# Web Content Accessibility Guidelines

Version 1.0

Edited by Maurice Chang

Table of Contents

[Web Accessibility Guidelines i](#_Toc166054309)

[Introduction 1](#_Toc166054310)

[1. Accessible Design 3](#_Toc166054311)

[a. Universal design 3](#_Toc166054312)

[b. Accessible design and WCAG 1.0 3](#_Toc166054313)

[c. Interface design and navigation 3](#_Toc166054314)

[d. Color and contrast 4](#_Toc166054315)

[e. Page structure 4](#_Toc166054316)

[2. Accessible Content 5](#_Toc166054317)

[a. Writing for accessibility 5](#_Toc166054318)

[b. Accessible text links 5](#_Toc166054319)

[c. Making images accessible for the visually disabled 8](#_Toc166054320)

[d. Accessible table content 10](#_Toc166054321)

[e. Accessible form content 10](#_Toc166054322)

[3. Accessible Content – HTML Markup 11](#_Toc166054323)

[a. Alternate Text Equivalents < … alt=””> <longdesc> 11](#_Toc166054324)

[b. TITLE and ALT attributes and child browser windows 11](#_Toc166054325)

[c. Abbreviations and acronyms <abbr> <acronym> 12](#_Toc166054326)

[d. Languages < … lang=”xx”> 12](#_Toc166054327)

[e. Quotes < blockquote> 12](#_Toc166054328)

[f. <strong> and <em> 13](#_Toc166054329)

[4. Access Alternatives 14](#_Toc166054330)

[a. ACCESSKEY attribute 14](#_Toc166054331)

[b. TABINDEX attribute 16](#_Toc166054332)

[c. Link bypassing and grouping 16](#_Toc166054333)

[5. Accessible Forms 17](#_Toc166054335)

[a. Form prompts and controls placement 17](#_Toc166054336)

[b. LABEL tag 17](#_Toc166054337)

[c. FIELDSET and LEGEND tags 18](#_Toc166054338)

[6. Accessible Tables 20](#_Toc166054339)

[a. Table Summaries <table … summary=”text here”> 20](#_Toc166054340)

[b. CAPTION Element <caption> 20](#_Toc166054341)

[c. id and headers Attributes 21](#_Toc166054342)

[<th id=”whatever”> 21](#_Toc166054343)

[<td headers=”whatever”> 21](#_Toc166054344)

[7. Accessibility and Cascading Style Sheets 23](#_Toc166054345)

[a. Fonts and positioning 23](#_Toc166054346)

[b. Heading tags 23](#_Toc166054347)

[8. Scripting 25](#_Toc166054349)

[a. Visible effects 25](#_Toc166054350)

[b. Child browser windows 25](#_Toc166054351)

[c. Event triggers 26](#_Toc166054352)

[d. Expanding menus 26](#_Toc166054353)

[e. Pulldown menus 27](#_Toc166054354)

[9. Accessibility of PDF Documents 28](#_Toc166054355)

[a. Testing PDFs for accessibility 28](#_Toc166054356)

[b. Creating accessible PDFs 29](#_Toc166054357)

[c. Modifying PDFs for accessibility 29](#_Toc166054358)

[d. Tips for optimizing PDFs pre- and post-conversion 31](#_Toc166054359)

[10. Flash Accessibility 33](#_Toc166054360)

[a. Adding accessibility to Flash content 33](#_Toc166054361)

[b. Making Flash movies accessible to screen readers 33](#_Toc166054362)

[c. Assigning text equivalents to visual elements 34](#_Toc166054363)

[d. Controlling read order 35](#_Toc166054364)

[e. Captioning audio content 36](#_Toc166054365)

[f. Working with changing content 37](#_Toc166054366)

[g. Flash components and accessibility 38](#_Toc166054367)

[h. Other Flash accessibility techniques 39](#_Toc166054368)

[i. Flash extensibility 39](#_Toc166054369)

[j. Testing and validating Flash content 40](#_Toc166054370)

[k. Flash accessibility limitations and benefits 40](#_Toc166054371)

[11. Accessibility Testing and Validation 42](#_Toc166054372)

[a. Testing and validation tools 42](#_Toc166054373)

[b. InFocus recommendations 43](#_Toc166054374)

[c. InFocus exceptions 44](#_Toc166054375)

[d. InFocus validation limitations 45](#_Toc166054376)

[e. Testing with JAWS 45](#_Toc166054377)

[f. Checklist for JAWS accessibility testing 45](#_Toc166054378)

[g. Basic JAWS shortcuts 48](#_Toc166054379)

[h. JAWS miscellanea 48](#_Toc166054380)

[Appendix A: Resources 49](#_Toc166054381)

# Introduction

These *Web Accessibility Guidelines* have been adopted for making web sites accessible to people with disabilities. They provide specific detail on the standards that should be used to achieve web accessibility, and they should be followed by all web designers, web developers, and content authors who provide or create materials for internet and intranet web sites. These guidelines are based on, and built upon the WCAG and Section 508. In many cases they are identical, but in some cases they may be explained differently to reflect the unique nature of internet applications.

***Web Accessibility***

Web accessibility is the ability of a user with a disability to use a web site and access information. It can be thought of as the usability of a web site to a person with a disability. Accessible Web content benefits a variety of people: the blind, the elderly, those with repetitive stress injury, or someone with a color deficiency. Web accessibility is generally concerned with ensuring that pages are designed in such a way that they interact appropriately with assistive technology devices.

***Types of Disabilities***

Though there are many different types of disabilities. We are primarily concerned with the following:

* **Dexterity Difficulties and Impairments**   
  Individuals with dexterity difficulties find it difficult to use a standard keyboard or mouse. Examples of dexterity impairments are carpal tunnel, arthritis, stroke, multiple sclerosis, and repetitive stress injury, among others.
* **Visual Difficulties and Impairments**   
  Visual difficulties and impairments include low vision, color blindness, and blindness. There are many options for individuals with vision difficulties to modify the computer displays and appearance so it is more legible, or receive information through sound or touch. Those who are blind cannot use a computer monitor and but have the option to receive information from their computers through hearing or touch offered through screen readers and Braille displays.
* **Hearing Difficulties and Impairments**   
  Hearing difficulties and impairments encompass a wide range of conditions—from slight hearing loss to deafness. There are no specific assistive technology products for individuals who have hearing impairments because they can interact with computers by receiving information visually or adjust sounds and volume options to meet their hearing needs.

***Assistive Technologies***

Assistive technology is a product, software or hardware, designed to provide the disabled with the means to use computers or computer programs. Here is a sampling of these technologies:

* **Screen enlargers** work like a magnifying glass. They enlarge a portion of the screen as the user moves the focus.
* **Screen readers** are software programs that “speak” everything on the screen.
* **Speech recognition systems**, allow people to give commands and enter data using their voices rather than a mouse or keyboard.
* **Refreshable Braille displays and Braille embossers** provide tactile output of information represented on the computer screen.
* **Alternative input devices** allow people to control their computers through means other than a keyboard or mouse (examples include electronic pointing devices, sip-and-puff systems, wands and sticks, joysticks and trackballs).

***AKQA and Web Accessibility***

The Web is an opportunity for people with disabilities. Some disabilities limit the type of work a person can do, but an accessible Web expands opportunities for communication, interaction, and employment for people with disabilities. The Web is also increasingly used for government services, education, commerce, news, and health care, among others. In some cases, the Web is replacing traditional resources. Therefore it is essential that the Web be accessible in order to **provide equal access and equal opportunity to people with disabilitie**s.

There are also legal considerations. Section 508 of the Rehabilitation Act of 1998 (effective June 2001) requires that Federal employees with disabilities have equal access to information and data when U.S. Federal agencies procure or use electronic and information technology. Section 508 requires that the U.S. Federal government, in a competitive bid, procure the product which is most compliant with the Section 508 standards, i.e. accessible to people with disabilities.Further, any products previously purchased are subject to review and replacement with an alternative product that carries a higher accessibility rating.

AKQA is committed to Web site accessibility on the web sites it helps build. Providing online access to people with disabilities gives more people access to the products, services, and information. Also, by making our sites accessible to people with disabilities, we make them highly usable and accessible to everyone else as well.

# 1. Accessible Design

## a. Universal design

Designing for people with disabilities employs the concept of “universal design.” Universal design aims to make products and environments usable by all people, to the greatest extent possible, without adaptation or specialized design.

Universal web design standards benefit not only those with disabilities, but everyone using a site. It considers a wide spectrum of users, including those who:

* May have a text-only browser, a slow internet connection, or a small screen. Given the increasing array of non-PC web browsing devices such as cell phones, more users may be using small, text-only screens.
* May be in a situation where their eyes, ears, or hands are busy or interfered with (for example, working in a noisy environment).

## b. Accessible design and WCAG 1.0

The W3C *Web Content Accessibility Guidelines* 1.0 (WCAG 1.0, <http://www.w3.org/TR/WCAG10/>), which are used as the foundation for the *AKQA Web Accessibility Guidelines*, break down accessible design practices into 14 basic checkpoints:

* Provide equivalent alternatives to auditory and visual content.
* Don't rely on color alone for visual cues.
* Use markup and style sheets, and do so properly.
* Clarify natural language usage.
* Create tables that transform gracefully.
* Ensure that pages featuring new technologies transform gracefully.
* Ensure user control of time-sensitive content changes.
* Ensure direct accessibility of embedded user interfaces.
* Design for device-independence.
* Use interim solutions.
* Use W3C technologies and guidelines (where applicable and possible).
* Provide context and orientation information.
* Provide clear navigation mechanisms.
* Ensure that documents are clear and simple.

The *AKQA Web Accessibility Guidelines* attempts to incorporate these checkpoints to the extent they are sound and practicable, considering relevant circumstances.

## c. Interface design and navigation

Well-designed pages that use a simple, intuitive interface and consistent navigation benefit everyone. However, for a visually disabled user who can only view a web page linearly and has to memorize much of its layout and navigation, consistently designed pages can make the difference between a workable or arduous web experience.

The needs of people with visual disabilities may seem to conflict with those of the cognitively impaired. However, use of graphics does not hurt the visually disabled if alternative text is given (when appropriate), and alternative text will not impede those with cognitive problems.

## d. Color and contrast

Estimates say that about eight percent of men and 0.5 percent of women have some form of color blindness – potentially around 12 percent of a web site's viewers. Most color-blind people can see black and white accurately, and they can see all shades of yellow and blue. Less color-blind safe colors can also be used if placed against an appropriately colored background. As most color-blind people have red-green color blindness, avoid using these two colors next to each other. **Avoid Red and Green whenever possible for hyperlinks and directional images and instructions**.

* A simple test is to print out your mockups in black and white.
* If you are unsure, check your page by installing [Vischeck Photoshop Plug-In](http://vischeck.com/) for Windows, a plug-in which simulates color blindness.
* Be wary of using a combination of CSS and HTML to control colors. Text may become unreadable with style sheets turned off. For instance, if a white font color is specified with a black background color in the BODY tag, with style sheets turned off the text would be black on a black background. In the example below, a user with style sheets disabled would see links reverted to default blue on a blue background.

A {font-family: Arial, Helvetica, sans-serif; color: #900;}

<td bgcolor="#00c"><a href=[http://www.AKQA.com](http://www.wellsfargo.com)>Visit AKQA Online</a></td>

* Whenever possible, do not remove the underlining from links, the underlining is an important, standard visual clue. The exceptions would be navigation menus and other lists of links. But for most of the modern designs that use the stylesheets, the underlines of the links have been removed unless you mouse over. So, depending on the types of audience you want to accommodate, this is highly debatable.
* Use other cues beside color to define clickable areas of image maps.
* Use [different colors for visited and unvisited links](http://www.useit.com/alertbox/20040503.html).

## e. Page structure

When composing HTML, authors should think about what the content *means* instead of how it should *look*. Using correct page structure adds meaning to the content. By using the heading element (H1- H6), paragraph element (P) list item element (LI) and blockquote element (BLOCKQUOTE), you provide logical structural information about the document to blind users.

Use HTML to mark up the structure of a web page, not for presentation or appearance. In particular:

* Heading level elements (<h1> through <h6>) should only be used for content headings and should be used hierarchically. Some users skim through a document by navigating its headings, and heading tags help non-sighted users understand content hierarchy. Thus, headings should not be used for presentation effects for other page elements.
* Use the STRONG and EM tags instead of B and I tags. The former are content-based style tags, whereas the latter are physical style tags that solely control how text appears on a page.
* Do not use empty paragraphs (<p>) to put a vertical space between consecutive blocks of text. Use CSS to achieve spacing between elements
* Do not use the <table> element to achieve columnar alignment of material that is not inherently tabular.

# 2. Accessible Content

This section focuses on content accessibility. For more information about the HTML markup mentioned here, see Chapter 3, Accessible Content – HTML Markup.

## a. Writing for accessibility

While good web writing – for example, writing for scannability and chunking copy into smaller units – benefits everyone, it is even more essential for people who have reading or cognitive disabilities.

Consider the following suggestions (from the W3C, Core Techniques for Web Content Accessibility Guidelines at [http://www.w3.org/TR/WCAG10-CORE-TECHS/ - comprehension](http://www.w3.org/TR/WCAG10-CORE-TECHS/#comprehension)):

* Use clear, accurate headings and link descriptions. Well-written headings allow users to scan a page quickly. And, as some users may browse by tabbing from link to link and listening only to link text, links should make sense when read out of context.
* State the topic of a sentence or paragraph at its beginning ("front-loading") to help people who skim visually and who use speech synthesizers.
* Limit each paragraph to one main idea.
* Avoid idiomatic language, technical jargon, and specialized meanings of familiar words, unless defined within the document or the document is meant to provide technical information..
* Use commonly used words. For example, write "for" instead of "for the purpose of" or "use" instead of "utilize".
* Use the active rather than passive voice.
* Avoid complex sentence structure. Instead use simple sentence construction and short, concise sentences.

## b. Accessible text links

Link text should be clearly written and should concisely express the target of the link. Avoid generic phrases such as “click here” (an especially egregious phrase as some disabled users cannot use a mouse).

As link content may be accessed independently of surrounding content, the link text itself should clearly inform the user of the link’s target. For example, to more quickly find information on a web site, visually-disabled people using screen readers may use the tab key to find and read the links on a page. They may also use a “links list” feature to read a list of all the links.

A link such as "[Vischeck](http://vischeck.com/) simulates colorblind vision" is more concise, clear content for both visually disabled and sighted users than "[Click here](http://vischeck.com/) to learn about Vischeck, which simulates colorblind vision". The latter link is useless for non-sighted users tabbing through links or for a screen reader links list.

If for some reason the link cannot be written so that it is meaningful within itself, use the TITLE attribute to clarify the link for screen readers users. The TITLE attribute is used within the link tag – for example: <a href=“…” title=“Learn More about Online Brokerage”>Learn More</a>.

Screen reader users can be alerted that TITLE attributes may be used. For example, many websites that are designed to accommodate the blind people have an invisible link at the top of the page, alerting JAWS (a screen reader, <http://www.freedomscientific.com/fs_products/software_jaws.asp>) users that they need to change a JAWS link setting to hear the TITLE attribute by default.

For the link situations below, either rewrite link for clarity or add a TITLE attribute:

|  | **Link Text** | **Option 1 –**  **Rewrite link(s)** | **Option 2 – Add**  **TITLE attribute** | **Comments** |
| --- | --- | --- | --- | --- |
| * **Unclear links** | * Apply Now” * Get Started | Apply Now for a membership  Get started with the tutorial | title=“Apply now for a membership.”  title=“Get started with the tutorial” |  |
| * **Links opening a new browser window** | * Contact Us | This link needs a TITLE attribute, unless the new browser information is included in the link:  Contact Us (This link will open a new browser window) | title=“Contact Us. This link will open a new browser window.” |  |
| * **Links opening to PDFs or Flash movies** | * 2003 Annual Report * Online Banking Through QuickBooks® Demo | 2003 Annual Report (PDF)  Online Banking Through QuickBooks® Demo (Flash) | Do not use TITLE attribute. Media information should be included in the link text. | To let users know they are going to a PDF or Flash movie, the link content should include “(PDF)” or “(Flash)” at the end of the link content.  Also inform user that to access the file a plug-in or application is needed, and provide a link to download it – for example, “You need Adobe® Reader® to read PDF files. Download Adobe Reader for free.” |
| * **Repetitive links:** Multiple links on a page share **same link text** and go to **same target**. | * Three links, both with text “Learn more” (These links - and other commonly used repetitive links such as “Apply Now” – are also unclear.) | Write the same text for all of the links – for example, Learn More about Online Banking | For all of the links:  title=“Learn More about Online Banking” | Link text or TITLE attribute should be the same for all of the links.  A sighted user can see the context for a series of unclear links such as “Learn More.” However, for “Learn More” links without TITLE attributes, a screen reader user using a links list command will hear: “Learn More. Learn More. Learn More….” |
| * **Unique links**: Multiple links on a page share **same link text** but go to **different targets**. | * Three links on a page say “More >>” and go to three different targets. (This is also an example of unclear links.) | Write different text for each link.   * More banking accounts and services * More loans and credit options * More investing accounts and services | * title=“More banking accounts and services” * title=“More loans and credit options” * title=“More investing accounts and services” | Link text or TITLE attributes should be unique for each link, reflecting the page linked to. |
| * **Redundant links**: Multiple links on a page have **different text** but go to **same target**. | * Sign up for Bill Pay * Apply Now | Write the same text for both links:  Sign up for Bill Pay | For both links:  title=“Sign up for Bill Pay” | Link text or TITLE attribute should be the same for all links that go to the same target. |
| A combination of text and image links could result in repetitive, unique, or redundant links. The above rules still apply. Use TITLE and ALT attributes where appropriate for clarity. | | | | |

## c. Making images accessible for the visually disabled

Providing appropriate ALT attributes for images is one of the most important (and easiest) ways to make visual web content accessible to visually-impaired people who use screen readers.

|  | **Image** | **Sample**  **ALT attribute** | **Comments** |
| --- | --- | --- | --- |
| **ALT and Title attribute**  The ALT attribute should contain the most relevant information for the user. |  |  | Service marks, trademark symbols, and registration marks are not required in ALT text.  Both ALT and Title work in IE. ALT doesn’t work in Firefox. To be compatible with both browsers (and others), it’s recommended to use both attributes. |
| * **Same text** as shown on the image | * Help * Stay organized with free 24/7 access to online statements. Sign up today. | alt=“Help”  alt="Stay organized with free 24/7 access to online statements. Sign up today.” | Do not add “Click here” to a “call-to-action.” A screen reader informs the user they are on an image link by saying, “link graphic” before reading the ALT attribute – for example, “link graphic Stay organized….” |
| * **Expanded version of the image text** (where necessary to provide context) | * Learn More * Apply Now | alt=“Learn More about Online Banking”  alt=“Apply for a Business Secured MasterCard now.” | Adding the appropriate ALT attributes is particularly important for multiple graphical links on a page that all have the same name. (See Repetitive and Unique links, below.) |
| * **Different version of the image text** (where appropriate) |  |  | If the logo image is a link leading to the client’s home page, it’s more important to inform the user of the target of the link rather than that the image is the logo. |
| * **Links within image maps** | * someshop.com intranet image map | For the state of California:  alt=”California  (or  title=”California”) | An image map is an image containing one or more invisible regions or “hotspots” that are links to other web content.  For each AREA tag for the image map, provide either an ALT or TITLE attribute, and maybe **not both, depending on the browsers and devices**. To find out what attributes the browsers currently support, it’s best to run some tests and develop an encompassing strategy to cover all the popular browsers (IE, Firefox, Opera, Safari, etc.) |
| * **Meaningful non-text images** | * images in someshop.com Bus Photo Gallery | alt=”[short description of image]” | Short ALT descriptions of the image links in the Photo Gallery state the target of the link, as well as give context to the text following each image. (However, the ALTs could describe the image more clearly.) |
| * **Decorative images** or images that do not provide meaningful information to a non-sighted user should have an ALT attribute with a null value. | * “spacer” image | alt=“”  Do not put a space between the two quotes. | Avoid meaningless ALTs such as ALT=”ad” or ALT=”photo” for content images or ALT=”blue line” for decorative images. |
| * For graphical links that **open a new browser window**, use the ALT attribute to alert the screen reader user. | * Help | alt=“Help. This link will open a new browser window.” |  |
| * **Repetitive links:** Multiple links on a page share **same link text** and go to **same target**. | * Three images, both with text “Learn more about Online Banking” | For all images:  alt=“Learn More about Online Banking” | ALT attribute should be the same for all of the links. |
| * **Unique links**: Multiple links on a page share **same link text** but go to **different targets**. | * Three links on a page say “Learn More” and go to three different targets. | alt=“Learn More about Online Banking”  alt=“Learn More about Bill Pay”  alt=“Learn More about Checking and Savings” | ALT attributes should be unique for each link, reflecting the page linked to.  A sighted user can see the context for a series of links such as “Learn More.” However, if the ALTs for all these links are “Learn More,” a screen reader user using a link list command will hear: “Learn More. Learn More. Learn More….” |
| * **Redundant links**: Multiple links on a page have **different text** but go to the **same target**. | * Sign up for Bill Pay * Apply Now | For both links:  alt=“Sign up for Bill Pay” | ALT attribute should be the same for all links that go to the same target.. |
| A combination of text and image links could result in repetitive, unique, or redundant links. The above rules still apply. Use TITLE and ALT attributes where appropriate for clarity. | | | |

## d. Accessible table content

|  | **Comments** |
| --- | --- |
| **CAPTION element**  Short description of table’s content that appears above table. | Can serve as title of a table. |
| **SUMMARY attribute**  Summary of table’s purpose – not visible, but read by screen reader. | If a table’s title or CAPTION does not clearly describe the table’s content, include a SUMMARY attribute to further summarize the content for a screen reader user. |
| **Table Header (TH) elements and TH coding attributes**  Coding that tells screen reader what table columns or rows to read as headers for the table’s data. | Table headers should always be at the top row and/or left column of the table. This assures that a screen reader will read the table correctly and avoids unnecessary coding.  If you need to lay out the table differently, contact a developer versed in accessibility to make sure your table will be accessible. |
| See Chapter 6, Accessible Tables, for more information about table accessibility. | |

## e. Accessible form content

|  | **Comments** |
| --- | --- |
| **Form prompt content and placement**  The form prompt is the text that comes before or after a form control (the form element that the user interacts with). | When screen reader users fill out a form, they most likely will use “forms mode” to move through the form. Besides knowing what the form control is (which the screen reader will note), the user also needs to know what its prompt is. Thus, the content of the form prompts should be clear, succinct, and directly relevant to the control.  The location of a prompt in relation to its control helps assure correct association by a screen reader of a form’s control with its prompt. |
| See Chapter 5, Accessible Forms, for more information about form accessibility. | |

# 3. Accessible Content – HTML Markup

## 

This section focuses on HTML markup for accessible content.

## a. Alternate Text Equivalents < … alt=””> <longdesc>

**Applies to:** images, advertisements, charts, diagrams

**Why it’s important:** The visually impaired can't see the image. Unless you provide a text alternative that their assistive technology can read, the information will be lost to them.

**What to do:** Provide a text equivalent for every non-text element (using **alt** or **longdesc**). Writers will provide text equivalents for images, animated gifs, charts, graphs, or any other item that requires descriptive content. **longdesc** should be used for images that convey information larger than 1024 characters.

**Hints/Examples:**

* Advertisements using animated gif’s – writers will provide the copy
* Use "" for spacers
* Use "**\***" for graphical bullets
* Don’t use “>” as an alt text for breadcrumbs. A screen reader will read “greater than.”
* **longdesc** should be used for images that convey information beyond simply raw text, such as an organization chart or a data graph – writers will provide the copy
* Complex images that convey information – writers will provide the copy
* Line-art inserted into a page mainly as eye-candy, which conveys no information, does not need a text alternative.
* Although it is not currently supported by any major browser, both JAWS and IBM Home Page Reader support LONGDESC
* For purely decorative or spacer GIFs that do not give significant information, use the "null" value (ALT=""). A screen reader will ignore the image, and the user does not hear the useless description "image" (which a screen reader might recite if the ALT attribute is not included).

## b. TITLE and ALT attributes and child browser windows

As child browser windows can be disorienting to visually disabled users, do not change the current window without informing the user. Use the TITLE attribute with the A element, the ALT attribute with an image link, and ALT or TITLE (**not** both) for an imagmap link to inform the user of the child window.

| <a href="javascript: var w=window.open('smallwin.html','smallwin', 'width=400, height=350, status=yes, resizable=yes');" title=”Contact Us. This link will open a new browser window.”>Contact Us</a>  <area coords="166,6,224,20" href="javascript: var w=window.open('/security.html','smallwin', 'width=400, height=350, status=yes, resizable=yes');" alt="Online Security. This link will open a new browser window." tabindex="8" /> |
| --- |

Screen readers read either the link text or the TITLE attribute, not both. Thus, be sure to include the link content in the TITLE attribute.

## c. Abbreviations and acronyms <abbr> <acronym>

**Applies to**: abbreviations or acronyms

**Why it’s important**: Screen readers will read as an abbreviation or acronym.

**What to do:** Code all abbreviations or acronyms. Writers will indicate whether the acronym or abbreviation tag should be used. They will also provide the expansion of the abbreviation or acronym.

**Hints/Examples:**

* If it isn’t coded as an abbreviation, the abbreviation ACH would be read by a screen reader as “ach” (rhymes with “batch”)
* An acronym is usually formed by taking the first initials of a phrase or compounded word and using those initials to form a word that stands for something. Thus NATO is an acronym for North Atlantic Treaty Organization, and LASER is an acronym for Light Amplification by Stimulated Emission of Radiation. FBI, then, is not really an acronym for the Federal Bureau of Investigation; it’s an abbreviation. AIDS is an acronym; HIV is an abbreviation.
* The following sentence: “When in New York, be sure to visit MOMA. MOMA is located at 11 W. 53 St.” would be coded:

When in New York, be sure to visit,   
<acronym title=" Museum of Modern Art">MOMA</acronym>.  
MOMA is located at 11 <abbr title="West">W.</abbr> 53 <abbr title="Street">St.</abbr>

Unfortunately, JAWS 4.5??????????? (and possibly other screen readers and versions of JAWS) does not support ACRONYM. It reads the acronym, not the TITLE content.

## d. Languages < … lang=”xx”>

**Applies to**: words or phrases from another language

**Why it’s important**: Screen readers will use the correct pronunciation. (Imagine a computer reading *"Vos enfants sont très beaux. Ils sont adoptes?"* with English pronunciation rules.)

**What to do:** Use the **lang** attribute with the two-letter ISO designation for that language. Writers will inform you if a word or phrase should be coded with the **lang** attribute.

**Hints/Examples:**

* A promotional message for AKQA’s participation in a Cinco de Mayo celebration should code “Cinco de Mayo” as a Spanish phrase.

<span lang="fr">&#34;Vos enfants sont très beaux. Ils sont adoptes?&#34;</span>

**Resources:**

<http://lcweb.loc.gov/standards/iso639-2/englangn.html#st> – 2 digit ISO codes for languages

HERE, HERE

## e. Quotes < blockquote>

**Applies to:** longer quotations that span several lines

**Why it’s important:** Speech-based user agents will employ changes in voice characteristics when they read anything in the **<blockquote>** tags.

**What to do:** Use **<blockquote>** for quotations that span several lines. Never use **<blockquote>** to create formatting effects for content that isn’t a quote.

**Hints/Examples:**

* Do not use **<q>** tag. It has little support in browser readers and is not supported by IE
* “To krosk your enemies, to see dem driven before you, and to hear de lamentations of der vomen.”

Conan the Barbarian

<blockquote>&#34;To krosk your enemies, to see dem driven before you, and to hear de lamentations of der vomen.&#34;<blockquote>

Conan the Barbarian

## f. <strong> and <em>

**Applies to:** content that requires semantic (meaning) emphasis

**Why it’s important:** Speech-based user agents will employ changes in voice characteristics when they read anything in the **<strong>** and **<em>** tags

**What to do:** Use **<strong>** and **<em>** to add semantic meaning for content. Never use **<strong>** and **<em>** for presentation effects. Use CSS instead.

**Hints/Examples:**

* Though not deprecated, the use of **<b>** and **<i>** is discourage
* *<em> for emphasis.* <**strong> for stronger emphasis.**
* Do not use **<strong>** when a header **<h1>**, **<h2>**, etc. is more appropriate. Use CSS if you cannot use a header tag for a title.

**Resources:**

<http://www.w3.org/TR/html4/struct/text.html#h-9.2.1> – w3c standard

# 4. Access Alternatives

## a. ACCESSKEY attribute

The ACCESSKEY attribute allows keyboard access to links and input fields in a Web page by pressing **ALT** (Windows) or CONTROL (Mac) plus a keyboard character. It can be used with the following HTML tags: A, AREA, BUTTON, INPUT, LABEL, LEGEND, TEXTAREA.

Access keys may be helpful for people who cannot use a mouse – screen reader users and people with limited mobility – and can also be used by users who prefer keyboard shortcuts.

With access keys, user can quickly go to important links and input fields. For example, for View Your Accounts on the yourshop.com Individual and Small Business home pages, ALT (CONTROL) + U takes one to the Username box, and ALT (CONTROL) + P goes to the Password box:

| logo1  <label for="userid">1. Username</label>:<br /><input type="text" accesskey="U" id="userid" name="userid" tabindex="1" size="13" maxlength="14" class="input" title="Enter Username" autocomplete="off" /></input>...  <label for="password">2. Password</label>:<br /><input type="password" accesskey="P" id="password" name="password" tabindex="2" size="13" maxlength="8" title="Enter Password" class="input" autocomplete="off" /></input>... |
| --- |

Unfortunately, there are numerous problems with access keys:

* **Access keys may conflict with browser, screen reader, and operating system keyboard commands/shortcuts.** Here are the keys used by some applications:

|  |  |
| --- | --- |
| **Application** | **Keys used** |
| IE 6.0.26 | F (File), E (Edit), V (View), A (Favorites), T (Tools), H (Help)  (In Print Preview only: P, U, Z, C); D (Select text in Address bar) |
| Netscape 7.0 | F (File), E (Edit), V (View), G (Go), B (Bookmark), T (Tools), W (Window), H (Help) |
| JAWS 4.5????????? | IE 5 and 6 – D (Move to address bar)  Windows and Menus – ALT (Move to menu bar in app and Cancel/close a cascading menu) |
| Window-Eyes 4.211 | Defaults in Hot Key Definitions window varied when opening Definitions window at different times.  Defaults using ALT were only for Window 0-9 and Attribute Changes Toggle (ALT-A) |
| Home Page Reader | C, L, O, 1 (read all headlines on a page) |

Note that some screen readers are not fully compatible with access keys. See <http://support.microsoft.com/default.aspx?scid=kb;en-us;306448>.

ACCESSKEY support should be tested with all client-supported browsers and assistive technology. For example, ALT-F will work with IE to go to the tag that contains the ACCESSKEY=”F” attribute. However, this key will open the File pull-down menu in Netscape. Access keys may also work differently when JAWS is opened.

* **Remaining keys available for access keys are limited.**

|  |  |  |  |
| --- | --- | --- | --- |
| **Summary of ALT (CONTROL) keyboard shortcuts** used in screen readers and client-approved browsers | | **What’s left** | |
| A  B  C  D  E  F  G  H  L  O  T  V  W  1 | (In Print Preview only for IE: P, U, Z, C)  This list does not consider other key assignments that future versions of browsers/screen readers may use. | I  J  K  M  N  P  Q  R  S  U  X  Y  Z | 2  3  4  5  6  7  8  9  0 |

* **Users may not know access keys are available and what the accesskey assignments are.**  
    
  For screen reader users, an invisible link could be used to lead to a list of access keys and their function. However, informing sighted users of access keys requires a site design or content change – for example, adding an accesskey legend or visual cues to links via styles or HTML (bolded or upper case letter in the link that denotes the accesskey, etc.)
* **Access keys may be difficult to memorize.**  
    
  Too many access keys may be hard to memorize, particularly if numbers are used, and users may forget them between site visits.  
    
  With the available letters left after eliminating browser and screen reader conflicts, there may not be an easily remembered accesskey for a link or input. For example, “H” is not available for “Home”. Even if it were available, would users assume that it’s for “Help”? Which one would be assigned the “H”? Or for “Site Map” and “Sign Off,” which one gets the “S”?
* **Does the accesskey give focus to or activate a link or input?**  
    
  Whether an accesskey only gives focus to or actually activates an element is variable. This should be tested thoroughly tested before implementing access keys.

Given these limitations access keys should be used sparingly. Keep numbers down to make them easier to memorize, and only use for global or frequently used content.

## b. TABINDEX attribute

The TABINDEX attribute controls the tabbing order of fields in a form. TABINDEX can also be used with links, including links in an image map. By pressing the Return/Enter key, a link is activated.

| <form action="submit" method="post">  First Name: <input tabindex="10" type="text" name="first\_name">  Middle Initial: <input tabindex="20" type="text" name="middle\_initial">  Last Name: <input tabindex="30" type="text" name="last\_name">  </form> |
| --- |

## c. Link bypassing and grouping

**Applies to:** navigation areas, collections of links

**Why it’s important:** A navigation bar is often the first thing that is come across on a page. For screen reader users, this usually means having to hear a large number of repeated links on every page before they arrive at any unique content.

**What to do:** Provide a link that will skip over these repetitive links and allow the user to start reading at the beginning of the main body of the page.

**Hints/Examples:**

Place an invisible image with an anchor tag before the nav group or at the very beginning of the page (in the header)

<a href="#skip"><img src="assets/spacer.gif" alt="" width="1" height="1" border="0" alt="skip to main page content" /></a>

Place the anchor immediately before the page content:

<a name="skip"></a><h1>Heading for Page Content</h1>

* The public site has implemented this in the global header. Simply add the anchor name **skip** to the main body of any new or modified pages

# <a name="skip"></a>5. Accessible Forms

Besides access keys and tab indexes, there are other more forms-specific tags that give greater accessibility to forms: LABEL, OPTGROUP, FIELDSET and LEGEND. However, for simple forms, accessibility can also be achieved by correct placement of form prompts and controls.

## a. Form prompts and controls placement

“Controls” are the actual form inputs, such as text fields, radio buttons, multiple-choice lists, and submission buttons. A “prompt” is the text or image that comes before or after a control that describes it or tells the user what to do with the control.

Screen readers should read form controls correctly with the following prompt placements:

|  |  |
| --- | --- |
| **Control** | **Location of prompt** |
| Edit fields  Select menus | To the left or above the control |
| Checkboxes  Radio buttons | To the right of the control |
| INPUT tag buttons | Within the control. The prompt content is entered in the VALUE attribute of the INPUT tag. Like Submit or Reset or Image. |
| BUTTON tag buttons | Within the BUTTON opening and closing tags (<button>………</button>) |

If prompt placement is different from the above or to be absolutely sure that a screen reader will read the correct text prompt for each control without relying solely on placement, use a LABEL tag with a FOR attribute that explicitly corresponds to the ID attribute of the control.

For multiple fields for a form item – for example, separate fields for the area code, prefix, and suffix for a phone number or the three parts of a Social Security number – use the TITLE attribute for all fields after the first one.

| ssn  <label for="ssn">Social Security Number</label><br>  <input id="ssn" value="" type="text" size="3" maxlength="3" tabindex="130"> -  <input title="Social Security Number middle two numbers" value="" type="text" size="2" maxlength="2" tabindex="135”> -  <input title="Social Security Number last three numbers" value="" type="text" size="4" maxlength="4" tabindex="140"> |
| --- |

## b. LABEL tag

**Applies to:** form controls and their labels

**Why it’s important:** Labels for form controls allow assistive technology to associate a form description to its associated for control.

**What to do:** Use the <label> tag for all form labels

**Hints/Examples:**

* Precede the input field code with the label code, and use the for="..." attribute to show which element this label corresponds to. The value of this should be the same as the value for the id="..." attribute that you'll put into the form you want to affect.

<form action="test.html">

<label for="labelexample1">Click here</label>

<input type="checkbox" name="check1" id="labelexample1"></input><br />

<label for="labelexample2">Input Text</label>

<input type="text" name="text1" id="labelexample2"></input>

</form>

* Each label element is associated with exactly one form control
* the label tag cannot span table cells. For example, this will not validate:

<label for="labelexample1"><td>Last Name</td><td>\* required field></td></label>

* If the prompt is also a link, be sure to nest the A tag within the LABEL tag, for example:

<label for="empStat"><a href="help.jhtml#employ">Employment Status</a></label>

| <label for="empStat"><a href="help.jhtml#employ">Employment Status</a></label>  rather than  <a href="help.jhtml#employ"><label for="empStat">Employment Status</label></a> |
| --- |

## c. FIELDSET and LEGEND tags

The FIELDSET tag allows one to arrange the individual controls and labels of a form into logically related groups. The LEGEND tag acts as a caption for the FIELDSET and is placed directly after the opening FIELDSET tag.

| fieldset | <fieldset>  <legend>Name</legend>  <table …>  <tr>  <td><label for="firstName">First name: </label></td>  <td><input type="text" name="name1" id="firstName"></td>  </tr>  <tr>  <td><label for="middleName">Middle initial: </label></td>  <td><input type="text" name="name2" id="middleName"></td>  </tr>  <tr>  <td><label for="lastName">Last name: </label></td>  <td><input type="text" name="name3" id="lastName"></td>  </tr>  </table>  </fieldset> … |
| --- | --- |

Unfortunately, browsers display FIELDSET and LEGEND inconsistently, and the major screen readers and voice browser – JAWS, Window Eyes, and Home Page Reader – do not support them. Also, the visual effect they render may not be appropriate for all forms.

# 6. Accessible Tables

If a **data table** coded correctly for accessibility, its content can be read and navigated by screen readers and other tools used by the visually disabled. **Layout** tables should not be coded for accessibility. Layout should always be controlled through the use of CSS.

Relative sizing rather than absolute sizing for tables ideally should be used to facilitate user-generated text size changes.

## a. Table Summaries <table … summary=”text here”>

**Applies to**: All data tables, such as a table of interest rates, or a table with balances.

**Why it’s important**: The visually impaired can't see the table, so it helps them to know the purpose of the table and its structure.

**What to do:** Writers will provide a summary of the table's purpose and structure.

**Hints/Examples:**

* the summary is hidden to normal browsers, but is read by assistive technologies

"The following table shows the final principal after 1 year, of an account initially with $10000, at 6% interest rate, with the given compounding. It shows that the method of compounding has little effect on the principle.”

Compound Interest

|  |  |
| --- | --- |
| **# of times per year**  **interest is compounded** | **Principle** |
| 1 (yearly) | $ 10600.00 |
| 2 (semiannually) | $ 10609.00 |
| 4 (quarterly) | $ 10613.64 |
| 12 (monthly) | $ 10616.78 |
| 52 (weekly) | $ 10618.00 |
| 365 (daily) | $ 10618.31 |
| continuous | $ 10618.37 |

## b. CAPTION Element <caption>

**Applies to:** the title of the table

**Why it’s important:** the caption helps the screen reader relate a title to a table. OtherAKQAe, the title may appear to simply be a sentence preceding a separate table.

**What to do:** Use **<caption>** to delineate the title of a table. The writers will provide the title.

**Hints/Examples:**

<table>

<caption>AKQA Coffee Study</caption>

<tr> …

## id and headers Attributes

## <th id=”whatever”>

## <td headers=”whatever”>

**Applies to:** headers and data cells in complex data tables

**Why it’s important:** By giving each table heading a unique label (using the **id** attribute) - and then associating each of the data cells with that label new Web browsers can give appropriate feedback for those using assistive technologies.

**What to do:** Simple tables should use the **scope** attribute (see below). Complex tables should use the **id** attribute. If appropriate a complex table may also benefit by using the **axis** attribute)

<xml id="cdcat" src="cd\_catalog.xml"></xml>

<table border="1" datasrc="#cdcat">

<thead>

<tr><th>Artist</th><th>Title</th></tr>

</thead>

<tfoot>

<tr><th colspan="2">This is my CD collection</th></tr>

</tfoot>

<tbody>

<tr>

<td><span datafld="artist"></span></td>

<td><span datafld="title"></span></td>

</tr>

</tbody>

</table>

**Hints/Examples:** *there should not be any more borders surrounding the caption***.**

|  |  |  |  |
| --- | --- | --- | --- |
| Cups of coffee consumed by selected individuals of the production team | | | |
| **Name** | **Daily Cups** | **Type of Coffee** | **Projected cups of coffee for 2007** |
| Content Strategy | | | |
| Mary | 4 | Latte | 1032 |
| Publishing | | | |
| John | 3 | Mocha | 774 |

<table cellspacing="0" cellpadding="2" border="1" width="100%" **summary**="This table displays the projected yearly cups of coffee consumed by selected individuals of AKQA. It shows that Mary and John consume an amazing amount of coffee.">

<caption>Cups of coffee consumed by selected individuals of the production team</caption>

<tr>

**<th id="hdr1">**Name</th>

<th id="hdr2" abbr="Cups">Daily Cups</th>

<th id="hdr3" abbr="Type">Type of Coffee</th>

<th id="hdr4" abbr="Projected">Projected cups of coffee for 2007</th>

</tr>

<tr>

<td colspan="4">Content Strategy</td>

</tr>

<tr>

**<td headers="hdr1">**Mary</td>

<td headers="hdr2">4</td>

<td headers="hdr2">Latte</td>

<td headers="hdr2">1032</td>

</tr>

<tr>

<td colspan="4">Publishing</td>

</tr>

<tr>

<td headers="hdr1">John</td>

<td headers="hdr2">3</td>

<td headers="hdr3">Mocha</td>

<td headers="hdr4">774</td>

</tr>

</table>

c. **Abbreviations for table headers**

**Applies to**: data tables that have long titles in a row or column header

**Why it’s important**: Saves time. Assistive technologies read every row and column header. This can get tedious with wordy headers.

**What to do:** Use an abbreviation for lengthy row or column headers. Writers will designate when an abbreviation should be used, and provide the abbreviation.

**Hints/Examples:**

* A data table with a column header "**Projected cups of coffee for 2007**" would be read every time the screen reader read a value in that row. Imagine having to listen to "**Projected cups of coffee for 2007**" over and over again. Provide an abbreviation of "Projected" to the AKQA. That way, the screen reader will read the abbreviated “Projected” when it comes across an associated value in that row.

|  |  |  |  |
| --- | --- | --- | --- |
| Cups of coffee consumed by selected individuals of the production team | | | |
| **Name** | **Daily Cups** | **Type of Coffee** | **Projected cups of coffee for 2007** |
| Mary | 4 | Latte | 1032 |
| John | 3 | Mocha | 774 |

<table cellspacing="0" cellpadding="2" border="1" width="100%" summary="This table displays the projected yearly cups of coffee consumed by selected individuals of the production team. It shows that Mary and John consume an amazing amount of coffee.">

<caption>Cups of coffee consumed by selected individuals of the production team</caption>

<tr>

<th **scope='col'**>Name</th>

<th **scope='col' abbr='Cups'**>Daily Cups</th>

<th scope='col' abbr='Type'>Type of Coffee</th>

<th scope='col' abbr='Projected'>Projected cups of coffee for 2007</th>

</tr>

<tr>

<td>Mary </td>

<td>4</td>

<td>Latte</td>

<td>1032</td>

</tr>

<tr>

<td>John</td>

<td>3</td>

<td>Mocha</td>

<td>774</td>

</tr>

</table>

# 7. Accessibility and Cascading Style Sheets

Cascading style sheets benefit accessibility primarily by separating document structure from presentation. CSS allows precise control, outside of markup, of a web page, including character spacing, text alignment, and font characteristics.

Controlling presentation with a style sheet allows users to override the default style sheet with their own styles. Some browsers allow visually-impaired people to set their own style sheets, and people with low vision can use a graphical browser by enlarging the text and choosing high contrast colors – for example, a user may choose to have all web content displayed in large, bright-yellow type against a black background.

Although separating style from markup simplifies a web page’s HTML, the document should be coded in a “logical” order and designed to make sense without style sheets for users who need to disable styles.

## 

## a. Fonts and positioning

Avoid using exact pixel measurement for font sizes and positioning. It restricts the user’s ability to control the size of the displayed text within the browser. Even users who are not visually impaired may wish to increase the display font for readability.

| Use relative or absolute sizes:  p { font-size: 120% }  .small { font-size: xx-small}  .indented { margin-left: 10% } | Rather than exact sizes:  p { font-size: 14px }  .small { font-size: 8px }  .indented { margin-left: 10px } |
| --- | --- |

## b. Heading tags

Structure should be controlled by the HTML coding, not the style sheet. For example:

| Do not define a style as:  subheading { font-size: 135%; font-style: bold }  and apply it in an HTML tag to designate subheaders:  <p class="subheading">Get pre-approved</p> |
| --- |

To a screen reader, or someone with style sheets turned off, the above example will not appear to be a subheader, but just another line of text. Instead, use the proper heading tag and apply styles to it.

| H2 { font-size: 135% }  <h2>Get pre-approved</h2> |
| --- |

# 8. Scripting

HERE June 8, 2007

Scripting poses unique issues for accessibility. Screen readers and other assistive technologies handle JavaScript with varying degrees of success. In addition, JavaScript tends to rely on mouse input more than on keyboard input, creating a problem for people who cannot use a mouse.

When using scripting, consider:

* Scripts must operate with both keyboard and mouse input (or just keyboard). Scripts triggered by device-dependent event handlers – for example, ONCLICK or ONMOUSEOVER – may be completely unusable with assistive technologies.
* Scripts embedded in images or links (e.g., a query embedded in a script) can result in unintelligible output by the screen reader, which may read a string of letters and numbers but not provide any usable information.
* Scripts should be tested on multiple platforms, browsers, and assistive technologies to ensure maximum user access. In particular, scripts **must be tested with keyboard-only input**.

Even an accessible script may produce problematic results, such as a child browser window or other unanticipated changes to the page. The user may not know what has happened or where they are on a page, or the screen reader may behave unpredictably.

Any scripting should be carefully evaluated before development. What will this script accomplish? Why is it necessary? Is there a way to achieve the same functionality or effect without using scripting?

## a. Visible effects

In general, any script that produces a visible effect must have a description attached to it or built into it.

* Include "functional" ALT text with scripts embedded in images (such as button images used to increase or decrease a quantity on a form), explaining the function of scripts (such as "add one item").
* Scripts generating dynamic objects can confound screen readers and disorient users. Newer screen readers can process dynamic objects, but users need to know where and/or what the change is before they know to re-read it. For instance, a calculator script that computes loan information could have default text in the "results" box that says "Your monthly payment will appear here when you have completed the form."
* Keep in mind the distinction between purely decorative effects and providing functionality or information to the user. A script that triggers a color change when a user mouses over an image is merely providing a visual embellishment. Non-sighted users will not be at a disadvantage if they are not told about this and in fact may find the page much less usable if descriptions of such "fluff" are intermingled with critical page content.

## b. Child browser windows

A child window is a separate browser window that opens when a user selects a link.

* Alert users that a link opens a new browser window. Otherwise, a blind user using a screen reader cannot tell that a new window has opened. Use the TITLE attribute of a text link to inform the user that the link will open a new window – for example, the TITLE of a link with the text "Contact Us" would be "Contact Us. This link will open a new browser window."
* Child windows can be hard to locate for people using screen magnifiers. These users are often zoomed-in on a document, so when the child window opens they have to zoom out and hunt it down. For this reason, child windows should open with a minimal offset from the parent window.
* If using "javascript:" as a link target keep in mind the limitations of device-dependent event handlers, such as ONCLICK (discussed below). Remember that any JavaScript written must be accessible for non-mouse users. In the example above, the redundant event handler ONKEYPRESS would accommodate keyboard users.

## c. Event triggers

Scripts should be tied to active user input. Triggers such as a key press or selection of an option from a list allow the user to be in control of the changes on the page, rather than at the mercy of them. Using "passive" triggers without notifying the user – for example, when the page loads, after a set time, or when a mouse passes over an object – can be disorienting for users with screen readers.

**Specific event handler issues and recommendations:**

* Do not use ONDBLCLICK. It does not have a keyboard equivalent.
* Do not use mouse coordinate-dependent event handlers.
* Other mouse-specific handlers can be paired with equivalent keyboard handlers:
* ONMOUSEDOWN with ONKEYDOWN
* ONMOUSEUP with ONKEYUP
* ONCLICK with ONKEYPRESS – for many users, it may be possible to provide ONCLICK alone, as it can be accessed from the keyboard. But this implementation would require careful testing.
* ONMOUSEOVER and ONMOUSEOUT – these event handlers must be paired with ONFOCUS and ONBLUR to allow for keyboard-only input. This is a problematic solution and may require the addition of TABINDEX. It will certainly require careful testing.

## d. Expanding menus

Using ONMOUSEOVER-triggered expanding menus for navigation has strong limitations. The current state of JavaScript and browser support makes a clear scripting solution nearly impossible. There are some differences between mouse and keyboard events involving focus, and the diversity of browser support (not to mention various assistive technologies) makes this a difficult proposition.

The benefits of expanding menus are far outweighed by their drawbacks. If ONMOUSEOVER is used in navigation menus, provide an accessible alternative, at least by giving immediate access to the submenu choices once the main menu link has been followed.

## e. Pulldown menus

Do not create pulldown menus that select using the ONCHANGE handler. A keyboard-only user will not be able to scroll through all the menu choices, but will be sent off immediately to the first item on the list.

The Windows/Internet Explorer key combination (ALT + Down Arrow key) will allow keyboard users to scroll through the list. But this does not work for Netscape users (Windows and Mac) or for IE/Mac, and novice keyboard users may not be aware of this key combination. In addition, pressing two buttons at once might be challenging or impossible for a disabled person.

**Solution:**

Program pulldown menus as forms, using server-side scripting and a "submit" button, rather than as "pseudo-forms" using JavaScript and no submit button. The pulldowns will then be accessible to everyone, with no loss of function. Also consider offering alternate forms of navigation, such as simple lists in which all links are visible without any action on the user's part.

# 9. Accessibility of PDF Documents

Newer versions of Adobe Acrobat (6.0 and above) offer accessibility features, including the ability to:

* Read PDF files aloud using standard operating system text-to-speech synthesis.
* Exercise greater control when using MSAA-compliant screen readers.
* Evaluate the accessibility of PDF files.
* Reduce the size of tagged Adobe PDF files for faster download times.

To assure the greatest accessibility, the latest version of Acrobat should be used for all PDF conversion and editing at AKQA.

## a. Testing PDFs for accessibility

PDFs need to be tested for accessibility, including read order (especially for data tables), alternative text for images and charts, and forms accessibility.

#### Accessibility checking using Acrobat

A PDF can be checked for accessibility within Acrobat. From the Advanced menu, choose Accessibility and select one of the following options:

* **Quick Check:** When selected, this option simply shows a message box with the Quick Check results, with no other information.   
    
  Quick Check only checks whether a document is structured or not (see testing results sample below). For example, for a 52-page tagged PDF that showed thousands of errors in Full Check, Quick Check noted: “Accessibility Quick Check is complete: No accessibility problems were detected in this quick check. Choose the Full Check command to check more thoroughly.”
* **Full Check:** Provides a thorough check of a PDF, including a report, accessibility errors highlighted on the PDF itself, and “tool tips” that explain the errors when the highlighted areas are hovered over.
* **Add Tags to Document:** Automatically adds tags, but may not be complete (see testing results sample below).

| Acrobat accessibility testing results for sample untagged PDF  **Quick Check** showed the message, “This document is not structured, so the reading order may not be correct. Try different reading orders using the Reading Preferences panel.”  **Full Check:**  The checker found problems which may prevent the document from being fully accessible.  + All of the text in this document lacks a language specification.  + 5 images with no alternate text.  + 5 words inaccessible because they contain characters with no reliable mapping to Unicode.  + This document is not structured; the reading order of the contents may be incorrect.  **Add Tags to Document:**  Acrobat was able to make this document accessible but found the following oddities:  Some unknown font encodings encountered. [5 of 4630 glyphs]. |
| --- |

For AKQA PDFs, always use Full Check. And be sure to recheck all documents after any changes, whether for content or accessibility.

#### Testing PDFs with a screen reader or with Acrobat’s Read Out Loud feature

It’s ideal to test PDFs by using a screen reader. However, if a screen reader is not available, a PDF can also be read aloud from within Acrobat or Acrobat Reader. Choose “Read Out Loud” from the View Menu. While “Read Out Loud” does not offer the myriad settings of a full-fledged screen reader, it does offer a few setting choices in the Reading panel of the Preferences dialogue box (Select “Preferences…” from the Edit menu).

## b. Creating accessible PDFs

For the greatest accessibility, PDF documents should be tagged. Besides containing the document’s structure, a tagged PDF includes a hierarchical structure tree where all the document’s content’s internal structure is interpreted. A screen reader can render a tagged PDF more accurately, and tagged PDFs also support text reflow, which is helpful for screen magnification and for viewing text on different-sized screens.

PDFs created within the latest versions of Microsoft Word, Excel, and PowerPoint will be tagged. However, PDF conversion in other applications may not. See the respective application’s Help, Acrobat’s Help, and the PDF Accessibility resources in Appendix B, below, for more information.

| Creating PDFs in Word  In Word 2000 for Windows, from the menu bar select “Adobe PDF” and choose “Convert to Adobe PDF.” “Convert to Adobe PDF and Email,” or “Convert to Adobe PDF and Send for Review.” You will then be asked to save the PDF. After clicking “OK” Acrobat PDFMaker will automatically create the PDF.  Make sure that, in the Conversion Settings dialogue box (Settings tab), the checkbox for “Enable accessibility and reflow with Tagged PDF is checked. |
| --- |

## c. Modifying PDFs for accessibility

Probably the most important aspect of making a PDF accessible is that a document’s structure is correctly exported to Acrobat. However, all the information necessary to make a document completely accessible in Acrobat may not make it through the PDF conversion process from the document’s original application, and further modification for accessibility will be needed.

#### Adding document language

While current screen readers are designed to read documents based on one language dictionary, future technologies may be able to access multiple language dictionaries. For this reason, Acrobat considers specifying the document’s language as an accessibility requirement, and Full Check will flag its omission as an accessibility error.

To select a language for a PDF, choose Document Properties from the File Menu, click on Advanced in the left pane, and then choose the language from the Language dropdown box.

#### Creating tags for an untagged document

Tags are an invisible part of the document structure. To see if a PDF is tagged or not, use the Tags palette. If it doesn’t appear on the Navigation Pane along the left side of the Acrobat window, open it from the View menu by selecting Navigation Tabs and then Tags. This will open a floating palette that can be docked on the Navigation Pane by clicking on and dragging the tab of the Tags palette to the Navigation Pane.

When a PDF is opened in Acrobat, a small icon will appear at the top left of the Tags palette. If the PDF is tagged, the icon will have a plus sign to the left and the word “Tagged” to the right. If the PDF is not tagged, the icon will note “No Tags available.”

If the document is not tagged, tags can be added in Acrobat. From the Advanced Menu, select Accessibility and then Add Tags to Document. Be sure the recheck the PDF for accessibility.

#### Modifying tag elements

The various tag elements in a PDFs structure tree represent page elements, such as an image or text block, and higher level elements, such as a section of the PDF that contains page elements. The properties of any element can be viewed and edited via an element’s TouchUp Properties dialog box, which can be opened in Windows by right clicking the element or by selecting Properties in the Options Menu at the top of the Tags palette.

The following element properties are particularly relevant for accessibility:

* **Actual Text.** This property is much like the TITLE attribute in HTML. Any text entered for the property will be read by a screen reader instead of the element’s text content shown in the PDF document.
* **Alternate Text.** This property corresponds the HTML’s ALT attribute. As in HTML, graphic representations in PDFs such as illustrations, charts, and graphs cannot be interpreted by a screen reader. However, a screen reader will read text entered in the Alternate Text box for a graphical element or any element, such as a formula, that does not translate naturally into text. As with the ALT attribute, the Alternate Text content should provide an appropriate text alternative to the visual information.
* **Text language.** In addition to the high-level language specified for a PDF as a whole, the language of individual elements can be set to a different language. A screen reader can then use alternate pronunciation or hyphenation schemes for the language.  
    
  Note that the language selected for an element applies to all the elements nested under it in the logical structure tree.

Is there any difference between Actual Text and Alternate Text? JAWS and Acrobat’s Read Out Loud both Actual and Alternate text whether added to an image or to text. However, the Alternate Text box offers a bit more space for entering content.

#### Clarifying abbreviations

Actual Text or Alternate Text can also be used to give the full text for an abbreviation. However, if the abbreviation is not separated into its own element, the entire content of the element in which the abbreviation appears will have to be included in the Actual or Alternate text box.

Rearranging a PDF’s Read Order

Tags support, but don’t guarantee, a correct read order. However, a PDF’s read order can be changed within the Tags palette.

To change the order of an element, in the logical structure tree, select the element to be moved and drag it to the desired position in the tree. The visible page content will not change, but a screen reader will read the content as ordered in the structure tree.

For simple documents, this is easily done. However, for more complex documents with a correspondingly complex tree, it may be difficult to find the element to be moved. Also, if a document is edited in its original application and converted to a new PDF, all the Structure Tree reorganization will be lost.

The following table offers ways to navigate a PDF’s structure tree more efficiently:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Navigating a Tagged PDF’s Structure Tree  **1. Accelerator keys**  Tags within the logical structure tree of a PDF can be opened or closed by clicking on the top level tag icon, but Acrobat also features “accelerator keys” to move more quickly through the tree.   |  |  | | --- | --- | | **Key** | **Function** | | Arrow keys  * Right (element selected) * Left (element selected) * Up * Down * Right * Left | Expand the element  Close the element  Move up the structure tree  Move down the structure tree  Move down a series of nested elements  Move up a series of nested elements | | **Plus and minus keys** (Num pad; element selected)   * Plus * Minus | Expand the element  Close the element | | **Star key / forward slash** (Num pad)   * Star key * Forward slash * Shift + star key | Open tree to show all elements  Close the tree  Open tree one level at a time |   **2. Highlighting elements**  Sometimes it’s difficult to relate an tag in the structure tree to the content in the document. To find the content related to a tag, go to the Options menu at the top of the Tags Palette and select Highlight Content. When you select a tag, the corresponding content will be highlighted with a black box.  **3. Viewing the logical structure tree in XML**  Another way to view the tag tree (and the only way for Acrobat Standard) is to save the PDF as an XML file. Choose “Save As” from the File menu, and select XML from the pull-down list.  The XML file shows more information than the Tags Palette in Acrobat. For example, Alternate Text is shown in the XML code, but not in the Tags Palette. |

## d. Tips for optimizing PDFs pre- and post-conversion

* Correct use of columns and tables in the original document will provide better structure in a PDF and help a screen reader correctly render the PDF.
* To decrease PDF file size, use styles or before/after paragraph spacing in Word rather than repeated returns for paragraph breaks or to make large spaces. Every hard return in Word adds a <Normal> tag.
* After the final editing of a PDF, choose Save As rather than Save. In the Save As process, Acrobat consolidates duplicate fonts and images, removes “unused objects,” and optimizes for Fast Web viewing, resulting in a significantly smaller file size.

See **Appendix B: Resources** for further information about PDF accessibility.

# 10. Flash Accessibility

??????????????????? **update information with Flash 8 or 9**

The information in this chapter is based on Adobe Flash 8, Flash CS3( both regular and Professional version) and Adobe Flash Player 8 and 9. To assure the greatest accessibility, the latest version of Flash should always be used at AKQA.

Adobe provides extensive information on Flash accessibility at:

* <http://www.adobe.com/accessibility/products/flash/>
* Flash 8 Help and Flash CS3 Help

Flash accessibility is inconsistent, and all techniques noted here should be thoroughly tested (see “Testing and validating Flash content,” below).

## a. Adding accessibility to Flash content

Accessibility information can be added to a Flash movie via the Accessibility panel or with ActionScript.

#### Accessibility panel

Developers can choose what movie elements will be accessible or not accessible by checking or unchecking the **Make Movie Accessible**, **Make Object Accessible**, and **Make Child Objects Accessible** checkboxes in the Accessibility panel.

The Accessibility panel allows the addition of text equivalents for a single Flash content element or for a group of elements. The panel includes a **Name** field for a brief descriptive text equivalent and a **Description** field for a longer text equivalent. A screen reader reads both by default. (See “Assigning text equivalents to visual elements” below.)

The Accessibility panel also offers the **Auto Label** option. With this option, text adjacent to or within another object can be associated as a label or title for the object – for example, an input text field. (See “Flash Components and Accessibility” below.)

#### ActionScript

All of the information that can be added via the Accessibility panel can also be assigned using ActionScript. See Flash MX 2004’s Help, “Creating accessible documents with ActionScript,” for more information on using ActionScript for accessibility.

According to Adobe, both Flash MX 2004 and Flash Player 7 support updating and generating accessibility properties via ActionScript to enable applications to update accessibility information for dynamic content.

For sample Flash accessibility ActionScript code samples, see “Flash MX Accessibility,” Digital Design World 2004 lecture notes, at <http://texturadesign.com/html/speaking_flash.htm>.

## b. Making Flash movies accessible to screen readers

To access Flash content with a screen reader, users need Flash Player 6 or higher and a screen reader that has integrated the Flash Player implementation of Microsoft ???????????Active Accessibility (MSAA, http://msdn2.microsoft.com/en-us/library/ms697707.aspx). When Flash Player 7 was released, this included JAWS and Window-Eyes.

* **Make Movie Accessible**When a screen reader encounters a Flash movie, it announces the beginning and end of the movie. To ensure that the movie is accessible, the **Make Movie Accessible** option in the Accessibility panel should be selected (this option appears in the Accessibility panel when nothing is selected on the Stage). This option should be selected by default and allows Flash Player to pass accessibility information nested inside a movie to a screen reader.  
    
  Even if this option is selected for the entire movie, other movie elements can still be hidden from screen readers by unselecting the **Make Object Accessible** or **Make Child Objects Accessible** options.
* **Flash audio and screen readers**Audio can enhance a screen reader user’s experience of a Flash movie. However, consider how it may interrelate with a screen reader’s speech rendering of other objects in the movie. Background music or event-triggered sounds may make the screen reader’s rendering hard to hear. If so, a blank button with the text equivalent “Turn off all Flash-generated sounds” could be added to the movie.
* **Offstage text equivalent**  
  In certain cases where it is difficult to make Flash content accessible, a single-column text version of the movie could be added primarily offstage. If all other objects in the movie are made inaccessible, then a screen would only read this text.   
    
  Although Adobe claims that a screen reader will read offstage text content, JAWS will not read text that is entirely offstage. However, it will read all the content of a text element if even a small portion of the area within the element bounding box is onstage. Thus, if the actual text content is offstage with the bounding box slightly onstage, it would be completely invisible in the movie yet also accessible to JAWS.  
    
  While this method solves the problem of the often random order of screen reader rendering of Flash objects, it would also be necessary to revise this text if other content in the movie is changed, a step that could be easily forgotten.

## c. Assigning text equivalents to visual elements

Screen readers can read all Flash text elements, text in buttons, and basic form elements. However, add text equivalents for graphic objects and icons, as well as for animations, by using the Accessibility panel’s **Name** and **Description** options.

The object must be saved as a symbol in the library. As text equivalents are not supported for graphic symbols, the object needs to be saved as a button symbol or movie symbol. Also provide a Name and/or Description when using a feature such as Break Apart for text.

The Name and Description options in the Accessibility panel can also be used to provide context for an object or to give the meaning of an abbreviation or acronym, as well as for adding headings, lists, and quotes. Also, acronyms could be written with spaces between the letters to ensure that the screen reader says them correctly – for example, “Internal Revenue Service (I R S)”.

A screen reader generally reads the Description after the Name. Be sure to consider how the two would flow together, or use only a Name or Description. The latter is best for longer content

As with ALT attributes, text equivalents for Flash elements are only needed if the elements are meaningful for a screen reader user. Repetitive and decorative elements should be hidden, as well as elements that are difficult to render using assistive technologies – for example, a movie that keeps looping.

If the relationship among elements in a movie is more significant than their individual identities, a single text equivalent can be given for the group by selecting Make Object Accessible and deselecting Make Child Objects Accessible. For example, for an animation of the solar system showing the movement of the planets, a single text equivalent describing the movement would be more significant than descriptions of the animation’s individual planet elements.

Invisible buttons and some pre-made forms components (such as combo boxes, list boxes, and scroll bars) cannot be given accessible text equivalents. However, form objects equivalent to Flash’s pre-made versions can be created manually and text equivalents added to sub-components of the objects.

JAWS reads button text by default. However, if Name text is assigned to a button in the Accessibility panel, JAWS will read the Name content and not the button text. If a Description is provided for a button, JAWS will read it after the button text or Name content. JAWS also reads the Description of dynamic text after it reads the text.

## d. Controlling read order

The default reading order in a Flash movie does not necessarily follow the same order of the visual layout of the page. The read order, however, should be understandable to a screen reader user.

The default reading order of a Flash movie is determined in the same way as the tab index. With no modification, the tab index of the movie should move roughly from left to right and top to bottom. A screen reader will read the content in the same order as the tabbing order. However, as one gets near to the end of a line, the tab order will jump back and forth between the end of the line and the beginning of the next line.

An appropriate order for a screen may be different than how the page appears visually. All related elements should be read as a group. For example, the navigation for the movie, content items, and other instructions should all read as their respective groups. If the groups were intermingled, the movie would be difficult for a screen reader to understand. In particular, it would be difficult for a screen reader to navigate the movie if the navigation elements were interspersed with other content.

Even if a screen reader reads the content differently than the visual order, there is no need to specify a different read order if the content makes sense to the screen reader user.

Read order can be controlled by:

* **Limiting stage size and keeping layout simple**  
  The simplest way to control read order is to keep the stage size small and the layout simple. For a simple movie on a small stage, Adobe claims there is no need to specify read order.
* **ActionScript**  
  ActionScript is the most precise way to control read order. Specify a .tabindex value for each instance on the stage – for example,   
    
  root.a\_mc.tabindex = 1;  
  root.b\_mc.tabindex = 2;  
  root.c\_mc.tabindex = 3;…  
  However, it is more difficult to use this method with a more complex movie, as the .tabindex value must be specified for every instance of the movie. If a single instance is missing (even one instance of static text), the movie will revert to the default read order. There are a few exceptions to this rule – for example, children of a button object or a button movie clip allegedly do not have to be included in the .tabindex order.  
    
  Adobe states that offstage, invisible, or obscured objects must also be included in the read order, as (alleges Adobe) screen readers do not distinguish between onstage/offstage or visible/hidden. For elements that should not be read at a given point in the movie, one could set the value for the .visible property to false or the .silent property to true.  
    
  While static text is accessible to screen readers, it cannot be given an instance name, which is required to control tab order and reading order.
* **Offstage list**  
  Develop a secondary control that places a linear version of the content offstage. If a screen reader is detected, for each element on the state a second instance of the same object is placed in a column just offstage. All of the onstage objects have the .silent property set to true. A screen reader will read the offstage content from top to bottom without specified .tabindex values.  
    
  For complex movies that import content from a dynamic data source, this method can help eliminate long read order lists. This method is especially helpful for movies that are generated on the fly and the read order is difficult to anticipate.

Read and tab order is tricky even for a simple movie. For example, the tab order for the movie illustrated below renders differently if JAWS is active:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| flash_sample | |  |  | | --- | --- | | **Tab order without JAWS loaded** | **Tab order with JAWS loaded** | | 1. dropdown menu 2. radio buttons (only one tab for both buttons) 3. First Name 4. Last Name 5. Submit | * 1. dropdown menu   2. First Name   3. Last Name   4. radio button for Single Selection   5. Submit button   6. radio button for Multiple Selection | |

Ironically, Flash MX 2004 provides this movie as a sample of how to use the accessibility features of Flash.

## e. Captioning audio content

Flash excels at providing audio content, and synchronized audio tracks can be used to provide narrative audio to the visually disabled. However, if the audio provides meaningful content not otherwise expressed or is necessary for interacting with a Flash movie, it will need to be captioned for the hearing impaired. “Decorative” audio content does not need to be captioned.

For example, a navigation bar that plays sounds when a mouse passes over the buttons does not need to be captioned, but an audio recording of a speech or a video of a presentation should be captioned.

Captions not only display spoken dialogue as printed words on the screen, but can also identify speakers, music, laughter, and on- and off-screen sound effects.

Captions can be created by:

* **Manually adding captions**  
  The simplest way to add captions is to add the caption text on the timeline and synchronize the text with the audio as it is played.   
    
  This method works well for brief audio segments. However, for audio lasting more than a few seconds, manually adding text objects to the stage is tedious and time consuming.
* **Importing captioned video**  
  Use external tools to generate the captions, add the captions to a video file, and import the captioned video file into the Flash movie. External tools for captioning include HiSoftware’s Hi-Caption and MAGpie from the National Center for Accessible Media.   
    
  This method can simplify the development of the user interface for captioning; however, it limits the flexibility and integration of the captioning controls within the user interface.
* **Adding captions with XML using Hi Caption SE**  
  This method uses Hi-Caption to create a caption data file, and the captions are imported into Flash 2004 using the Hi-Caption Viewer component. The component also provides display options for the captions control interface, and multiple instances of the captioning component can deliver captioning data for multiple languages at the same time.

## f. Working with changing content

* **Screen readers and animation**When content on a web page changes, most screen readers will begin reading the content again from the top of the page. For example, a Flash movie with an animation may cause the screen reader to begin reading the page again every time the animation changes.  
    
  A Flash movie that never stops moving will cause screen readers to refresh frequently. No matter where the movie is on a page, the screen reader can interpret motion as a page update, return to the top of the page, and start reading it again.   
    
  To avoid multiple refreshes, hide the animation from the screen reader by making the child objects inaccessible by deselecting the Make child objects accessible option in the Accessibility panel. If the movie is meaningful to a screen reader user, provide equivalent text in the Accessibility Panel for the movie.
* **Motion and cognitive disabilities**Transitions and loading sequences in a movie should stop once a page loads. Constant motion can be distracting to people with cognitive disabilities and impede them from reading or seeing other elements on a page.  
    
  For example, a person with dyslexia may have to turn off any moving elements of a web page to access the text content, and elements that blink at certain rates can trigger photo-sensitive epilepsy in some people (the blink range that may do so is difficult, if not impossible, to predict).
* **User control of movement**Information in a movie that only remains on the screen for a short time can present problems for both screen reader users and people with learning disabilities. It may be difficult for screen readers to keep up with quick changes in a movie, and the learning disabled may not be able to process information quickly. “Next” or “Pause” buttons could be added to control movement.

## g. Flash components and accessibility

Accessible Flash components include:

* Button
* CheckBox
* RadioButton
* Label
* TextInput
* TextArea
* ComboBox
* ListBox
* Window
* Alert
* DataGrid

Use the Accessibility panel to assign names to buttons and input text fields to enable a screen reader to identify them correctly. This can be done two ways:

* **Use Auto Label to assign text adjacent or within the object as a label.**   
    
  The **Auto Label** option associates text next to another object – for example, an input text field – as a label or title for the object. Unfortunately, Adobe does not precisely define what is meant by “adjacent.” Flash MX 2004’s Help for “Choosing names for buttons, text fields, and entire Flash applications” only says, “Flash automatically gives an appropriate name to a button or input text field in your document, as a text label that you have placed on top of, inside, or near a button or another text field. Labels for buttons must appear within the bounding shape of the button.”   
    
  However, at a workshop at the Technology and Persons with Disabilities conference in March 2004, Bob Regan, senior product manager for accessibility at Adobe, said that text to the left of or above the form element is the label for the element.  
    
  But, how close is “near” in order for the text label to be used as the label for a text field? What if there is more than one “text label” near a text field?   
    
  Flash’s Help acknowledges that there are locations for the label that Flash Player cannot detect and that a name will need to be specified for the button or text field. As an example, Flash’s Help shows a button with the text describing the button outside and to the right of the button and notes that Flash Player (or a screen reader) will not detect this text. In this case,   
  the name of the button should be entered in the Accessibility panel for the button and, to prevent repetition, the text object made inaccessible.
* **Enter a specific label in the Accessibility panel Name field.**   
    
  Automatic labeling can be turned off for the entire movie and names assigned to objects via the Accessibility panel. Even if Auto Label is selected for the movie, if a name is assigned to a specific object, the name will be used instead of the object text label.  
    
  An object’s accessibility name is unrelated to the ActionScript instance or ActionScript variable name associated with the object.

“Hit areas” (button symbols with nothing in them except a hit state) are not recognized by screen readers, nor are invisible buttons placed over a background with text. An invisible image can be put in the button. Also, invisible buttons with a Name or Description may be accessible.

## h. Other Flash accessibility techniques

* **Ensure keyboard access to all controls**   
  Some visually and mobility impaired people may not be able to use a mouse. Thus, all controls in a Flash movie need to be navigable using only the keyboard. To facilitate keyboard access, scripts should be kept within frames instead of attaching them directly to objects.   
    
  Keyboard shortcuts added to commonly used buttons will facilitate access to a Flash movie. However, any keyboard shortcuts should not conflict with browser or assistive technology keyboard commands.   
    
  Avoid the use of components that are device-dependent, such as only usable with a mouse. Instead, keyboard controls can be included in components that are created from scratch.
* **Expose state of controls**For all controls in a Flash movie, the user needs feedback on any changes in the controls. For example, when activated a Play button may change to a Pause button. For screen reader users the accessibility information for the button should be updated when the state of the button changes.
* **Describe structure of complex movies**A description of a complex movie will help orient screen reader users to its structure and content. Either use the root-level Description or a separate information screen. A separate information screen allows users to get to the content without hearing the description at each refresh.
* **Use accessible color**  
  Color usage in Flash movies should follow the guidelines given in the “Color and contrast” section of Chapter 1, Accessible Design. Designers can use Flash’s Color Swatches panel to create custom accessible color palettes.
* **Allow user control of presentation**ActionScript can be used to allow users to change presentational aspects of a Flash movie. For example, the ActionScript Color object can be used for modifying or applying color to Flash content and the TextFormat Object for modifying text properties, with buttons added to the Flash movie to allow user control of these objects.

## i. Flash extensibility

The authoring environment for Flash MX 2004 features a new extensibility layer that allows third-party developers to create extensions for Flash. Accessibility extensions includes Hi-Caption™ (closed caption generation system and captioning interface), AccRepair® (Flash accessibility validation and repair tool), and OnScreen Keyboard (component for easier form and comment field entry by people with low vision, color blindness, and mobility disabilities).

Flash extensions can be found at the Adobe exchange ([http://www.Adobe.com/  
cfusion/exchange/index.cfm](http://www.macromedia.com/cfusion/exchange/index.cfm)).

## j. Testing and validating Flash content

At the time of this writing, there were no tools that completely validate Flash content automatically. Flash content is more complex than HTML, so it is more difficult to construct measurable rules and be able to validate these rules.

HiSoftware’s AccRepair for Flash does offer validation for text equivalents and tab order.

Justin Cone at the University of Texas, Austin, tested 11 sample Flash movies using JAWS 4.50 (see <http://ficp.engr.utexas.edu/cone/flash/files/Flash%20and%20JAWS.pdf>). He concluded that:

* Static text offstage elements were not read by JAWS (contradicts Adobe documentation that says screen readers make no distinction between “onstage or off, visible or hidden.” – see workaround described in “Offstage text equivalent” above).
* Read order is complex and somewhat unpredictable.
* JAWS does not re-read the contents of a movie after an update has occurred.
* JAWS handles simple, automated slideshows fairly well.

To determine if a Flash movie is accessible:

* Use a screen reader such as JAWS or Window-Eyes to access the movie.
* As keyboard access differs when a screen reader is opened, access the movie using only the keyboard when a screen reader is not running.
* Possibly use third-party repair and validation tools for Flash to identify common errors.
* Ideally test with users with disabilities.
* At the very least tab through the movie and press enter to go to a link. This will give an idea of the read order and if a person using only a keyboard can navigate the movie.

## k. Flash accessibility limitations and benefits

While Flash has made significant progress in accessibility, Adobe acknowledges that some Flash content is not accessible with Flash Player 7. This includes dynamic text variables and custom navigational elements, as they are not exposed to the operating system using the same methods as HTML.

Bob Regan’s list of things that Flash does “badly” includes:

* Read order is confusing.
* Flash doesn’t have a good sense of what data to pass to screen reader – for example, onstage or offstage.
* As Flash objects have a parent/child relationship rather than being linear like HTML, the structure of a Flash movie is hard to discern.
* Animation forces constant render refreshes.

However, Flash may be helpful for certain disabilities. While accessibility is often perceived in terms of helping visually disabled people access the web with screen readers and voice browsers (thus the focus on adding text content to images and text not clear to the visually disabled), images and animation may benefit users with other disabilities such as dyslexia and other cognitive disabilities.

The PBS site on Zoot Suit Culture ([http://www.pbs.org/wgbh/amex/zoot/eng\_sfeature/ sf\_zoot\_mx.html](http://www.pbs.org/wgbh/amex/zoot/eng_sfeature/%20sf_zoot_mx.html)) includes a Flash movie that is highly accessible. It is entirely navigable using the tab key, the video clips in the movie are captioned, and the background music can be turned on or off, making the video content and screen reader rendering easier to hear. It is also nicely designed and is relatively lightweight (113K) considering its rich content (15 “pages” – each one with a video, captions, photos, and navigation). The site also offers a text version of the Flash movie.

See Appendix B: Resources for more information about Flash accessibility.

# 11. Accessibility Testing and Validation

Accessibility testing and validation is an integral part of the process for making web pages accessible for the disabled. Though many tools exist for web page testing, validation and repair, there is no one tool that can conclusively do the entire job. And there is no substitute for human judgment. The developer needs to have a thorough knowledge of web accessibility to effectively use any tool and to understand its strengths and limitations.

## a. Testing and validation tools

The following list offers a small sampling of the many tools available. For links to other accessibility tools, go to WAI’s web tools page at <http://www.w3.org/WAI/ER/existingtools.html>.

**HTML validation**

HTML validation is perhaps the most important, and the easiest, thing one can do to aid accessibility. A validator checks the document's HTML against a document type definition (DTD) to ensure that the syntax of the HTML is correct. Though there are server-based validators, such as HTML Tidy (<http://www.w3.org/People/Raggett/tidy/>) and the W3C HTML Validation Service ([http://validator.w3.org](http://validator.w3.org/)), the most useful tools are locally installed

Homesite ??????????????????

[http://www.Adobe.com/software/homesite/](http://www.macromedia.com/software/homesite/)

Homesite HTML Editor comes with a built-in validator which is basic, but functional. Homesite will also integrate with the more powerful CSE HTML Validator

CSE HTML Validator ??????????????????

<http://www.htmlvalidator.com/>

This is the standard HTML Validator used by the AKQA Publishing Team front-end developers. It is the most robust Validator and integrates with Adobe’s Homesite ????????????????HTML editor

**Accessibility validation**

**InFocus**

<http://www.ssbtechnologies.com/>

InFocus is the validator used by AKQA teams to check web pages for accessibility compliance. It supports rules customization, batch checking, reporting, and automated correction of errors.

**AccVerify, AccRepair**<http://www.hisoftware.com/>  
The standard version of AccVerify verifies web pages for Section 508 accessibility compliance and reports on the W3C’s recommended guidelines; AccVerify Professional also supports batch files processing and automated checklist and report building. AccRepair uses the reporting/verification components of AccVerify to launch a repair wizard that allows correction of accessibility errors.

**Bobby**<http://www.cast.org/bobby/>  
The online version tests individual web pages for accessibility, and the downloadable application checks multiple local files or an entire web site. Bobby can also analyze web pages for compatibility with various browsers.

**Lift**<http://www.usablenet.com/>  
Evaluates usability and accessibility features of a web site, including site navigability, download speed, graphic quality, and searchability. Lift Online tests live web sites, and the LIFT Onsite software for the Mac tests locally stored sites.

**aDesigner**

<http://www.alphaworks.ibm.com/tech/adesigner>

aDesigner is a disability simulator that helps Web designers ensure that their pages are accessible and usable by the visually impaired. It looks at such elements as the degree of color contrast on the page, the ability of users to change the font size, the appropriateness of alternate text for images, and the availability of links in the page to promote navigability.

**Adobe accessibility tools**[http://www.Adobe.com/Adobe/accessibility/](http://www.macromedia.com/macromedia/accessibility/)

* + **Adobe Accessibility Starter Kit**Includes Dreamweaver and UltraDev Accessibility Kit (extensions, guidelines, and Check Page for Accessibility extension), 508 Accessibility Suite extension; Adobe Flash Accessibility Kit, tutorials, and trial versions of Dreamweaver 4, UltraDev 4, and Adobe Flash 5.
* **Adobe Flash Accessibility Kit**[http://www.Adobe.com/software/flash/productinfo/accessibility/](http://www.macromedia.com/software/flash/productinfo/accessibility/)  
  Provides free sample code to help make Flash (SWF) movies accessible to users with disabilities.   
  For more information about Flash accessibility see:  
  1. Appendix B: Flash Accessibility  
  2. “[Making Flash Accessible](http://www.webaim.org/Articles/accessibleflash)” – summarizes the contents of the Flash Accessibility kit and discusses the limitations of Flash accessibility.  
  3. [Voluntary Product Accessibility Template: Adobe Flash](http://www.macromedia.com/macromedia/accessibility/tools/vpat/flash.html) – offers a few concrete suggestions for Flash accessibility.

**Color Blindness Simulation**

**Vischeck**  
<http://vischeck.com/index.php3>  
Vischeck simulates color blind vision, showing how people with different kinds of color blindness see web pages and images. A Photoshop plug-in or Java version of Vischeck can be downloaded, or single images or an entire web page can be checked directly online

## b. InFocus recommendations ?????????????????????????

In Autumn 2007, AKQA????????.com developers extensively used InFocus to test for accessibility the top 185 AKQA???????.com web pages (based on high traffic volume and importance for product lines).

Based on their experience, they found that:

**InFocus version**: It’s imperative that everyone in a work group use the same version of InFocus. OtherAKQAe, configuration files will not work, and errors will not match up.

**Configuration File**: One configuration file customized to AKQA standards ???????should be created and maintained within a department. It can and should be modified to department specifics (for example, business decisions) and to different platforms and technologies that might trip errors in InFocus as violations.

**Validation**:

* If errors exist even after the configuration file is customized, manually check all these violations and check the code. Not all violations reported by InFocus are true.
* An exceptions list needs to be developed and maintained within a department. For example, InFocus may not recognize data tables or layout tables correctly – need to view in browser, look at code, and make accessibility corrections.
* Do not use InFocus as sole validation for proof of compliance.
* In a dynamic environment, InFocus should only be used as a validation tool, not as a tool to fix violations. JAWS testing is needed.

**Memory**:

* Large URLs may time out an InFocus session. If this happens, save the page as a static HTML page to the hard drive and run it locally. Or clear history and try again.
* Sometimes when reloading a page after making changes to the code, all violations will disappear and the page becomes inaccessible. InFocus will show “no page loaded”. If this happens, the entire history has to be cleared to run the URL again. (This is an InFocus bug that has not been addressed.)

**Speed issues**:

* InFocus will act very slow or time out if history is not cleared.
* The more complex the configuration file, the slower InFocus can navigate in pulling files.
* Use C:\ or local drive for testing URLs. InFocus runs faster locally.

## c. InFocus exceptions

When testing with InFocus, the following exceptions may occur:

* InFocus may report a violation, stating that LABEL is placed too far from the input. However, JAWS will read the form correctly. In this case, InFocus can be disregarded.
* On some forms it was not possible to place the LABEL tags next to the form field, which gave errors with InFocus but tested fine in JAWS.
* If InFocus gives an error for hidden fields, try moving them out of the LABEL tag or placing them somewhere else. Since they are hidden, it should not be a problem to move them around.
* Use clear language warning should be ignored.

## d. InFocus validation limitations

InFocus cannot test for:

* **Content.** InFocus checks a page by checking the code and looks for certain attributes when it encounters an HTML tag. It does not validate content.
* **Correct ALT text.** It only lets one know if the ALT is included.
* **Correct placement of skip navigation anchors.**
* **Correct coding of data tables -** InFocus cannot tell if a table is a data table or a layout table, as it cannot check the meaning of the table content. It also cannot check the content of a table’s SUMMARY attribute. InFocus cannot verify the correct use of TD and TH tags and SCOPE, ID, HEADERS, and AXIS attributes for data cells.
* **Correct use of form labels -** InFocus will check if LABEL tags re used for form controls, but it will not be able to inform the developer if the labels are applied correctly. The developer needs to decide what part of a text needs to be the label for a form field.   
    
  Only listening to the page with JAWS will show if the labels for the form fields are chosen correctly (or make sense to the user). On some forms – for example, Social Security numbers – one label is meant to be used for more than one form field. However, technically this is not possible. The developer will need to add a TITLE attribute to the remaining form fields in the same group. Only use of a screen reader will determine if the code works.

## e. Testing with JAWS

JAWS is the screen reader most commonly used by blind and other visually disabled people to access their computers, including accessing web pages. Reading web pages with JAWS is “real world experience” – the actual experience of a visually disabled person.

JAWS testing is an essential adjunct to the use of any accessibility validator and will catch accessibility errors that other accessibility validators may miss.

#### How JAWS reads web pages

JAWS renders web pages in linear speech and (for English) will read a page from left to right and top to bottom in a logical, linear order. It looks at the HTML code of the page, not at what is visually displayed on the screen. Thus, poorly written code can result in JAWS reading a page incorrectly.

Once a web page is loaded, JAWS creates a buffered version of it based on the document object model in Microsoft Internet Explorer. JAWS uses a buffer so that a point of focus – the Virtual Cursor – can be tracked. Otherwise a web page would be static text with no cursor or point of focus at which interaction with text can take place. If speech is stopped at any point while reading a page, JAWS remembers the point of focus because of the buffered information.

## f. Checklist for JAWS accessibility testing

* **ALT attributes** – Do all images have ALT attributes? Do the ALTs make sense?  
    
  Before testing a page with JAWS, a quick way to check for ALTs is (in Internet Explorer) to mouse over the page’s images to check ALT content. If no ALT appears for an image, the ALT is either missing or is set to an empty value (ALT=“”). Use JAWS or check the code to determine which is true. (For a missing ALT, JAWS will read the file name.)  
    
  All images must have an ALT attribute; otherwise, JAWS will read the file name of the image. If the image is not relevant to a screen reader user – for example, shims or decorative images – the ALT attribute must be null (alt=””).  
    
  The ALT text needs to make sense, especially for images that are also links. You can use the JAWS link list (see below) to determine if the ALT text is useful. Because of the frequent use of link lists by screen reader users, the ALT text also has to make sense by itself and not depend on surrounding content.
* **Skip navigation links** – If a page has a “skip navigation” link, does it work correctly and go to the correct anchor (usually the main content of the page)?   
    
  A page that features sub-navigation may also need to include a “skip secondary navigation” link to skip over the sub-navigation to the main content of the page. Be sure to test for this. But, if the secondary navigation uses a template, testing once for a group of pages using the same sub-navigation should be OK.
* **Repetitive and redundant links** – Do all repetitive links have a unique TITLE attribute that clearly describes the link’s destination? Do redundant links that go to the same destination all have the same link text (and, if included, TITLE attribute)?

| **Testing links: JAWS Links List feature**  Screen reader users often use a links list feature to hear a list of links rather than listening to the whole page content. For JAWS the shortcut “Insert + F7” will list all the links on a page, and the shortcut “CTRL + Home” will focus the cursor on the first link of the list (which can be set either in link order or alphabetically).  This feature is very helpful to test for:   * Invisible “skip links” to make sure they are included where appropriate. * Redundant and repetitive links. |
| --- |

* **Child browser windows** – Does the page have any links that open child browser windows? If so, is the user informed that “This link will open a new browser window” for all child window links?  
    
  If a link opens a child browser window, the user needs to be informed. This is done by using a TITLE attribute in the link tag. Check to see if the TITLE attribute is present by mousing over the link. If nothing is displayed, then the TITLE attribute is not used.  
    
  To make sure JAWS reads the TITLE attribute, check the JAWS Text Link Verbosity setting: JAWS Utilities menu 🡪 Configuration Manager 🡪 Set Options 🡪 HTML Options. For JAWS 4.02, for “Text Link verbosity” make sure “Speak Alt Tag or Title” is selected (this should be the default). With JAWS 4.5?????????, select “Title Only”, click “OK” , and then click the save icon to save the configuration.
* **Content read order** – Is the content read order correct? In most cases, JAWS read order should match what is displayed on the screen.
* **Data tables** – Are there any data tables on the page? Are they accessible?  
    
  Reading data tables with JAWS is more complicated than just reading text, and testing a page with data tables will be slower than testing a page with just text, links and images.  
    
  JAWS users move from cell to cell in a table and hear table cell header values by pressing “CTRL + ALT + down/up/left/right arrow”. If JAWS reads “not in a table,” the table is not accessible.  
    
  However, when reading a table, JAWS will only say either the vertical or horizontal header values in a table, not both at once.   
    
  If JAWS is reading a table moving down from row to row and it encounters a cell containing only a spacer image, it will say “bottom of row” and stop reading. The user will need to use “CTRL+ down arrow” to continue to the next row. However, JAWS will then begin reading from the first cell of the next row, even if the user was previously in another column.
* **Forms** – Are there any forms on the page? Are they accessible?  
    
  Filling out forms using a screen reader is a very complex and slow process. However, all forms on a page need to be tested by turning JAWS forms mode on and filling out the form using the screen reader.  
    
  In JAWS 4.5????? “CTRL + INSERT + HOME” will take the JAWS cursor near the first form field of the first form on the page. Then by pressing “/” key on the number pad the cursor will focus on the form field, and forms mode will be turned on.   
    
  In forms mode, the user can move the focus forward to another form field or a link in the form area by using the TAB key. SHIFT + TAB will move the focus backward. When the focus is on a field, a screen reader should read the label of the field and follow with the type of the field – for example, a text field with the label “First name” should read “First Name Edit.”  
    
  Things to pay attention to when testing forms:
* Verify that JAWS reads the label of the form field
* Be sure the label makes sense when it is read.
* Check the tab order.
* If there is a question with a set of radio buttons and if the label tags are used for the text next to the buttons, the question will not be read by JAWS since the text itself is not a form element. The screen reader user would have to turn the forms mode off the read the question and then turn the forms mode on the answer that question. This is correct JAWS behavior, and it is technically impossible to associate the question with the set of radio buttons without losing one of the labels.
* Field groups like phone numbers or SSN are also tricky. The first text field can be associated with the label using the label tags, and the title attribute can be used for the rest of the fields in the group. For example, an acceptable result for a phone number with three fields could be “Home Phone, Home Phone 2, Home Phone 3” when read by JAWS.
* Drop down boxes can be read by using the down arrow when the focus is on the drop down box. If a drop down menu is used for navigation, the JAWS user needs to click on a button to activate the navigation. (not automatically go to the destination)

## g. Basic JAWS shortcuts

* **CTRL–** Stops JAWS without quitting program.
* **CTRL + HOME –** Brings the cursor to the beginning of the page. For a web page, JAWS will start with the TITLE element content.
* **INSERT + DOWN ARROW –** JAWS will start to read from where the cursor is until the page ends or the CTRL key is pressed.
* **CTRL + DOWN/UP ARROWS –** Will move the cursor to the next paragraph and will only read that paragraph.
* **ENTER and / on NUM PAD –** These shortcuts simulate the left click of a mouse. Either one works to submit a form or choose a link.
* **INSERT + F7 –** Lists all the links on a page.
* **CTRL + INSERT + HOME –** Takes the JAWS cursor to the first field of the first form on the page and will read the label and the form field. After the form field is read using the “\” key on the number pad, this shortcut will focus the cursor on the form field and turn forms mode on.
* Another way to turn forms mode on (works better with JAWS 4.02) is to press the “\” key on the number pad when JAWS is reading the form field.
* **TAB –** Can be used to navigate between form fields or links.
* **PAGE DOWN** – increase voice rate
* **PAGE UP** – decrease voice rate

## h. JAWS miscellanea

* Table cell header values can be heard when navigating data cells by pressing CTRL + ALT + down/up/left/right arrow. However, when reading a table, JAWS will render either the vertical or horizontal header values, not both at once.   
    
  If JAWS is reading a table moving down from row to row and encounters a cell containing only a shim gif for spacing, the screen reader will say “bottom of row” and stop reading. The user will need to use CTRL + down arrow to continue to the next row. However, JAWS will then begin reading from the first cell of the next row, even if the user was previously in another column besides the first one.
* JAWS will read more than one header for a single table data cell if the AXIS attribute is used. However, the AXIS values are read for either the vertical or horizontal header values, not both at once.
* As the ID attribute refers to the **content** within table header tags (<TH>….</TH>), JAWS reads the content referred to. However, JAWS reads the **value** of the AXIS attribute.
* The default link setting for JAWS 4.5 ????? does not read the TITLE attribute of a text link.  
    
  For the websites that want to be compliant with Section 508 or just up to the standards for the accessibility, their home pages code should include an invisible shim link after the corporate logo that links to a page that informs JAWS users that, to best access the site, they need to change their link setting in the JAWS Configuration Manager to read the TITLE attribute of a link rather than the link text. ?????????

| <a href="/help/jaws\_setting.jhtml"><img width="1" src="//a248.e.akamai.net/7/248/1856/5472601571f31e/www.AKQA??????????????.com/img/s.gif" height="1" border="0" alt="JAWS 4.5 users can change text link setting to hear clearer link descriptions." /></a> |
| --- |

.

# Appendix A: Resources

General

Clark, Joe, *Building Accessible Websites,* New Riders Publishing, 2003

<http://www.joeclark.org/book/sashay/serialization/>

Paciello, Michael G., *Web Accessibility for People with Disabilities,* CMP Books, 2000

Excellent overview of web accessibility issues, including legal requirements, standards, and accessible web site design.

Pilgrim, Mark, *Dive Into Accessibility:30 Days to a More Accessible Web Site*, 2007

Online book (http://diveintoaccessibility.org/) that includes disabled user profiles and tips for creating accessible web sites sortable by user, disability, design principle, web browser, and publishing tool.

Legal and other guidelines

Americans with Disabilities Act Document Center

<http://www.jan.wvu.edu/links/adalinks.htm>

Repository for U.S. legal documents regarding disability regulations and standards.

Web Accessibility Initiative

<http://www.w3.org/WAI/>

Major resource for web accessibility development. Provides in-depth guidelines, checklists, detailed techniques, and other resources. A good starting point is the WAI Resources page (<http://www.w3.org/WAI/Resources/>).

Section 508

<http://www.section508.gov/>

Official US government site for Section 508.

PDF accessibility

Acrobat Accessibility web site at adobe.com

<http://www.adobe.com/products/acrobat/solutionsacc.html>

Information about Acrobat 6.0 accessibility features and other resources. In particular, note “Advanced Techniques for Creating Accessible Adobe PDF Files” (<http://www.adobe.com/products/acrobat/pdfs/CreateAccessibleAdvanced.pdf>). Although “Advanced Techniques” was written for Acrobat 5.0, the document includes detailed information on document structure and modification still relevant for Acrobat 6.0.

Baker, Donna L., *Adobe® Acrobat® 6 Tips and Tricks: The 100 Best*, Adobe Press, 2004

This book only covers Acrobat 6 Standard, but it offers a chapter on making documents accessible that includes tips that are also relevant for Acrobat 6 Professional.

Padova, Ted, *Adobe® Acrobat® 6 PDF Bible*, Wiley Publishing, Inc., 2003.

Thorough coverage of Acrobat, including a entire chapter on PDF accessibility and tagged PDF files.

Flash accessibility

Accessibility and Adobe Flash CS3

<http://www.adobe.com/accessibility/products/flash/>

Extensive resource for Flash accessibility.

Celic, Sofia, and Andrew Arch, “Multimedia Accessibility – Flash and the Web Content Accessibility Guidelines,” AusWeb 03 Conference, 2003.

<http://ausweb.scu.edu.au/aw03/papers/arch__with_celic_/>

Excellent resource that describes the applications of the *Web Content Accessibility Guidelines 1.0* checkpoints to the Flash MX development platform and provides techniques for making Flash accessible.

Andrew Kirkpatrick & Bob Regan, “In search of... a perfect plugin technique,” undated

[http://weblogs.Adobe.com/accessibility/archives/2005/08/in\_search\_of\_a.cfm](http://weblogs.macromedia.com/accessibility/archives/2005/08/in_search_of_a.cfm)

Methods, testing criteria, and results are listed

Accessibility

<http://blogs.adobe.com/accessibility/index.xml>

Information and news about accessibility for people with disabilities in Adobe products. There are a few good articles regarding Flash and the Videos.

Flash 8 Help

Includes a “How do I…” section on “Creating Accessible Content.” Can also find much information on accessibility and Flash by searching for “accessibility or accessible.”

Other resources

Brewer, Judy (ed.), “How People with Disabilities Use the Web,” World Wide Web Consortium, W3C Working Draft, 4 January 2001 (update in progress)

<http://www.w3.org/WAI/EO/Drafts/PWD-Use-Web/>

General introduction to how people with different kinds of disabilities use the web, including scenarios of people with disabilities using accessibility features of web sites and web-based applications.

HCI Bibliography: Accessibility Resources

<http://www.hcibib.org/accessibility/>

Almost 200 links to information on making computers, software, and the web more accessible to people with disabilities.

Adobe Accessibility Center

http://www.adobe.com/accessibility/

Accessibility information for Adobe products and some good general accessibility information.

Musciano, Chuck, and Kennedy, Bill, *HTML & XHTML: The Definitive Guide*

(5th Edition), O’Reilly, 2007.

Not an accessibility resource *per se,* but an invaluable resource for in-depth coverage of HTML and XHTML, with related comments about accessibility.

TIA Access, *Extend Their Reach: Marketing to Consumers with Disabilities – Interpreting the Numbers,* online report © 1998-2007.

<http://www.tiaonline.org/access/etr_brochure.html>

Statistical and demographic information about people with disabilities.

Trace Center, University of AKQAconsin: Designing More Usable Web Sites

[http://trace.AKQAc.edu/world/web/](http://trace.wisc.edu/world/web/)

Comprehensive list of web usability and accessibility resources, including web access tools, disability and web use, color blindness, and accessibility organizations and discussion groups.

WebAIM

<http://www.webaim.org/>

Features web accessibility articles, tutorials, and links to tools and products.

Accessify.com

<http://www.accessify.com/>

Tutorials, discussion forums and articles.

Microsoft Accessibility Center

Microsoft Active Accessibility

http://msdn2.microsoft.com/en-us/library/ms697707.aspx

Microsoft Accessibility

http://msdn2.microsoft.com/en-us/library/aa286482.aspx

Accessibility at Microsoft

http://www.microsoft.com/enable/