Accessibility Guidelines for OpenACS/.LRN Developers

Emmanuelle Raffenne & Héctor Romojaro

Agenda

- Defining Accessibility and Usability
- Getting what we need: tools & references
- Putting stuff in the right place: how to organize/layer the front-end code
- The Web Content Accessibility Guidelines
- Satisfying the checkpoints/success criteria
- Open issues in OpenACS
- Conclusions

Accessibility & Usability Definitions

Usability - Definition

"Usability is the extent to which a system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use"

ISO 9241-11

Accessibility - Definition

"The art of ensuring that, to as large an extent as possible, facilities (such as, for example, Web access) are available to people whether or not they have impairments of one sort or another."

Tim Berners-Lee

Accessibility: Why

- improves code and pages quality
- improves level of usability for **<u>everyone</u>**
- most countries included accessibility as a legal requirement now

Accessibility & Usability are linked

Accessibility: How

- Making a website accessible is not like turning a switch on
- But it starts by turning a switch on in our minds:
  - use technologies as they were meant to be used
  - remove the barriers people with special needs usually encounter while surfing the web
    - understand those barriers
    - understand how WCAG address them
  - keep the above in mind along the whole process of building a website

Getting what we need

Getting what we need: W3 references

- Guidelines:
  - http://www.w3.org/TR/WCAG20/
- Understanding WCAG 2.0:
  - http://www.w3.org/TR/UNDERSTANDING-WCAG20/
- Techniques:
  - http://www.w3.org/TR/WCAG20-TECHS/

Getting what we need: Section508

- The federal law:
  - http://www.section508.gov
  - See § 1194.22 “*Web-based intranet and internet information and applications*”.

- A guide to section 508 § 1194.22:
  - http://www.access-board.gov/sec508/guide/1194.22.htm

Getting what we need: Firefox add-ons

- Web developer:
  - HTML and CSS validation
  - WCAG/508 reports
  - many others goodies …
- Colour Contrast Analyzer:
  - Luminosity Contrast Ratio
  - Color Difference
- Fire Vox:
  - http://clc4tts.clcworld.net/clc-firevox_doc.html

Getting what we need: Scripts

- Scripts to find **objective** defects in snippets:
  - missing mandatory attributes for markup tags
  - misuse of markup
  - use of absolute units in CSS
  - inline styles
  - blinking, refresh and redirect
  - deprecated and/or presentation elements
  - etc ...

- Will be available at the openacs.org file-storage shortly

Putting stuff in the right place

Data – Structure - Style

Putting stuff in the right place: Data → Tcl

- **Data**: information to be output in the document
- **Metadata**: title, content-type and charset, lang, keywords, stylesheets, etc.
- optionally javascript

- Don't build HTML in Tcl scripts
- Markup is adp business

- header_stuff : DEPRECATED!
- use template::head
- use the “doc” array for document properties

Putting stuff in the right place:
structure → adp

Don't use presentation markup in adp
Avoid inline styles
Styles and layout is CSS business

- **Structure** the document using <u>semantic</u> markup
- **ID**: to IDentify blocks of information (has to be unique!).
- **CLASS**: to define the style (format) to be applied

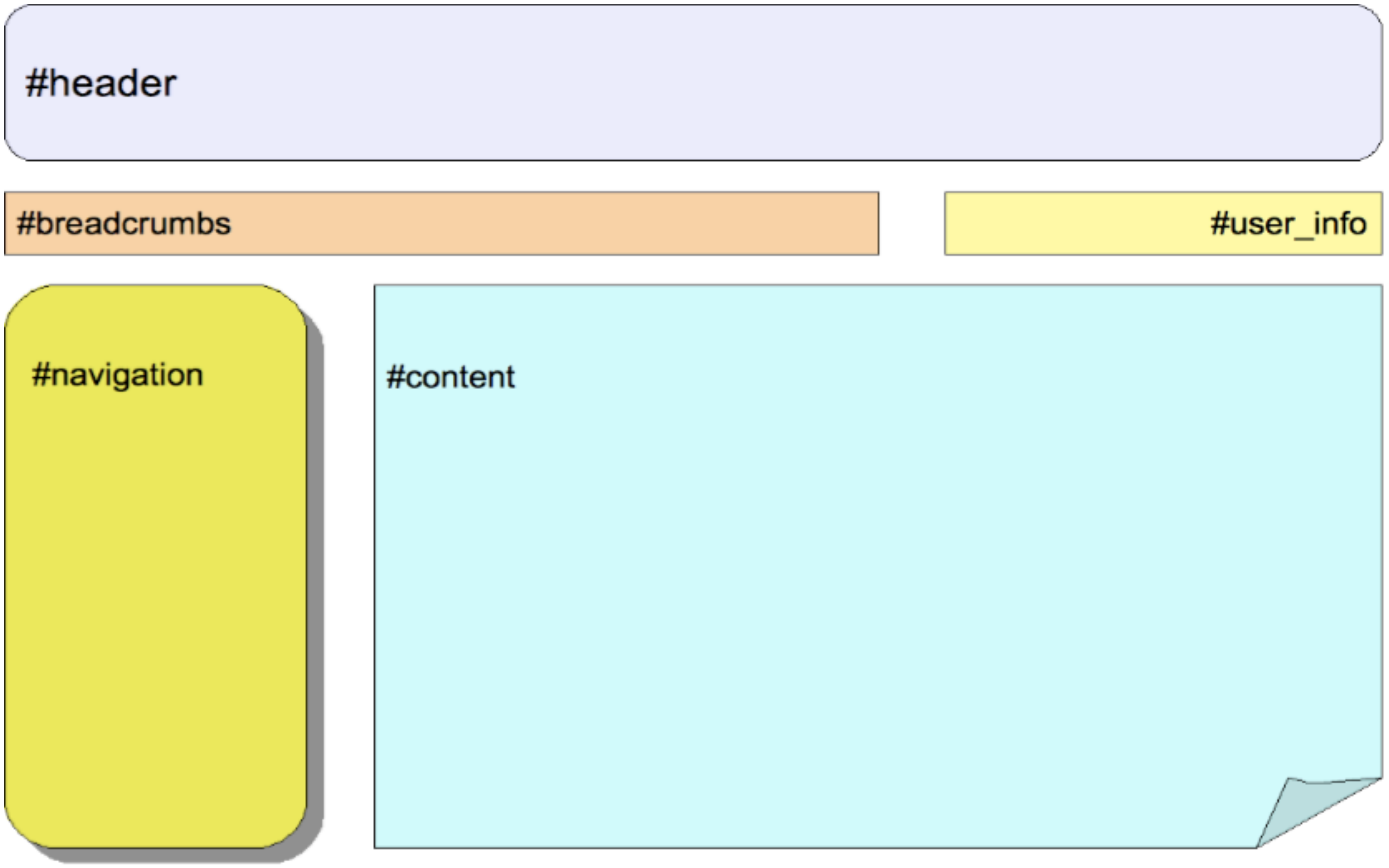
Putting stuff in the right place:
styles → CSS

“Divide and Conquer”
Don't put everything in a 1000s lines CSS

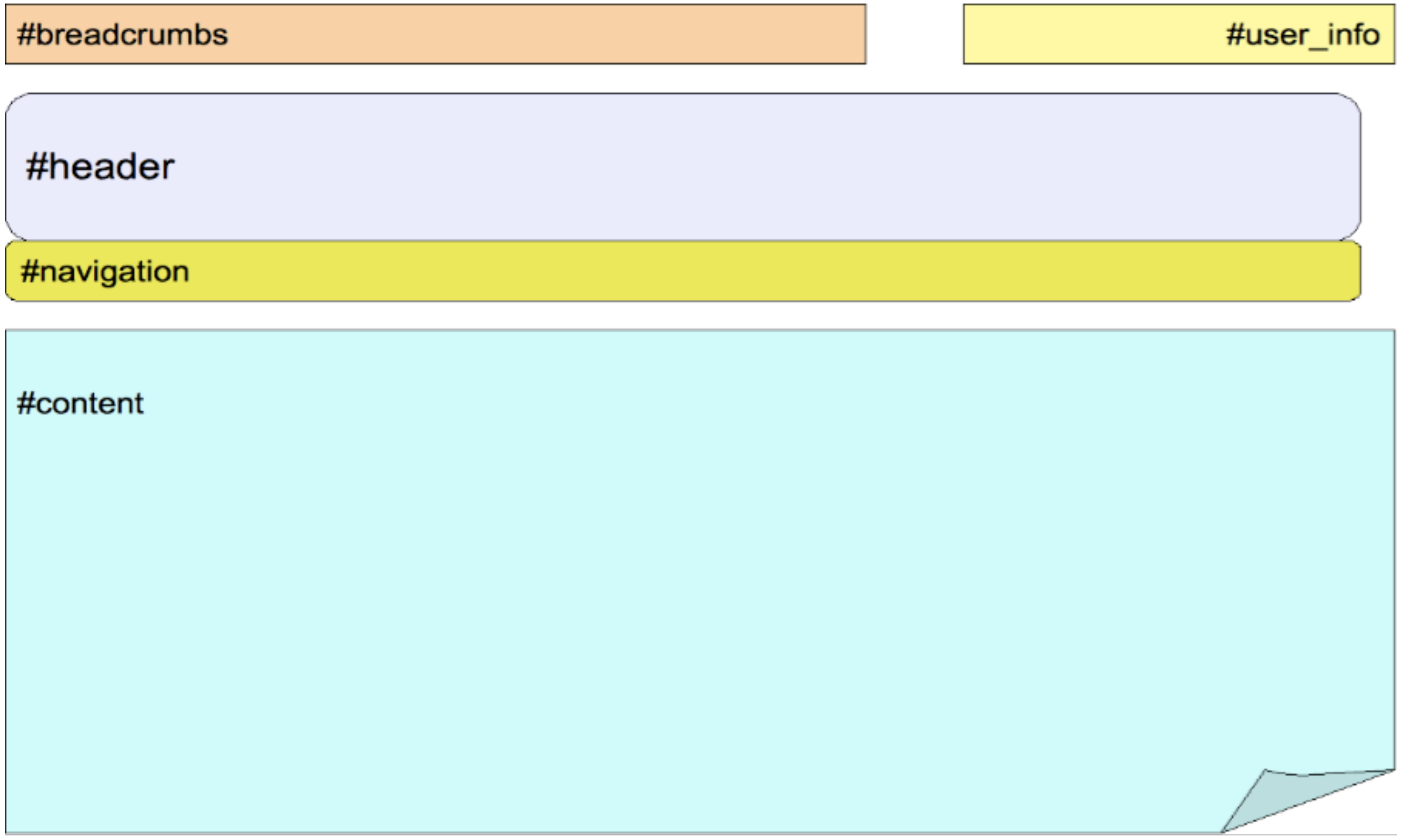
- **main.css**: common styles
- **type.css**: fonts, colors, backgrounds, etc… (using CSS selectors). Can be refined: fonts and <u>colors</u> (e.g. for high contrast, skining).
- **layout.css**: to position blocks in the page . Can be refined by media (print, handheld, etc.)

Will make it easier to maintain and customize

Laying out using IDs (option 1/n)



Laying out using IDs (option 2/n)



Formating and Laying out - Examples

http://www.dotlrn.org ☹

http://www.csszengarden.com/ ☺

⚠ These are NOT examples of accessible sites but of how to lay out and format using CSS

WCAG 2.0

W3C Recommendation 11 December 2008

WCAG 2.0

- **4 Principles**: The guidelines and Success Criteria are organized around four principles, which lay the foundation necessary for anyone to access and use Web content.
- **12 Guidelines**: Overall objectives. Provide the basic goals that authors should work toward in order to make content more accessible to users with different disabilities.
- **Success criteria**: testable success criteria are provided for each guideline. Assigned to 1 of the 3 levels of conformance (A, double-A, triple-A)
- **Sufficient and Advisory Techniques**: provided for each of the guidelines and success criteria

WCAG 2.0 - The 4 principles

1. **Perceivable**: Information and user interface components must be presentable to users in ways they can perceive.
2. **Operable**: User interface components and navigation must be operable
3. **Understandable**: Information and the operation of user interface must be understandable
4. **Robust**: Content must be robust enough that it can be interpreted reliably by a wide variety of user agents, including assistive technologies

WCAG 2.0 – Guidelines for principle 1

- **Perceivable**
  - Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.
  - Provide alternatives for time-based media.
  - Create content that can be presented in different ways (for example simpler layout) without losing information or structure.
  - Make it easier for users to see and hear content including separating foreground from background.

WCAG 2.0 – Guidelines for principle 2

- **Operable**
  - Make all functionality available from a keyboard.
  - Provide users enough time to read and use content.
  - Do not design content in a way that is known to cause seizures.
  - Provide ways to help users navigate, find content, and determine where they are.

WCAG 2.0 – Guidelines for principle 3

- **Understandable**
  - Make text content readable and understandable.
  - Make Web pages appear and operate in predictable ways.
  - Help users avoid and correct mistakes.

WCAG 2.0 – Guidelines for principle 4

- **Robust**
  - Maximize compatibility with current and future user agents, including assistive technologies.

Satisfying the Checkpoints/Success Criteria

Priority/Level 1 and 2

Important Note

- **WCAG 2.0** has been released last december: very recent
- This tutorial has been written initially based on **WCAG version 1.0** for checkpoints **level A and level double-A**
- This tutorial has been adapted to WCAG 2.0 by <u>mapping</u> the WCAG 1.0 checkpoints for level A and double-A
- A few WCAG 2.0 success criteria are not included in this tutorial

Transition from WCAG 1.0 to 2.0

- Differences Between WCAG 1.0 and WCAG 2.0
  - http://www.webaim.org/standards/wai/wcag2.php
- **Mapping of WCAG 1.0 checkpoints to WCAG 2.0 success criteria**
  - http://www.w3.org/WAI/GL/2005/11/23-mapping.html
- Transition 1.0 → 2.0:
  - http://www.w3.org/WAI/EO/changelogs/cl-transition1to2

Guideline 1.1

Text Alternatives

Provide text alternatives for any non-text content so that it can be changed into other forms people need, such as large print, braille, speech, symbols or simpler language.

Guideline 1.1 – Text alternatives

- **Techniques**:
  - provide ALT text for images
  - move background/decoration images to CSS

- **Tools**:
  - Webdev:
    - Images → display alt
    - Images → disable all images
    - Tools --> Validate local HTML
  - Scripts: look for missing ALT in IMG tags

Guideline 1.2

Time-based Media

Provide alternatives for time-based media.

Techniques:

- **Audio content**: provide transcription
- **Video content**: provide captions, audio description and transcription

Guideline 1.3

Adaptable

Create content that can be presented in different ways (for example simpler layout) without losing information or structure.

Guideline 1.3 - Adaptable

Techniques:

- Use header elements to convey document structure: H1, H2, ..., in sequence (don't skip a level)
- Mark up lists and list items properly: UL, OL, DL
- Don't use semantic markup for formating purpose: BLOCKQUOTE and Q

Tools:

- Webdev: Tools → Outline → Outline headings (with “show elements name” checked to display heading level)

⚠ Do NOT use template::list for a one column list

Guideline 1.3 - Adaptable

Example:

```
<!-- block -->
<BLOCKQUOTE cite=”http://...”>
  <p>Cited text...</p>
</BLOCKQUOTE>

<!-- inline -->
<p>
  As X said
  <Q cite=”http://...”>Me too!</Q>
</p>
```

Guideline 1.3 - Adaptable

Techniques:

- For data tables, identify row and column headers and use markup to associate data cells with their header cells:
  - Use list builder, it will output a well-formed table
  - Do NOT use display_template
- Do NOT use TH or TD for presentation purpose
- Do NOT use tables for layout.

Tools:

- webdev: Outline → Outline Tables → Table Cells
- Scripts: look for TH without ID attribute and TD without HEADERS attribute

Guideline 1.3 - Adaptable

Example:

```
<TABLE>
  <!-- header cells -->
  <thead>
   <th id="name">Name</th>
   <th id="email">Email</th>
  </thead>
  <!-- tfoot would go here -->
  <!-- data cells -->
  <tbody>
   <td headers="name">Me</td>
   <td headers="email">me@acme.com</td>
  </tbody>
</TABLE>
```

Guideline 1.3 - Adaptable

Techniques:

- Do NOT use inline styles
- Do NOT use presentation elements: B, I, S, PRE, TT
    - B -> STRONG (very important) or EM (emphasis)
    - I and S -> in CSS
    - PRE and TT -> CODE

Tools:

- Webdev: CSS → Disable styles → All styles
- Webdev: Tools → Validate **local** HTML
- Scripts: look for deprecated and presentation tags and attributes

Guideline 1.4

Distinguishable:

Make it easier for users to see and hear content including separating foreground from background.

Guideline 1.4 - Distinguishable

Techniques:

- Ensure that foreground and background color combinations provide sufficient contrast
  - in CSS: set explicitly the color for text, background (inherit if necessary), border, links (all states)
  - Provide an alternative CSS for high contrast
- Use relative rather than absolute units:
  - “em” or “%” rather than “px” or “pt”
- Use image of text for decoration only
  - set the background image in CSS

Tools:

- Colour Contrast Analyzer: run “All tests” on all color schemes
- Increase font size to check the fluidity of the page
- Scripts: look for absolute units in CSS and literal color names (black, blue, etc.)

Guideline 2.1

Keyboard Accessible:

Make all functionality available from a keyboard.

Techniques:

- Use "onmousedown" with "onkeydown".
- Use "onmouseup" with "onkeyup"
- Use "onclick" with "onkeypress"
- Do NOT use “ondblclick”: no keyboard equiv.

Guideline 2.2 & 2.3

***Enough Time:** Provide users enough time to read and use content.*

***Seizures**: Do not design content in a way that is known to cause seizures.*

Techniques:

- Avoid motion or provide the user with options to control it
- Avoid causing the screen to flash or blink

Tools:

- Scripts: look for BLINK tag

Guideline 2.4

***Navigable**:*

Provide ways to help users navigate, find content, and determine where they are.

Guideline 2.4 - Navigable

Techniques:

- Provide a title and an accurate context for each page
  - Page title == last element of breadcrumbs (context)
- Structure the content using headings (H1, H2, …)
- Clearly identify the target of each link
  - link text should be meaningful, if not use the "title" attribute
  - use the link_html property for `template::list` elements `{link_html {title "more descriptive"}}`
- Provide a site map or table of contents
- Title each frame to facilitate frame identification and navigation.
  - `<frame src="page" title="A meaningful title">`
- Provide ways to skip over navigation links

Tools:

- Webdev: Information → Display title attributes
- Scripts: look for missing title in FRAME and A tags

Guideline 3.1

***Readable**:*

Make text content readable and understandable.

Guideline 3.1 - Readable

Techniques:

- Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions):
  - Specify the lang of the document `<HTML lang="en">`
  - Markup text that appears in a different language (still an open issue in OpenACS)
- Use the clearest and simplest language (very subjective)
- Localize **numbers**, **dates and times** and use **long format** to avoid confusion (only for display purpose):
  - `[lc_time_fmt "2008-11-03 14:00:00" "%Q %X"]` → Monday November 03, 2008 02:00 PM
  - `[lc_time_fmt "2008-11-03 14:00:00" "%q %X"]` → November 03, 2008 02:00 PM

Guideline 3.2

Predictable:

Make Web pages appear and operate in predictable ways.

Guideline 3.2 - Predictable

Techniques:

- do not change the current window without informing the user.
  - don't popup windows
  - don't resize windows
- Use server-side redirects rather than client-side ones
- Use navigation mechanisms in a consistent manner.
- Consistent style of presentation (e.g. “button” style for actions)
- Consistent links:
  - link text should match partially or entirely the title of the page it points to
  - link text should be consistent across the site

Tools:

- Scripts: look for refresh directives in META tags

Guideline 3.3

Input Assistance:
Help users avoid and correct mistakes.

Techniques:

- Forms: associate labels with their controls.
  - Use labels and place them next to the form field
  - Use implicit + explicit association:

```
<label for="nameId">
   Name
   <input id="nameId"...>
</label>
```

- Use the form builder !
  - Provide help text using the "help_text" property of form elements `{help_text "localized help"}`

Guideline 4.1

Compatible:

Maximize compatibility with current and future user agents, including assistive technologies.

Guideline 4.1 - Compatible

Techniques:

- Create documents that validate to published formal grammars.
    - Declare the document type (DTD) and content type + charset
- use list and form builders
- `template::list` for tabular data (not lists)
    - avoid `display_template`
    - use `display_col, link_url_col, link_url_eval` instead
- Blocks should **start and end** in the same template, and in the same conditional/loop block.

Guideline 4.1 - Compatible

Techniques:

- Build valid URLs:
  - In Tcl script: `set var_url [export_vars -base $url $arg_list]`
  - In ADP template: `<a href="@var_url@" ...>...</a>`
- Be careful with lists that contain a variable number of items (e.g.: based on conditions, empty multirow output using <multiple>): `<ul></ul>` will NOT validate!
- Enclose form elements in block elements (div, p, etc.), even hidden ones (form builder does it automagically).

Tools:

- Webdev: Tools → Validate **local** HTML
- List of valid tags and attributes (w3.org)

Open Issues

- To guarantee doc structure (headings sequence) when using includelets
- web2.0 technologies: Ajax => ARIA (xhtml 1.1 → IE8...)
- To markup change in natural language (lang=”en” when no translation for the current locale)
- SCORM and IMSLD players accessibility
- others?

Conclusions

- Accessibility is not a switch that can be turned on and off
- Has 2 aspects:
  - **subjective**: to be addressed (mostly) at design time → needs human testing
  - **objective**: to be addressed (mostly) at coding time → testing can be automated
- Requires awareness at all phases of the development
- Improves usability and quality of a website
- May be extensive but not so complex after all ;-)

Thank You!

¡Gracias!