength `/text_input/input_illegal_max.xhtml`

Assertion: If the *maxlength* attribute contains a negative value, the attribute value SHOULD be ignored.

Note: *Maxlength* is defined as a number which must contain digits [0-9]. The behavior for UA's which encounter a negative *maxlength* value is undefined. Presumably UAs would attempt to render the page and allow the user to enter text into the fields, perhaps using the default *maxlength* (i.e. no limit) for the field.

References: (*inferred*)
[HTML401] Section 17.4, 6.6

Actions:

1. Verify that the page "Illegal Maxlength" loads without crashing the browser.
2. Verify that there are two fields, each of which allow the user to enter text.
3. Submit the form.
4. Verify that the response page loads without crashing and that the text entered is displayed along with the contents of the hidden field.

45.07.15 Label Explicit `/label/label_explicit.xhtml`

Assertion 1: When the *label* element is encountered within the contents of a *form* element and the *for* attribute is set with a value and the label receives focus, the user agent MAY pass the focus on to the form control having the same value for its *id* attribute as that of the *for* attribute of the *label* element, but MUST NOT crash.

Note: The *label* element may not be supported by all browsers. This test ensures that the browser does not crash when encountering typical uses of this element.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.9.1

Actions:

1. Verify that a page entitled "Label Explicit" is loaded, and that the page contains a form with "First Name", "Last Name", and "Password", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script and the browser does not crash.

45.07.16 **Label Implicit** `/label/label_implicit.xhtml`

Assertion: When the *label* element is encountered within the contents of a *form* element and another form control is enclosed within the *label* element and the label is given focus, the user agent MAY pass the focus on to the enclosed form control, but MUST NOT crash.

Note: The *label* element may not be supported by all browsers. This test ensures that the browser does not crash when encountering typical uses of this element.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.9.1

Actions:

1. Verify that a page entitled "Label Implicit" is loaded, and that the page contains a form with "First Name", "Last Name", and "Password", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script and the browser does not crash.

45.07.17 **Multiple Labels** `/label/label_multiple.xhtml`

Assertion: When more than one *label* element is encountered within the contents of a *form* element and the *for* attribute is set with the same value for all of them, the user agent MAY associate all those elements as labels with a single form control having its *id* attribute set to the value specified by the *for* attribute, thereby allowing the user agent to pass the focus on to this control whenever any of these labels receive focus. The browser MUST NOT crash.

Note: The *label* element may not be supported by all browsers. This test ensures that the browser does not crash when encountering typical uses of this element.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.9.1

Actions:

1. Verify that the page "Label Multiple" is loaded, and that the page contains a form with "First Name", "Last Name1", and "Password".
2. Verify that the form contains the text "Last Name2", a submit button, "Last Name3", a clear button, and "Last Name4".
3. Submit the form and verify that the field values are passed to the result script and the browser does not crash.

45.07.18 Labels Nested `/label/label_nested.xhtml`

Assertion: When the *label* element is encountered within the contents of another *label* element, the user agent MAY report an error, but MUST NOT crash.

Note: The *label* element may not be supported by all browsers. This test ensures that the browser does not crash when encountering typical uses of this element.

References:

[XHTMLBasic] Section 3
[XHTML10] Appendix B

Actions:

1. Verify that the page "Labels Nested" is loaded.
2. Select the "GO" link and verify that the browser does not crash.

45.07.19 Label Multiple Controls `/label/label_multi_controls.xhtml`

Assertion: When the *label* element is encountered within the contents of a *form* element and there is more than one control element contained within the label element, the user MAY report an error, but MUST NOT crash.

Note: The *label* element may not be supported by all browsers. This test ensures that the browser does not crash when encountering typical uses of this element.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.9.1

Actions:

1. Verify that the page "Label Multi-controls" is loaded.
2. Select the "GO" link. Verify that the browser does not crash.

45.07.20 Checkboxes `/checkbox/checkbox_name.xhtml`

Assertion 1: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "checkbox", the user agent MUST display a selectable checkbox control.

Assertion 2: When the *input* element is encountered within the contents of a *form* element, the *type* attribute is set to "checkbox", the *name* and *value* attributes are set, this *checkbox* element is selected and the enclosing form is submitted, the user agent MUST associate the value of the *name* attribute with the value of the *value* attribute and submit the pair to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.4.1, 17.13.2

Actions:

1. Verify that the page "Checkboxes" is loaded.
2. Verify that there are five "Colors" checkboxes (note: the label for the fifth checkbox is simply a white-space) and four "Cars" checkboxes displayed and that none of them are pre-checked.
3. Check a few colors and cars and submit the form.
4. Verify that the results page reports the same colors and same cars as the ones you checked.

45.07.21 Checkbox Pre-checked 1 /checkbox/checkbox_checked1.xhtml

Assertion: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "checkbox" and the *checked* attribute is set, the user agent MUST render a pre-selected box.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.4

Actions:

1. Verify that the page "Checkbox-checked1" is loaded.
2. Verify that the checkboxes for colors "Red" and "Green" are pre-checked.
3. Submit the form without changing the selections and verify that "Red" and "Green" are reported as the colors that were chosen.

45.07.22 Checkbox Pre-checked 2 /checkbox/checkbox_checked2.xhtml

Assertion: When the *input* element is selected within the contents of a *form* element and the *type* attribute is set to "checkbox", and the form is reset, the user agent MUST revert the checkbox selections back to their default settings. That is those checkbox controls with the *checked* attribute included will be displayed as selected. Those checkbox controls with the *checked* attribute omitted will be displayed as deselected.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.2.1

Actions:

1. Verify that the page "Checkbox-checked2" is loaded.
2. Change the selections from their defaults, "Red" and "Green", select the Clear button, and verify that the controls were reset to their default values. Submit the form.

45.07.23 Checkbox No Name /checkbox/checkbox_no_name.xhtml

Assertion: When the *input* element is encountered in the contents of the *form* element, the *type* attribute is set to "checkbox", the *name* attribute is not set and the *value* attribute is set, and this checkbox is selected and the enclosing form is submitted, the user agent MUST ignore this checkbox and not submit its value to the server.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.4.1, 17.13.2

Actions:

1. Verify that the page "Checkbox No Name" is loaded.
2. Select from both groups and submit the form. Verify that in results page, only selected cars will be displayed.

45.07.24 **Radio Buttons** `radio_name.xhtml`

Assertion 1: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "radio", the user agent MUST display a selectable button.

Assertion 2: When more than one *input* elements are encountered in the content of a single *form* element and the *type* attribute for all of them is set to "radio" and the *name* attribute is set with the same value, the user agent MUST allow only one selectable button to be selected at any given time. Selecting an un-selected button will cause the selected button with the same *name* attribute to become un-selected.

Assertion 3: When the *input* element is encountered within the contents of a *form* element, the *type* attribute is set to "radio", the *name* and *value* attributes are set, this *radio* element is selected by the user and the enclosing form is submitted, the user agent MUST associate the value specified by the *name* attribute with the value specified by the *value* attribute and submit the pair to the server.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.2.1, 17.4.1

Actions:

1. Verify that the page "Radio Buttons" is loaded.
2. Verify that there are four "Colors" radio boxes and four "Cars" radio buttons displayed, and each category has one pre-selected item. Select several different colors and cars and verify that the user agent only allows one color and one car to be selected at any time.
3. Submit the form.
4. Verify that the results page reports the same color, car, and cylinders as the ones you selected.

45.07.25 **Radio Pre-checked 1** `/radio/radio_checked1.xhtml`

Assertion: When the *input* element is encountered within the contents of a *form* element and the *type* attribute is set to "radio" and the *checked* attribute is set, the user agent MUST display this item as pre-selected.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.4

Actions:

1. Verify that a page entitled "Radio-checked 1" is loaded.
2. Verify that the button "Green" appears pre-selected.
3. Submit the form.
4. Verify that the response reports that you have selected the color Green.

45.07.26 **Radio Pre-checked 2** **/radio/radio/checked2.xhtml**

Assertion: When the *input* element is encountered within the *form* element, the *type* attribute is set to "radio", and the form is reset, the user agent MUST revert the radio button selections to their default settings. That is the radio button with the *checked* attribute included, the user agent will show the radio button selected. Those radio button(s) with the *checked* attribute omitted, the user agent will show the radio buttons deselected.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.2.1, 17.4.1

Actions:

1. Verify that the page "Radio-checked2" is loaded.
2. Change the selection from its default, "Green", select the Reset button, and verify that the controls were reset to their default values.
3. Submit the form.

45.07.27 **Radio No Name** **/radio/radio_no_name.xhtml**

Assertion: When the *input* element is encountered within the contents of a form element, the *type* attribute is set to "radio", the *value* attribute is set and the *name* attribute is not set, this radio button is selected and the form is submitted, the user agent MUST ignore this radio button and its value will not be submitted to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.4.1, 17.13.2

Action:

1. Verify that the page "Radio No Name" is loaded.
2. Select from both groups and submit the form.
3. Verify that in results page, only selected cars will be displayed.

45.07.28 **Select Single** **/select/select_single.xhtml**

Assertion 1: When a *select* element is encountered within the contents of a *form* element, the user agent MUST display a set of options (identified by enclosed *option* elements) from which the user can select.

Assertion 2: When a **select** element is encountered within the contents of a **form** element and the **multiple** attribute is not set, the user agent MUST display a set of options, from which only a single entry can be selected.

Assertion 3: When the **option** element is encountered within the content of a **select** element, the user agent MUST display the contents as a choice in the set of options.

Assertion 4: When a list of **option** elements are encountered within the content of a **select** element, and the **multiple** attribute is not set, and no options are pre-selected using the **selected** attribute, the user agent SHOULD render the first item in the list as pre-selected.

Assertion 5: When the **option** element is encountered in the contents of a **select** element and the **value** attribute is set and this **option** element is selected and the enclosing form is submitted, the user agent MUST associate the value of the **value** attribute with the value of the **name** attribute of the **select** element and submit the pair to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.6

Actions:

1. Verify that the page "Select single" is loaded.
2. Verify that five color options are available and that the first one is pre-selected.
3. Select several options in turn and verify that only one of them can be selected at any time.
4. Select any color and submit the form.
5. Verify that results page displays only the color name as defined by the "value" attribute (e.g. 'Red' not 'Red Color')

45.07.29 **Select Size**

/select/select_size.xhtml

Assertion: When the **select** element is encountered within the contents of a **form** element and the **size** attribute is set as a positive integer, it MAY define the number of selections that the user agent will make visible at a time, even if there are more option items included.

Note: Not all browsers display options as a "pull-down" style selection list. Many mobile devices render these as radio buttons or checkboxes, in which case displaying only **size** of the options (without a way to select the non-displayed items) would not make sense. While the **size** attribute may be ignored, all other functionality defined by the document must still be supported.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.6

Actions:

1. Verify that the page "Select-size" is loaded without any errors. (If the color options are displayed as a select list, only 2 options may be visible at a time, but this is optional.)
2. Verify that only one option can be selected.

3. Select and submit the form. Verify that the option chosen is displayed on the results page.

45.07.30 Select Multiple /select/select_multiple.xhtml

Assertion 1: When a ***select*** element is encountered within the contents of a ***form*** element and the ***multiple*** attribute is set, the user agent MUST display a set of options from which one or more items can be selected.

Assertion 2: When more than one ***option*** elements are encountered in the content of a ***select*** element and the ***multiple*** attribute is set for the ***select*** element and the ***selected*** attribute for the ***option*** elements is set, the user agent MUST render all those elements as pre-selected.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.6, 17.6.1

Actions:

1. Verify that the page "Select-Multiple" is loaded.
2. Verify that both 'Blue' and 'Green' options are pre-selected.
3. Verify that more than one option can be selected.
4. Select a few colors and submit the form.
5. Verify that the colors you selected are displayed on the results page.

45.07.31 Select Pre-select 1 /select/select_selected1.xhtml

Assertion: When the ***option*** element is encountered within the content of a ***select*** element and the ***selected*** attribute is set, the user agent MUST render the element as pre-selected.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.6.1

Actions:

1. Verify that the page "Select-selected 1" is loaded.
2. Verify that the option "None" is pre-selected.
3. Submit the form without changing the selection and verify that the correct option is displayed on the results page.

45.07.32 Select Pre-select 2 /select/select_selected2.xhtml

Assertion: When the ***option*** element is encountered within the content of a ***select*** element and the ***selected*** attribute is set (i.e. one or more options are pre-selected), and the user changes the selections, and the form is reset, the user agent MUST revert the ***option*** selections back to their pre-selected default settings.

References:

[XHTMLBasic]	Section 3
--------------	-----------

[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.2, 17.3, 17.6.1

Actions:

1. Verify that the page "Select-selected 2" is loaded.
2. Verify that the option "None" is pre-selected.
3. Change the default selection and activate the "Reset" button.
4. Verify that the selection is reset to its default value.
5. Submit the form.

45.07.33 **Option No Value** /select/option_no_value.xhtml

Assertion: When the *option* element is encountered in the contents of a *select* element, the *value* attribute is not set, this *option* element is selected, and the enclosing form is submitted, the user agent MUST associate the contents of the element with the value of the attribute named by the *name* attribute of the *select* element, and submit the pair to the server.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.6.1

Actions:

1. Verify that the page "Option no value" is loaded.
2. Verify that five color options are available.
3. Select any color and submit the form.
4. Verify that the results page displays the complete option text, e.g. "Red Color".

45.07.34 **Select No Name** /select/select_no_name.xhtml

Assertion: When the *select* element is encountered within the contents of a *form* element, the *name* attribute is not set and any of the enclosed options is selected and the form is submitted, the user agent MUST ignore this selection list and the chosen option will not be submitted to the server.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.5.1
[HTML401] Section 17.2, 17.6

Actions:

1. Verify that the page "Select No Name" is loaded.
2. Make your selection and submit the form.
3. Verify that the results page will not show your selection.

45.07.35 **Select No Options** /select/select_no_options.xhtml

Assertion: When a *select* element is encountered within the contents of a *form* element and there is not even a single *option* element included in the content, the user agent MUST report an error.

Note: This assertion applies only to VALIDATING UAs. Non-validating agents MAY simply ignore the select and attempt to render the document, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.3, 17.6

Actions:

1. Verify that the page "No options" is loaded.
2. Activate the "GO" link to load a page, which contains a select element with no options.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the start page.
4. For non-validating UAs, verify that the browser does not crash.

45.07.36 Textarea **/textarea/textarea.xhtml**

Assertion 1: When a *textarea* element is encountered within the contents of a *form* element, the user agent MUST display a rectangular input area into which the user can enter text data.

Assertion 2: When the *textarea* element is encountered within the contents of a *form* element and the *rows* attribute is set as a positive integer, the user agent MUST display that many rows initially in the text area for input.

Assertion 3: When the *textarea* element is encountered within the contents of a *form* element and the *cols* attribute is set as a positive integer, the user agent MUST display that many number of columns initially for text input.

Assertion 4: When more text input is entered into a *textarea* than can be displayed, the user agent MUST provide a mechanism to move through all of the entered data.

Assertion 5: When the *textarea* element is encountered within the contents of a *form* element, the *name* attribute is set, some text is entered as input and the enclosing form is submitted, the user agent MUST associate the value of the *name* attribute with the text entered as input and submit the pair to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.7, 17.2

Actions:

1. Verify that the page "Textarea" is loaded. Verify that two textarea fields of different sizes are displayed.
2. Enter more text than can all be displayed in the textareas at once (multiple lines).
3. Verify that they provide some scrolling mechanism to scroll through the data.
4. Submit the form and verify that the results page displays all of the data entered.

45.07.37 Textarea Default Text1 **/textarea/textarea_default1.xhtml**

Assertion 1: When a *textarea* element is encountered within the contents of a *form* element, the user agent MUST display the enclosed contents as the initial text in the textarea.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.7

Actions:

1. Verify that the page "Textarea-Default Text 1" is loaded.
2. Verify that the "Description" field displays the default text "An interesting book for children and elders alike."
3. Submit the form without modifying the data and verify that this data is displayed on the results page.

45.07.38 Textarea Default Text2 /textarea/textarea_default2.xhtml

Assertion: When the *textarea* element is encountered within the contents of a *form* element, some text is enclosed within the element, some other text is entered as input and the user resets the form, the user agent **MUST** reset the contents of the textarea to the initial contents.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.7

Actions:

1. Verify that the page "Textarea-Default Text 2" is loaded.
2. Change the default text and activate "Reset" button.
3. Verify that the initial value is displayed again.
4. Modify the text again and submit the form.
5. Verify that the text you entered is displayed on the results page.

45.07.39 Textarea No Name /textarea/textarea_no_name.xhtml

Assertion: When the *textarea* element is encountered within the contents of a *form* element, the *name* attribute is not set and some text is entered as input and the form is submitted, the user agent **MUST** ignore this textarea and the value will not be submitted to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.7, 17.13.2

Actions:

1. Verify that the page "Textarea No Name" is loaded.
2. Enter text into both fields and submit the form.
3. Verify that results page displays only the text entered in "Title" field.

45.07.40 Multipart/Form-Data /form_elem/multipart.xhtml

Assertion: When the **form** element is encountered and the **enctype** attribute is set to "multipart/form-data", the user agent MUST use the encoding mechanism specified to transmit the form responses to the server.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.5.1
[HTML401]	Section 17.13.4
[RFC2388]	Entire document

Actions:

1. Verify that the page "Multipart/Form-Data" is loaded.
2. Fill in the field and submit the form by selecting "Send".
3. On the response page, verify that the data was transmitted in multipart/form-data format as follows:
 - The data contains two parts, each preceded by a boundary delimiter line beginning with "--".
 - The data concludes with a closing boundary delimiter line ending with "--".
 - All three delimiters within this document must match.
 - Each part contains a content-type, set to "text/plain", a Charset parameter, and a content-disposition, set to "form-data".
 - The first part must contain Name=UserID, followed by the data that was entered into that field; the second must contain Name=Send, followed by the data "send".
4. Select the "Back" link to return to the start page and post your results.

4.8 Basic Tables

4.8.1 Testing Objective

This module verifies the rendering and functionality of elements and attributes related to tables, such as <td>, <tr>, and <th>.

4.8.2 Test Content

Tests for this module begin with the number 45.8.x. The xbasic/tables_module/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages for these tests. The directory within Test Suite which contains these tests is /xhtml_tables. Starting pages are described below. Each test is described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
45.08.01	Table Structure	table_struct.xhtml

Assertion 1: When the *table* element is encountered, the user agent MUST render a table so that it is set apart from the surrounding content.

Assertion 2: When a *table* element is encountered, the user agent will calculate the number of columns as the number of columns required by the row with most columns, including the cells that span multiple columns and will render the same number of columns for all the rows.

Assertion 3: When a *table* element is encountered, the user agent MUST pad the end of the row which has a lesser number of columns than number of columns calculated by the user agent, with empty cells.

Assertion 4: When a *table* element is encountered, the user agent MUST calculate the number of rows by the number of *tr* elements contained by the *table* element and render the table with that many rows.

Assertion 5: When a *caption* element is encountered within a *table* element, the user agent MUST render its contents as a caption for the entire table.

Assertion 6: When the *td* element is encountered within a table and it has no contents, the user agent MUST display an empty cell.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.1, 11.2.2, 11.2.4, 11.2.6

Actions:

1. Verify that the page "Table Structure" is displayed.
2. Verify that the table is set apart from the surrounding content, has a caption 'My Table', and contains 3 rows.
3. Verify that the browser renders the same number of columns (5) for all rows.
4. Verify that cols 1 and 2 for row 2 are combined.
5. Verify that the UA creates the rows 2 and 3 for col 5 as padding (they should be empty, and may appear combined.)
6. Verify that the first and fourth cols in the third row contain empty content.

45.08.02 Table Padding

table_padding.xhtml

Assertion 1: When a **table** element is encountered, the user agent will calculate the number of columns as the number of columns required by the row with most columns, including the cells that span multiple columns and will render the same number of columns for all the rows.

Assertion 2: When a **table** element is encountered, the user agent MUST pad the end of the row which has a lesser number of columns than number of columns calculated by the user agent, with empty cells.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.1, 11.2.4

Actions:

1. Verify that the page "Table Padding" is loaded. Each row contains a different number of cells: the first contains one cell, the second two, the third three and the fourth four.
2. Verify that the UA renders all cells and that the first cell is padded with enough space for three cells (to the right of its content); the second is padded with two cells; the third with one; and the fourth has no padding (i.e. the table should appear like a staircase).

45.08.03 Table Width Px

table_width_pixels.xhtml

Assertion: When the **width** attribute of the **table** element is specified as a non-negative integer representing a number of pixels, the user agent MUST display a table of width equal to the specified number of pixels.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.1

Action: Verify that the page "Width Integer" is loaded. Verify that the "Table 1" is displayed in a lesser width than "Table 2" .

45.08.04 Table Width %

table_width_percent.xhtml

Assertion: When the **width** attribute of the **table** element is specified as a non-negative integer between 0 and 100 inclusive followed by a percent character, the user agent MUST display a table of width equal to the percentage of the user agent's available horizontal space

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.1

Actions:

1. Verify that the page "Width Percent" is loaded.
2. Verify that "Table 1" is displayed at 50% of the screen width and "Table 2" is displayed at 100% of the screen width.

45.08.05 Long Table

table_long.xhtml

Assertion: When a *table* element is encountered, the user agent MUST avoid clipping any of its contents or MUST provide a means to access all parts by vertical or horizontal scrolling.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.2

Actions:

1. Verify that the page "Long Table" is loaded.
2. Verify that the table contains a caption, three rows, and three columns, and that the content is not clipped.
3. If the table extends beyond the screen area, verify that the user agent provides a mechanism to allow you to view all parts of the table, such as by horizontal or vertical scrolling.

45.08.06 Summary Attribute

summary_attrib.xhtml

Assertion: When a *summary* attribute is encountered within the *table* element, the UA MUST NOT crash and must render the table normally.

Note: This attribute is used to provide a summary about the contents of the table, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.1

Actions:

1. Verify that the page "Summary Attrib" is loaded.
2. Verify that the table contains a caption, two headers, and two cells, and that the page does not crash the browser.
3. Select the "Next" link.
4. Verify that the page "Long Summary" loads without errors. This test uses a very long summary value.
5. Verify that the table contains a caption and four cells.

45.08.07 th td Defaults

th_td_defaults.xhtml

Assertion 1: When the *th* element is encountered within a *table* element, the user agent MUST render the header information in a visually distinct manner from the rest of the text.

Assertion 2: When the *th* element is encountered within the table and the *align* attribute of the *th* element is not set, and the *align* attribute of the enclosing *tr* element is not set, then the user agent MUST horizontally align the cell's contents with the default value, "center".

Assertion 3: When the *th* element is encountered within a table and the *valign* attribute is not set, and the *valign* attribute of the enclosing *tr* element is not set, the user agent MUST vertically align the cell's contents with the default value, "middle".

Assertion 4: When the **td** element is encountered within a table and the **align** attribute of the **td** element is not set, and the **align** attribute of the enclosing **tr** element is not set, then the user agent MUST horizontally align the cell's contents with the default value "left".

Assertion 5: When the **td** element is encountered within a table and the **valign** attribute is not set, and the **valign** attribute of the enclosing **tr** element is not set, the user agent MUST vertically align the cell's contents with the default value, "middle".

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "th td Defaults" is loaded.
2. Verify that the headers "H1", "H2", "H3" are visually distinct from the rest of the text and are center aligned horizontally, and middle aligned vertically.
3. Verify that the content in the "A1", "A2", and "A3" cells is left aligned horizontally and middle aligned vertically. (Note that the fourth column is a spacer column, used only to force the height of all cells to be large enough to be able to verify vertical alignment. Similarly, the third row is a spacer row, used to force the cells to be wide enough to verify horizontal alignment.)

45.08.08 **th td Align** **th_td_align.xhtml**

Assertion 1: When the **th** element is encountered within a table and the **align** attribute is set, the user agent MUST horizontally align the cell's contents according to the value specified.

Assertion 2: When the **td** element is encountered within a table and the **align** attribute is set, the user agent MUST horizontally align the cell's contents according to the value specified.

Note: Legal values are "left" (the contents are aligned against the left side of the cell), "right" (the contents are aligned against the right side of the cell), and "center" (the contents are centered horizontally within the cell).

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "th td Align" is loaded.
2. Verify that the text "H1" is left-aligned, "H2" is center-aligned and "H3" is right-aligned.
3. Verify that in the second row, the text "B1" is left aligned, "B2" is center aligned and "B3" is right aligned. The third row is left aligned by default and functions as a spacer.

45.08.09 **th td vAlign** **th_td_valign.xhtml**

Assertion 1: When the **th** element is encountered within the table and the **valign** attribute is set, the user agent MUST vertically align the cell's contents according to the value specified.

Assertion 2: When the **td** element is encountered within the table and the **valign** attribute is set, the user agent MUST vertically align the cell's contents according to the value specified.

Note: Legal values are "top" (the contents are aligned against the top of the cell), "bottom" (the contents are aligned against the bottom of the cell), and "middle" (the contents are centered vertically within the cell).

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "th td vAlign" is loaded.
2. Verify that the text "H1" is top-aligned, "H2" is middle-aligned and "H3" is bottom-aligned. Verify that in the second row, the text "A1" is top-aligned, "A2" is middle-aligned and "A3" is bottom-aligned. The fourth column functions as a spacer.

45.08.10 **th Rowspan** **th_rowspan.xhtml**

Assertion: When a *th* element is encountered within a table and the *rowspan* attribute is set as a positive integer, the user agent MUST render the contents of the cell spanning that many rows. When the attribute is not specified, the cell occupies a single row.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6

Actions:

1. Verify that the page "th Rowspan" is loaded.
2. Verify that Row 1 and 2 in Column 1 are combined in cell 1 and that there are 8 individual cells.

45.08.11 **td Rowspan** **td_rowspan.xhtml**

Assertion: When a *td* element is encountered within a table and the *rowspan* attribute is set as a non-negative integer, the user agent MUST render the contents of the cell spanning that many rows. When the attribute is not specified, the cell occupies a single row in height.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6

Actions:

1. Verify that the page "td Rowspan" is loaded.
2. Verify that Row 1 and 2 in Column 1 are combined in cell 1 and that there are 8 individual cells.

45.08.12 **th td Colspan** **th_td_colspan.xhtml**

Assertion 1: When the *th* element is encountered within a table and the *colspan* attribute is set as a positive integer, the user agent MUST render the contents of the cell, spanning that many columns. When the attribute is not specified, the contents are rendered in a single cell.

Assertion 2: When the **td** element is encountered within a table and the **colspan** attribute is set as a non-negative integer, the user agent **MUST** render the contents of the cell, spanning that many columns. When the attribute is not specified, the cell occupies a single column in width.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6

Actions:

1. Verify that the page "th td Colspan" is loaded.
2. Verify that the header for columns 1 and 2 spans both columns and column 3 has its own header.
3. Verify that cell containing text "Mary Sue" spans two columns.

45.08.13 **th td Abbr** **th_td_abbr.xhtml**

Assertion 1: When the **th** element is encountered within the table and the **abbr** attribute is set, the user agent **MAY** render its value, as an abbreviated form of the cell's content.

Assertion 2: When the **td** element is encountered within the table and the **abbr** attribute is set, the user agent **MAY** render its value as an abbreviated form of the cell's content

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6

Actions:

1. Verify that the page "th td Abbr" is loaded.
2. Verify that the table is rendered with column headers and that none of table cells are empty.
3. For user agents that support the **abbr** attribute, and use it in place of the cell content when the cell content is large and being rendered on a small screen, the headers may be rendered as "h1", "h2", etc. instead of "This is Column 1", "This is Column 2", etc. Similarly, the second row may be rendered as "a1" instead of". This is A1. The third row does not include the **abbr** attribute.

45.08.14 **th Misc Attribs** **th_misc_attrib.xhtml**

Assertion: When the axis, header, and scope attributes are encountered within the **th** element within a table, the UA **MUST NOT** crash and **MUST** render the table normally.

Note: These attributes are used to provide additional information about the table, but may be ignored by some browsers.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6

Actions:

1. Verify that the page "th Misc Attrib" is loaded without errors.

2. Verify that the table contains a caption ("Stocks:"), three right aligned headers at the top ("Num", "Price", "Total"), three right aligned columns of numbers, and one left aligned vertical header column of stock symbol headers.

45.08.15 **td Misc Attribs** **td_misc_attrib.xhtml**

Assertion: When the axis, header, and scope attributes are encountered within the **td** element within a table, the UA MUST NOT crash and MUST render the table normally.

Note: These attributes are used to provide additional information about the table, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.6

Action:

1. Verify that the page "td Misc Attrib" is loaded without errors.
2. Verify that the table contains a caption ("Stocks:"), three right aligned cells at the top ("Num", "Price", "Total"), three right aligned columns of numbers, and one left aligned leading column of stock symbol headers.

45.08.16 **tr Align 1** **tr_align1.xhtml**

Assertion 1: When the **th** element is encountered within the table and the **align** attribute is not set, and the **align** attribute of the enclosing **tr** element is set, the user agent MUST horizontally align the cell's contents according to the value specified for the **tr** element.

Assertion 2: When the **td** element is encountered within a table and the **align** attribute is not set, and the **align** attribute of the enclosing **tr** element is set, the user agent MUST horizontally align the cell's contents according to the align value specified for the **tr** element.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "tr align 1" is loaded.
2. Verify that the column headers are right aligned.
3. Verify that the entries in next three rows are left, center and right aligned respectively. The last row renders with default alignment.

45.08.17 **tr Align 2** **tr_align2.xhtml**

Assertion 1: When the **th** element is encountered within the table and the **align** attribute is set, and the **align** attribute of the enclosing **tr** element is also set, the user agent MUST horizontally align the cell's contents according to the value specified for the **th** element.

Assertion 2: When the **td** element is encountered within the table and the **align** attribute is set, and the **align** attribute of the enclosing **tr** element is also set, the user agent MUST horizontally align the cell's contents according to the value specified for the **td** element.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "tr align 2" is loaded.
2. Verify that the column headers are center and right aligned respectively, and that the second row (A1, A2) is left and center aligned. The third row functions as a spacer only.

45.08.18 tr vAlign 1 tr_valign1.xhtml

Assertion 1: When the *th* element is encountered within the table and the *valign* attribute is not set, and the *valign* attribute of the enclosing *tr* element is set, the user agent MUST vertically align the cell's contents according to the value specified for the *tr* element.

Assertion 2: When the *td* element is encountered within the table and the *valign* attribute is not set, and the *valign* attribute of the enclosing *tr* element is set, the user agent MUST vertically align the cell's contents according to the value specified for the *tr* element.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "tr vAlign" is loaded.
2. Verify that the column headers for columns 1,2 are bottom aligned.
3. Verify that the entries in the next three rows are top, middle and bottom aligned respectively. Note that the third column functions as a spacer only.

45.08.19 tr vAlign 2 tr_valign2.xhtml

Assertion 1: When the *th* element is encountered within the table and the *valign* attribute is set, and the *valign* attribute of the enclosing *tr* element is also set, the user agent MUST vertically align the cell's contents according to the value specified for the *th* element.

Assertion 2: When the *td* element is encountered within the table and the *valign* attribute is set, and the *valign* attribute of the enclosing *tr* element is also set, the user agent MUST vertically align the cell's contents according to the value specified for the *td* element.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.6.1
[HTML401]	Section 11.2.6, 11.3.2

Actions:

1. Verify that the page "tr vAlign 2" is loaded.
2. Verify that the column headers are bottom and middle aligned, and that the second row is top and middle aligned. The third column functions as a spacer only.

45.08.20 Illegal Spans illegal_spans.xhtml

Assertion: When a table contains rowspan and colspan attributes that are defined to be larger than the number of rows or columns in the table which contains them, the UA MUST NOT crash.

Note: The behavior for UA's which encounter a rowspan or colspan that is too large is undefined.

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.6, 11.3.2

Action:

1. Verify that the page "Illegal Spans" loads without crashing the browser. The table may not be displayed.

45.08.21 **Illegal Width px** **illegal_width_pixels.xhtml**

Assertion: When a negative number is given as the value of the attribute **width** of the table element, the attribute SHOULD be ignored.

Note: The behavior for UA's which encounter a negative **width** value is undefined. Presumably UAs would attempt to render the table, perhaps using the default width.

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.1

Action:

1. Verify that the page "Illegal Width px" loads without crashing the browser. The table may be displayed or ignored.

45.08.22 **Illegal Width Percent** **illegal_width_percent.xhtml**

Assertion: When a number greater than 100% is given as the value of the attribute **width** of the table element, the attribute SHOULD be ignored.

Note: The behavior for UA's which encounter a **width** value that is greater than 100% is undefined. Presumably UAs would attempt to render the table, perhaps using the default width.

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2.1

Actions:

1. Verify that the page "Illegal Width Percent" loads.
2. If a table is displayed, verify it contains six cells, and that all cell content is visible.

45.08.23 **Malformed Table1** **malformed_table1.xhtml**

Assertion: When a table contains malformed content, the user agent MAY ignore the table, but the page MUST NOT crash the browser.

Note: The behavior for non-validating UA's which encounter a malformed table is undefined. UAs may ignore the table or attempt to render as much of it as possible.

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2

Actions:

1. Verify that the page "Malformed Table1" loads without crashing.
2. The tables will probably not be displayed. Verify that the browser does not crash.

45.08.24 **Malformed Table 2** **malformed_table2.xhtml**

Assertion: When a table contains malformed content, the user agent MAY ignore the table, but the page MUST NOT crash the browser.

Note: See the note for "Malformed Table 1".

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2

Action:

1. Verify that the page "Malformed Table2" loads without crashing. The table may or may not be displayed.
2. Verify that the browser does not crash.

45.08.25 **Multiple Tables** **multiple_tables.xhtml**

Assertion: When a UA encounters a page with multiple tables, the UA MUST render each table without crashing the browser.

References: (*inferred*)

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.6.1
[HTML401] Section 11.2

Action:

1. Verify that the page "Multiple Tables" loads without crashing.
2. Verify that four short tables are displayed: the first should contain letters, the second numbers, the third character entities, and the fourth numeric entities.

45.08.26 **Embedded Tables** **embedded_tables.xhtml**

Assertion: When the UA encounters a table which contains tables within its cells, the UA MUST render the embedded tables as well as the table which contains them.

References: (*inferred*)

[XHTMLBasic] Section 3

[XHTMLMOD] Section 5.6.1

Action:

1. Verify that the page "Embedded Tables" loads without crashing.
2. Verify that there is a table with the caption "Table One:" which contains two embedded tables, "Tbl Two:" and "Tbl Three:".
3. Verify that the first row of "Table One" contains two cells, "A", and "B".
4. Verify that the second row, first column, of "Table One" contains "Tbl Two" and the second row second column contains "Tbl Three".
5. Verify that "Tbl Two" contains two cells, "1" and "2", in its first row, and the cells "3" and "4" in its second row.
6. Verify that "Tbl Three" contains two cells, "5" and "6", in its first row, and the cells "7" and "8" in its second row.
7. Verify that the third row of "Table One" contains two cells, "C", and "D".
8. Verify that the following elements of "Table One" are left aligned: caption, "A", "B", "C", "D".
9. Verify that the following elements of the embedded tables are right aligned: both captions, all cells "1" - "8".

4.9 Image Module

4.9.1 Testing Objective

This module verifies the rendering and functionality of elements and attributes related to images, such as <image> and src.

4.9.2 Test Content

Tests for this module begin with the number 45.9.x. The xbasic/image_module/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages for these tests. The directory within Test Suite which contains these tests is /xhtml_image. Starting pages are described below. Each test is described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.09.01	Image Element	image_element.xhtml
----------	---------------	---------------------

Assertion: When an *img* element is encountered and the *src* attribute is set, the user agent MUST load and display the image specified by the value of the *src* attribute, inline with the surrounding text of the current document.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.2

Action:

1. Verify that the page "Image Element" is loaded, and that the image is displayed inline with the surrounding text.

45.09.02	Image as Link	image_as_link.xhtml
----------	---------------	---------------------

Assertion 1: When an *img* element is encountered within the content of an *a* element, the user agent MUST render the image inline and visit the resource specified by the destination anchor of the link, when the link associated with the image is activated.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.1

Actions:

1. Verify that the page "Image as Link" is loaded.
2. Select the image to visit the link. Verify that the link takes you to the page "Image Link Works".

45.09.03	Image with Content	image_content.xhtml
----------	--------------------	---------------------

Assertion: When the *img* element is encountered and some content is enclosed within the element, the user agent MUST report an error.

Note: This assertion is applicable for VALIDATING UAs only. Non-validating agents MAY ignore the content within the tags, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.7
[HTML401] Section 13.2

Actions:

1. Verify that the page "Image Content" is loaded.
2. Selected the "Go" link.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the start page.
4. For non-validating UAs, verify that the browser does not crash.

45.09.04 **Image Corrupt** **image_corrupt.xhtml**

Assertion: When the *img* element is encountered and the image file pointed to by the src attribute is corrupt, the UA MUST display the alt text instead of the image.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.7
[HTML401] Section 13.2

Action:

1. Verify that the page "Image Corrupt" loads without crashing the browser and that the UA displays the text "corrupt cart image".

45.09.05 **Alt Text** **image_alt.xhtml**

Assertion: When the *img* element is encountered and the *alt* attribute is set as a string and the image cannot be loaded, the user agent MUST display that string as an alternate text.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.7
[HTML401] Section 13.2, 13.8

Actions:

1. Verify that the page "Image Alt Text" is loaded, and the alt text, "time", for the image is displayed.

45.09.06 **Missing Alt** **missing_alt.xhtml**

Assertion: When the *img* element is encountered and the *alt* attribute has not been set, the user agent MUST report an error.

Note: This assertion is applicable to VALIDATING UAs only. Non-validating agents MAY attempt to render the document without the *alt*, but MUST NOT crash when loading the test page.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.2, 13.8

Action:

1. Verify that the page "Missing Alt" is loaded.
2. Select the "Go" link. For validating UAs, verify that the user agent reports an error and allows the user to return to the start page.
3. For non-validating UAs, verify that the browser does not crash.

45.09.07 Missing Src **missing_src.xhtml**

Assertion: When the *img* element is encountered and the *src* attribute has not been set, the user agent **MUST** report an error.

Note: This assertion is applicable to VALIDATING UAs only. Non-validating agents **MAY** attempt to render the document with a "broken image" icon in place of the image, but **MUST NOT** crash when loading the test page.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.2

Action:

1. Verify that the page "Missing Src" is loaded.
2. Select the "Go" link.
3. For validating UAs, verify that the user agent reports an error and allows the user to return to the start page.
4. For non-validating UAs, verify that the browser does not crash.

45.09.08 Height-Integer **ht_integer.xhtml**

Assertion 1: When the *img* element is encountered and the *height* attribute is set as a positive integer, the user agent **MUST** render the image with its natural height overridden and scaled to the value specified.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.2, 13.7.1

Actions:

1. Verify that the page "Height-Integer" is loaded. The first image uses its natural height. The second one is increased to a 96px height.
2. Verify that the second one appears longer in the vertical dimension than the first one does.

45.09.09 Height-Percent **ht_percent.xhtml**

Assertion: When the *img* element is encountered and the *height* attribute is set as a percentage value of the available screen height, the user agent MUST render the image with its natural height overridden and scaled to the value specified.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.7
[HTML401] Section 13.2, 13.7.1

Actions:

1. Verify that the page "Height-Percent" is loaded. The first image uses its natural height. The second one is set to 50% of the screen or window height.
2. Verify that second image is scaled down to about 50% of the height of the first image.

45.09.10 **Width-Integer** **width_integer.xhtml**

Assertion: When the *img* element is encountered and the *width* attribute is set as a positive integer, the user agent MUST render the image with its natural width overridden and scaled to the value specified.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.7
[HTML401] Section 13.2, 13.7.1

Actions:

1. Verify that the page "Width-Integer" is loaded. The first image uses its natural width. The second image is increased to 96px width.
2. Verify that the second image appears wider than the first image does.

45.09.11 **Width-Percent** **width_percent.xhtml**

Assertion: When the *img* element is encountered and the *width* attribute is set as a percentage value of the available screen width, the user agents MUST render the image with its natural width overridden and scaled to the value specified.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.7
[HTML401] Section 13.2, 13.7.1

Actions:

1. Verify that the page "Width-Percent" is loaded. The first image uses its natural width. The second one is set to 50% of the screen or window width.
2. Verify that second image is scaled down to about 50% of the width of first image.

45.09.12 **Animated Image** **img_w_animated_gif.xhtml**

Assertion: When an *img* element is encountered and the *src* attribute is set to show an animated gif, the user agent MUST load and display the animated gif image specified by the *src* attribute, inline with the surrounding content.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.2

Actions:

1. Verify that the page "Animated Image" is loaded. This page contains an animated gif of the word "OK" being drawn in black, as if with a paint brush.
2. Verify that the gif is rendered correctly and that it is displayed inline with the rest of the text (i.e. there should not be an extra line break or block-style break before or after the image).

45.09.13 Image Longdesc image_longdesc.xhtml

Assertion: When the *longdesc* attribute is encountered within the *image* element, the UA MUST NOT crash and must render the image normally.

Note: This attribute is used to provide a link to a text file which describes the image, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.7
[HTML401]	Section 13.2

Action:

1. Verify that the page "Image Longdesc" loads without crashing and that the image of a globe is displayed.

4.10 Link Module

4.10.1 Testing Objective

Most elements of the link module are UNSUPPORTED, with the exception of those required to link to an external style sheet.

4.10.2 Test Content

Tests for this module begin with the number 45.10.x. The xbasic/link_module/default.xhtml page serves as a menu to the local versions of the tests, but currently links to only one test, "Linking Stylesheets", which is described below. The directory within Test Suite which contains these tests is /xhtml_link. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
45.10.01	Linking Stylesheets	link_css.xhtml

Assertion: When the *link* element is encountered and the *rel* attribute is set as "Stylesheet", the user agent MUST render the current document according to the stylesheet information available in the web resource specified as the value of the *href* attribute.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.19
[HTML401]	Section 12.3, 6.12

Actions:

1. Verify that the page "Linking Stylesheets" is loaded. The style information described in the external style sheet will be applied to the paragraph beginning "This text should appear..."
2. On color user agents this will appear in red color and italicized; on black and white UAs it will appear italicized only.
3. Verify that the style information is applied correctly.

4.11 Base Module

4.11.1 Testing Objective

This module tests relative links as affected by the <base> element.

4.11.2 Test Content

Tests for this module begin with the number 45.11.x. The xbasic/base_module/default.xhtml page serves as a menu to the local versions of the tests, and contains links to all starting pages, described below. The directory within Test Suite which contains these tests is /xhtml_base. Starting pages are described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.11.01	No Base	no_base.xhtml
----------	----------------	----------------------

Assertion: When the *base* element is not present, the user agent MUST use the document's URL as the base URL to dereference relative URLs within the current document.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.20
[HTML401]	Section 12.4.1

Action:

1. Verify that the page "No Base" is loaded.
2. Select the "Go" link to load a page without a base tag.
3. Verify that the "No Base 2" page loaded, and that the "Back" link takes you back to the starting page.

45.11.02	Abs Base	abs_base.xhtml
----------	-----------------	-----------------------

Assertion: When the *base* element is encountered within the head section of the document and the *href* attribute is set with an absolute URL as the value, the user agent MUST use that URL instead of the base URL.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.20
[HTML401]	Section 12.4

Actions:

1. Verify that the page "Abs Base" is loaded.
2. Select the "Start" link to load a page with a base tag. Verify that the page "Abs Base Start" loaded.
3. Select the "Go" link. Verify that the page "Abs Base 2" loaded.
4. Verify that selecting the "Back" link takes you back to "Abs Base Start".
5. Verify that selecting "Back" from this page returns you to the starting page.

45.11.03	Base With Content	content_base.xhtml
----------	--------------------------	---------------------------

Assertion: When the **base** element is encountered in the head section of the document and the **href** attribute is set and some content is enclosed within the start and the end tags of the element, the user agent **MUST** report an error.

Note: This assertion applies to **VALIDATING** UAs only. Non-validating UAs **MAY** simply ignore the content, but **MUST** load the page correctly.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.20
[HTML401] Section 12.4

Actions:

1. Verify that the page "Base With Content" is loaded.
2. Select the "Go" link to load a page with content within the base tag.
3. For validating UAs, verify that the UA reports a document error and allows you to return to the previous page.
4. Non-validating UAs may load the page; if the page loads, the "Back" link must work correctly.

45.11.04 **Base in Body** **body_base.xhtml**

Assertion: When the **base** element is encountered in the body section of a document, the user agent **MUST** report an error.

Note: This assertion applies to **VALIDATING** UAs only. Non-validating UAs may interpret the base element anyway, and **MUST** load the page correctly.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.20
[HTML401] Section 12.4

Actions:

1. Verify that the page "Base in Body" is loaded.
2. Select "Go" to load a page with the base tag within the body.
3. For validating UAs, verify that the UA reports a document error and allows you to return to the previous page.
4. Non-validating UAs may load the page; if the page loads, the "Back" link must work correctly.

45.11.05 **No HREF** **no_href.xhtml**

Assertion: When the **base** element is encountered and the **href** attribute has not been set, the user agent **MUST** report an error.

Note: This assertion applies to **VALIDATING** UAs only. Non-validating UAs **MAY** simply ignore the base element, and **MUST** load the page correctly.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.20

[HTML401] Section 12.4

Actions:

1. Verify that the page "No HREF in Base" is loaded.
2. Select the "Go" link to load a page with an invalid BASE tag.
3. For validating UAs, verify that the UA reports a document error and allows you to return to the previous page.
4. Non-validating UAs may load the page; if the page loads, the "Back" link must work correctly.

45.11.06 **Rel Base** **rel_base.xhtml**

Assertion: When the **base** element is encountered within the head section of the document and the **href** attribute is set with a relative URL as the value, the user agent **MUST** use the URL as the base URL for resolving URLs within the current document.

References:

[XHTMLBasic]Section 3
[XHTMLMOD]Section 5.20
[HTML401] Section 12.4

Action:

1. Verify that the page "Rel Base" is loaded.
2. Select the "Start" link to load a page with a base tag. Verify that the page "Rel Base Start" loaded.
3. Select the "Go" link. Verify that the page "Rel Base 2" loaded.
4. Verify that selecting the "Back" link takes you back to "Rel Base Start".
5. Verify that selecting "Back" from this page returns you to the starting page.

4.12 Common Attributes Module

4.12.1 Testing Objective

This module verifies the rendering and functionality of attributes which are common to many elements, such as `accesskey` and `tab`.

4.12.2 Test Content

Tests for this module begin with the number 45.12.x. The `xbasic/common_attr_module/default.xhtml` page serves as a menu for the local versions of the tests, and contains links to the starting pages for these tests. The directory within Test Suite which contains these tests is `/xhtml_common_attr`. Starting pages are described below. Each test is described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
---------------	------------------	----------------------

45.12.01	Accesskey Test 1	accesskey1.xhtml
----------	-------------------------	-------------------------

Assertion: When the **accesskey** attribute is set for a supporting element, the user agent MUST assign an access key to the element and when the accesskey is activated, it will give focus to that element.

Note: The Supporting Elements checked in this test are **INPUT** (text input, password, submit and reset) and **LABEL**.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 17.11.2

Actions:

1. Verify that the page "Accesskey Test 1" is loaded.
2. Verify that using the accesskeys 1 - 3 will move the focus to the designated input field. Verify that accesskey "5" will reset the form and "4" will submit it.

45.12.02	Accesskey Test 2	accesskey2.xhtml
----------	-------------------------	-------------------------

Assertion: When the **accesskey** attribute is set for a supporting element, the user agent MUST assign an access key to the element and when the accesskey is activated, it will give focus to that element.

Note: The Supporting Elements checked in this test are **A**, **TEXTAREA**, and **INPUT** (radio buttons and checkboxes).

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 17.11.2

Action:

1. Verify that the page "Accesskey Test 2" is loaded.
2. Verify that using the accesskeys 1 - 5 will move the focus to the designated control. Verify that accesskey "6" will activate the Post Results link.

45.12.03 Unsupported Accesskeys

unsupported_access.xhtml

Assertion: When non-numeric accesskeys, including letters, asterisk, and pound, are activated, the UA MUST ignore these accesskeys.

Note: While the XHTML-B and -MP specs support these keys, some UAs may not, due to the fact that some of these keys may be assigned to other functions.

References: (*inferred*)

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.3
[HTML401]	Section 17.11.2

Actions:

1. Verify that the page "Unsupported Accesskeys" is loaded.
2. Verify that using the accesskeys "A", "B", "*" and "#" causes nothing to happen, i.e. the corresponding fields DO NOT receive focus and are not activated.

45.12.04 Tabbing Order

tab_order.xhtml

Assertion : When the *tabindex* attribute is set for a supporting element, with an integer value between 0 to 32767, the page MUST load without crashing the browser.

Note: Tabbing may not be supported by all browsers. In UAs with a tab key, the tab key is used to move focus to a series of elements according to the increasing numeric order of the value - where numerically greater numbers indicate a later position than numerically lesser numbers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.10
[HTML401]	Section 17.11.1

Actions:

1. Verify that the page "Tabbing Order" loads without error, and that the page contains a form with "First Name" set to the value "Joe", "Last Name" to "Turner", and "Password" to "*****", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script (Password: rose).

45.12.05 Equal Tabindex

tab_equal.xhtml

Assertion: When the *tabindex* attribute is set for more than one supporting element, and the value of this attribute for these elements is the same, the page MUST load without crashing the browser.

Note: Tabbing may not be supported by all browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.10
[HTML401]	Section 17.11.1

Action:

1. Verify that the page "Equal Tabindex" is loaded, and that the page contains a form with "First Name" set to the value "Joe", "Last Name" to "Turner", and "Password" to "*****", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script (Password: rose).

45.12.06 Negative Tabindex `tab_negnum.xhtml`

Assertion: When the *tabindex* attribute is set for a supporting element with a numeric value less than 0, the page MUST load without crashing the browser.

Note: Tabbing may not be supported by all browsers. Submit the form.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.10
[HTML401]	Section 17.11.1

Action:

1. Verify that the page "Negative Tabindex" is loaded, and that the page contains a form with "First Name" set to the value "Joe", "Last Name" to "Turner", and "Password" to "*****", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script (Password: rose).

45.12.07 Too High Tabindex `tab_toohigh.xhtml`

Assertion: When the *tabindex* attribute is set for a supporting element with a numeric value greater than 32767, the page MUST load without crashing the browser.

Note: Tabbing may not be supported by all browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.10
[HTML401]	Section 17.11.1

Actions:

1. Verify that the page "Too High Tabindex" is loaded, and that the page contains a form with "First Name" set to the value "Joe", "Last Name" to "Turner", and "Password" to "*****", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script (Password: rose).

45.12.08 Non-numeric Tabindex `tab_nonnum.xhtml`

Assertion: When the *tabindex* attribute is set for a supporting element with a non-numeric value, the page MUST load without crashing the browser.

Note: Tabbing may not be supported by all browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.10

[HTML401] Section 17.11.1

Actions:

1. Verify that the page "Non-numeric Tabindex" is loaded, and that the page contains a form with "First Name" set to the value "Joe", "Last Name" to "Turner", and "Password" to "*****", and that the form contains a submit and clear button.
2. Submit the form and verify that the field values are passed to the result script (Password: rose).

45.12.09 **Title Input** **title_attrib.xhtml**

Assertion 1: When a *title* attribute is defined for an *input* element, and the element receives focus, and the user attempts to enter text, the value of the title MAY be used as a header in that text input dialogue.

Note: The title attribute may not be supported by all UAs.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.1
[HTML401] Section 7.4.3

Actions:

1. Verify that the page "Title Input" is loaded without error, and that the "First Name", "Last Name", "Password", "Login", and "Reset" fields are rendered normally.
2. Enter the text input dialogue for each text field by selecting the field.
3. Enter text into each field. Verify that the header of the dialogue for each field is as follows: for "First Name", the title should be "client's first name"; for "Last Name", "client's last name"; for "Password", "intranet password".
4. Submit the form and verify that the response page reports the text you entered correctly.

45.12.10 **Title Select** **title_select.xhtml**

Assertion: When a *title* attribute is defined for a *select* element, and the element receives focus, and the user attempts to make a selection, the value of the title MAY be used as a header in that selection dialogue.

Note: The title attribute may not be supported by all UAs.

References:

[XHTMLBasic] Section 3
[XHTMLMOD] Section 5.1
[HTML401] Section 7.4.3

Actions:

1. Verify that the page "Title Select" is loaded without error.
2. Select the "location" to bring up the selection dialogue.
3. Verify that the title "OMA Campus" is rendered as the title in that dialogue.
4. Make a selection and submit the form. Verify that the response page reports your selection correctly.

45.12.11 Title Textarea `title_textarea.xhtml`

Assertion 1: When a *title* attribute is defined for a *textarea* element, and the element receives focus, and the user attempts to enter text, the value of the title MAY be used as a header in that text input dialogue.

Note: The title attribute may not be supported by all UAs.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.1
[HTML401]	Section 7.4.3

Actions:

1. Verify that the page "Title Textarea" is loaded without error.
2. Select the "OMA Location" field to bring up the text input dialogue.
3. Verify that the title "OMA Location" is rendered as the title in that dialogue.
4. Enter your text and submit the form. Verify that the response page reports your input correctly.

45.12.12 Title Misc `title_misc.xhtml`

Assertion: When a *title* attribute is defined for an element, the user agent MAY use the attribute to provide additional information about the element, such as in a status bar or help balloon.

Note: The title attribute may not be supported by all UAs.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.1
[HTML401]	Section 7.4.3

Action:

1. Verify that the page "Title Misc" is loaded without error. This page contains numerous title attributes within various elements, but these will probably be ignored.

45.12.13 ID Attribute `id_attrib.xhtml`

Assertion 1: When the *id* attribute is used within an element, and a style is defined which references that *id*, and the element supports styles, the user agent MUST apply the style properties defined by the stylesheet to the element's contents.

Assertion 2: When an element that is not allowed to have an id attribute has one, the browser MUST NOT crash.

Note: While the Stylesheet Module and Style Attribute Module are not included in XHTML Basic, some UAs may support styles.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.2.2
[HTML401]	Section 7.5.2

Actions:

1. Verify that the page "id Attribute" is loaded.
2. Verify that only the text "Boston" appears italicized.
3. For color UAs verify that the word also appears in red text.
4. Select "Next" to load a page with an illegal id attribute. For validating UAs, verify that an error is reported (you will have to go back to the start page to post your result). For non-validating UAs, verify that the page "ID in Title" loads.
5. Select the "Link to Title" link and verify that the browser does not crash.

45.12.14 **Class Attribute** `class_attrib.xhtml`

Assertion 1: When the *class* attribute is used within an element, and the name of the class corresponds to a defined stylesheet class, and the element supports styles, the user agent MUST apply the style properties defined by that class to the element's contents.

Assertion 2: When an element that is not allowed to have a class or style attribute has one, the browser MUST NOT crash.

Note: While the Stylesheet Module and Style Attribute Module are not included in XHTML Basic, some UAs may support styles.

References:

[XHTMLMOD]	Section 5.18
[HTML401]	Section 7.5.2

Actions:

1. Verify that the page "Class Attribute" is loaded.
2. Verify that the test paragraph appears with large text.
3. For color UAs verify that the paragraph also appears in red text.
4. Select "Next" to load a page with an illegal class and style attribute. For validating UAs, verify that an error is reported (you will have to go back to the start page to post your result). For non-validating UAs, verify that the page "Illegal Class Style" loads and does not crash the browser.

45.12.15 **Xml:Lang** `xml_lang.xhtml`

Assertion: The browser MUST NOT crash when the *xml:lang* attribute is encountered within an element.

Note: This attribute is used to provide a hint about the language of the content within the page, but may be ignored by some browsers.

References:

[XHTMLBasic]	Section 3
[XHTMLMOD]	Section 5.1
[XHTML10]	Section C.7

Actions:

1. Verify that the page "Xml:Lang" loads without error. This page contains a title element, which is using the "lang" attribute of the "xml" namespace.

45.12.16 Char Ref in ID

char_ref_id.xhtml

Assertion: An *id* attribute MUST NOT contain character references.

Note: The behavior for this assertion is undefined. Presumably UAs will simply ignore the id and render the page.

References:

[HTML401] Section 12.2.3

Action:

1. Verify that the page "Char Ref in ID" loads without crashing.
2. Verify that the test paragraph does not have any style properties applied to it, as the id reference is illegal.

4.13 XHTML Mobile Profile

4.13.1 Testing Objective

This module tests additional functionality included in the XHTML Mobile Profile specification which is not part of XHTML Basic. This includes a few elements/attributes from the Forms Module (fieldset, optgroup), the Legacy Module (start attribute on OL, value attribute on LI), the Presentation Module (b, big, hr, i, small), the Style Sheet, and Style Attribute Modules. (Note that all Mobile Profile tests use the WapForum XHTML Mobile 1.0 DTD, rather than the standard XHTML Basic DTD)

4.13.2 Test Content

Tests for this module begin with the number 45.13.x. The xbasic/xhtmll_mp/default.xhtml page serves as a menu for the local versions of the tests, and contains links to the starting pages for these tests. The directory within Test Suite which contains these tests is /xhtml_mp. Starting pages are described below. Each test is described below.

<u>Test #</u>	<u>Test Name</u>	<u>Starting Page</u>
45.13.01	Content Type	content_wap_xhtml.xhtml

Assertion: The User Agent MUST accept documents with the content type of "application/vnd.wap.xhtml+xml" and render them normally.

References:
[XHTMLMP] Section 7.2

Action:

1. Verify that the page titled "Content-type: application/vnd.wap.xhtml+xml" is loaded.
2. Select the GO link to load a page with that content type. Verify that that page loads without errors and that the content header is not displayed at the top of the page.

45.13.02	B Element	b_element.xhtml
----------	-----------	-----------------

Assertion: When encountering the *b* element, the user agent MUST render its text content in boldface.

References:
[XHTMLMP] Section 5
[XHTMLMOD] Section 5.4.1
[HTML401] Section 15.2.1

Action:

1. Verify that the page titled "B Element" is loaded and that the words "The New England Patriots" appears in boldface.

45.13.03	I Element	i_element.xhtml
----------	-----------	-----------------

Assertion: When encountering the *i* element, the user agent MUST render its text content in italics.

References:
[XHTMLMP] Section 5
[XHTMLMOD] Section 5.4.1

[HTML401] Section 15.2.1

Action:

1. Verify that the page titled "I Element" is loaded and that the words "The New England Patriots" appears in italics.

45.13.04 **Big Element** **big_element.xhtml**

Assertion: When encountering the *big* element, the user agent MUST render its text content in a larger font size than the surrounding text.

References:

[XHTMLMP] Section 5
[XHTMLMOD] Section 5.4.1
[HTML401] Section 15.2.1

Action:

1. Verify that the page titled "Big Element" is loaded and that the words "The Boston Red Sox" are rendered in a larger font than the rest of the paragraph.

45.13.05 **Small Element** **small_element.xhtml**

Assertion: When encountering the *small* element, the user agent MUST render its text content in a smaller font size than the surrounding text.

References:

[XHTMLMP] Section 5
[XHTMLMOD] Section 5.4.1
[HTML401] Section 15.2.1

Action:

1. Verify that the page titled "Small Element" is loaded and that the words "The Boston Red Sox" are rendered in a smaller font than the rest of the paragraph.

45.13.06 **Fieldset** **fieldset.xhtml**

Assertion: When encountering the *fieldset* element which contains form controls within its content, the user agent MAY visually group the enclosed set of controls in a visually distinct way.

References:

[XHTMLMP] Section 5
[XHTMLMOD] Section 5.5.2
[HTML401] Section 17.10

Action:

1. Verify that the page titled "Fieldset" is loaded without errors. This page contains two groups of fields, "Billing Address" fields and "Shipping Address" fields, which may be visually grouped in two distinct ways (i.e. they might be enclosed in two separate boxes or blocks).

45.13.07 **Option Group** **option_group.xhtml**

Assertion 1: When the *optgroup* element is encountered within the content of a *form* element, the user agent MUST logically group and display all *option* elements within the *optgroup* element.

Assertion 2: When the *optgroup* element is encountered and the *disabled* attribute is set, the user agent MUST NOT allow the control to receive focus.

References:

[XHTMLMP]	Section 5
[XHTMLMOD]	Section 5.5.2
[HTML401]	Section 17.6, 17.6.1, 17.12.1

Action:

1. Verify that the page titled "Option Group" is loaded without errors. This page contains two selection lists, where the first selection list is a single selection list (allows only 1 selection) and the second is a multiselection list (allows 1 or more selections).
2. Open the first selection list then verify that the selection list contains two option groups, "Standard" and "Custom".
3. Verify that the "Standard" group contains the options "Red", "Blue", "Green", and "Yellow", and that the options are grouped together visually (i.e. they should be distinct from the second group.)
4. Verify that the "Custom" group contains the options "Lime", "Aqua", "Peach" and "Slate", and that the options are grouped together visually.
5. Verify that the "Standard" set of options only is disabled, i.e. one cannot make a selection from that group.

45.13.08 **Horizontal Rule** `horiz_rule.xhtml`

Assertion 1: When the *hr* element is encountered, the user agent MUST render a horizontal line on a new line.

Assertion 2: When the *hr* element is encountered, and the element contains an *align* attribute, the user agent MUST align the line to the left, right, or center, depending on the value of the attribute.

Assertion 3: When the *hr* element is encountered, and the element contains a *width* attribute with a value in pixels, the user agent MUST render the line to be that number of pixels wide.

Assertion 4: When the *hr* element is encountered, and the element contains a *width* attribute with a value as percentage, the user agent MUST render the line at a width which equals that percentage of the screen width.

Assertion 5: When the *hr* element is encountered, and the element contains a *size* attribute with an integer value, the user agent MUST render the line with a height equal to the number of pixels specified.

Assertion 6: When the *hr* element is encountered, and the element contains a *noshade* attribute, the user agent MUST render the line without shading.

References:

[XHTMLMP] Section 5
[XHTMLMOD] Section 5.4.1
[HTML401] Section 15.3

Action:

1. Verify that the page titled "Horizontal Rule" is loaded, and that the alignment, width, size, and shading of the rendered lines correspond to the descriptions in the test case.

45.13.09 **LI-Value**

li_value.xhtml

Assertion 1: When encountering the *ol* element, the *value* attribute, when set to an integer value, n, MUST reset the current value in the labeling scheme to the nth value in the labeling sequence. (For example, the third item in a capital letter series (e.g. A, B, C, D) can be reset to "G" with a value of "7".)

Assertion 2: Once reset, the values MUST continue to sequence from the new value forward.

Assertion 3: The *value* attribute MUST override the starting value set by *ol*.

References:

[XHTMLMP] Section 5
[XHTMLMOD] Section 5.22
HTML 4.011 Section 10.1

Action:

1. Verify that the first list item bullet is set to "Z", the second is "AA", third is "Y", fourth is "Z".

45.13.10 **OL - Start**

ol_start.xhtml

Assertion 1: When the *start* attribute of the *ol* element is encountered, the user agent must start the ordered list from the number specified.

References:

[XHTMLMP] Section 5
[XHTMLMOD] Section 5.4.1
HTML 4.011 Section 15.2.1

Action:

1. Verify that the page titled "OL-Start" is loaded, and that the first bullet item is labeled as "20", the second is "21", the third is "99", and the fourth is "100".

45.13.11 **Style**

style.xhtml

Assertion 1: When the **style** element is encountered which associates a class with style properties, the class defined MUST be available for use by elements within the document that support styles.

Assertion 2: When the **class** attribute is used within an element, and the name of the class corresponds to a defined stylesheet class, and the element supports styles, the user agent MUST apply the style properties defined by that class to the element's contents.

Assertion 3: When the **style** attribute is used within an element that supports it, the properties contained within the **style** must be applied to the content of that element..

References:

[XHTMLMP]	Section 8.1.2, 8.1.3
[XHTMLMOD]	Section 5.5.2
[HTML401]	Section 14.2.2, 14.2.3

Action:

1. Verify that the page titled "Style" is loaded.
2. Verify that the text in the test paragraph appears in red (on color UAs), with center alignment, and that the word "OMA" is rendered at a larger font size than the rest of the text in the paragraph.