aDeNu

Accessibility in OpenACS and .LRN

Tutorial

Emmanuelle Raffenne & Héctor Romojaro

Agenda

- Introduction
- Get what we need:
  - references
  - tools
- Put stuff in the right place:
  - organize the code
- Satisfy the checkpoints
- Open issues in OpenACS
- The (near) future

Accessibility What

"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."

“The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.”

Tim Berners-Lee

Accessibility
Why

- low level of accessibility = many people won't be able to browse the site
- it matters to us (we'll all get old at some point)
- higher code and pages quality
- higher level of usability for everyone
- you might get a nice job...
- it's not optional anyway...

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

Get what we need from w3

- Guidelines:

  http://www.w3.org/TR/WCAG10/

- Techniques:

  http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/

- Checklist:

  http://www.w3.org/TR/WCAG10/full-checklist.html

Get what we need from section508

- The standards:

  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

Get what we need firefox add-ons

- Web developer:
  - HTML validation
  - CSS validation
  - WCAG/508 reports
  - others goodies … we'll see them later ...
- Colour Contrast Analyzer:
  - Luminosity Contrast Ratio
  - Color Difference
- Fangs (ff2) – Fire Vox (ff3):
  - screen reader [emulator]
  - http://clc4tts.clcworld.net/clc-firevox_doc.html

Get what we need from H

- Scripts to find snippets that **may** not be compliant
  - missing mandatory attributes for HTML tags
  - misuse of markup
  - use of absolute units in CSS
  - inline styles
  - blinking
  - refresh and redirect
  - deprecated and/or presentation elements (see markup organized by type)
  - and more to come...

Get what we need Access to a Server

- http://devel.adenu.ia.uned.es:8080
- **Admin user**:
  - admin@test.test / admin
- **Regular user**:
  - user@test.test / user

Put stuff in the right place

Data – Structure - Style

Put stuff in the right place
Tcl → Data

Don't build HTML in Tcl

Markup is adp business

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

header_stuff : DEPRECATED!

use template::head

use the “doc” array property

Put stuff in the right place

adp → structure

Don't use presentation markup in adp

Avoid inline styles

Styles and layout is CSS business

- **Structure** the document using semantic markup
- **ID**: to IDentify blocks of information (has to be unique!).
- **CLASS**: to apply styles defined in stylesheets

Put stuff in the right place
styles → CSS

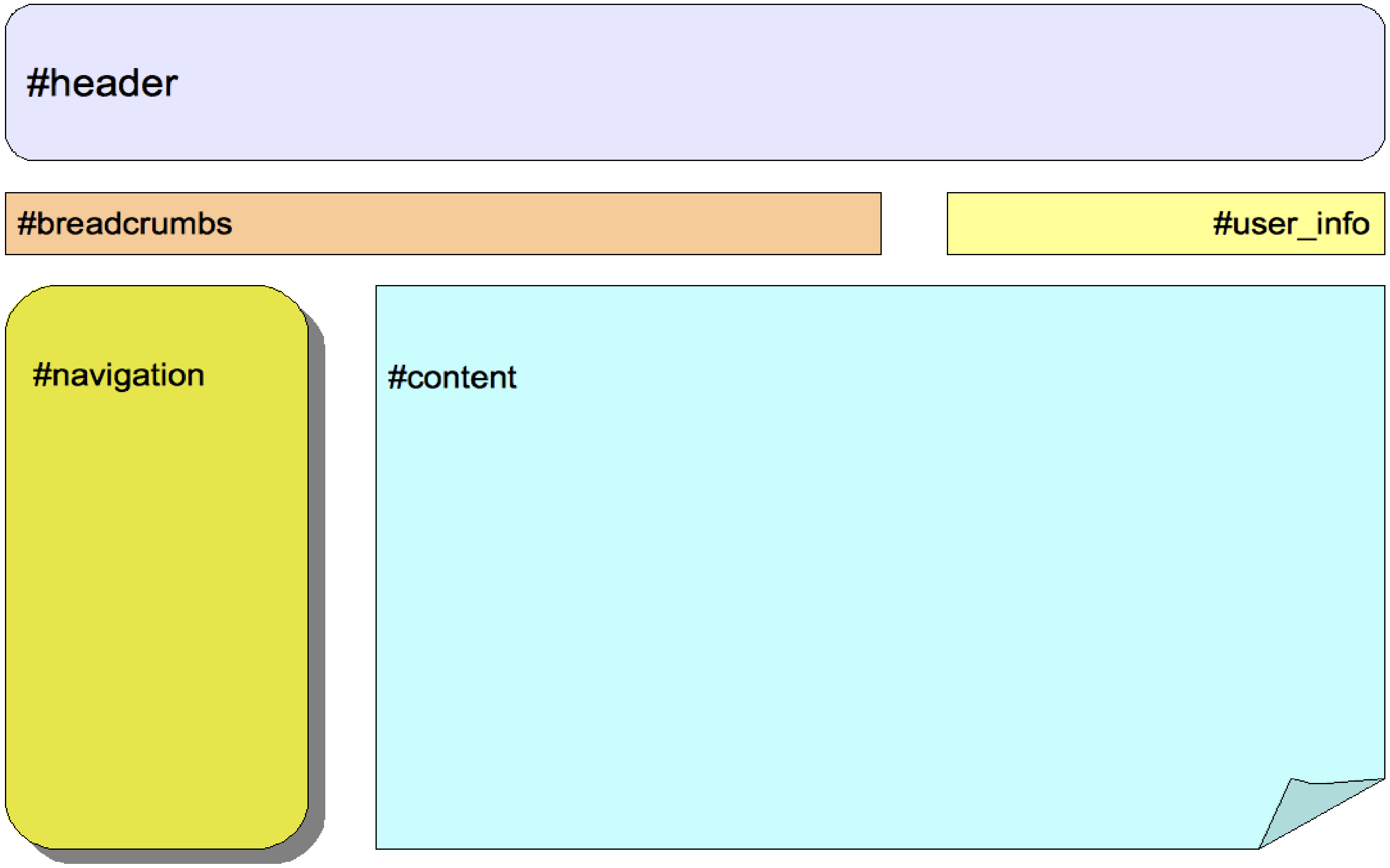
"Divide and Conquer"

Don't put everything in a 1000s lines CSS file

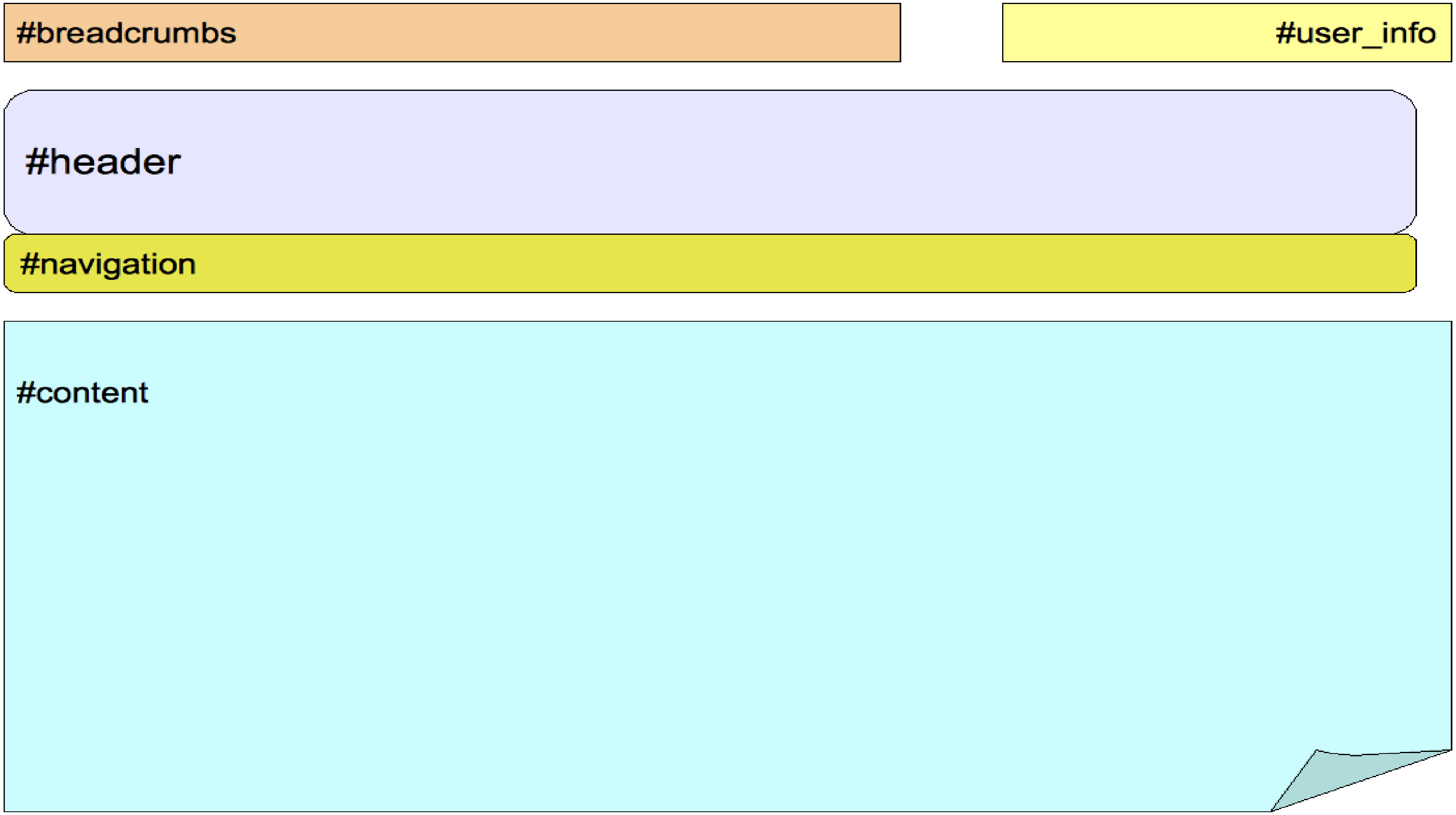
- **html.css**: HTML tags redefinitions
- **style.css**: fonts, colors, backgrounds, etc… (using class definitions and combinations of class+id). Can be divided in 2: fonts and colors (HC).
- **layout.css**: to position blocks in the page (one for screen, PDA, cell, printer, etc.)

Will make it easier to maintain and customize

Laying out using IDs Example



Laying out using IDs Example



Laying out
Live Example

http://www.csszengarden.com/

WCAG 1.0

14 Guidelines

1. Provide equivalent alternatives to auditory and visual content.
2. Don't rely on color alone.
3. Use markup and style sheets and do so properly.
4. Clarify natural language usage
5. Create tables that transform gracefully.
6. Ensure that pages featuring new technologies transform gracefully.
7. Ensure user control of time-sensitive content changes.

14 Guidelines

8. Ensure direct accessibility of embedded user interfaces.
9. Design for device-independence.
10. Use interim solutions.
11. Use W3C technologies and guidelines.
12. Provide context and orientation information.
13. Provide clear navigation mechanisms.
14. Ensure that documents are clear and simple.

Satisfying the Checkpoints

WCAG 1.0 - Priority 1 and 2

In General

1. Provide equivalent alternatives to auditory and visual content

[P1] 1.1: *Provide a text equivalent for every non-text element (e.g., via "alt", "longdesc", or in element content).*

This includes: images, graphical representations of text (including symbols), image map regions, animations (e.g., animated GIFs), applets and programmatic objects, ascii art, frames, scripts, images used as list bullets, spacers, graphical buttons, sounds (played with or without user interaction), stand-alone audio files, audio tracks of video, and video.

Text Equivalent for Images

- Techniques:
  - provide ALT
  - move image to CSS
  - replace image with text
- Tools:
  - Webdev:
    - Images → display alt
    - Images → disable all images
  - H's script

In General

2. Don't rely on color alone

[P1] 2.1: *Ensure that all information conveyed with color is also available without color, for example from context or markup.*

- Structure documents using semantic markup
- Webdev: CSS → Disable Styles → All styles

In General

2. Don't rely on color alone

[P2] 2.2: *Ensure that foreground and background color combinations provide sufficient contrast when viewed by someone having color deficits or when viewed on a black and white screen.*

- CSS: color for text, background, border, links (all states)

- Provide an alternative CSS for high contrast (title=”highContrast”)

- Colour Contrast Analyzer: All tests (for both color scheme: default and HC)

In General

3. Use markup and style sheets and do so properly

[P2] 3.1: *When an appropriate markup language exists, use markup rather than images to convey information.*

- e.g: use MathML for formula (not an image)

In General

3. Use markup and style sheets and do so properly

[P2] 3.2: *Create documents that validate to published formal grammars.*

- Declare the document type (DTD) and content type + charset
- Webdev: Tools → Validate **local** HTML (of each frames)
- 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

In General

3. Use markup and style sheets and do so properly

- Build link URL in Tcl (data):

```
      export_vars -base $url $arg_list
```

- Careful with lists that contain variable number of items (e.g.: based on conditions or empty <multiple>): <ul></ul> won't validate...
- Enclose form elements in block elements (div, p, etc.), even hidden ones (form builder does it automagically).
- HTML blocks should **start and end** in the same template, and in the same conditional/loop block.

In General

3. Use markup and style sheets and do so properly

[P2] 3.3: *Use style sheets to control layout and presentation*

- Don't use inline styles
- Don't use presentation elements (H's script)
- Webdev: CSS → Disable styles → All styles

[P2] 3.4: *Use relative rather than absolute units in markup language attribute values and style sheet property values.*

- Don't use “px” nor “pt” but “em” or “%”
- Increase font size to check the fluidity

In General

3. Use markup and style sheets and do so properly

[P2] 3.5: *Use header elements to convey document structure and use them according to specification.*

- Use H1, H2 etc and in sequence (don't skip a level)

- Webdev: Tools → Outline → Outline headings (with “show elements name” checked it's better)

[P2] 3.6: *Mark up lists and list items properly.*

- Don't use template::list for a one column list (it outputs a table!)

- Use UL, OL or DL

- e.g.: forums portlet

In General

3. Use markup and style sheets and do so properly

[P2] 3.7: *Mark up quotations. Do not use quotation markup for formatting effects such as indentation.*

- BLOCKQUOTE and Q (inline): semantic markup!!

```
<BLOCKQUOTE cite="http://...">
   <p>Cited text...</p>
</BLOCKQUOTE>

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

In General

4. Clarify natural language usage

[P1] 4.1: *Clearly identify changes in the natural language of a document's text and any text equivalents (e.g., captions).*

This is about the “lang” attribute.

- Specify the lang of the document <HTML lang=””>

- Markup text that appears in a different language (see Open Issues regarding acs-lang)

In General

6. Ensure that pages featuring new technologies transform gracefully

[P1] 6.1: *Organize documents so they may be read without style sheets. For example, when an HTML document is rendered without associated style sheets, it must still be possible to read the document.*

- Webdev: CSS → Disable Styles → All Styles
- See http://www.csszengarden.com again :)

[P1] 6.2: *Ensure that equivalents for dynamic content are updated when the dynamic content changes.*

[P2] 6.5: *Ensure that dynamic content is accessible or provide an alternative presentation or page.*

In General

7. Ensure user control of time-sensitive content changes

[P1] 7.1: *Until user agents allow users to control flickering, avoid causing the screen to flicker.*

[P2] 7.2: *Until user agents allow users to control blinking, avoid causing content to blink (i.e., change presentation at a regular rate, such as turning on and off).*

[P2] 7.4: *Until user agents provide the ability to stop the refresh, do not create periodically auto-refreshing pages.*

[P2] 7.5: *Until user agents provide the ability to stop auto-redirect, do not use markup to redirect pages automatically. Instead, configure the server to perform redirects.*

In General

10. Use interim solutions

[P2] 10.1: *Until user agents allow users to turn off spawned windows, do not cause pop-ups or other windows to appear and do not change the current window without informing the user.*

- don't popup windows
- don't resize windows

In General

11. Use W3C technologies and guidelines

[P2] 11.1: *Use W3C technologies when they are available and appropriate for a task and use the latest versions when supported.*

- HTML 4.01

- XHTML 1.1 ?

[P2] 11.2: *Avoid deprecated features of W3C technologies.*

- Use strict DTDs rather than transitional ones

- See deprecated elements list

- Webdev: Tools → Validate **local** HTML

In General

12. Provide context and orientation information

[P2] 12.3: *Divide large blocks of information into more manageable groups where natural and appropriate.*

- use FIELDSET to group form elements
- group table rows and cols with THEAD, TBODY...
- nest list with UL, OL, DL
- break text into paragraph P
- group related links with MAP

In General

13. Provide clear navigation mechanisms

[P2] 13.1: *Clearly identify the target of each link.*

- e.g.: manage memberships
- link text should be **meaningful**, if not use the “title” attribute
- Webdev: Information → Display title attributes
- `template::list::element::create`: use the link_html property `{title “more descriptive” class “button”}`

In General

13. Provide clear navigation mechanisms

[P2] 13.2: *Provide metadata to add semantic information to pages and sites.*

- content type and charset: very important
- keywords and author: when there's content and it can be read by the public or to index the site
- use `template::head` API to add metadata
- Webdev: Information → View meta tags information

In General

13. Provide clear navigation mechanisms

[P2] 13.3: *Provide information about the general layout of a site (e.g., a site map or table of contents).*

- [Added in .LRN 2.4.1]
- OpenACS one might need to be improved

In General

13. Provide clear navigation mechanisms

[P2] 13.4: *Use navigation mechanisms in a consistent manner.*

- Consistent style of presentation (e.g. “button” style for actions)
- Mechanisms to skip navigation
- 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
  - e.g.: Manage memberships or control panel

In General

14. Ensure that documents are clear and simple

[P1] 14.1: *Use the clearest and simplest language appropriate for a site's content.*

- Avoid jargon (e.g. subsite, portlet, applet, etc.)
- Use a consistent vocabulary across the site
- Use concepts that can be understood by everyone (e.g.: “minimize…” means nothing to the blind)
- Localization: use existing `acs-kernel.common_*` keys for common objects and actions

In General

14. Ensure that documents are clear and simple

[P1] 14.1: *Use the clearest and simplest language appropriate for a site's content.*

- Localize numbers, dates and times
- Use long format:

```
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

If you use Images and Maps

1. Provide equivalent alternatives to auditory and visual content.

9. Design for device-independence.

- **[P1] 1.2**: *Provide redundant text links for each active region of a server-side image map.*
- **[P1] 9.1**: *Provide client-side image maps instead of server-side image maps except where the regions cannot be defined with an available geometric shape.*

If you use Tables for data

5. Create tables that transform gracefully

- **[P1] 5.1**: *For data tables, identify row and column headers.*
- **[P1] 5.2**: *For data tables that have two or more logical levels of row or column headers, use markup to associate data cells and header cells.*

- Use list builder, it will output a well-formed table
- Don't use display_template (e.g. weblogger)
- Don't use TH nor TD for presentation purpose
- Relate TD with their corresponding headers: very important for screen-readers
- webdev: Outline → Outline Tables → Table Cells

If you use Tables for layout

5. Create tables that transform gracefully

- **[P2] 5.3**: *Do not use tables for layout unless the table makes sense when linearized. Otherwise, if the table does not make sense, provide an alternative equivalent (which may be a linearized version).*
- **[P2] 5.4**: *If a table is used for layout, do not use any structural markup for the purpose of visual formatting.*

- See structural markup elements
- See Calendar as an example
- webdev: Outline → Outline Tables → Table Cells

If you use Frames

12. Provide context and orientation information

- **[P1] 12.1**: *Title each frame to facilitate frame identification and navigation.*
- **[P2] 12.2**: *Describe the purpose of frames and how frames relate to each other if it is not obvious by frame titles alone.*

- <frame src=”...” title=”...” longdesc=”desc.html#thisframe”>

If you use Forms

10. Use interim solutions

12. Provide context and orientation information.

- **[P2] 10.2**: *Until user agents support explicit associations between labels and form controls, for all form controls with implicitly associated labels, ensure that the label is properly positioned.*
- **[P2] 12.4**: *Associate labels explicitly with their controls.*

- Use form builder!
- Use labels and place them next to the form field
- Use implicit + explicit association:

  <label for=”nameId”> Name <input id=”nameId”...> </label>

If you use Applets and Scripts - 1/2

6. Ensure that pages featuring new technologies transform gracefully.

7. Ensure user control of time-sensitive content changes

- **[P1] 6.3**: *Ensure that pages are usable when scripts, applets, or other programmatic objects are turned off or not supported. If this is not possible, provide equivalent information on an alternative accessible page.*
- **[P2] 6.4**: *For scripts and applets, ensure that event handlers are input device-independent.*
- **[P2] 7.3**: *Until user agents allow users to freeze moving content, avoid movement in pages.*

If you use Applets and Scripts - 2/2

8. Ensure direct accessibility of embedded user interfaces.

9. Design for device-independence

- **[P2] 8.1**: *Make programmatic elements such as scripts and applets directly accessible or compatible with assistive technologies*
- **[P2] 9.2**: *Ensure that any element that has its own interface can be operated in a device-independent manner.*
- **[P2] 9.3**: *For scripts, specify logical event handlers rather than device-dependent event handlers.*

Scripts and Applets

- **Applets and scripts:**
  - **avoid motion** or provide the user with options to stop it
  - provide **text equivalent**
  - page should work when js disabled (e.g. forums)
- **Events:**
  - Use "onmousedown" with "onkeydown".
  - Use "onmouseup" with "onkeyup"
  - Use "onclick" with "onkeypress"
  - Don't use “ondblclick”: no keyboard equiv.

If you use Multimedia

1. Provide equivalent alternatives to auditory and visual content

- **[P1] 1.3**: *Until user agents can automatically read aloud the text equivalent of a visual track, provide an auditory description of the important information of the visual track of a multimedia presentation.*
- **[P1] 1.4**: *For any time-based multimedia presentation (e.g., a movie or animation), synchronize equivalent alternatives (e.g., captions or auditory descriptions of the visual track) with the presentation.*

If all else fails

11. Use W3C technologies and guidelines

- **[P1] 11.4**: *If, after best efforts, you cannot create an accessible page, provide a link to an alternative page that uses W3C technologies, is accessible, has equivalent information (or functionality), and is updated as often as the inaccessible (original) page.*

Open Issues

- How to guarantee doc structure when using includelets
- web2.0 technologies: Ajax => ARIA (xhtml 1.1 → IE8...)
- List template: aggregate function output
- Markup change in natural language (acs-lang)
- SCORM and IMSLD players
- Convention for IDs
- …

Introducing WCAG 2.0

WCAG 2.0

The near future

Web Content Accessibility Guidelines (WCAG) 2.0 was published as a W3C ***Proposed Recommendation*** *on* ***3 November 2008****. This means that the technical material of WCAG 2.0 is complete and it has been implemented in real sites. The next stage is the* ***final publication****, which is expected in* ***December 2008****.*

And now what?

WCAG 2.0 References

- The URL: http://www.w3.org/TR/WCAG20/
- Guide to understand WCAG 2.0: http://www.w3.org/TR/2008/WD-UNDERSTANDING-WCAG
- Techniques for WCAG 2.0: http://www.w3.org/TR/2008/WD-WCAG20-TECHS-2008110

WCAG 2.0 in a nutshell

- **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

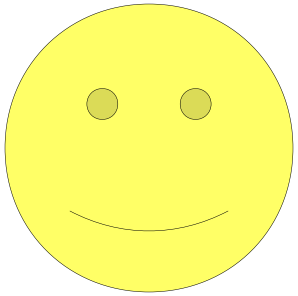
WCAG 2.0 The 4 principles

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

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

The End



Thanks Everybody !