# Screen readers

**Chapter** · January 2009

CITATIONS
4

READS
642

**2 authors**, including:

Barbara Leporini
Università di Pisa
**142** PUBLICATIONS **1,812** CITATIONS

SEE PROFILE

# CHAPTER 36:
# SCREEN READERS

Table of contents:



## 36.1. Introduction

In this chapter we discuss the main interaction and design issues to consider when using a screen reader, the main Assistive Technology used by people with little or no functional vision when interacting with a computer. Assistive technologies are comprised of hardware and software offerings that serve the needs of persons with disabilities. More specifically, assistive technologies include screen readers, screen magnifiers, alternative keyboards, single switches, voice recognition, and a wide variety of input and output devices that meet the needs of people with disabilities. Usually a screen reader is for blind people while a screen magnifier for low vision users. However, depending on the vision residual several low vision users prefer to rely on a screen reader rather than a magnifier.

### *36.1.1 What is a screen reader*

A screen reader is an assistive technology used by visually impaired people to interact with the computer and other technological devices, such as mobile phones or Personal Device Assistant (PDA). In practice, A screen reader is a software which mediates between the user and the operating system (including its applications), assisting the visually impaired by interpreting the user interface. A screen reader can read aloud the content displayed by means of a voice synthesizer, or it can provide the same content in a written format by using a Braille display. In brief, when a user interacts via a screen reader, the rendered content must be interpreted and then converted into

spoken or haptic output. That is, the screen reader recites in voice synthesizer modality the content of what is displayed on the screen, word by word, line by line, etc., or it can write in a tangible braille modality, so that the users can read it directly by themselves. Braille is a code that maps the characters of a writing system to the six, or in some cases eight, raised dots, which can be read with the fingers by people who are blind. Braille format is mainly adopted and preferred by blind persons who learned the system at an early age and used it regularly over the years for studying, reading and so on. Most users are elderly people who, like elderly sighted people, are more reluctant to learn a new technology. Anyway, Braille output is even slower when working with a computer and with the Internet in particular. With the development of new technologies, the screen reader with voice synthesizer has become the most appreciated tool for blind people. On the other hand, it must be pointed out that using a screen reader is not so easy and, in the beginning, it requires particular effort for the user to learn. In fact, even blind people who are skilled in this technology may not know all the advanced features implemented by this assistive technology. It is also important to emphasize that page content perceived in vocal or Braille format is quite different from that perceived in visual modality.

### *36.1.2 Screen reader classifications*

Several screen readers have been developed, such as Jaws for Windows (http://www.freedomscientific.com/), Windows Bridge (http://www.synthavoice.on.ca/), Hal for Windows (http://www.dolphincomputeraccess.com/products/hal.htm), Windows eyes (http://www.gwmicro.com/), and others. Jaws for Windows [Jaws] is the screen reader most often used and preferred by blind users.

Some programs speak or make other sounds so that they can be used by blind people or people who cannot see the screen. These programs are called self-voicing and can be considered a form of assistive technology. However, it is important not confusing a self-voice application with a screen reader. A self-voicing application is an application that provides an aural interface without requiring a separate screen reader. One its limitation is that the user can just interact with such application and not with others. Differently a screen reader is a software able to work with several general applications, provided they are well-developed (i.e. they are accessible). A prominent group of self-voicing applications are talking Web browsers. Examples of this kinds of applications are Home Page Reader, Connect Out-loud, etc. A more recent trend has seen the self-voicing capabilities added to browsers (e.g. opera) or specific applications (e.g. Acrobat Reader). Such specific applications might be not particular useful for blind people considered they have limited features and provide no navigation reading commands. For example, the user can just read the content in a sequential way. Whereas, in this sense a screen reader allows to move more freely.

Finally, a new recent category of screen readers are developed for mobile devices such as smart phones or handled computer (PDA). Examples are Mobile Speak and Mobile Speak Pocket (http://www.codefactory.es/index2.htm), Talks (http://www.nuance.com/talks/), and so on.

### *36.1.3 History*

As described above, a screen reader is defined as a software component which outputs the information on the screen to a speech synthesis and/or a Braille display as controlled by keyboard operations. It has been a key technology to make computers accessible to blind people. Even before computer displays appeared, the Optacon {??}, which translated print into tactile vibrating images

could be used by the first blind programmers. The Optacon was used until screen readers were developed, even after computer displays were in use.

Before screen readers became available, there was some previous work in the early 1980s on alternative approaches {Microvox, SAID}. For example, SAID became an IBM product as the 3270 Talking Terminal in 1981, but only as a remote terminal for certain IBM mainframe computers. The basic SAID design concept was transferred to the first screen reader for DOS, which became an IBM product in 1988. Later, several other screen readers for DOS became commercially available and they helped blind people around the world use computers.

However, the DOS era ended and the graphical user interface (GUI) became widespread. This created various problems for screen reader developers and users, because GUIs were originally inaccessible without vision. For several years, new computers and new operating systems could not be used by the blind. There were many research challenges to make GUIs accessible, but now the research work in this area has become more proactive. We can see much related work in academia, some of which was funded by the European Union or the US government {GUIB, Mercator}.

Taking full advantage of the fundamental research contributions, screen reader products for GUIs gradually became available in the market. The IBM Screen Reader/2 was the one of the earliest screen readers for a GUI, and the first version of JAWS became a product in 1995. The developers of these screen readers invented various methods to read aloud the text on a screen. The off-screen model (See 36.5) is one of the most important methods, which contributed to make GUIs accessible in the mid-1990s. The scripting capability was also an important invention to make complicated GUIs accessible non-visually (See 36.3.1). Some screen readers handle multiple languages and multiple operating systems. Others were developed in certain countries for local languages (e.g. Japanese, Chinese, Korean, etc.) as well for certain operating system (e.g. UNIX, Mac/OS).

The counterparts of screen reading technologies are the operating systems and applications. On one side, accessibility APIs (application programming interfaces) were developed to make the development of screen readers much easier by exposing the information required for screen reading via an official API channel (See 36.5).

A screen reader is an approach to create a read-aloud interface for the non-sighted treating the display as it is for sighted users. It provides full (or partial) access to existing applications, but sometimes there are limitation on the usability compared to the degree of control enjoyed by sighted users. The “self-talking” approach tries to provide an ideal interface optimized for visually impaired users. The IBM Home Page Reader was an example in this category, providing usable access to Web content. The product offered an advanced interface, which other screen readers have since assimilated.

Currently, screen reading technologies are becoming much more popular and playing an important role in the information technology industry. Various accessibility APIs have been developed and standardized, and also many applications are capable of supporting these APIs. Screen readers are also becoming much more powerful to read the latest GUIs out loud. The challenge of screen reading is still continuing and will never end, since interfaces for sighted people are always evolving. This evolution will not stop, and it means that screen readers must be enhanced to read the latest highly graphical user interfaces.

**Table 1 - Brief History of Screen Readers**

| | |
|---|---|
| 1970 | Optacon |
| 1976 | First Kertzweil reading machine |
| 1982 | Microvox (general-purpose speech machine) |
| 1984 | Talking Terminal for IBM 3270 |
| 1988 | IBM Screen Reader for MS DOS |
| 1989 | JAWS, first version for MS DOS |
| 1992 | IBM Screen Reader/2 for IBM OS2 [Thatcher] |
| 1995 | outSPOKEN for Apple Machintosh |
| 1995 | JAWS 1.0 for Microsoft Windows |
| 1995 | Window-eyes, first version for Microsoft Windows |
| 1997 | IBM Home Page Reader first version [Asakawa] |

## 36.2 Auditory perception VS visual layout (Barbara)

For users interacting by means of assistive technologies, the user interface layout and structure are crucial. When navigating by screen reader the user perceives window and/or page content different from what is on the screen. In this section, we discuss the main differences between visual rendering and aural perception. More details on how a blind user perceives Web content as well as information displayed on the screen is further described in sections 36.3 and 36.4.

### *36.2.1 Overview*

To understand the differences between visual rendering (basically with mouse interaction) and aural perception (usually with keyboard interaction), it is fundamental to know how the screen reader deals with Web contents or displayed information. Additionally, how a blind user works with the application affects user interface perception.

When designing a user interface (UI), developers pay attention to structuring the content in a way the user easily identifies main parts and functions. Moreover, user interfaces are increasingly complex and rich in information in order to make content rendering more attractive and full of functions. If no appropriate technique is used to place and highlight main functions and content, accomplishing a task could be difficult to do. To this end, elements such as position, form, font and colour, component arrangement, styles (bold, underlining, etc.) are used to visualize information. Thus, if such elements are appropriately used, the user is able to easily identify the wanted text and functions.

Design attention is focused on how to place content and components in the user interface. Usually design principles are based on a visual rendering so that the user interface is pleasant. Frequently no consideration on logical flow of elements and content is adequately taken into account. Most assuredly how a user interface is built can heavily affect interaction by users who are constrained to interact through assistive technologies, such as a screen reader.

In order to understand how a screen reader interprets rendered content, it is necessary to differentiate between a Graphical User Interface (GUI) and Web interface case (see sections below for more details).

In software applications, general speaking the screen reader follows the system focus. That means the focal point skips component by component in a limited area. When jumping element by element (usually through Tab key) a blind user perceives each single component outside the context. If the logical flow is not appropriate, a user could be constrained to skip many useless elements before achieving the needed components (i.e. a button, an edit field, etc.). Moreover, in this way, in order to get an overview of the overall user interface (e.g. the current window), the user has progressively to build the mental map by experience. The exploration of a user interface might require a lot of time. In this context, specific visual features used to get attention, such as position, styles and so on, do not take effect on blind users. To allow an easier identification of main components by people who interact through keyboard, alternative strategies and features should be applied. For example, functions available in toolbars - i.e. shown through intuitive icons - should be replicated in menus with a textual labels as well as with shortcuts.

For Web interfaces - i.e. Web pages - the screen reader rearranges the content in order to give it a logical flow. In this case the visual structure (frames, columns, etc.) are lost. That means to logically structure the content so that blind users also are able to perceive others techniques should be used. for instance, when a result is shown in a specific point within the page, specific visual properties are probably used to immediately focalize that result. Others alternative modalities could be used to provide the same feature. Elements like hidden labels, shortcuts can improve Web navigation and content structure perception.

### *36.2.2 Aspects affecting perception*

As already mentioned, a screen reader is a software application that attempts to identify and interpret what is being displayed on the screen. Then, the content reported on the screen can be presented to a blind user as speech through a voice synthesizer or as a braille display. People with some vision often use screen magnifiers, which have various issues in common with a screen reader. For this reason, some problems and difficulties encountered when interacting through a screen reader can also occur when operating via screen magnifier. Herein we will aim to explain the problems encountered by people with vision impairment (blind and low vision) when working with a software application through screen reader or magnifier. However, the screen reader is the assistive technology which apparently causes the majority of content interpreting problems. Thus, the present chapter is mainly focused on screen reader.

In order to simplify user interactions, the developers who design a product should be aware of the principle needs of users as well as their interaction modalities with the system [Leporini]. In fact, proper knowledge on how the screen reader works can make the design process easier.
In sections 36.3 and 36.4 how a blind user works with a GUI or Web page is described by giving appropriate interaction examples. Herein we summarise the main difficulties encountered by a visual-impaired user. Those problems are mainly due to differences between visual layout and aural/haptic perception.

The main problems for a blind person interacting through screen reader are:

- Lack of context – Using a screen reader (or magnifier) the user accesses only small portions of content and may lose the overall context.
- Information overload - The unchanging portions of the site (menu, frames with banners, etc.) may overload the reading because the user has to read through all the items nearly every time, thus slowing down navigation. Moreover, in software application user interfaces are more and more complex; it is mainly due to so much

icons, symbols, toolbars, child-windows, etc. Arrangement of those elements certainly affects the user interface complexity.

- Excessive sequencing in reading the information - The command for navigating and reading can force the user to follow the page content sequentially. This may provoke frustration in the user.
- Keyboard navigation – Usually blind users do not use the mouse function (i.e. pointing, scrolling, selecting, etc.) for moving around the page; but instead move by means of keyboard commands, such as Tab key, arrow keys, and so on. Consequently, navigation around a page is slowed.
- Screen reader interpretation - The screen reader deals with Web page content in a manner that differs greatly from visual rendering. This requires a certain expertise in advanced screen reader and browser commands and orienting oneself within the page content can require considerable effort.

In conclusion, there are several differences between visual layout and aural perception due to the way the screen reader works. Designers should apply specific guidelines and appropriate designs based on the interface's purpose as well as on the need for assistive technologies. They should concentrate on the best possible design in order to improve interface usability for both visual layout and hearing navigation.

## 36.3 Working with a Graphical User Interface

### *36.3.1 Interacting with a Graphical User Interface*

When personal computers appeared in the market, they only had text-based (or character-based) user interfaces. Users could interact with computers with textual commands by using keyboards and this interaction process could easily be verbalized for screen readers. In contrast, systems using a graphical user interface (GUI) require users to "see" the screen and to "point" at targets. This interface is very intuitive for the sighted majority, but it was barely accessible for screen reader users. In the late 1990s, the developers of screen readers faced challenges in dealing with GUIs, seeking methods and architectures to control GUIs using text-to-speech and a keyboard instead of the screen and a pointing device.

The basic concept of an alternative to vision is a keyboard-based exploration method. Sighted users use their eye movements to scan a graphical user interface and find the necessary information for their operations. Blind users move the focal position on the screen by using the keyboard, and the screen reader must read aloud information about the cursor position.

The focus is an operating system property, provided the application is well-designed. The focus indicates the component of the graphical user interface which is currently selected. For example, text entered at the keyboard or pasted from a clipboard is sent to the component which currently has the focus. When the user moves through Tab key, the focus moves among interactive components, also called controls. More precisely, the focus only moves among the interactive interface elements, such as edit fields, buttons, combo-boxes, checkboxes, etc. All content shown on the screen is not interactive, thus it does not receive the focus. what is merely displayed (i.e. what is not interacting) is to be just read (i.e. no action by user is required). Consequently, when moving through Tab key the focus directly skips to interactive elements, thus avoiding non interactive parts (e.g. text). In this way a blind user might have a partial perception of the user interface.

For frequently used commands, applications usually provide shortcut keys. A short cut key is a combination of keys to invoke a command without moving the cursor to the specific GUI component assigned to invoke the command. For Microsoft Windows, lists of keyboard shortcuts are published in places such as their online help pages and their Website {url}. Shortcut keys can drastically reduce the required time for operations, but users need to learn how the keyboard shortcuts work and memorize the frequently used ones to use them effectively with GUIs.

If all applications were developed using only the Windows standard controls and also provided shortcut keys, then the focus movement and shortcut keys would provide a minimal level of access to applications. However, because the Windows standard controls are not visually rich enough and usability for sighted users, modern applications and popular applications may be designed to be visually usable with their own user interface components. This trend requires screen readers to be continually updated to access some applications.

In order to make non-standard visual interface accessible, various techniques are used. Additional exploratory methodologies have been developed to allow blind users to get information on the current window. Screen readers usually make available specific commands to freely explore the content visualized on the screen. For example, Jaws has two modalities: "jaws cursor" and "invisible cursor". The JAWS Cursor corresponds to the system mouse. The invisible cursor is similar to jaws cursor, but it is not interconnected to mouse pointer. Such cursors are used to read static text to which the system focus does not have access, and to access other parts of windows to which the system cursor cannot be moved. Hence, as well as driving the mouse pointer the JAWS Cursor is also used to provide a quick overview of the information currently available on the screen. Usually that cursor is limited to the current window - main application or a dialogue window - in order to avoid confusion due to mixed partial contents.

These exploration operations are extremely time consuming and tedious, so each screen reader provides functions to create “scripts” to automate these operations. Scripts are powerful tools for constructing appropriate readable text from the available information in applications (see Section 36.5). These scripts can be assigned not only to standard operations, but also to special shortcut keys. Screen reader vendors continue to update these scripts for major applications, and to package them with their screen reader products.

Users need to learn how the keyboard shortcuts work and they also need to memorize at least the frequently used ones to effectively work with GUIs. This is the most significant difference between the sighted and the blind in using GUIs, and the difference is extremely critical, especially for beginners. GUIs were originally developed for sighted users to help them use the OS intuitively while minimizing the learning needed to start using a computer. These helpful visual effects in the GUIs cannot be conveyed easily to blind users. Therefore they need to learn how to operate the GUIs by spending more time when they first start using the computer, while sighted beginners can frequently start using it very quickly. The learning curves for sighted and blind users are completely different.{}

### *36.3.2 An interactive example: basic operations*

A navigation method for basic Windows operations while using a screen reader is outlined below.
Note: This is not the only method for non-visual operations with a screen reader.

The start-menu can be opened with Ctrl+Esc or by pressing the Windows key. Screen reader users often register the frequently used applications on the start menu so they can quickly find the

application to be opened. After the start menu is opened, the up and down cursor keys can be used for moving between items. To make this faster, alphabetic shortcut keys can be used. In Figure 1, if "M" is pressed once, the focus moves to Microsoft access, and with the next press of "M", it moves to Microsoft Excel. The screen reader reads the text information at the focus, synchronized with the keyboard movements. When the Enter key is pressed on Microsoft Word, that program is opened.

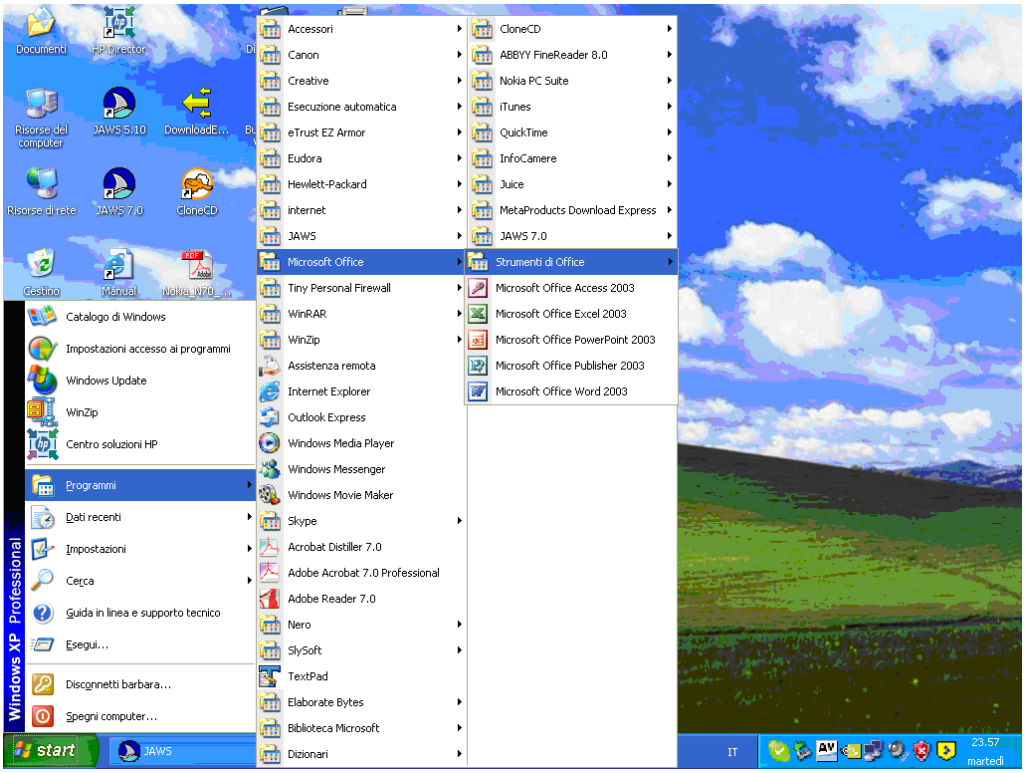


**Figure 1 - Start Menu: the Microsoft Office 2003 menu**

On the initial screen of Word, the focus starts in the editing area, and the screen reader can read the focal information along with the keyboard movements. The Alt key is used to move to the menu and Esc is used to go back to the editing area from the menu area. If the focus is deeper in the menus, for example in the second level menus below the top level, the esc key must be pressed twice to go back to the editing area. The Esc key will go back to a previous layer until the editing area is reached. When the menu structure is logically aligned, the four cursor keys can be used to move in the corresponding directions.

Dialog box navigation is different from the cursor key operations. It basically uses the Tab key. Figure 2 is the dialog to open a file in Microsoft Word. The initial position is in the File name field. And to move to the next field, the Tab is used. To go back to the previous field Shift+Tab is used. In a dialog box, there can be various types of controls, such as another dialog box, a list box, a combo box, an edit box, buttons, or trees. Moving between items in a combo box or list box, the cursor up and down keys are used. In accord with all of the keyboard movements, the screen reader outputs the information on the focal position. The dialog box can be closed by pressing Enter, Esc, or an OK button (if there is one).

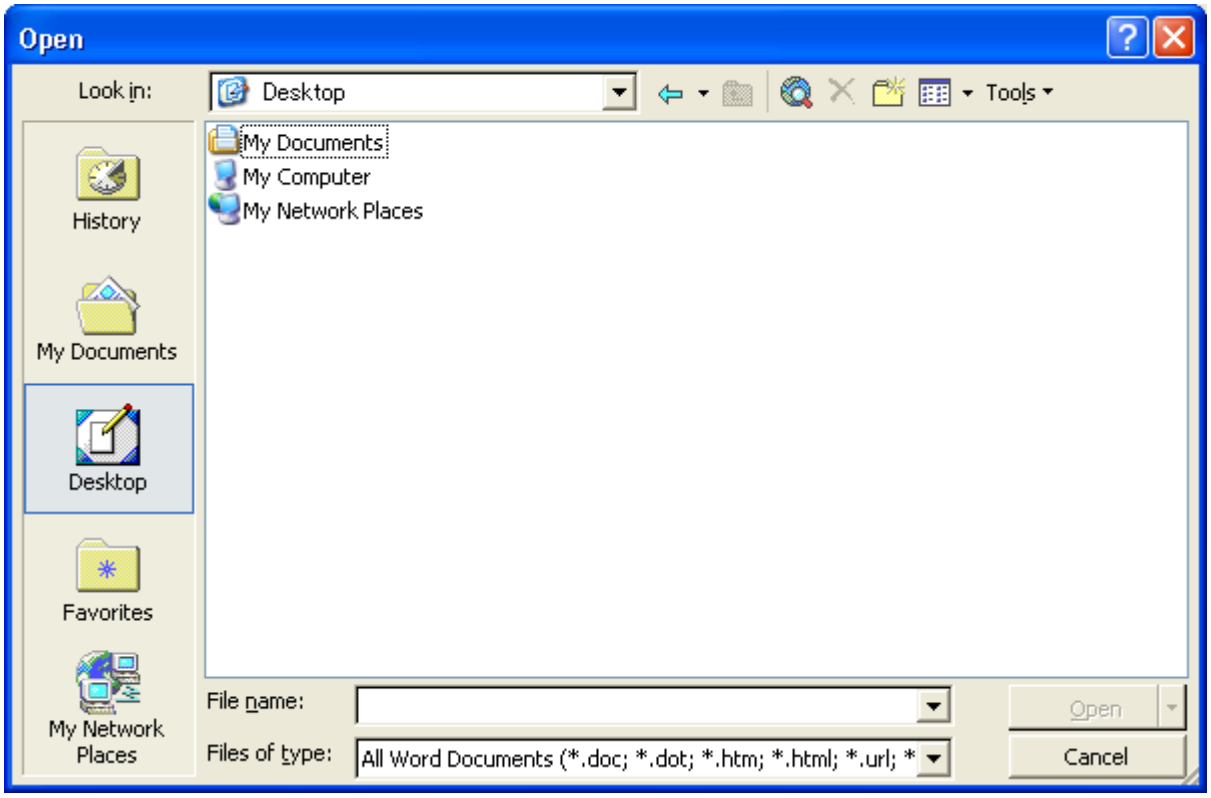


**Figure 2 - Microsoft word: "Open a file" dialogue window**

The basic operations can mainly be done by using the cursor keys, Tab key, Enter, Esc and Alt. For more advanced and effective operations, more keyboard shortcuts should be used and for more effective and quicker operations, screen reader commands be combined.

### *36.3.3 An interacting example: E-mail*

To give an example to better understand how a blind user can interact with a user interface, we consider a common mailer application: Eudora application.
Next figure shows a composition window where an outgoing message is editing. The screenshot shows the Eudora main window (title, menu bar, toolbar, etc.), a child window (i.e. composing window), and on the left, a vertical list of mailboxes is rendered.

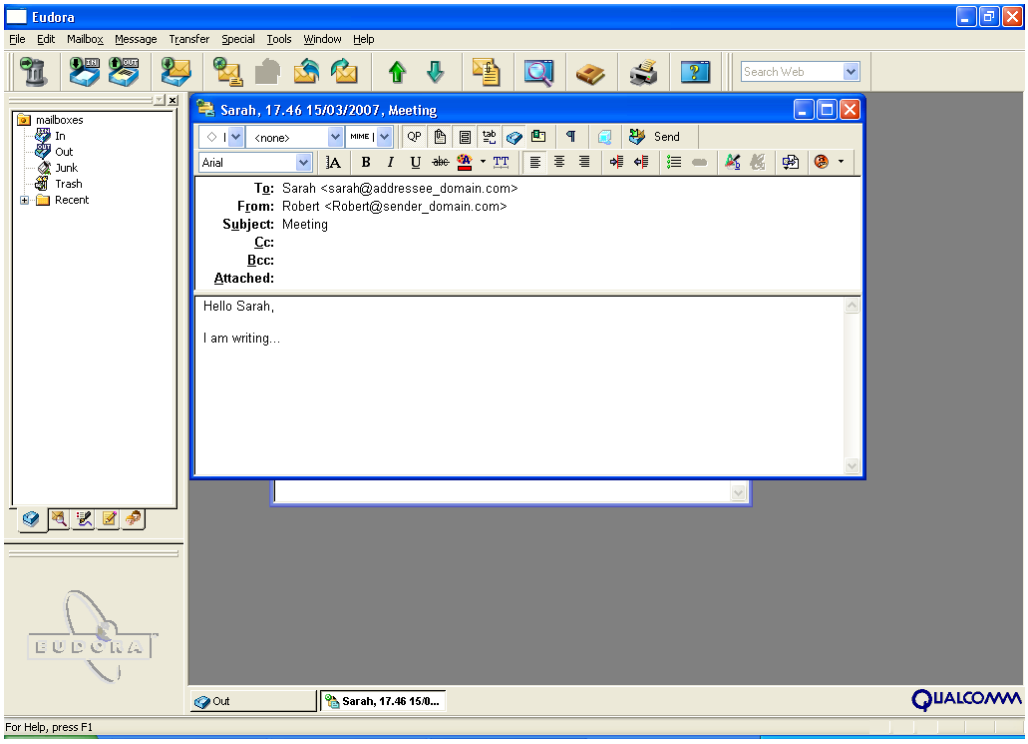


**Figure 3 - Eudora mailer application where the composing window is opened to edit a new outgoing message**

If the user moves through Tab key, the focus shifts among the edit fields: "To:", "Subject", "CC:", "BCC:", and message body. In addition, through a command to activate the menu bar (e.g., Alt key), the user can also move over the menus and submenus. However, in such a situation, the mailboxes shown on the left, the toolbar available below the menu bar and message features (e.g. property, font, signature, and so on) are not reached by the user, because those elements do not receive the system focus.

To explore those parts an exploratory modality is needed. Thus, if the user uses the jaws or invisible cursor, also the toolbar and mailboxes can be read.

In specific, just to give an idea, when the jaws/invisible cursor passes over the menu bar, by moving line by line (up-down arrow), the entire line is read; moving word by word, each component is announced. Thus, when the menu bar is read, the user perceives all menu elements: "file, edit, mailbox, message, transfer,... help". The entire line is read in a single time. Differently, when moving through the focus, each menu item is only read per time; to read the next one, the user must move the focus.

Let us see another case. When in the exploratory mode the cursor is over the two lines corresponding to addressee and sender fields "TO:" or ("From: "), the screen reader announces: "Trash To: Sarah <sarah@addressee_domain.com>" and "Recent From: Robert <robert@sender_domain.com". Actually, the two words "trash" and "recent" are read as first words of each line, because they are shown on the left in the vertical list of mailboxes. The screen reader captures the entire line, including what is displayed outside the child window open to compose a message, but what it is inside the Eudora main window.

In conclusion, when exploring a page through focus movement, the user is in a more functional modality; when using an exploratory mode, the user is able to better understand how the content is placed as well as all those non interacting parts (i.e. elements not receiving the focus). In order to well orientate in a user interface, both modalities are useful and necessary. At beginning, when not

knowing well the interface, the user probably explores through jaws/invisible cursor; next, when he/she has to effectively works with the UI, moving by focus (Tab key) is certainly more functional and effective. However, the exploratory method can also be used to get information on some elements which provide some information to the user (e.g. reading the status bar).

## 36.4 Working with a Web interface

### *36.4.1 Introduction of Web Accessibility*

In recent years, the use of Web sites has been expanding, and the number of users accessing them is steadily increasing. For this reason, it is important that the information be easily reachable by all, including people with special needs. A Web site can be said to be accessible if it can be used by everyone, including people with disabilities. Moreover, in addition to people who explore Web pages by using screen readers (with voice synthesizer or Braille display) or others assistive technologies, we can include those using low-bandwidth technology like cellular phones, black and white screens, speaking browsers via telephone, etc. For this reason accessibility is complicated by the fact that a Web site is not a published piece of work so much as a living document that can be interpreted in different ways by different browsers and on different platforms.

In the context of Web site design, accessibility is a measure of how easy it is to access, read, and understand the content of a Web site. Web applications feature significant differences with respect to classical graphical user Interfaces. Basically the structure of the Web site (e.g., number of pages, links between pages, etc.) is directly related to the information that is available on the Web site (amount, type, etc), while the structure of Graphical user interfaces is usually static. Therefore the user interaction as well as the screen reader interpretation differ from stand-alone applications.

In order to render a Web site more accessible, the Web designer should follow some simple criteria and guidelines which do not limit the graphical features (such as images and icons). To improve accessibility it would better to provide a text equivalent information combined to each multimedia content. There are various organizations that issue standard guidelines: some are international (such as the W3C/WAI), others have national standards (such as the section 508 [section 508]). Often they share the same body of knowledge but are slightly different, which designers have to consider in some contexts.

The Web Accessibility Initiative Interest Group[1] (WAI-IG), of the World Wide Web Consortium, investigates problems in accessing Web resources and produces guidelines for Web content, authoring tools and user agent accessibility. A set of 14 main guidelines, called Web Content Accessibility Guidelines 1.0 [WCAG 1.0], has been created by the group which is now working on version 2.0 of the Recommendation [WCAG 2.0]. In the United States other accessibility guidelines partially overlapping WCAG 1.0, have been defined by the Government[2] and other guidelines are specified by several national or international organizations.

General speaking, making a Web site more accessible and usable requires considerable effort by developers in handling Web page code and many specific design guidelines for various reasons: they have to decide which principles to use for the specific case, how to apply them, and when. Evaluating Web pages requires a lot of effort as well. From this perspective, several tools are

[1] The Recommendation developed by the WAI-IG and new works of the W3C group, are available at http://www.w3.org/TR/WCAG10/ (dated 5 May 1999) and http://www.w3.org/TR/2004/WD-WCAG20-20040730/

[2] Further information available at: http://www.section508.gov.

proposed in literature in order to support designers and evaluators in handling guidelines with Web sites.

### *36.4.2 Interface of voice Web browsers*

The screen reader deals with Web page content in a manner very different from visual rendering. Interaction requires a certain expertise in advanced screen reader and browser commands, and orienting oneself within the page contents can require considerable effort. However, user interaction is greatly improved and simplified if the content of Web interfaces is appropriately arranged and structured according to logical flow.

Screen readers are capable of various navigation commands to provide full access to accessible Web content. The framework of Web accessibility helps support non-visual access by providing three main types of information about the Web pages. The first one is text equivalents for visual objects, such as alternative texts for images. The second type is landmarks to support non-visual navigation, such as heading tags. The third type is semantic structures, such as table headers for a table. Any screen reader is capable of verbalizing such information, and that allows users to navigate the page by using these landmarks and semantic structures.

Concerning Web pages, the new generation of screen readers interprets the HTML code as it is structured, thus considering different tags, while the first generation could not, generating a continuous line of information without any kind of separation. Despite the possibility of recognizing different tags in a page (such as tables, headings, lists, etc) some common problems are still present.

The JAWS screen reader takes the page's tag structure and serializes the content (text, links, edit fields, buttons, cells, and so on). Also frames or blocks  are lined up, without taking into account any specific positions assigned by CSS properties. The Web page is dealt with by the screen reader as a document with a virtual cursor that allows moving line by line or word by word using the arrows keys, or link by link using the Tab key.

Screen readers such as JAWS interpret the code as it was written and lines up the page content in the form of a single column. Thus, the order in which the blocks , the frames and the paragraphs <p> are written is very important. Consequently, the content flow interpreted by the screen reader could be very different from its rendering on the screen. A left menu, for example, could be read after the content on the right if its code is written afterward corresponding to the content on the right. Its rendering on the left can be arranged by CSS properties (e.g. flow: left). Also tables are linearized. When a table is linearized, the contents of the cells become a series of paragraphs, one after another. Cells should make sense when read in row order and should include structural elements (creating paragraphs, headings, lists, etc.) so the page makes sense after linearization.

Usually a Web page is explored by the Tab key, link by link, by arrow keys, line by line or word by word; or through special commands to jump towards certain page blocks. However, it is important to remember that a blind user usually prefers to visit the page link by link (by Tab key) or using arrow keys to read in a sequential way like in a document. Special commands are mostly used by experienced users to move quickly around the pages. Furthermore, many special screen reader commands operate well only if the developer has applied specific tags or attributes, or appropriate criteria have been followed. Hence, it is important to favour navigation via keyboard by assigning a

scale of importance to the links, applying shortcuts to main elements, using specific tags such as <Hn>, and so on.

Also for Web pages, what is offered by a visual layout differs from one provided for aural perception. Often when developers design a Web page they provide some useful information by means of visual features, such as position, colour, separating blank spaces, formatting features, and so forth. For example, some secondary information is put on the side so that users can recognize it immediately. It is important to provide the same "message" to a blind user by another means (e.g. using a table, a heading, a hidden label, etc.).

In conclusion, the screen reader announces every word on a page, line by line, word by word, link by link, in a sequential way. Due to the drawbacks described in 36.2, reading a Web page can be somewhat laborious. Therefore, by following specific design criteria as well as by applying appropriate principles the Web navigation can result more efficiently and satisfactorily.

### *36.4.3 An example of Web interaction*

In order to understand how appropriate tags can positively affect the Web interaction, we report an example. let us assume that this chapter is available in a poorly structured HTML format (e.g. with no heading styles). In order to find the main sections of the document, such as 36.1, 36.2, etc., a blind user can try to look for numbered sections by searching for ".1", ".2", ".3", and so on; if the paragraphs are numbered, the reader could get an idea of the document structure by searching for progressive paragraph numbers. Although this approach would allow users to find the main sections of the page, it is not at all suitable. On the other hand, if sections and pages are not numbered, all the drawbacks described above would have a negative impact on both Web page navigation and reading Web content. To get an overview of the page, the user has to read it in a sequential way to get information on its structure. Thus, even if a page is available in HTML format is considered accessible, the reader can still find it difficult to read it efficiently. When reading a document sequentially (line by line), the user may only obtain an idea of the document structure after reading the entire content. But that process could require a lot of effort.
Whereas, the inclusion of appropriate features, such as heading styles when the document is prepared, enables the reader to obtain an overview of the content. Figure 4shows the chapter structure. The list of paragraphs and sub-paragraphs is generated on the fly through a Jaws command (Insert+F6). The user can understand the chapter structure by reading the list of paragraphs through the arrow keys. By pressing the Enter key on a title the focus skips to the corresponding content in the chapter. So the navigation through the chapter is easier and faster as well.

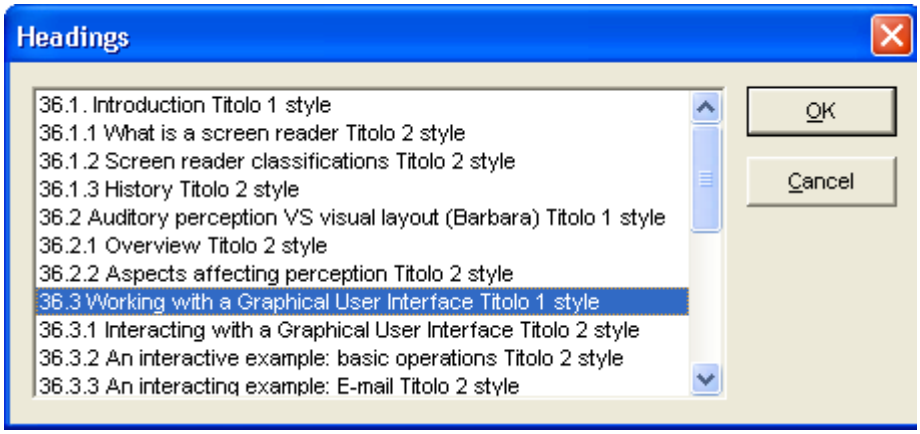


**Figure 4 - Chapter structure generated through a Jaws command**

## 36.5 Basic Technologies for Screen reading

In order to support screen reading, it is necessary to obtain the information that is on the screen, in synchronization with the user's keyboard operations. Over the years, screen readers have used various methods to obtain visual information. Figure 5 shows a typical architecture of a GUI-based operating system and some methods for screen reading. One of the oldest methods is the "off-screen model", which is a method to intercept the display driver function calls or to inspect video memory, and then create a text-based data model for screen reading. This technology was the key technology to obtain text information from GUI components, especially in the early days of GUI accessibility efforts. However with this approach sometimes the screen reader needs to use unsupported functions of the operating systems. That is why this method is not recommended by operating system venders. This technical conflict between accessibility enablement and the use of unsupported functions led to fully supported API-based methods.

An accessibility API (application programming interface) is an official API for screen reading supported by GUI operating system venders. Each modern GUI system provides an API for screen readers. Examples include Microsoft Active Accessibility (MSAA) for Windows, the Java Accessibility API for Java, and the Gnome Accessibility API for GNOME systems. IAccessible2 is a platform-independent standard API. These APIs share the following sets of functions:

- Event mechanisms
Mechanisms to notify the system of events calling for verbalizations, such as when a button has been pressed, the focus on a widget has been changed, or a key has been pressed.

- Widgets role information
Information about the types of GUI widgets, such as buttons, check boxes, slidebars, and pull-down menus. The roles are read to notify the user of the types of operations available for an active widget.

- State of widgets
Information about the status of each GUI widget. Each Widget has a predefined set of available "states", such as checked or unchecked for check boxes and open or closed for nodes in tree views.

Screen readers can obtain this information through the supported API channel. One of the problems with this approach is that each application should behave properly to expose the information. Each application vendor is responsible to appropriately implement the API capability in their applications, but they are not accessibility professionals. Screen reader vendors have expertise, but they cannot change the non-visual interfaces implemented by application vendors. And also it is hard to improve the non-visual usability independently of the application release cycle when using this accessibility API approach. That is why the off-screen-model approach is still widely used by screen readers.

The "internal model" approach is the other method used to obtain information. The internal model is an abstract data structure underlying the graphical representation. It is useful to create a semantically meaningful non-visual interface, since the internal model is a structured and semantically rich internal representation of the information on the screen. This approach is widely used for reading content, such as Web pages and word processor documents. Usually, the internal models are not visible outside of an application, but some applications provide access to internal models to allow application developers to compose applications. One example is in Internet Explorer (IE). IE provides access to the internal content model, the HTML document object model (DOM)) to allow other custom applications (e.g. Visual Basic applications) to use IE each component as a general Web container as a part of their user interface. That is how screen readers

can obtain live HTML DOM, and provide non-visual access based on it. Before this method became available, each screen reader could only track the Web browsers by using the off-screen-model. It was impossible to provide usable interfaces based on tag structures as described in Section 36.4.

Modern screen readers are using these three methods (off screen model, accessibility API, and internal model) simultaneously to provide maximum access to visual information. Because of progress in the visualization of graphical user interfaces, the off-screen model approach is becoming less and less successful in obtaining meaningful information. Therefore, it is expected that future screen readers will fully utilize the internal models and the accessibility APIs.

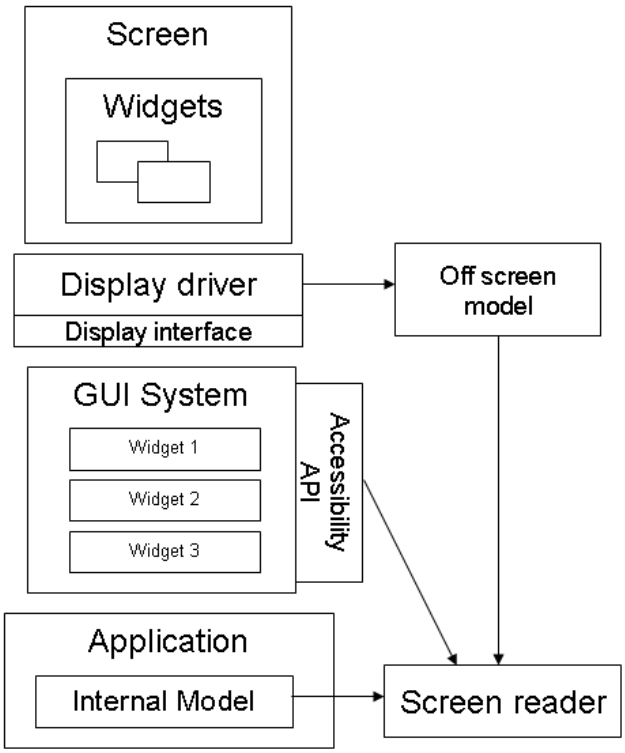


**Figure 5 - Typical architecture of GUI (graphical user interface) systems and methods for screen reading**

## 36.6 Designing a UI: some suggestions

### *36.6.1 General issues*

Usually the analysis of digital obstacles for the disabled only addresses accessibility, although usability is fundamental for simplifying both navigation and interaction for users using assistive devices or those with special needs.
The difficulties in providing universal access can be addressed through the application of the principles of accessibility and usability. Usability is a multidimensional concept, since it can refer to several aspects whose importance depends on the application domain. Although accessibility and usability are closely related, accessibility is aimed more specifically at making a Web site or application available to a wider user population, while the goal of usability is to make users' experience with the Web site more efficient and satisfying.

Therefore, for a blind person it is very important that an application is not only accessible, but also usable. An assistive technology – like a screen reader – gets information from the User Interface (UI) of the application or Web page. Therefore, it is necessary that a user interface is firstly accessible to allow the screen reader to obtain the information needed. Additionally, in order to facilitate the interaction by a blind user it is also important that the interface is usable. Thus, an appropriate design of the user interface is crucial for improving the accessibility and the usability of a system.

The user interface plays a crucial role in the correct and productive use of a search engine. It is not sufficient for the interface to be accessible - it must also be user-friendly, i.e. easy to use and navigate by all, regardless of any devices used. Accessibility guarantees use to anyone; accessible design ensures graceful transformation, as well as understandable and navigable content. Usability renders Internet navigation or application interaction more effective, efficient and satisfactory.

In order to make a graphical interface and Web sites accessible and usable, several guidelines and criteria have been proposed. However, when designing and developing an application, various aspects should be taken into account. Firstly, designers should be aware which UI features are affected by accessibility and usability issues. In addition, developers should be aware of how an assistive technology such as a screen reader works and interprets the user interface.

### *36.6.2 GUI Design*

When designing a GUI interface the developer should keep in mind how a blind user usually works with the interface. In section 36.3 interaction with a GUI has been described and explained. Starting from this point of overview, when designing a GUI, the main aspects to consider could be:

- *Focus*: Which component that should have the default focus and how focus should move between components is a difficult but important problem in user interface design. Giving the wrong thing focus means that the user has to waste time moving the focus. Whereas, giving the right thing focus can significantly enhance the users' experience. When defining the logical flow assigned to interactive components - i.e. the Tab order visiting - the designer should think of the more appropriate order. The Tab order of the controls on a UI determines the sequence in which the focus will change when the user strikes the tab key. Usually the tab order is right to left within each row of controls. Thus a logical order is assigned to interactive elements. When designing a UI, the developer should think about the most logical order according to the task that the user wishes to accomplish.
- *Functions* and commands *selection*: A non-visually impaired user typically performs a simple function by simply clicking on a graphical icon. A graphical toolbar composed of several intuitive icons is probably the selection modality preferred by the sighted user. As selecting modalities should be provided in order to make easier function selection, typically in user interfaces there are several toolbars composed of many graphical icons. While selecting one by a click is generally easy, even if various kinds of screen readers are allowed to simulate a mouse pointer, reaching a toolbar icon is not simple for a blind person. Moreover, in some cases it is not possible because the screen reader is not able to correctly recognise the graphical icons. Hence, an alternative modality should be developed. Firstly, explicit menu items in the menu bar should be added to the user interface. Furthermore, shortcuts should be added to the main functions. In this way, a user who interacts through a keyboard can select a function or command by exploring menus and submenus or alternatively by simply pressing a key combination (e.g. Ctrl+s for Saving).

- *Arrangement of content:* this point is quite relevant since value-enhancing features are more "visible" when located in an area that is rapidly encountered by eye movement and does not require repeated mouse clicks to open sub panels or windows. In the case of the visually-impaired the most relevant features/functions should be placed in a “relevant” position which means at the top of the page or in an easily-reached point. Alternatively a quick modality to reach main UI elements should be designed. To this end, application content could be placed in child windows which can be more accessible for all. Actually they can be placed side by side (for sighted users) and at the same time they can be arranged in a cascade modality which allows blind users to quickly shift one to another. Effectively, when multiple panels are used to split content by using only a main window, moving the focus from one part to another could not be easy and it could cause confusion.
- Additional multimedia features: a visual representation can communicate certain kinds of information much more rapidly and effectively than other methods. For this reason the interface design should try to maintain the same degree of expression in both the visual and aural versions. For example, to inform the user about a particular successful or failed event, in addition to a visual graphical or textual message, a short sound could be played. Moreover, if different sounds are used for diverse events, a blind user can easily understand process status. In practice, we suggest communicating information as well as event status through various modalities: textual, graphical and aural.

### *36.6.3 Web UI design*

The developer should be aware of how the screen reader handles the Web page layout, and how blind or visually-impaired users perceive page content and interact with the interface.
To make a Web page and sites accessible, developers have to apply accessibility guidelines available in literature. In addition, in order to make Web sites and services more accessible and also usable, some additional aspects should be considered. To this end, the main issues in UI design to take into account in order to facilitate Web navigation are:

- *Page content structure*. The page content should be structured in sections and sub-sections by using heading levels. In some cases also adding hidden labels, which have no visual impact, but which are captured by the screen reader.
- *Quick navigation*. Assign a scale of importance (using the tab index attribute) so users can reach the most important elements quickly. A lower value should be assigned to secondary links. Furthermore, shortcuts may be associated with main elements (e.g. navigation bar, login edit fields, etc.) and links to pages of results.
- *Additional multimedia features*. What is offered by a visual layout differs from one provided for aural perception. Often when developers design a Web page they provide some useful information by means of visual features, such as position, colour, separating blank spaces, formatting features, and so forth. For instance, some secondary information is put on the side so that users can recognize it immediately. It is important to provide the same “message” to a blind user by another means (e.g. using a table, a heading, a hidden label, etc.).

## 36.7 Future Directions

Screen readers have been evolving to provide better user interfaces for visually impaired users. It is a great success that users can access and control visual interfaces by using non-visual methods. Basic access has established and various efforts are continuing to maintain the level of access, even

as visual interfaces are becoming much more complicated and rich. However, usability is limited because of the large number of necessary key combinations and the complexity of the operations.

The primary future challenge for screen readers is obviously the improvement of usability. For sighted users, GUIs can be operated by applying the simple principle of "see and click". This simplicity drastically expanded the usage of computer from engineers and scientists to anyone in society. At the same time, the non-visual access provided by screen readers is too complex and there are no general principles for non-visual operations. It is clear that next generation screen readers should provide simpler and more usable interfaces for a wider variety of visually impaired people.

We already see possibilities for the next generation. The framework of Web accessibility is one of the avenues leading to these advances. In this framework, voice browsers obtain information from the active HTML structure, and they can provide logical navigation interfaces for users. The interface will be totally different from the "see and click" world, but optimized for non-visual use with a limited number of key combinations. Simple user interfaces will contribute to shortening the learning curves for new applications. These advances will contribute to improve education and job opportunities for visually impaired people. We hope screen readers will evolve in these directions, and finally implement simple principles for non-visual access in the near future.

## References

[Jaws] Freedom Scientific. Low Vision, Blindness and Learning Disability Adaptive and Assistive Software and Hardware Technology. http://www.freedomscientific.com/

[Leporini] Leporini B., Paternò F. Increasing usability when interacting through screen readers. In: International Journal Universal Access in the Information Society, Vol. 3 n. 1 (2004), pp. 57-70. Springer Verlag, 2004.

[Section 508] Section 508: The Road to Accessibility. http://www.section508.gov/

[WCAG 1.0] Web Content Accessibility Guidelines 1.0. Web Accessibility Initiative (WAI), World Wide Web Consortium, 1999. Accessible at http://www.w3.org/wai

[WCAG 2.0] Web Content Accessibility Guidelines 2.0, W3C Working Draft February/March 2007 available at http://www.w3.org/WAI/GL/WCAG20/