MOVING TOWARD XHTML 2

John Cowan
Associated Press

Copyright

Roadmap

- Introduction (10 slides)
- Goals and Principles (8 slides)
- New and Nifty (8 slides)
- What's History (7 slides)
- The Elements (27 slides)

Roadmap

- The Common Attributes (19 slides)
- XML Events (11 slides)
- XForms (19 slides)
- To Be Continued... (1 slide)

INTRODUCTION

Versions of HTML

- HTML 1.0 – simple hypertext
- HTML 2.0 – images and forms
- HTML 3.2 – the union of the browsers
- HTML 4.01 – the kitchen sink
- XHTML 1.0 – it's XML time
- XHTML 1.1 – modularization
- XHTML 2 – simple and *powerful* hypertext

What is XHTML 2?

- An evolving standard currently being worked on by the HTML Working Group of the W3C.
- The first major change to the semantics of HTML since HTML 4.0.
  - Obsolete syntax has been discarded
  - Separation of content and presentation essentially complete
  - Genuinely new features being introduced

XHTML 2 Abstract

XHTML 2 is a general purpose markup language designed for representing documents for a wide range of purposes across the World Wide Web. To this end it does not attempt to be all things to all people, supplying every possible markup idiom, but to supply a generally useful set of elements.

Why design XHTML 2?

- More usable
- More accessible to different kinds of users and display devices
- Easier to write by hand or with tools
- Less reliant on embedded scripting languages for commonly used functionality

Why plan for XHTML 2?

- Predicting the degree of market uptake not yet possible
- Support for legacy HTML will probably be provided forever for practical reasons
- Avoiding most deprecated and obsolete features is already possible
- The planner's rule of thumb: “Plan or be planned for”

Making early adoption easy

- Much HTML is already generated using XSLT from one of many XML formats
- XHTML 2 can be generated just as easily
- Early-adopting clients can get a better browsing experience

Who supports XHTML 2?

- Too soon for formal commitments
- Server-side support is straightforward
- Cross-platform open-source browsers will almost certainly support XHTML 2 fairly quickly and easily
- The WG contains several major browser companies

Constructing document types

- XHTML 2 has a fully modular definition
- Defining subsets for use within other document types will be easy, for:
  - rich textual annotations
  - embedded presentation-ready material

Everything is subject to change

- This presentation is based on the July 2006 public working draft of XHTML 2
- Disclaimer: not every slide is 100% updated to the very latest draft
- There are still open issues, which are rather poorly documented at present

Working Group(s)

- XHTML 2 has been developed by the W3C's HTML WG
- I am not on the WG, so I have limited inside information
- XHTML 2 is being handed off to a new WG soon, according to the latest reports

GOALS AND PRINCIPLES

Goals: ease of use

- Generic XML when possible
- Less presentation, more structure
- Make the language easy to write
- Make the resulting documents easy to use
- Be as inclusive as possible (“design for our future selves”)

Goals: universal access

- Better internationalization for the *World Wide* Web
- Device independence:
  - Author once, not once per device
  - Render differently on different devices

Goal: reduced use of scripting

- Writing scripts is programming, not document authoring
- Only high-powered user agents (lots of memory and CPU speed) can process scripts
- XHTML 2 tries to make declarative, not procedural, pages straightforward

Breaking backward compatibility

- Earlier versions of HTML were special-purpose languages
- Backward compatibility required so that new documents would still be usable in older browsers
- Modern browsers can render arbitrary XML with stylesheets

Breaking backward compatibility

- Much of XHTML 2 already works in existing browsers
- XML Events and XForms still require user agents that understand that functionality
- Clever use of Javascript under the table can help here

Modular design

- XHTML 2 is divided into modules which together define the elements, attributes, and content models of the language
- Modules are meant to be reusable independently
- If your XML design needs a bit of rich text, recycle some XHTML 2!

Namespaces

- XHTML 2 introduces a new namespace name: *http://www.w3.org/2002/06/xhtml2/*
- Ruby, XForms, and XML Events have their own namespace names

Schema language independence

- The working drafts define XHTML 2 syntax *normatively* in RELAX NG *(www.relaxng.org)*.
- DTD and W3C XML Schema normative definitions will be added in later drafts

NEW AND NIFTY

Ubiquitous hyperlinks

- Any element can be the target of a hyperlink (through the id attribute)
- Any element can be the *source* of a hyperlink (through the href attribute)
- The a element is kept for backward compatibility

Hyperlinking example

```
<p href="#security">Security
     Considerations</p>
           ...
<p id="security">This system
 is just inherently insecure.
      Deal with it.</p>
```


Generalized transclusion

- Any element can have a `src` attribute providing external content to be rendered instead of the element
- This means rich text can be embedded as well as images and objects
- The `img` element is likewise kept for backward compatibility

Transclusion example

```
<p src="http://www.example.com/intro">
   Hmm, transclusion isn't working.
                Sorry.  <a
 href="http://www.example.com/intro">
  Click</a> here and then use the Back
              button.</p>
```


How to do images

```
<img src="..." alt="text">
```

becomes

```
<span src="...">text</span>
```

or

```
<object data="...">text</object>
```


External modules

- Ruby provides support for Japanese-style annotation
- XML Events is a greatly extended events module based on DOM Level 2 Events
- XForms is a completely new forms language
- Other languages (MathML, SVG) aren't part of XHTML 2 but can be used with it

XForms

- Widgets can appear anywhere in a XHTML document
- Arbitrary XML is embedded in the document as a template
- Widgets are bound using XPaths to appropriate parts of the template
- The filled-out template is submitted

Streamlined language

- HTML 4.0: 91 elements, 188 attributes (counting shared attributes only once)
- XHTML 2: 59 elements (plus 1 for XEvents and 29 for XForms), 59 attributes
- Almost any attribute can be attached to almost any element
- XHTML 2 is the *true* (refactored) second version of HTML

WHAT'S HISTORY

It's XML, you know

- Make sure all start-tags have end-tags and empty-tags end in />
- Lower-case elements and attributes
- Quote attribute values
- Wrap script and style content in CDATA sections
- HTML Tidy *(tidy.sf.net)* is your friend

Presentational markup is dead, dead, dead

- Originally, HTML was designed to represent the structure of a document, not its presentation
- Presentation-oriented elements were later added to the language by browser manufacturers
- XHTML 2 takes HTML back to its roots

Presentational markup is dead, dead, dead

- All presentational elements and attributes are gone
- Even old warhorses like `b`, `i`, and `br` bite the dust
- HTML Tidy is still your friend
- XSLT is also your friend

Presentational markup is dead, dead, dead

- Presentation handled solely by stylesheets
- Standardizing on stylesheets means enhanced flexibility of presentation
- CSS can do more than presentational HTML ever did
- A standard stylesheet will be provided to give the default (traditional) presentation

The `img` and `applet` elements

- Superseded by the new and generalized `object` element, which is now used to embed any non-XHTML content
- XSLT is your friend once again

No more frames

- Everything to do with frames is gone
- The `target` attribute of hyperlinks remains, but its use is not defined

New forms and event sublanguages

- The new XForms language completely supersedes HTML Forms
- It's still possible for forms to return attribute-value pairs, but full XML documents are the recommended method
- Event handling now uses XML Events

THE ELEMENTS

The Document Module

- Elements: `html`, `head`, `title`, `body`
- Attributes specific to the `html` element:
  - `version` replaces `DOCTYPE` declaration as an effective version signature
  - `xsi:schemaLocation` points to W3C XML Schemas for namespaces

The Structural Module

- Elements: `p`, `div`, `separator`, `address`, `blockquote`, `pre`, `blockcode`, `section`, `h`, `h1`-`h6`
- `blockcode` is a hybrid of `blockquote` and `pre`, intended to set off code
- `separator` replaces `hr`
- `p` can now contain lists and tables, but not nested paragraphs

The `section` and `h` elements

- `section` elements
  - contain major divisions of text
  - can be nested to convey subdivisions, sub-subdivisions, etc.
- The `h` element works like `h1` within a main document, like `h2` within a top-level section, like `h3` within a sub-section, etc.
- `h1`-`h6` are no longer necessary

The Inline Text Module

- Elements: `abbr`, `cite`, `code`, `dfn`, `em`, `kbd`, `l`, `q`, `samp`, `span`, `strong`, `sub`, `sup`, `var`
- *No* presentation elements: not even `b` and `i` survive
- `q` is for quotations, but no longer tries to supply quotation marks automatically (the rules depend on language and country)

The `l` element

- Container for a single line (of verse, computer code, etc.)
- Replaces HTML's `br`

```
<p><l>Take what you need,</l>
   <l>need what you take.</l>
</p>
```


What are those elements for, anyhow?

- `abbr`: an abbreviation (`full` attribute is an IDREF to the full name)
- `cite`: a citation or source reference
- `code`: inline computer code
- `dfn`: a definition
- `kbd`: keyboard input
- `samp`: sample output
- `var`: a variable

The Hypertext and Image Modules

- Elements: `a`, `img`
- Redundant thanks to ubiquitous hypertext (same as `span`)
- Being kept in XHTML 2 because they:
  - are familiar
  - have well-known semantics
  - have well-known default presentations

The List Module

- Elements: `dl`, `dd`, `dt`, `dl`, `ol`, `ul`, `li`, `nl`, `label`
- Definition lists are as before
- Ordered and unordered lists are as before
  - the `value` attribute on `li` tells where to start numbering
  - specialized numbering only in stylesheet)

The List Module

- The new `label` child element can be used to label any list (it must be the first child)
- The new `di` element groups term(s) with definition(s) in a `dl` list, showing what defines what

Navigation lists

- `nl` elements are initially displayed in collapsed form, with just the required `label` child shown
- Clicking on the label causes the `li` children to be popped up/pulled down
- Nested navigation lists supported
- No more scripting just to get decent menus!

Access Module

- `access` element specifies keyboard access to elements in the document
- `access` must be in the head
- `key` attribute specifies the single-character key mapping
- `targetid` and `targetrole` specifies the target element either by id or by matching roles

The Metainformation Module

- Elements: `meta`, `link`
- RDF is a standard way of making statements about a resource
- RDF statements contain a *subject*, a *predicate*, and an *object*
- `link` and `meta` elements can only appear in the head

The Metainformation Module

- The `meta` element can contain plain text, possibly decorated with Text Module elements
- The `link` element can only contain other `link` and `meta` elements
- Browsers are encouraged to render `link` (perhaps as part of a dropdown list of links)
- The metainformation attributes control the semantics of both kinds of elements

The Object Module

- Elements: `object, param, caption, standby`
- Supersedes `img` and `applet`
- `src` specifies the object's source
- The content of `object` is used if the object can't be loaded
- Nested `object` elements provide for a series of fallbacks

Attributes of `object`

- `content-length` is a hint about the size of the object
- `archive` specifies resources (jar files, e.g.) to be preloaded
- `declare="declare"` means:
  - object will only be loaded
  - must be launched by a user action

The `param` element

- Embedded in an object element, but not part of the fallback
- Passes a name-value pair to the object
- Attributes: `name`, `value`, `valuetype`, `type` only; does *not* support Common Attributes

The `param` element

- `valuetype` specifies the kind of value:
  - `data` for a simple string
  - `ref` for an external reference
  - `object` for the ID of an object element elsewhere in the document
- When `valuetype` is `ref`, `type` specifies the content type of the external resource

The `caption` and `standby` elements

- `caption` specifies a caption; this is borrowed from the Tables Module
- `standby` provides textual content to be displayed while an object is being loaded

The Ruby module

- Supports annotations written above or next to the base text
- A Japanese requirement to explain obscure kanji characters
- The `ruby` element is added to Text Module
- Child elements: `rbc`, `rtc`, `rb`, `rt`, `rp`
- For details, see *www.w3.org/TR/ruby*

The Handler Module

- Elements: `handler`
- `handler` specifies a script or other event handler
- Can appear in head or body as either a Structural or a Text element
- If `src` is not given, the content is a fallback handler if it is wrapped in a child `handler` element, or else the script itself

Attributes of `handler`

- `type` specifies the scripting language and is now required
- `encoding` gives the character set of an external script, since scripting languages typically aren't self-identifying
- `declare="declare"` means don't run the script until an event activates it

The Style Sheet Module

- Element: `style`
- Must be in the `head` element
- Used for embedded sheets only
- External stylesheets use the `link` element from the Linking Module
- The `<?xml-stylesheet ... ?>` processing instruction is also supported

External stylesheets

- Appropriate attributes on the `link` element specify different flavors of external stylesheet
- `href` specifies the location of the stylesheet (if omitted, the content is the stylesheet)
- `type` specifies the stylesheet language

Types of stylesheets

- Persistent: `rel="stylesheet"`
- Preferred: `rel="stylesheet" title="..."`
- Alternate: `rel="alternate stylesheet" title="..."`
- User agents should allow different stylesheets to be selected

The Style Attribute Module

- Attribute name: `style`
- Added to the Common Attributes Collection
- Specifies the style of a single element using an unspecified style language
- Strongly discouraged in favor of the Style Sheet Module

The Tables Module

- Elements: `table`, `caption`, `summary`, `col`, `colgroup`, `thead`, `tfoot`, `tbody`, `tr`, `th`, `td`
- Essentially the same structure as older versions of HTML
- As always, presentation is done via stylesheets; width, border, rules, spacing, padding attributes are gone

Tables Module attributes

- `col` and `colgroup` support the `span` attribute (number of columns spanned)
- `th` and `td` continue to support the HTML attributes `abbr`, `axis`, `headers`, `scope`, `rowspan`, `colspan` (all fairly advanced features)

THE COMMON ATTRIBUTES

Common Attribute Collection

- In XHTML 2, all elements support the same list of attributes!
- Exceptions: `param`, elements from external modules
- `html`, `abbr`, `li`, `style`, `handler`, `access`, `th`, `td`, `col`, and `colgroup` have their own special-purpose attributes

List of Attribute Modules (including some attributes)

- Core: `class`, `xml:id`, `title`
- Internationalization: `xml:lang`, `dir`
- Hypertext: `href`, `cite`
- Embedding: `src`, `type`
- Role: `role`
- Editing: `edit`, `datetime`

Attribute Modules (including some attributes)

- Media: `media`
- Meta: `about`, `content`, `property`
- Image Map: `usemap`, `ismap`, `shape`
- Events: `event`, `handler`, `observer`, `target`
- Style: `style`

Core attributes

- `class` used for stylesheet control and general-purpose text processing; multiple names are allowed
- `id` (alternatively `xml:id`) used for:
  - Hypertext anchors (replaces `name`)
  - Stylesheet control

Core attributes

- `title` used for tooltips, multiple stylesheets
- `layout` indicates whether whitespace is `relevant` or `irrelevant` to the element's meaning; default is `irrelevant`.

Internationalization attributes

- `xml:lang` specifies the language of content (supersedes HTML's `lang`)
- `dir` specifies directional properties of text (for use with Hebrew, Arabic, Syriac scripts)
  - values are `ltr`, `rtl`, `lro`, `rlo`
  - supersedes the HTML `bdo` element

Hypertext attributes

- `href` is unchanged, but any element can bear it
- `hreflang`, `hrefmedia`, `hreftype` are the language, CSS2 media type, and MIME type of the referenced document
- `cite` points to a source document that gives more information about the element

Hypertext attributes

- `nextfocus` and `prevfocus` are IDREFs to the element that will get the focus when this element has the focus and the user tabs or shift-tabs away
- `target` specifies the destination of the hyperlinked resource (values not defined)
- `xml:base` sets the base URI

Embedding attributes

- XHTML 2 allows general-purpose transclusion, including hypertext
- `src` specifies a resource to be embedded in the document in place of the element to which it is attached
- `srctype` and `encoding` give the media type (and, if needed, the character encoding) of the resource

Role attribute

- Elements can have roles, which are arbitrary QNames that specify the purpose of parts of a document
- `role` specifies one or more QNames
- Standard values: `main`, `secondary`, `navigation`, `banner`, `contentinfo`, `definition`, `note`, `seealso`, `search`

Editing attributes

- Provide a simple changelog facility
- `edit` can specify `inserted`, `deleted`, `changed`, or `moved`
- `datetime` specifies time of last change
- `edit="deleted"` defaults to invisible

Media attribute

- Some elements should only be presented for particular media
- `media` specifies a CSS2 media type such as `print` or `screen`
- If the current presentation medium matches the attribute, the element is presented; otherwise, it is ignored

Metainformation attributes

- Allows RDF statements to be made locally to arbitrary elements
- `about` specifies an RDF *subject* (if missing, the first ancestor with an `about` or `id` attribute is the subject)
- `property`, `rel`, and `rev` specify the RDF *property* value (can be a QName like `dc:creator`)

Metainformation attributes

- If a `rel` attribute is present, the RDF *object* is specified by the `href` attribute
- If a `rev` (reverse relationship) attribute is present, the RDF subject is specified by `href` and the RDF object by `about`
- If a `property` attribute is present, the RDF object is the content of the element
  - `datatype` is the XML Schema datatype of the RDF object

Standardized `property` values

- `description`: a basic description
- `generator`: identifies the software used to generate the document
- `keywords`: comma-separated list
- `title`: specifies a title
- `robots`: advice to web-crawlers
- `reference`: default property

Standardized `rel/rev` values

- `alternate`: alternate version
- `start`, `next`, `prev`, `up`: adjacent resources in the collection this resource belongs to
- `contents`, `index`, `glossary`, `copyright`, `appendix`: a resource serving as a specialized part of the whole collection

Standardized `rel/rev` values

- `chapter`, `section`, `subsection`: the current part(s) of this collection
- `help`: resource offering help
- `bookmark`: key entry point within a collection
- `meta`: a resource that provides metadata
- `icon`: icon for this resource or collection

More Common Attributes

- Image Map: `usemap`, `ismap`, `shape`, `coords` are basically unchanged
- Events (XML Events namespace): `event`, `handler`, `observer`, `phase`, `propagate`, `target`, `defaultAction`
- Style (*strongly discouraged*): `style`

Repurposing Attributes

- HTML attributes attached to HTML elements are in no namespace
- However, if you want to attach an HTML attribute like `href` to a non-HTML element, put it in the XHTML 2.0 namespace using `html:href`
- This rule applies both in XHTML 2.0 documents and other documents

XML EVENTS

Basic XML Events model

- Every event has a *target* element (the one being clicked on, inserted, removed, gaining or losing focus, etc.)
- Any element can have one or more *handlers* for event types (typically scripts)
- Handlers are attached to *observer* elements
- The observer must be either the same as the target or an ancestor of it

Capturing and bubbling

- Most events can be *captured* by an ancestor of the observer before the observer even sees the event
- If the observer element has no handler for the event, it *bubbles* up the tree until some ancestral element handles it
- Capturing and bubbling handlers can be distinct, even on the same element

Stopping propagation and canceling

- A handler can stop the propagation of an event during either capturing or bubbling
- Events have associated default behaviors
- The default behavior can be canceled or allowed to happen (after all handlers are processed)

The XML Events Module

- Uses a separate namespace: *http://www.w3.org/2001/xml-events*
- Element: `ev:listener`
- Attributes: `event`, `observer`, `handler`, `target`, `phase`, `propagate`, `defaultAction`, `id`

Attributes of `listener`

- `event` is the name of the event
- `handler` is the URI of a script
- `observer` is the ID of the observer
- `target` is the ID of the target
- If `target` is not specified, any descendant of the observer is allowed as a target

Attributes of `listener`

- `phase` is `capture` for capturing-phase handlers, `default` for bubbling-phase
- `propagate` is `stop` to stop propagation after this handler runs, `continue` to continue looking for handlers
- `defaultAction` is `cancel` to cancel the default action for this event or `perform` to do it

Putting attributes on the observer

- Attributes other than `observer` can be placed directly on the observer element
- Such attributes must be qualified with the XML events namespace

Putting attributes on the handler

- Attributes other than `handler` can be placed directly on the handler element
- Such attributes must be qualified with the XML events namespace
- If `observer` is omitted too, the observer is the parent of the handler

Standard event types

- UI events: `DOMFocusIn`, `DOMFocusOut`, `DOMActivate`
- Mouse events: `click`, `mousedown`, `mouseup`, `mouseover`, `mousemove`, `mouseout`
- Key events: not yet defined

Standard event types

- Mutation events: `DOMSubtreeModified`, `DOMNodeInserted`, `DOMNodeRemoved`, `DOMNodeInsertedIntoDocument`, `DOMNodeRemovedFromDocument`, `DOMAttrModified`, `DOMCharacterDataModified`

HTML-specific event types

- Events: `load`, `unload`, `abort`, `error`, `select`, `change`, `reset`, `focus`, `blur`, `resize`, `scroll`
- Some of these are redundant with previously specified events
- Some are probably obsolete

XFORMS

Goals

- Controls and values are internationalized, accessible, and device-independent
- Submissions are type-validated XML
- Validity is defined by XML Schema, augmented by XForms-specific declarations
- Reduced dependence on scripting

XForms models

- Models contain:
  - Instance data
  - Bindings from UI controls to data
  - Submission instructions
  - Optional W3C XML Schema reference used to validate the instance data
- There can be any number of models in a document

Instance data

- A shell or skeleton of what is going to be returned to the server when the form is submitted
- Provides the XML structure and any fixed attributes or character data
- Can be loaded from an external resource
- Data already in the instance becomes the initial value of bound controls

Bindings

- Associate part of the instance document with a specific forms control
- XPaths specify the relevant part of the instance
- The association is bidirectional: change either, the other changes too
- Can be a separate element or an attribute on the control

Submissions

- Not to be confused with the submit control
- Specifies the URI, method, and encoding where the form is to be submitted
- Specifies what to do with data returned by the server:
  - Replace the whole page
  - Replace just the instance data
  - Discard the returned data

Form controls

- Can occur anywhere in a document – `form` is dead
- Represent abstract input devices, not specific UI widgets
- Rendering is up to the browser
- Can have labels, help, hints, and alerts attached to them

Input controls

- `input` specifies simple data; browsers should provide datatype-specific input methods
- `secret` is similar, but conceals its value
- `textarea` is similar, but supports multi-line text with no datatype

Selection controls

- `select1` allows the user to select a single item from a list
- `select` is similar, but allows multiple selections
- Selections can be grouped (this replaces `optgroup`)
- `range` provides a range of values to choose one from

Other controls

- `trigger` signals a single event when activated (replaces `button`)
- `submit` provides a submit widget
- `upload` provides a file browser for uploading bulk content
- `output` displays the value of an XPath expression, often referring to the instance

Grouping of controls

- `group` sets off a group of controls
- `switch` is a container for sets of controls (enclosed in case elements) to be displayed alternatively, controlled by toggle elements
- `repeat` allows groups of items to be added or deleted, allowing for shopping-cart-like behavior

Controlling control values

- Constraints apply to a value and are extended to all controls bound to that value
- Values can be:
  - Required
  - Disabled or made read-only
  - Constrained by an XPath expression
  - Constrained by an XForms datatype
  - Calculated from other values

XForms datatypes

- All simple datatypes except `duration`, `ENTITY`, `ENTITIES`, `NOTATION`
- Durations use `yearMonthDuration` and `dayTimeDuration`, which are not in XML Schema but are in XQuery as well as XForms

XForms datatypes

- XForms-specific `listItem` and `listItems` are like `NMTOKEN` and `NMTOKENS` but without lexical restrictions
- Derivation by restriction and by list are supported

XPath functions for XForms

- Boolean: `boolean-from-string`, `if`
- Numeric: `avg`, `min`, `max`, `count-non-empty`, `index`
- Date/time: `now`, `days-from-date`, `seconds-from-dateTime`, `seconds`, `months`
- Nodeset: `instance`

XForms events

- Too many event types to list here
- General types:
  - Standard events
  - Initialization
  - Interaction
  - Notification of changes
  - Errors and exceptions
- The sequence of events is carefully defined

XForms events

- The most important events:
  - `xforms-ready` (initialization)
  - `xforms-focus` (when a control gains focus)
  - `xforms-submit` (before submission)
  - `xforms-value-changed` (changed data)
  - `xforms-invalid` (data is invalid)

XForms actions

- Events can trigger built-in XForms actions as well as scripts
- Some of the actions:
  - Submit
  - Reset
  - Display message box
  - Traverse a link
  - Set an instance value

XForms actions

- More actions:
  - Recalculate instance data
  - Revalidate instance data
  - Refresh form controls from instance data
  - Change focus
  - Redispatch event to another control
- A single event can cause multiple actions to occur

Some existing XForms implementations

- X-Smiles browser (open source, written in Java)
- FormsPlayer (a plugin for IE6 SP1)
- Novell's XForms (a testbench browser analogous to `appletviewer`)
- Mozquito DENG (an XML + CSS + XForms + SVG browser written in Flash MX)

TO BE CONTINUED...

Some open issues

- Currently anything can have image-map attributes, but it only works if `src` specifies an included image
- There's no document-type independent way to detect inline style attributes
- Character entities: staying or going?
- Further review will probably create more issues

MORE INFORMATION

`http://www.w3.org/TR/xhtml2/`

`http://www.ccil.org/~cowan/xhtml2{.ppt,.sxi,.pdf}`