Web-based Help Systems that Support Accessibility

Ian Jacobs

This presentation available at:
`http://www.w3.org/2001/Talks/winwriters-20010305`

Slides, but not demo, available in a single HTML file and six-to-a-page in PDF.

Summary

- I: Background: WAI, rationale for accessibility
- II: Accessible help content
- III: Fix this site!
- IV: Accessible user interface
- V: Tools

I-a: The Web Accessibility Initiative (WAI)

- Guidelines
- Technical review
- Evaluation, repair, and transformation tools
- Education and outreach, training

Check out WAI resources.

I-b: Who benefits?

- Visual disabilities (blindness, low vision, color deficiencies)
- Hearing disabilities (deafness, hard of hearing)
- Physical and speech disabilities
- Cognitive, learning, and neurological disabilities (dyslexia, memory impairments, photo-sensitivity, etc.)

"How People with Disabilities Use the Web" explains characteristics of disabilities, provides scenarios, explains assistive technologies.

For statistics (U.S.), refer to "Falling through the Net: Toward Digital Inclusion" (from NITA)

I-c: Cross-disability strategies

Challenge of producing guidelines to address different needs. Some users:

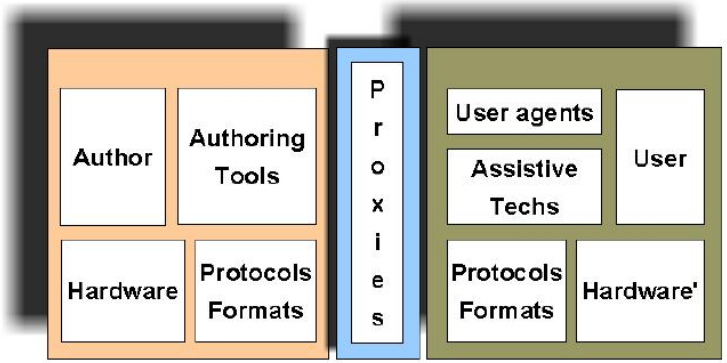
- Need audio and synthesized speech;
- Need images, video, and graphics;
- Require time-independent control (e.g., those using serial access or with physical disabilities);
- Require space-independent control (e.g., can't use the mouse to navigate in two dimensions);
- May have more than one disability.

Note: Plain text help content is **not** the best solution.

I-d: Accessibility guidelines model

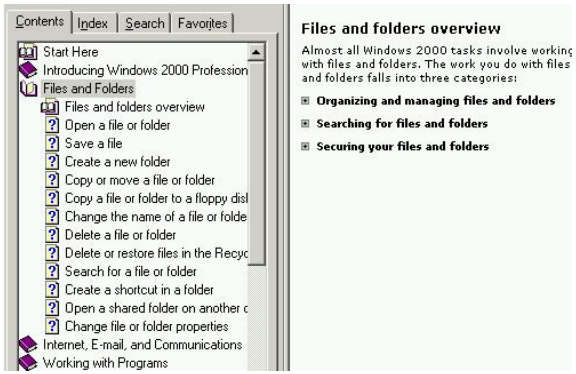
- Players: protocols and formats, authors, authoring tools, user agents.
- Each player has responsibilities to meet needs of users with disabilities.
- Some repair required due to errors and inevitable limitations.

I-e: Accessibility guidelines model diagram



I-f: Example of who does what

User need: Users need to know what level they are at in a nested list.



I-g: Example of who does what (continued)

- Format: Support nested lists. E.g., HTML `ul` lists.
- Help author: Encode table of contents using nested `ul` elements (e.g., not just using different font sizes and indentation).
- Authoring tool: Generate proper structured markup, provide an accessible UI and documentation.
- Help viewer: Display nested structure visually, communicate with assistive technologies (e.g., through the DOM and Microsoft Active Accessibility (MSAA) to allow specialized rendering.

In documentation, explain how end users can promote accessibility (e.g., how to use an authoring tool to create accessible content).

I-i: Why accessible help content?

- Help is particularly important to users with disabilities (e.g., who may not be able to use the "intuitive" graphical user interface).
- Increased market share.
- Legal requirements in many countries. Refer to policies relating to Web Accessibility.

Excerpt from Section 508 final rule, effective 20 February 2001 in the US:

Section 508 requires that when Federal agencies develop, procure, maintain, or use electronic and information technology, they shall ensure that the electronic and information technology allows Federal employees with disabilities to have access to and use of information and data that is comparable to the access to and use of information and data by Federal employees who are not individuals with disabilities, unless an undue burden would be imposed on the agency.

II-a: Accessibility topics to include in help content

For example, in the help content for the hypothetical "CoolEditor":

- Describe CoolEditor's user interface features that are covered by the User Agent Accessibility Guidelines (UAAG)1.0.
- Document the default keyboard shortcuts of the CoolEditor. E.g., keyboard shortcuts for Windows Help Viewer.
- Explain the configuration and control features, notably those that benefit accessibility.
- Link help topics to step-by-step processes (e.g., how to increase the font size).
- Integrate documentation of accessibility features.
- Provide a centralized view of accessibility features (e.g., features of Amaya and Opera, Office 2000).
- Include accessibility keywords in the index.

II-b: Guidelines for accessible help content

Create content that conforms to the Web Content Accessibility Guidelines (WCAG) 1.0. Refer also to Microsoft documentation on making Help more accessible.

For instance:

- Include accessible graphics and multimedia (text equivalents, auditory descriptions, text transcripts, resizable).
- Use simple, straightforward language. Front-load paragraphs.
- Design content that works when scripts have been turned off (e.g., for security reasons).

II-c: Guidelines for accessible help content

- Create a library of accessible templates.
- Ensure that examples illustrate accessibility features and conform to WCAG 1.0.
- Ensure that language in help content is accessibility aware (e.g., device-independent).
- Document changes between software releases if those change affect accessibility.
- Ask users with disabilities to test the accessibility of your content.

II-d: Guidelines for accessible help content

Use formats that support accessibility.

- W3C specifications undergo review for accessibility:
  - Techniques documents for WCAG 1.0
  - *Accessibility Features of SMIL*
  - *Accessibility Features of SVG*
  - *Accessibility Features of CSS*
  - QuickTips.
- For formats other than XML, HTML look for resources at vendor Web sites.

II-e: Example of a glossary

Divide content into sensible pieces (here, a-to-z).

III-a: Fix this site!

While fictional, this page accurately illustrates common practices.

- Inaccessible page example
- Lynx view

III-b Fix 1: Ensure scripts are accessible

- Content must not rely on scripts alone to provide information.
- Use markup instead of scripts when possible.
- Don't prevent access through browser-sniffing.

Make other scripts accessible:

```
<a href="javascript:popout('miles')"
         onmouseover="highlight()">
     <img src="blood.gif"
         alt="earn miles - donate blood">
     </a>
```

Becomes

```
<a href="miles" target="_blank"
         onfocus="highlight()"
    onmouseover="highlight()">
     <img src="blood.gif"
         alt="earn miles - donate blood">
  </a>
```

Related WCAG 1.0 checkpoitns: 6.3 and 6.4.

- Improved example
- Lynx view

III-c Fix 2: Provide equivalent alternatives

Since text can be rendered visually, as speech, or as braille it helps many users.

In HTML, provide short functional equivalent text with the "alt" attribute:

```
<img ... alt="Navigation Bar..." >
<area .. alt="More Articles" >
```

Provide content for applets:

```
<applet code="Lake.class">
    <param name="image" content="lh.gif">
    <img src="lh.gif" alt="Latest News">   </applet>
```

Provide long descriptions for complex content (charts, graphs, etc.)

Related WCAG 1.0 checkpoints: 1.1 and 6.3. ATAG 1.0 checkpoints: 3.1, 3.3, 3.5.

- Improved example
- Lynx view

III-d Fix 3: Make frames accessible

Warning: It's easy to use frames in a way that causes usability problems. Advance with caution.

To provide navigation alternatives, assign titles to frames and add a "noframes" alternative:

```
<frame ... name="TOC"
           title="Table of Contents" >
 <frame ... name="Home"
           title="Home Page" >
  <noframes>
     <body>
     <p><a href="a">
        <img src="vnav.gif" border=0 usemap="#map"
             alt="Navigation Bar..."></a>
     ...
     </body>
  </noframes>
```

Related WCAG 1.0 checkpoint: 12.1.

- Improved example
- Lynx view

III-e Fix 4: Specify colors appropriately

- Use color, but don't rely on color alone (e.g., don't say "click on the red bar").
- Specify all colours for a color scheme. If only a background is given it might be the same as user-specified defaults for a foreground.
- Ensure sufficient contrast (for readability) when specifying colors (including in images).

So, in HTML:

```
<body background=bg.gif>
```

becomes

```
<body background="bg.gif"
      bgcolor="#8e0851" text="white"
      link="3fa" vlink="5f5">
```

Related WCAG 1.0 checkpoints: 2.2.

- Improved example
- Lynx view

III-f Fix 5: Use meaningful link text

To provide information about link targets to users who surf link text only, write:

```
<a href="a">
    Coping With Asthma's Relentless Attacks</a>
```

instead of:

```
Coping With Asthma's Relentless
  Attacks:<a href="a">Click Here</a>
```

Related WCAG 1.0 checkpoint: 13.1.

- Improved example
- Lynx view

III-g Fix 6: Add structure

To convey a logical model other than visually, use markup correctly. For example:

```
  <b>9 May 1999</b> <img src="l1.gif">
```

becomes:

```
<h1>9 May 1999
     <img src="l1.gif" alt="Latest News"></h1>
```

Related WCAG 1.0 checkpoints: 3.5 and 3.6. ATAG 1.0 checkpoint: 3.2.

- Improved example
- Lynx view

III-h Fix 7: Use style sheets

- Use style sheets instead of background images for color.
- Replace text in images with real text.
- Use a style sheet instead of FONT element / inline styles

Related WCAG 1.0 checkpoints: 3.1 and 3.3.

- Accessible example
- Lynx view
- Inaccessible page example

IV-a: Guidelines for an accessible user interface

For more information, refer to the User Agent Accessibility Guidelines (UAAG)1.0.

Design for space independence:

- Don't rely on two-dimensional frame layout alone to convey relationships.
- Provide keyboard shortcuts so users don't have to navigate through space.
- Don't rely on absolute font sizes and window sizes.

IV-b: Guidelines for an accessible user interface

Design for device independence:

- Don't rely on mouse-driven popups alone to convey information.
- Ensure that the user interface can be operated entirely with the keyboard.

Design for time independence:

- Don't rely on user input within a time limit.

IV-c: Guidelines for an accessible user interface

Allow configuration, control, and override:

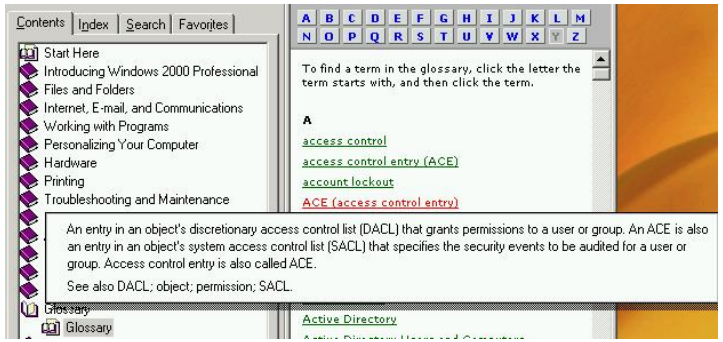
- Colors
- Font sizes
- Multimedia presentation rates

Follow operating environment conventions:

- APIs
- User interface controls
- Documentation and installation

IV-d: Example of problematic interface

Selecting a glossary entry causes window to pop-up with definition. This window is initially part of the window list, conventionally reached using Alt-tab. But when you press any key, the window closes.



V-a: What tools do you use to author help?

According to the Winwriters Skills and Technologies Survey on Tools (2000), top tools in use are:

1. Microsoft Word: "Save as HTML".
2. Notepad: Create HTML (or other formats) by hand.
3. RoboHelp HTML: Interactive Help authoring tool.
4. Paint Shop Pro: Accessible graphics

V-b: Guidelines for accessible help authoring tools

Design authoring tools that conform to the Authoring Tool Accessibility Guidelines 1.0.

- Ensure that templates are accessible.
- Generate valid markup.
- Prompt the user to include equivalents (e.g., text for images).
- Ensure that prepackaged content conforms to WCAG 1.0.

V-c: Guidelines for accessible help authoring tools

- Check for and inform the author of accessibility problems.
- Assist in correcting accessibility problems.
- Ensure that users with disabilities can use the authoring tool.

V-d: Validation and repair tools

- W3C HTML/XML validator
- W3C CSS validator
- HTML Tidy
- WAI list of evaluation and repair tools (e.g., Bobby, the Wave, Lynx-me, WCAG 1.0 checklist, etc.)

Questions and discussion

- Authoring scenarios you think might be difficult to make accessible?
- Your experience using tools to author accessible help?