

From the web page
<http://www.ehealth.va.gov/508/flash/>

Contents

- [Creating Accessible Flash Course – Introduction](#)
- [Introduction](#)
- [Section 508 Standards](#)
- [Assistive Technology \(AT\)](#)
- [Flash](#)
- [Testing for Accessibility](#)
- [About this Course](#)
- [VHA Section 508 Help](#)
- [Course Overview](#)
- [Introduction](#)
- [Accessible Flash](#)
- [Adobe Flash](#)
- [The Accessibility Panel](#)
- [Overview](#)
- [Overview, continued](#)
- [How to Access the Accessibility Panel](#)
- [Enabling Accessibility](#)
- [Introduction](#)
- [User Perspective: Incorrect WMode](#)
- [User Perspective: Application-Level Accessibility Not Enabled](#)
- [Enable Accessibility](#)
- [Provide an Accessible Name](#)
- [Example: Accessibility Enabled](#)
- [Conceptual Knowledge Checks](#)
- [How to Enable Accessibility at the Application Level](#)
- [Using ActionScript](#)
- [How to Set the Window Mode Parameter to Enable Accessibility](#)
- [Technical Knowledge Checks](#)
- [Hiding Flash](#)
- [Introduction](#)
- [User Perspective: Invisible Controls](#)
- [How Much to Hide](#)
- [When is Flash Decorative?](#)
- [Individual Elements to Hide](#)
- [Animation](#)
- [Conceptual Knowledge Checks](#)
- [How to Hide Flash Applications from AT](#)
- [How to Hide Individual Flash Elements from AT](#)
- [How to Hide Child Objects from AT](#)
- [VHA Section 508 Checkpoints: Animation](#)
- [How to Limit Looping](#)
- [Technical Knowledge Checks](#)
- [Working with OS/AT Accessibility Features](#)
- [Introduction](#)

- [Windows OS Built-in Accessibility Features](#)
- [Windows OS Accessibility Utilities](#)
- [Interoperability with AT](#)
- [Conceptual Knowledge Checks](#)
- [How to Test Flash for Interoperability with Windows OS Accessibility Features and Utilities](#)
- [How to Test Flash for Interoperability with AT](#)
- [VHA Section 508 Checkpoints: Operating System Interoperability](#)
- [The best way to test your Flash application for compatibility with AT is to try it out yourself. Remember that interoperability with AT is a basic requirement and does not eliminate the need to meet other Section 508 accessibility requirements.](#)
- [Designing for Accessibility](#)
- [Introduction](#)
- [User Perspective: Inaccessible Link Text](#)
- [Provide Meaningful Link Text](#)
- [Provide Accessible Instructions](#)
- [VHA Section 508 Checkpoints: Accessible Instructions](#)
- [Use Images Consistently](#)
- [Use Standard Character Sets](#)
- [VHA Section 508 Checkpoints: Standard Character Sets](#)
- [Settle Screen Transitions Quickly](#)
- [VHA Section 508 Checkpoints: Screen Transitions](#)
- [Conceptual Knowledge Checks](#)
- [How to Create Accessible Links](#)
- [How to Ensure Consistent Use of Images](#)
- [Technical Knowledge Checks](#)
- [Avoiding Flicker](#)
- [Introduction](#)
- [Flicker Requirements](#)
- [Conceptual Knowledge Checks](#)
- [How to Ensure Safe Flicker Rates](#)
- [VHA Section 508 Checkpoints: Flicker](#)
- [It is best to avoid flickering content whenever possible. If your Flash application must include content that blinks or flickers, make sure that it flickers in a range that will not cause seizures. You can manually test your content or you can use a tool designed for this purpose.](#)
- [Technical Knowledge Check](#)
- [Using Color](#)
- [Introduction](#)
- [User Perspective: Insufficient Contrast](#)
- [Provide Sufficient Contrast](#)
- [VHA Section 508 Checkpoints: Sufficient Contrast](#)
- [Links to color and contrast analysis tools are provided in the Resources section of this course. Procedures for using one of these tools are provided later in this lesson.](#)
- [Honor User-Selected Color and Contrast Settings](#)
- [Provide Options for Color and Contrast](#)
- [User Perspective: Color Coding](#)
- [Avoid Color Coding](#)
- [Conceptual Knowledge Checks](#)
- [How to Ensure Sufficient Contrast](#)
- [VHA Section 508 Checkpoints: Sufficient Contrast](#)
- [When you have entered foreground and background colors and selected Luminosity as your algorithm, the Colour Contrast Analyser displays results in a ratio that you can compare to the functional criteria in the VHA Section 508 checkpoints. In the examples above:](#)
- [How to Provide Color and Contrast Options](#)
- [How to Ensure There Are Alternatives for Color-Coded Content](#)
- [Technical Knowledge Checks](#)
- [Providing Controls for Audio, Video and automatically Updating Content](#)
- [Introduction](#)

- [User Perspective: No Video Controls](#)
- [Provide Video Controls](#)
- [User Perspective: No Audio Controls](#)
- [Provide Audio Controls](#)
- [Conceptual Knowledge Checks](#)
- [How to Provide Accessible Multimedia Controls](#)
- [How to Provide Step-Through Controls for Animation](#)
- [VHA Section 508 Checkpoints: Animation](#)
- [Testing your step-through animation controls in Flash includes ensuring that the appropriate controls exist and function appropriately and ensuring that the controls are accessible. To test an animation for appropriately functioning step-through controls:](#)
- [How to Provide Controls for Auto-Updating Content](#)
- [VHA Section 508 Checkpoints: Auto-Updating Content](#)
- [Testing controls for automatically updating content in Flash includes ensuring that the controls exist and function appropriately and ensuring that the controls are accessible. To test for appropriately functioning controls for auto-updating content:](#)
- [How to Provide Accessible Audio Controls](#)
- [VHA Section 508 Checkpoints: Audio Controls](#)
- [To ensure that your application has appropriate audio controls:](#)
- [Technical Knowledge Checks](#)
- [Providing Captions and Visual Indicators for Sound Cues](#)
- [Introduction](#)
- [User Perspective: No Audio](#)
- [User Perspective: Transcript Only](#)
- [The Role of Transcripts](#)
- [Captioning Requirements](#)
- [Approaches to Captioning](#)
- [Overview of One Approach to Captioning: Embedding FLV Cue Points](#)
- [Overview of Another Approach to Captioning: Using a Timed Text File of Captions](#)
- [Captioning Tools and Services](#)
- [Providing Visual Alternatives to Sound Cues](#)
- [Conceptual Knowledge Check](#)
- [How to Use the FLVPlaybackCaptioning Component](#)
- [VHA Section 508 Checkpoints: Captioning](#)
- [One way to add captions in Flash is to use the FLVPlaybackCaptioning component. To use this component, you must add it to the FLVPlayback component and set the properties for each instance of the component, using the Property Inspector or the Component Inspector.](#)
- [How to Provide Visual Alternatives to Sound Cues](#)
- [Technical Knowledge Checks](#)
- [Using Audio for Visual Information](#)
- [Introduction](#)
- [User Perspective: No Visuals](#)
- [User Perspective: No Response](#)
- [Choosing Audio or Text for Describing Visual Information](#)
- [Audio Description Requirements for Visuals with Synchronized Audio](#)
- [VHA Section 508 Checkpoints: Audio Description for Animation](#)
- [Note that there is no exemption for Flash files playing videos previously recorded without implicit audio description. The only exemption is for multimedia created for some purpose other than supporting the agency's mission, such as a video of an employee's birthday party.](#)
- [Approaches to Audio Description for Visuals with Synchronized Audio](#)
- [Audio Equivalent Requirements for Purely Visual Elements](#)
- [VHA Section 508 Checkpoints: Audio Equivalents for Animation](#)
- [Approaches to Audio Equivalents for Purely Visual Elements](#)
- [Conceptual Knowledge Checks](#)
- [How to Add Explicit Audio Description to a Video](#)
- [How to Add Audio Equivalents to the Timeline](#)
- [VHA Section 508 Checkpoints: Audio Equivalents for Animation](#)

- [Animations and slide shows often use a timeline to time visual displays and movements. Audio equivalents can be added to play as the animations or slide shows play. If the slide show or animation has its own soundtrack, you can see both waveforms in the timeline to synchronize them. If it is difficult to synchronize them, consider adding user controls, covered in the next topic.](#)
- [How to Add Audio Equivalents to a User Interaction](#)
- [Technical Knowledge Checks](#)
- [Providing Text Equivalents](#)
- [Introduction](#)
- [User Perspective: Missing Information](#)
- [Choose Text or Audio for Describing Visual Information](#)
- [Identify Content Requiring Equivalents](#)
- [Create Good Text Equivalents](#)
- [VHA Section 508 Checkpoints: Text Equivalents](#)
- [These can be summarized as follows.](#)
- [Conceptual Knowledge Check](#)
- [How to Provide Text Equivalents for Graphic Symbols](#)
- [How to Include Hierarchy Information in Text Equivalents](#)
- [How to Provide Text Equivalents for Animated Content](#)
- [How to Provide Text Equivalents for Progress Bars](#)
- [How to Provide Text Equivalents for Window & Dialog Titles](#)
- [Technical Knowledge Checks](#)
- [Ensuring Keyboard Accessibility](#)
- [Introduction](#)
- [User Perspective: Mouse Required](#)
- [User Perspective: Need for Shortcuts](#)
- [Provide Keyboard Accessibility](#)
- [Objects that Must Be Keyboard Accessible](#)
- [VHA Section 508 Checkpoints: Objects Requiring Keyboard Access for Animation](#)
- [The lesson on Maintaining Focus also contains content related to this section. Objects that must be keyboard accessible can be summarized as follows.](#)
- [Functionality that Must Be Keyboard Accessible](#)
- [VHA Section 508 Checkpoints: Functionality Requiring Keyboard Access](#)
- [These can be summarized as follows.](#)
- [Creating Good Keyboard Shortcuts](#)
- [Conceptual Knowledge Check](#)
- [How to Make a Movie Clip Button Keyboard Accessible](#)
- [How to Add Keyboard Shortcuts](#)
- [How to Make Read-only Text and Controls Keyboard Accessible](#)
- [How to Make Changing Dynamic Text Keyboard Accessible](#)
- [Technical Knowledge Checks](#)
- [Controlling Reading Order and Tab Order](#)
- [Introduction](#)
- [User Perspective: Incorrect Reading Order](#)
- [Definitions](#)
- [Requirements for Accessibility](#)
- [Control in Flash](#)
- [Allow Users to Skip Repetitive Navigation Links or Buttons](#)
- [Conceptual Knowledge Checks](#)
- [How to Control Reading Order and Tab Order](#)
- [How to Control Reading Order for Rollovers](#)
- [How to Control Reading Order for Error Messages](#)
- [How to Allow Users to Skip Repetitive Navigation Links or Buttons](#)
- [VHA Section 508 Checkpoints: Skipping Repetitive Navigation Links](#)
- [To allow users to skip repetitive navigation links or buttons, you can change the reading order, or you can provide a Skip Navigation link or button.](#)
- [Technical Knowledge Checks](#)
- [Maintaining Focus](#)

- [Introduction](#)
- [User Perspective: No Visual Focus](#)
- [User Perspective: Forced Focus Changes](#)
- [Focus in Flash](#)
- [Requirements for Accessibility](#)
- [When to Set Focus Explicitly](#)
- [Provide User Control: Avoid Forced Focus Changes](#)
- [Return Focus to Open Applications](#)
- [When Not to Provide Focus](#)
- [Keep Visual Focus](#)
- [VHA Section 508 Checkpoints: Focused Controls in Scrollable Areas](#)
- [Focus for Custom Components](#)
- [Conceptual Knowledge Checks](#)
- [How to Set Focus Explicitly](#)
- [VHA Section 508 Checkpoints: Setting Focus Explicitly](#)
- [Ensuring that focus is set properly includes checking for visual, keyboard and programmatic focus. You can check for programmatic focus using a screen reader or the Object Inspector.](#)
- [How to Avoid Forced Focus Changes](#)
- [VHA Section 508 Checkpoints: Avoiding Forced Focus Changes](#)
- [To ensure that you have avoided forced focus shifts:](#)
- [How to Return Focus to an Open Application](#)
- [VHA Section 508 Checkpoints: Returning Focus](#)
- [To ensure that focus is returned to the last-focused element in an open Flash application:](#)
- [How to Ensure that Inactive/Disabled Elements Do Not Receive Focus](#)
- [How to Change the Default Indicator of Visual Focus](#)
- [VHA Section 508 Checkpoints: Visual Focus](#)
- [To ensure that keyboard focus is indicated visually:](#)
- [Technical Knowledge Check](#)
- [Providing Accessible User Interface Controls](#)
- [Introduction](#)
- [User Perspective: Unseen Requests and Double-Talk](#)
- [User Perspective: Improperly Grouped Radio Buttons](#)
- [Make Interface and Form Controls Accessible](#)
- [Ensure Updates to Controls and Instructions are Accessible](#)
- [Ensure Users Have Time to Complete a Form or Use a Control](#)
- [VHA Section 508 Checkpoints: Timed Responses](#)
- [Best Practice](#)
- [Conceptual Knowledge Checks](#)
- [How to Enable Accessibility for User Interface Controls](#)
- [How to Properly Form, Group and Label Controls](#)
- [VHA Section 508 Checkpoints: Forms](#)
- [Using a Screen Reader](#)
- [How to Provide an Accessible Name, Description, Role, State or Value](#)
- [VHA Section 508 Checkpoints: Name, Role, Description, State, Value](#)
- [Using a Screen Reader](#)
- [How to Indicate Form Field Constraints and Feedback](#)
- [VHA Section 508 Checkpoints: Form Field Constraints and Indicators](#)
- [Using a Screen Reader](#)
- [How to Indicate Errors](#)
- [VHA Section 508 Checkpoints: Error Handling](#)
- [Using a Screen Reader](#)
- [Technical Knowledge Checks](#)

Creating Accessible Flash Course – Introduction

Introduction

Welcome to the Creating Accessible Flash Course.

Background

Section 508 of the Rehabilitation Act requires Federal agencies to make electronic and information technology (E&IT) accessible to users with disabilities, including:

- * Blindness, color blindness, visual impairment
- * Deafness, hearing impairment
- * Speech impairment
- * Mobility, strength, dexterity or reach impairment

The law includes standards for software applications, operating systems (OS), web-based applications and multimedia. These standards apply to Flash, a software application increasingly used to create rich interactive presentations and training.

Those involved in the design, development, procurement and use of Flash products are responsible for ensuring that they comply with Section 508, but it is not always clear how to do so.

Purpose

The goal of this course is to enable developers and other personnel involved in Flash development projects to apply the principles of accessibility to creating Section 508-compliant Flash presentations.

This course references commercial products likely to be familiar to those taking the course. References to commercial product functionality and providers are included to illustrate application of techniques described, and not intended as either endorsements or critiques of specific providers or products.

This lesson introduces Section 508, assistive technologies and tools for creating, playing and testing accessible Flash. It also explains how the course is structured and the assumptions on which the course is based.

Select Next to learn more about the Section 508 standards that apply to Flash.

Section 508 Standards

Section 508 includes four subparts:

- A. General Standards
- B. Technical Standards
- C. Functional Performance Criteria
- D. Information, Documentation and Support

This course references Subpart B, Technical Standards and Subpart C, Functional Performance Criteria.

Subpart B, Technical Standards

Section 508 technical standards are organized by technology, as follows:

- * §1194.21: Software Applications and Operating Systems
- * §1194.22: Web-based Intranet and Internet Information and Systems
- * §1194.23: Telecommunication Products
- * §1194.24: Video and Multimedia Products
- * §1194.25: Self-contained Closed Products
- * §1194.26: Desktop and Portable Computers

Since Flash is a software application that can be used to deliver multimedia presentations on the web, you will find Flash-related technical standards in §1194.21, §1194.22 and §1194.24. This course focuses on the technical standards for software applications found in §1194.21, Software Applications and Operating Systems.

Subpart C, Functional Performance Criteria

Section 508, Subpart C, Functional Performance Criteria (§1194.31) highlights two approaches to making E&IT accessible to users with disabilities:

- * Provide multiple ways to operate the technology and retrieve information so that users can choose alternatives based on their physical capabilities.
- * Provide support for the assistive technology (AT) that people with disabilities use for improved accessibility.

Select Next to learn more about assistive technology.

Assistive Technology (AT)

Quality software development requires a solid understanding of user needs. To develop Section 508-compliant software, you need to understand the needs of users with disabilities. This includes a familiarity with how users with different disabilities interact with their computers and the types of assistive technology they use to enhance their experience.

Many strategies and assistive technologies enable users with disabilities to access electronic information. Following is a brief introduction to some of them. As you proceed through this course, you will learn more about how you can support assistive technologies in Flash.

AT for Users with Visual Disabilities

Users who are blind usually use a keyboard, rather than a mouse, to interact with the computer. They may also use a Braille display or screen reader to access information from the computer. Screen reader software, such as JAWS or Window-Eyes, converts computer text to speech and reads it out loud.

People who have a visual disability other than blindness may use screen magnification software, such as ZoomText or Magic, to enlarge the computer display or to adjust the contrast.

Strategies for Users with Hearing Disabilities and Speech Impairments

Users who are deaf rely on captioning or a transcript to access the audio portions of software applications. Users with a hearing disability other than deafness may use assistive hearing devices to amplify sound, or they may rely on an application's volume control.

Currently, speech impairments are unlikely to impede computer access, as few applications rely on voice input. However, voice input technologies are becoming more widespread, and although they can improve accessibility for users with mobility or dexterity impairments, it is important to remember that users with speech impairments may need an alternative way to access information.

AT for Users with Mobility Impairments

People with mobility impairments may not be able to manipulate a mouse or a standard keyboard. These individuals may use alternative input devices to control the computer. Such devices include speech recognition software, like Dragon NaturallySpeaking, which substitutes voice commands for keyboard and mouse input; and on-screen keyboards that are activated by pointing devices, hardware switches or eye gaze technology.

How ATs Work

Assistive technologies typically obtain information from an application (like Flash) through an application programming interface (API) such as Microsoft Active Accessibility (MSAA) and then convey that information to the user. Although MSAA recognizes and automatically conveys some types of information to the user, in most cases, Flash developers must perform specific steps to expose the information to MSAA so that it can be passed along to the user. Many of the accessibility techniques in this course are, simply put, techniques for ensuring that you expose the right information to AT.

Although this course focuses on one particular application framework — Flash — it is important to remember that assistive technology is more than an interface between Flash and the user. Using specialized but well-documented channels and techniques, AT interacts with the computer hardware, the operating system and other applications, as well as with the user. AT can monitor events generated by the system; it can monitor and intercept text and lines written through the computer's video card and sound played through the computer's sound card; and it can intercept keystrokes and act on them before they reach an application.

Flash

Flash is a multimedia platform that allows developers to incorporate animation, sound, video and interactivity into a standalone product or onto a web page. Its uses range from simple web page decorations and banner advertisements to fully interactive training and electronic forms.

Developers may use the term Flash to refer to many different products that incorporate these features, regardless of how they are created or played back. This can be confusing for non-technical personnel; it also complicates a discussion of accessibility. It is helpful, therefore, to introduce some basic terminology and to establish parameters for this course.

Flash: The Author and the Player

Creating Flash and playing Flash require two different kinds of software:

- * Developers need an authoring tool to create Flash content and to package it into the product the end user sees.
- * Users generally need a player to run the end product on their computers as a standalone product or within a web page.

Developers first use an authoring tool to create Flash content in a format, such as an .fla file, that developers can edit, but which users cannot play. They then use the same tool to package the content into an end product, an .swf (or Shock Wave Format) file that the user can play, but which can no longer be edited, regardless of the authoring tool that was used to create it. The .swf file may be compiled as an .exe or .app file to avoid the need for a player on the user's computer.

Writing Flash

Flash authoring tools include:

- * Adobe Flash (Adobe Systems, Inc.)
- * Flex (open source framework by Adobe Systems, Inc.)
- * Adobe Captivate (screen recording software by Adobe Systems, Inc.)
- * Camtasia Studio (screen recording software by TechSmith Corp.)
- * Articulate Studio (Articulate Global, Inc.)
- * Viewlet and Composica (Qarbon, Inc.)

Developers can extend the capabilities of these authoring tools by adding commands in ascripting language, such as:

- * ActionScript
- * JavaScript
- * VBScript
- * JScript

Playing Flash

To display Flash content, a user needs to have a player for Flash installed on his or her computer. Most use Adobe Flash Player, which is available in multiple versions as a free download.

Users with disabilities use assistive technology in conjunction with a Flash Player to experience Flash presentations.

Versions Assumed in this Course

As developers know, software applications can be unpredictable when they are combined in new or different ways. In order to focus on applying the principles of accessibility to Flash, this course assumes that you are using the following software:

- * Adobe Flash CS4 Professional for authoring
- * ActionScript, version 3.0
- * Adobe Flash Player, version 10

We have found Adobe Flash CS4 produces accessible results more consistently than earlier versions of Adobe Flash and more reliably than other authoring tools. In addition, Adobe Flash CS4 works more readily with ActionScript, which provides additional accessibility capabilities. Regardless of which software you use, there is no substitute for testing.

Testing for Accessibility

One way to test an application for accessibility is to familiarize yourself thoroughly with the AT you want to use for testing, and then use the application with the AT running. Since this is not always possible, it's a good idea to familiarize yourself with other tools that allow you to verify that accessibility information is being exposed correctly to assistive technologies.

Testing with a Screen Reader

In this course, the instructions for testing with a screen reader assume some familiarity with screen readers and how they are used to test for accessibility. If you are a developer for whom screen reader testing is new, you may wish to explore some of the links available in Resources before you proceed.

As you continue in the course, you may also wish to refer to the list of screen reader keystrokes that is available on the VHA web site:

Keystrokes for Using JAWS and Window-Eyes to Test Web Pages

Testing with Inspect32

Microsoft offers an MSAA software development tool kit that includes an Object Inspector, known as Inspect32, that allows you to see what is being exposed to AT. You can download the Active Accessibility 2.0 Software Development Kit Tools by activating the link in Resources or read more about Object Inspector.

About this Course

In this training, you will learn how to apply the principles of accessible Flash to creating Section 508-compliant Flash presentations.

The main content of this course is presented in 15 lessons, which follow this Course Introduction. All students should take the Accessible Flash Overview lesson first. This lesson introduces basic terminology and presents an overview of Flash accessibility requirements that will help you find what you need as you proceed through the course.

Two-Track Lesson Structure

Each of the remaining 14 lessons provides both conceptual and technical instruction.

Flash Accessibility Principles

The first half of each lesson presents principles of Flash accessibility in clear, non-technical language. All personnel involved in Flash development projects for VA, from project managers to software developers, should proceed through this part of the instruction, which includes:

- * An introduction to one or more principles of Flash accessibility
- * Examples that illustrate how users are affected when those principles are violated
- * Non-technical explanations of related accessibility requirements and approaches to meeting them
- * Conceptual knowledge checks

When you have completed this part of each lesson, you may choose to proceed through the technical procedures that follow. If you are not a developer, you may wish to skip the technical portion and proceed to the beginning of the next lesson.

Flash Accessibility Procedures

The second half of each lesson includes one or more technical procedures for meeting the Flash accessibility requirements explained earlier in the lesson. This part of each lesson includes:

- * One or more specific techniques for meeting each accessibility requirement
- * Suggestions for self-testing to ensure an accessible product
- * A link to VHA Section 508 Office checkpoints related to each requirement
- * Technical knowledge checks

You can also access the procedures covered in a lesson by selecting them from the Topic menu from within the lesson or by selecting them from the Index.

Navigation

To navigate within a lesson, you may use the forward arrow (Next) and the back arrow (Back) at the top and bottom of each page; you may also use the topic menu at the bottom of each page. A progress indicator on each page (Page x of y) will help you keep track of where you are within each lesson. When you complete a lesson, you may proceed directly to the next lesson by selecting Next on the last page. You can also use the menu at any time to move to a different lesson or to review material in a lesson you have already studied.

User Perspectives

Throughout this course, you will find examples designed to provide you with first hand experience of the challenges faced by AT users in Flash. Many of these examples are demonstrations that open in a separate window and have their own controls. The controls that you will find in the user perspective demonstrations are labeled on the image that follows:

Glossary, Resources and Index

This course includes some additional resources for you to use during the training and later on the job. The navigation bar at the bottom of each page includes two links that look like little books with the letters G and R on them. From these links, you can access the course Glossary, which has the definitions of some terms and acronyms used in this course, and Resources, which includes documents you may wish to refer to from time to time.

In Resources, you will find a link to a Microsoft Word document of the complete Glossary content, a list of frequently-used keyboard commands for screen readers, a link to the General Services Administration's Section 508 web site and several Section 508 development checklists that are referenced in this course. This course is indexed to help you find information and procedures quickly when you need them on the job. The Index for this course is available from the menu.

Select Next for information on how to get VHA Section 508 help.

VHA Section 508 Help

VHA Section 508 Office personnel use checklists to facilitate and document their own compliance testing of Section 508 standards. Throughout this course, you will be able to access these checklists. Checklists, however, are not always able to keep pace with rapidly changing technology, and these checklists were designed to facilitate compliance testing, not software development.

If you have questions about how Section 508 standards apply to a Flash application you are developing, please contact the VHA Section 508 Office. This office partners with project teams to create Section 508-compliant e-learning products for VA and to make VHA web communications Section 508 compliant.

Course Overview

Introduction

Since there are many ways to achieve the same end result in Flash, there is no single formula that guarantees accessibility for every application. If you understand the principles of accessibility as they apply to Flash and are familiar with available tools, you will be able to implement accessibility principles within the context of your own projects.

This lesson introduces terminology used throughout the course, highlights features in Adobe Flash that may help you design for accessibility and presents an overview of Flash accessibility requirements covered in the remainder of this course.

Accessible Flash

In Flash, Section 508 compliance is often hindered by misperceptions about what is necessary, what is required and what is feasible.

Users with Disabilities Need Accessible Flash

Users with disabilities need access to the information that is being conveyed visually and aurally in Flash, and they need to be able to ignore or skip over Flash that is purely decorative.

Flash is no longer used primarily for pizzazz. With its multimedia capabilities, it is, in fact, an ideal tool for conveying important information to audiences with differing needs and abilities. What's more, even decorative Flash that contains no content can cause major, unacceptable disruptions to assistive technology (AT) users when principles of accessibility are ignored.

The Law Requires Accessible Flash

Past technical challenges notwithstanding, Section 508 is the law and it does apply to Flash. Furthermore, providing an accessible alternative such as a separate HTML page does not usually provide the equal access to content that is required by Section 508.

You Can Develop Accessible Flash

Because Flash content is dynamic and time-based, it can be challenging to make it accessible to assistive technology. Careful developer involvement is required to ensure that the Flash content is exposed in a logical manner. On the other hand, many of the steps required to achieve accessibility are not technically difficult, and one of the biggest challenges for software developers is to understand the perspective of

users with disabilities and to take that perspective into account during the early stages of development. Accessible Flash is not only feasible, but increasingly practicable. Many software producers, recognizing both a growing market among users with disabilities and increased enforcement of Section 508 requirements, are providing tools for creating accessible products, including accessible Flash.

Adobe Flash

Although Adobe has included many features in Flash CS4 to support accessible design, the process is by no means automated or foolproof.

CS4 Support for Accessible Design

Adobe Flash CS4 includes numerous features designed to help developers create more accessible Flash applications, including:

- * Support for Microsoft Active Accessibility (MSAA), the interface through which accessibility information is conveyed to assistive technology
 - * The Accessibility panel, a tool that allows developers to provide accessibility information for an object
 - * Scalable graphics and magnification capability for users with visual impairment
 - * Accessible video playback controls that allow users to stop, forward, rewind and pause presentations
 - * Customized color swatches that allow developers to design for users who are color blind
 - * Mouse-free navigation that allows users to navigate via the keyboard
 - * Synchronized audio track capability to help developers synchronize narrative audio in multimedia applications
 - * Support for closed captioning for users who are deaf
 - * A set of accessible standard user interface components, such as buttons, checkboxes, labels and alerts
- When used properly, these features can help developers create more accessible Flash applications.

Overcoming Challenges

The limitations of these featured capabilities, combined with many developers' limited understanding of how to use them, continue to pose accessibility challenges. In this course, designers and developers will learn to better understand what CS4 can and cannot do. For example:

- * Although the Accessibility panel is easy to use, it is not always enough. Developers must often augment its use with ActionScript to ensure that it works as intended.
- * CS4 allows developers to create scalable graphics; it does not create them automatically or guarantee scalability for the application as a whole.
- * Similarly, developers can use CS4 to create products with mouse-free navigation, but keyboard accessibility is not automatic; it requires the developers to make additional efforts.

In addition to understanding the principles of accessibility and the capabilities of Flash, developers can improve the chances of an accessible product by:

- * Incorporating an awareness of the needs of AT users into the early stages of design and development
- * Testing, testing, testing — using AT to test development efforts early and often
- * Contacting the VHA Section 508 Office for help when necessary

The Accessibility Panel

The Accessibility panel is an Adobe Flash tool that you can use to provide accessibility information about Flash objects to assistive technology. The options that appear on the panel depend on the type of object with which you are working. Two examples are shown here.

This screen introduces the concepts underlying the options that appear on the panel. Procedures for accessing this panel are provided later in this lesson.

Parent/Child Accessibility

In Flash, you can structure objects hierarchically in parent/child relationships. Parent objects are like containers, into which you can place child objects. Child objects typically inherit the properties of their parent objects. The Flash Accessibility panel allows you to indicate when you want to make a parent object accessible and whether or not to make its child objects accessible. This capability can be very useful as

long as you understand how it works and remember that it is not foolproof in practice.

Why Not Always Make Child Objects Accessible?

Sometimes it is better to hide child objects from AT, such as when the overall relationship among the objects is more important than the individual objects, or when animated child objects are disruptive to AT. Excessive enabling of accessibility can even make some Flash products inaccessible! You will learn more about this concept in the Hiding Flash lesson of this course.

The Importance of an Accessible Name

During development, you must specifically assign an accessible name property to each Flash object to expose it to assistive technology. Although there are several types of name properties in Flash; only the accessible name property is exposed to AT. You can use the Accessibility panel to assign an accessible name property. Convey the object's meaning or function as briefly as possible in the Name field, and use the Description field for additional information if necessary.

Accessible names are sometimes used as text alternatives or text equivalents. You will learn more about text equivalents in the Providing Text Equivalents lesson of this course.

Auto Labeling: Use with Care

The Accessibility panel includes an auto labeling option for Flash movies. This option automatically creates alternative text for buttons in the movie by labeling them with nearby text. In theory, if you include text on or near all of your buttons, you can use this option instead of providing text alternatives. In practice, this option does not always work well, especially with complex applications. If you use this capability, it is very important to test it.

Shortcuts

Keyboard shortcuts minimize the number of keystrokes necessary to access objects in Flash. The Accessibility panel allows you to notify AT when shortcuts are available. Shortcuts must be activated separately using ActionScript.

Tab Index

The Tab index field on the Accessibility panel allows you to control the tab order, which is the order in which keyboard users, including screen reader users, tab through a program, and the reading order. Note that tab order is not the same as reading order (the order in which a screen reader reads the information on a page) because the reading order may include objects that do not need to be accessible via the Tab key. You will learn more about reading order and tab order in the Controlling Reading and Tab Order lesson of this course.

Overview

The remainder of this course organizes the requirements for creating accessible Flash into 14 lessons. For a thorough understanding of accessibility in Flash, you should proceed in the order in which the lessons are presented on the menu. Depending on your prior experience and the type of applications you are developing, you may need to refer to some lessons more often than others. This overview will orient you to the remainder of the course so that you can focus on what you need most.

Overview, continued

To create accessible Flash, you must start with foundational knowledge about the interplay among Flash, AT, and operating systems (OS). Glossing over the basic concepts and techniques covered in the three lessons that follow this overview may undermine hours of accessibility work later.

In *Enabling Accessibility*, you will learn how to ensure that your Flash applications and elements are exposed properly to AT.

In *Hiding Flash*, you will learn how to determine which elements should not be exposed to AT and how to hide them to create a more accessible product.

In *Working with OS/AT Accessibility Features*, you will learn how Section 508 interoperability requirements apply to Flash. This lesson includes techniques for ensuring that Flash applications work with the accessibility features of operating systems (OS) and AT.

Overview, continued

Next, you must understand general concepts of accessible design that apply to all applications and learn how to apply them to Flash. If you have taken other courses on accessible software design, you may be familiar with the concepts in the next three lessons.

In *Designing for Accessibility*, you will learn how to apply principles of accessible design — including consistent use of images and link text, consistent naming and accessible instructions — to Flash.

In *Avoiding Flicker*, you will learn how to make sure your Flash applications do not flash or flicker in a way that can induce life-threatening seizures in users with epilepsy.

In *Using Color*, you will learn how to use color and contrast appropriately to create Flash that is accessible to users with visual disabilities, including blindness, color blindness and lack of visual acuity.

Overview, continued

You do not have to sacrifice Flash's multimedia capabilities for accessibility, but you do need to provide a way for users to control the audio and the video. If your application conveys information aurally, you must also ensure that users with hearing disabilities can access the information. Captioning is the most common way to accomplish this.

In *Providing Audio and Video Controls*, you will learn approaches to providing user control over audio and video so that users with disabilities can experience your rich media content.

In *Providing Captions and Visual Indicators for Sound Cues*, you will learn about captioning requirements, tools, services and other requirements for making Flash applications accessible to users who are deaf or have hearing impairments.

Overview, continued

In addition, you must ensure that users with visual disabilities can access visual content and that your application is keyboard accessible.

You can make visual content accessible by providing the same information in your audio description and/or by providing text equivalents that a screen reader can read.

Your application is keyboard accessible when users can access all of its functionality from the keyboard — without the use of a mouse. Keyboard accessibility is important for users with mobility impairments as well as for screen reader users.

Concepts and techniques for meeting these core accessibility requirements are covered in the following three lessons.

In *Using Audio for Visual Content* you will learn how to use audio descriptions and narration and how to use sounds as cues to visual information.

In *Providing Text Equivalents*, you will learn how to use text equivalents that can be read by AT for a variety of Flash elements, including graphics, animation, tables, window titles and hierarchical text.

In *Ensuring Keyboard Accessibility*, you will learn about all of the elements, controls and functions that must be accessible via the keyboard. You will also learn how to use keyboard shortcuts to make your application more accessible to users with manual dexterity impairments.

Overview, continued

To fully understand text equivalents and keyboard accessibility, you must also grasp the closely related concepts of reading order, tab order and focus.

Reading order is the sequence in which a screen reader reads the information on a page. Tab order is the sequence the Tab key follows when a user accesses the program via the keyboard. If you want your product to be accessible to screen reader and other keyboard users, you must control the reading order

and the tab order.

Focus refers to the place where user interaction takes place. When focus is properly indicated and maintained, the user knows where he is in a program and what his next keystroke will do. When an application does not maintain focus properly, most users will be confused and frustrated; AT users may be lost completely.

If you implement the techniques for providing text equivalents and ensuring keyboard accessibility without also controlling reading order, tab order and focus, the result may be a very inaccessible product! These concepts and techniques for implementing them in Flash are covered in the next two lessons:

In Controlling Reading and Tab Order, you will learn how to ensure that screen readers read your Flash applications and that keyboard users can tab through them in a logical sequence.

In Maintaining Focus, you will learn how to ensure that AT users know where they are and can control navigation through your Flash applications.

Overview, continued

Finally, if your application includes user interface controls, such as buttons or form elements, you must ensure that they are accessible.

In Providing Accessible User Interface Controls, you will learn how to provide accessible instructions and components; how to name, describe and label components; how to indicate field constraints, errors, changes and completion; and how to handle time constraints.

Overview, continued

This course also includes a comprehensive Course Index so that you can find information quickly when you need it on the job.

Select Next to learn how to access the Accessibility panel.

Select Enabling Accessibility from the menu to skip the technical part of this lesson and continue with the training.

How to Access the Accessibility Panel

The Flash Accessibility panel is available for Movie, MovieClip, Component, Button and Text objects. It is not available for Graphic objects. Since objects that you create or draw in Flash are, by default, established as Graphic objects, you must first convert them to Movie or MovieClip objects before you will be able to access the Accessibility panel for them.

There are two ways to access the Accessibility panel in Flash:

1. Select an object (Movie, MovieClip, Component, Button or Text) on the stage.
2. Press Shift+F11.

Or:

1. Select an object (Movie, MovieClip, Component, Button or Text) on the stage.
2. From the Window menu, select Other Panels; then select Accessibility.

When you select a Movie, the Accessibility panel displays the following options:

When you select a MovieClip or a Component, the Accessibility panel displays the following options:

When you select a Button, the Accessibility panel displays the following options:

When you select Text, the Accessibility panel displays the following options:

Enabling Accessibility

Introduction

Flash applications are not automatically accessible. Developers must take specific steps to enable accessibility at the application level and at the individual component level to let assistive technology (AT) know that the application and components are there. They must also provide informative accessible name properties for the AT to report. In addition, if a Flash application is to be deployed on the web, developers must ensure that the application will interact accessibly with the host web page.

In this lesson, you will learn about enabling accessibility and providing accessible name properties in Flash. You will learn how users with disabilities are affected when accessibility is not enabled. The technical part of this lesson explains how to enable accessibility at the application level, including additional instructions for ensuring that Flash that is destined for the web is accessible, and how to provide Flash applications with an accessible name.

As you proceed, remember that enabling accessibility is only a first step. It does not, by itself, provide enough information to assistive technology to make the application 508 compliant.

User Perspective: Incorrect WMode

Tom Rankin, a developer in Human Resources, created an orientation package for new employees. The package includes a web-based Flash presentation designed to provide important information to all new employees. Mary Woods is the first new employee to try to access Tom's presentation with a screen reader. To experience the presentation from Mary's perspective, select the Example link below. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Tom is a well-intentioned developer, but he was not familiar with the steps for ensuring that his Flash application would be accessible on the web. As a result, Mary's screen reader did not even report the existence of the Flash presentation; it read only the HTML content on top and bottom of the page, completely bypassing the Flash content. You will learn more about the procedures for making Flash accessible on the web later in this lesson.

User Perspective: Application-Level Accessibility Not Enabled

No amount of accessibility work will make a Flash application accessible if the developer forgets to enable accessibility at the application level. This is a common violation of Section 508 standards.

To understand what a screen reader user experiences when accessibility is not enabled at the application level, select the Example link below. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

In this example, the browser recognized the existence of a Flash application and reported "Flash Movie Start" and "Flash Movie End" to the screen reader, but nothing inside the Flash application was reported to the user. This is what happens when accessibility is not enabled at the application level, even if the developer has worked hard to make individual components accessible.

Enable Accessibility

In Flash, accessibility may need to be enabled at three different levels:

- * On the web

- * At the application level
- * At the individual component level

On the Web

When a Flash application is designed to be deployed on the web, enabling accessibility includes ensuring that the application interacts accessibly with the host web page. The only way for a Flash application to be accessible on the web is for it to be embedded in an HTML file; it cannot interact with other content on the page. Transparent applications that allow the web page to show through and opaque applications that obscure the page are not accessible.

To make Flash applications play correctly, developers must set the Window Mode (or WMode) parameter correctly. Procedures for setting this parameter are provided later in this lesson.

At the Application Level

Flash developers must enable accessibility at the application level. Enabling accessibility is not difficult, but it is frequently forgotten, resulting in Flash products that are useless to users with disabilities. Specific procedures for enabling accessibility at the application level are provided later in this lesson.

At the Component Level

Enabling accessibility at the application level does not automatically enable accessibility for all the objects within that application. Enabling accessibility for individual objects is covered later in this course. Sometimes it is better to hide individual components from assistive technology. You will learn more about how and when you should hide individual objects from AT in the Hiding Flash lesson.

Provide an Accessible Name

Each Flash application or movie should have a unique accessible name, the purpose of which is to identify the application as a whole for AT users. This is especially important when there are multiple Flash applications on a single web page.

The accessible name is a specific accessibility property. It may be the same as the movie title, but it is different from other types of name properties, such as instance names, used in Flash. For very simple Flash applications, an accessible name may also serve as a text equivalent. You will learn more about text equivalents in the Providing Text Equivalents lesson of this course.

Although some screen readers do not currently report accessible names properly in Flash, providing an accessible name is still an important best practice. It is also a simple step in the procedure for enabling accessibility at the application level, so developers can meet this functional performance criterion easily.

Example: Accessibility Enabled

Let's look once again at Tom's new employee orientation. In the following example, Tom has enabled his application to play it on the web, he has enabled accessibility at the application level, and he has provided an accessible name for the application.

Select the Example link below. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Conceptual Knowledge Checks

Check your knowledge of enabling accessibility. Use the Answer buttons to check your selections.

Top of Form

Once you have enabled accessibility at the application level, on the web and at the individual component level, you can be sure that your Flash content will be accessible to assistive technology (AT) users.

- A. True
- B. False

Bottom of Form

Top of Form

A developer has several ideas for Flash applications that returning Veterans can use to learn about their benefits. Which approach(es) will be accessible?

- A. An animation that overlays the existing VA web site, allowing the web site to show through the animation; the animated objects refer to items on the VA web site.
- B. A slide show that hides the existing VA web site while it is running; it includes all the information Veterans need without reference to the VA web site.
- C. A presentation that is launched from the VA web site within a designated area of the web site; it provides all the necessary information within its own area of the site and does not interact with the rest of the site.
- D. All of the above approaches can be made accessible.

Bottom of Form

How to Enable Accessibility at the Application Level

You can use either the Accessibility panel or ActionScript to enable accessibility at the application level. You must use the Accessibility panel to provide an accessible name for your Flash application. If your application is to be deployed on the web, you must also follow the procedures for setting the Window Mode parameter. Remember to check your work.

Using the Accessibility Panel

To use the Accessibility panel to enable accessibility at the application level and to provide an accessible name:

1. Open the Accessibility panel for the Movie:
 - a. Select or place focus on the stage.
 - b. Press Shift+F11.

OR

Select Other Panels from the Window menu; then select Accessibility.

2. Select the Make movie accessible checkbox.
3. Select or deselect the Make child objects accessible checkbox, depending on your application. Refer to the Course Overview for additional information about parent/child objects.
4. Select or deselect Auto label, depending on your application. Refer to the Course Overview for additional information about this feature.
5. Enter an accessible name in the Name field.
 - a. The accessible name should be a brief description of the purpose of the movie.
 - b. Aim for no more than 50 characters whenever possible.
 - c. Do not use quotation marks.
6. If it is necessary to enter supplementary information, beyond the purpose of the movie, you may do so in the Description field. Screen readers may read both fields together, so make sure they flow without repetition.

Using ActionScript

To enable accessibility at the application level with ActionScript 3 instead of the Accessibility panel use the following code:

```
if (!this.accessibilityProperties)
this.accessibilityProperties = new AccessibilityProperties();
```

```
this.accessibilityProperties.silent = false;  
this.accessibilityProperties.forceSimple = false;  
Accessibility.updateProperties();
```

Checking Your Work

The VHA Section 508 checkpoints related to enabling accessibility at the application level and providing an accessible name are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

You can use the Object Inspector or a screen reader to ensure that you have enabled accessibility at the application level. Remember that if you are deploying your application on the web, you must also set the Window Mode parameter.

1. On the Inspect32 Options menu, select the following:

- * On Top
- * SPI_SCREENREADER flag
- * Watch Focus
- * Watch Cursor
- * Show Highlight Rectangle (optional)

2. Open the Flash movie in Internet Explorer or Firefox.

3. Place the mouse on the movie but not on a particular element.

4. Make sure the application has a Name property.

Using a Screen Reader

To use a screen reader to ensure that accessibility is enabled at the application level and that the application has an accessible name:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. In virtual cursor mode, press the Down arrow to enter the application.
4. Make sure there is content between the Flash Movie Start and Flash Movie End statements.

How to Set the Window Mode Parameter to Enable Accessibility

To expose a Flash application when it is hosted on a web page, you must set it to play embedded in an HTML file by setting the Window Mode parameter of the Flash movie to Window. Although there are some limitations to development techniques in this mode, you cannot make your application accessible in the Opaque or Transparent mode.

You can set this parameter from within Flash (using the Publish Settings dialog) or by setting the Object tag in HTML.

Setting the WMode from within Flash

To set the WMode from within Flash:

1. Select Publish Settings from the File menu.
2. Select the HTML tab on the Publish Settings dialog box.
3. Select Window from the dropdown list in the Window Mode field.
4. Select OK.

Setting the WMode in HTML

To set the Window Mode parameter to Window in HTML, use the HTML Object tag with a line that reads:

```
<PARAM NAME="WMode" VALUE="Window">
```

The resulting HTML would appear as follows:

```
<OBJECT id=custom_cursor  
codeBase=http://download.macromedia.com/pub/shockwave/  
cabs/flash/swflash.cab#version=7,0,0,0  
height=200 width=400 align=middle  
classid=clsid:d27cdb6e-ae6d-11cf-96b8-444553540000  
name=custom_cursor>
```

```
<PARAM NAME="_cx" VALUE="10583">  
<PARAM NAME="_cy" VALUE="5292">  
<PARAM NAME="FlashVars" VALUE="">  
<PARAM NAME="Movie" VALUE="custom_cursor.swf">  
<PARAM NAME="Src" VALUE="custom_cursor.swf">  
<PARAM NAME="WMode" VALUE="Window">  
<PARAM NAME="Play" VALUE="0">  
<PARAM NAME="Loop" VALUE="-1">  
<PARAM NAME="Quality" VALUE="High">  
<PARAM NAME="SAlign" VALUE="">  
<PARAM NAME="Menu" VALUE="-1">  
<PARAM NAME="Base" VALUE="">  
<PARAM NAME="AllowScriptAccess" VALUE="sameDomain">  
<PARAM NAME="Scale" VALUE="ShowAll">  
<PARAM NAME="DeviceFont" VALUE="0">  
<PARAM NAME="EmbedMovie" VALUE="0">  
<PARAM NAME="BGColor" VALUE="FFFCF0">  
<PARAM NAME="SWRemote" VALUE="">  
<PARAM NAME="MovieData" VALUE="">  
<PARAM NAME="SeamlessTabbing" VALUE="1">  
<PARAM NAME="Profile" VALUE="0">  
<PARAM NAME="ProfileAddress" VALUE="">  
<PARAM NAME="ProfilePort" VALUE="0">  
<PARAM NAME="AllowNetworking" VALUE="all">  
<PARAM NAME="AllowFullScreen" VALUE="false">  
  
</OBJECT>
```

Technical Knowledge Checks

Check your knowledge of the procedures for enabling accessibility. Use the Answer buttons to check your selections.

Top of Form

How do you open the Accessibility panel in Flash?

- A. Select Publish Settings from the File menu.
- B. Select Other Panels from the File menu; then select Accessibility.
- C. Select Toolbars from the Window menu; then select Accessibility.
- D. Select Other Panels from the Window menu; then select Accessibility.

Bottom of Form

Top of Form

What is the appropriate Window Mode setting for an accessible Flash application that will be deployed on the web?

- A. Window
- B. Opaque Windowless
- C. Transparent Windowless
- D. Accessible

Bottom of Form

Hiding Flash

Introduction

To create accessible Flash, you must ensure that some elements are not exposed to assistive technology (AT). But knowing how to hide elements from AT is not enough to create a truly accessible product. It is more important to be able to identify what, when and how much to hide. This may require you to make judgment calls about the purpose of an application, the intent of the designer and the perspective of the user. It also requires a solid understanding of how Flash interacts with AT.

In this lesson, you will learn when it is appropriate or necessary to hide all or part of a Flash application from AT. You will learn how users with disabilities are affected when Flash elements are not hidden appropriately. The technical part of this lesson explains how to hide Flash applications from AT, how to hide individual Flash elements from AT, how to hide child objects from AT and how to limit disruptive looping in Flash.

Important Caveat about Hiding Flash

When you use the procedures provided in this lesson to hide Flash, you are preventing applications or elements from being exposed to AT through Microsoft Active Accessibility (MSAA); this creates a more accessible product for screen reader and screen magnifier users. This hiding process does not hide the applications or elements from all users. Users with photosensitive epilepsy and users who are color blind can still see them, and users of other types of AT can still interact with them. Even "hidden" applications and objects must comply with Section 508 requirements for interoperability, consistent use, flicker, color and contrast.

User Perspective: Invisible Controls

Let's consider Tom Rankin's New Employee Orientation presentation again. Select the Example link below and pay close attention when the screen reader announces the controls. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Did you notice that the screen reader announced a Prior button to the user even though no Prior button appeared on the screen? This is an example of a control that is visible to sighted users only when it is actually available. It is being exposed to AT even when it is not available because Tom did not take the necessary steps to hide it from AT when it is invisible on the screen. Imagine how frustrated a screen reader user might be if she tried to use this nonexistent button!

If you think that one non-existent Prior button on the first page of a presentation is a minor inconvenience, consider the fact that Flash allows developers to maintain many such objects, including:

- * Objects hidden behind other objects or behind popup dialogs
 - * Objects designed to appear at certain points along the timeline
 - * Objects designed to appear only in response to specific user actions
 - * Remnants of objects that remain onstage after a developer has moved the objects off screen
- These objects are exposed to AT unless the developer explicitly hides them.

How Much to Hide

In Flash, you can hide an entire application or specific elements within an application from AT; you can also expose a parent object to AT while hiding its child objects.

Entire Applications

You should hide Flash at the application level only if the application conveys no useful information, as in the case of a purely decorative banner that loops across a web page. The procedure for hiding a Flash application from AT is straightforward. Making the determination that an application conveys no useful

content, however, requires careful consideration. You will learn more about making this determination in the discussion of decorative Flash on the next screen.

Individual Elements

In general, you should hide individual Flash elements from AT if they are purely decorative, designed to be invisible to other users, likely to disrupt AT or redundant. You will learn more about how to determine if an element is purely decorative in the discussion of decorative Flash that follows. Then you will learn how to identify other types of elements that you should hide from AT.

Child Objects

Sometimes, you should hide child objects of an otherwise accessible parent object. This is especially true for certain types of movie clips or animations embedded in larger Flash presentations. In these situations, you can use an accessible name to convey the overall meaning of the parent movie clip or animation and prevent the moving child objects from disrupting screen readers by hiding them from AT.

Select Next to learn how to determine when Flash is purely decorative.

When is Flash Decorative?

You should hide purely decorative Flash — that is, Flash that conveys no useful information — from AT because it creates an unnecessary distraction. In some cases, an entire application is purely decorative; in others, there are individual elements within an application that are decorative.

Before you classify an application or element as decorative and hide it from AT, make sure it does not convey information that you would be hiding from AT users. To make this determination, consider the content, the context in which it is used, the designer's intent and the user's perspective.

Content

A good designer can convey a wealth of information in a simple, visually appealing presentation. Before you determine that a presentation is purely decorative, think carefully about whether it has conveyed information to you that you would not have received if you had not seen it.

Context

The same presentation may convey information in one context but not in another. Consider, for example, a slide show that depicts pictures of VA buildings around the country:

- * As a backdrop during a new VA employee orientation, this presentation probably doesn't convey useful information. It is just decoration and you could hide it from AT.

- * As part of a presentation on the variety of architectural styles used in government buildings, the same presentation may convey quite a bit of information that you should expose to AT.

Designer Intent

Sometimes designer intent is the clearest indicator of whether Flash conveys useful information. Consider, for example, how two designers might use the same elements featuring a catchy melody and an image of fireworks:

- * As a banner designed to attract attention on a web site, these elements would be purely decorative; you should hide them from AT.

- * As an indication that a user has answered a question correctly, the same elements convey important information that you should expose to AT.

User Perspective

To decide what to hide from and what to expose to AT, you must try to understand how AT users will experience an application. Then you must ensure that all users have equivalent access to all the information conveyed. Note that this process does not include making judgments about which users need access to which information. Procedures for hiding decorative Flash applications and elements are provided later in this lesson.

Individual Elements to Hide

In addition to decorative elements, you should avoid exposing to AT elements that are designed to be invisible to all users, elements that may be disruptive to AT and elements that are redundant.

Invisible Elements

Screen readers can often detect Flash elements that are designed to be hidden from or invisible to sighted users. Examples of these elements include:

- * Elements designed to appear only at specific times during a presentation, such as icons that indicate when help is available, or Next buttons that appear only when the user has the option to move forward
- * Elements designed to appear only in response to specific user actions, such as Submit buttons that appear only after a user has completed all the fields on a form, Answer buttons that appear only after a user has answered a question or images that appear only when a users answers a questions correctly
- * Elements that a designer has moved off-screen, but not completely off-stage
- * Elements designed to be obscured by other elements, such as when another window or tab pops up

If you do not hide these elements from screen readers when they are invisible to other users, the result can be very confusing for screen reader users.

In the following example, a popup dialog appears in front of and partially obscures a registration form. If the form is not hidden from AT when the dialog appears, the screen reader will treat the form and the dialog as if they are on the same plane. Screen reader users will not be able to tell that they have to answer the question on the popup to dismiss the dialog before they can submit the form.

Disruptive Elements

Screen readers tend to interpret any movement on the screen as a change that must be reported to the user. This can cause the screen reader to start over at the top of the page every time there is a movement on the screen. Examples of elements that can be disruptive to screen readers when accessibility is not implemented correctly include:

- * Animations
- * Elements that display weather or time in real time
- * Elements that display scores
- * Progress displays such as those that are displayed when loading a program
- * Timers

To make these types of elements accessible, you must hide the movement from AT, but still provide the user with the information that the movement conveys. Approaches to accessible animation are covered later in this lesson.

Redundant Elements

When a screen reader encounters a Flash component, such as a form field, it reads the component's accessible name. Screen reader users do not require visual labels for form fields that have an accessible name property, but other users need the visual label. A problem arises for screen reader users when both the visual label and the accessible name are exposed to AT.

In these cases, the screen reader reads both. If they are identical, the user hears everything twice; if they are not identical, the results can be even more confusing. In Flash, this is likely to happen with TextInput, ComboBox and TextArea components. To avoid this redundancy, it is best to hide the visual label from the screen reader.

In the following example, the First Name field is labeled with a visual label, but it also has an accessible name associated with it. To avoid having the screen reader read both the visual label and also report the accessible name associated with the component, it is best to hide the visual label from AT.

Animation

When accessibility is implemented poorly, animated content can disrupt AT. There are several approaches to making animated content accessible, but no hard and fast rules that work for every animation. In choosing your approach, you must exercise appropriate judgments about the design and purpose of each animation.

If the Animation is Purely Decorative

If your animation does not convey any useful information, you should hide it, as you would hide any decorative Flash. This may mean hiding an entire application, or it may mean hiding individual animated objects. The procedures for hiding applications and individual objects are provided later in this course.

If the Animation Conveys Information

If your animation conveys information, you must make that information accessible. You can deal with the potentially disruptive nature of the movement on the screen by hiding the movement or by allowing the

user to control it.

Convey Information while Hiding Movement

When an animation is used to convey information that is easily described, it is usually best to hide the animation as a child object and convey the necessary information in the accessible name, description or in a text equivalent at the parent level. The procedures for hiding child objects are provided later in this course.

Provide User Control

For narrative animations, such as those used in e-learning or procedural demonstrations, it is better to provide the user with step-through video controls that allow him to start and stop the animation at his own pace. This is also the best way to allow a user to access the information provided in an automatically updating display, such as a scoreboard or timer. You will learn more about providing controls in the Providing Audio/Video Controls lesson.

Limit Looping

It is a good practice to limit the number of times an animation loops, as endless looping can be distracting to users with cognitive disabilities. Procedures for limiting looping animations are provided later in this lesson.

Conceptual Knowledge Checks

Check your knowledge of important concepts related to hiding Flash. Use the Answer buttons to check your selections.

Top of Form

Which of these Flash applications, used on weather forecasting websites, should you hide completely from AT?

- A. An animated loop that depicts the percentage of cloud cover for the last two hours
- B. A slide show of weather-related images, such as lightning storms and tornadoes
- C. A weekly calendar that depicts beach scenes for sunny days and library scenes for rainy days
- D. A dancing pig used to indicate a severe weather alert

Bottom of Form

Top of Form

Flash animations should always be made completely inaccessible to AT because they are so disruptive to screen readers.

- A. True
- B. False

Bottom of Form

Top of Form

Even purely decorative Flash applications that are completely hidden from AT must comply with Section 508 requirements.

- A. True
- B. False

Bottom of Form

Top of Form

Screen reader users are not affected by Flash elements that are invisible on the screen.

- A. True
- B. False

Bottom of Form

Top of Form

When Flash animation conveys information, which of the following best describes how to make it accessible?

- A. Use parent/child accessibility features to convey the information while hiding the movement.
- B. Provide the user with step-through control of the animation.
- C. Either A or B, depending on the animation.
- D. Neither A nor B. You cannot make Flash animations accessible.

Bottom of Form

How to Hide Flash Applications from AT

The procedures for completely hiding a Flash application from AT are similar to the procedures for enabling accessibility at the application level. You can use the Accessibility panel, ActionScript or the Window Mode parameter. The application will still be visible on the page, so it must adhere to other Section 508 requirements. Remember to check your work.

Using the Accessibility Panel

To use the Accessibility panel to hide a decorative Flash application from AT:

1. Open the Accessibility panel for the Movie.
2. Deselect the Make movie accessible checkbox.

Using ActionScript

To hide a decorative Flash application from AT with ActionScript 3 instead of the Accessibility panel, use the following code:

```
if (!this.accessibilityProperties)
this.accessibilityProperties = new AccessibilityProperties();
```

```
this.accessibilityProperties.silent = true;
```

```
this.accessibilityProperties.forceSimple = true;
Accessibility.updateProperties();
```

Using the Window Mode Parameter

To use the WMode parameter to hide a web-based Flash application, set the WMode to Opaque. You can set this parameter from within Flash (using the Publish Settings dialog) or by setting the Object tag in HTML.

Setting the WMode from within Flash

To hide an application from within Flash:

1. Select Publish Settings from the File menu.
2. Select the HTML tab on the Publish Settings dialog box.
3. Select Opaque from the dropdown list in the Window Mode field.
4. Select OK.

Setting the WMode in HTML

To set the WMode parameter Opaque in HTML, use the HTML Object tag with a line that reads:

```
<PARAM NAME="WMode" VALUE="Opaque">
```

The resulting HTML would appear as follows:

```
<OBJECT id=custom_cursor
codeBase=http://download.macromedia.com/pub/shockwave/
cabs/flash/swflash.cab#version=7,0,0,0
height=200 width=400 align=middle
classid=clsid:d27cdb6e-ae6d-11cf-96b8-444553540000
name=custom_cursor>
```

```

<PARAM NAME="_cx" VALUE="10583">
<PARAM NAME="_cy" VALUE="5292">
<PARAM NAME="FlashVars" VALUE="">
<PARAM NAME="Movie" VALUE="custom_cursor.swf">
<PARAM NAME="Src" VALUE="custom_cursor.swf">
<PARAM NAME="WMode" VALUE="Opaque">
<PARAM NAME="Play" VALUE="0">
<PARAM NAME="Loop" VALUE="-1">
<PARAM NAME="Quality" VALUE="High">
<PARAM NAME="SAlign" VALUE="">
<PARAM NAME="Menu" VALUE="-1">
<PARAM NAME="Base" VALUE="">
<PARAM NAME="AllowScriptAccess" VALUE="sameDomain">
<PARAM NAME="Scale" VALUE="ShowAll">
<PARAM NAME="DeviceFont" VALUE="0">
<PARAM NAME="EmbedMovie" VALUE="0">
<PARAM NAME="BGColor" VALUE="FFFCF0">
<PARAM NAME="SWRemote" VALUE="">
<PARAM NAME="MovieData" VALUE="">
<PARAM NAME="SeamlessTabbing" VALUE="1">
<PARAM NAME="Profile" VALUE="0">
<PARAM NAME="ProfileAddress" VALUE="">
<PARAM NAME="ProfilePort" VALUE="0">
<PARAM NAME="AllowNetworking" VALUE="all">
<PARAM NAME="AllowFullScreen" VALUE="false">
</OBJECT>

```

Checking Your Work

Using a Screen Reader

The best way to ensure that you have hidden a Flash application is to use a screen reader, as follows:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. In virtual cursor mode, press the Down arrow to enter the application.
4. Make sure the application content is not reported by the screen reader.
 - a. If you have used the Accessibility panel of ActionScript to hide the application, the screen reader should not report content between the Flash Movie Start and Flash Movie End statements.
 - b. If you have set the WMode to Opaque, the screen reader should ignore the Flash movie completely.

How to Hide Individual Flash Elements from AT

To hide simple Flash elements and keep them hidden from AT, you can use the Accessibility panel. More often, however, you will need to hide and expose elements based on the timeline or on user action. In these cases, you must use ActionScript.

Making sure off-stage elements are not exposed to AT

Note that AT determines whether a Flash element is on- or off-stage based on the location of the element's registration point. If you move an element so that it is visibly off-screen but leave the element's registration point onstage, the element will still be visible to AT. To prevent off-screen elements from being reported to AT, make sure their registration points are completely off-stage.

Using the Accessibility Panel

To use the Accessibility panel to hide a specific Flash element from AT:

1. Open the Accessibility panel for the specific object.
2. Deselect the Make object accessible checkbox.

In this example, a decorative element is being hidden from AT.

Using ActionScript

You cannot use the Accessibility panel to hide Flash elements at specific points on the timeline or in

response to user actions. You must use ActionScript to perform these tasks. In the example you saw earlier in this lesson, the Prior button was designed to be invisible on entry to the presentation and appear only when it became active.

The following ActionScript 3 code can be inserted in Key frames within the timeline of a Flash movie to ensure that an object is not accessible when the object should not be recognized. Notice that you must not only set the accessibility properties for the object, but also update the accessibility properties. You need to add appropriate ActionScript commands whenever an object is supposed to appear and disappear.

```
if (!mc_PriorButton.accessibilityProperties)
```

```
mc_PriorButton.accessibilityProperties =
new AccessibilityProperties();
```

```
mc_PriorButton.accessibilityProperties.silent = true;
```

```
mc_PriorButton.accessibilityProperties.forceSimple = true;
```

```
Accessibility.updateProperties();
```

Checking Your Work

The VHA Section 508 checkpoints related to hiding individual elements from AT are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

You can use the Object Inspector or a screen reader to check your work.

Using the Object Inspector

To use the Inspect32 tool to ensure that specific objects are hidden from AT:

1. On the Inspect32 Options menu, select the following:

- * On Top

- * SPI_SCREENREADER flag

- * Watch Focus

- * Watch Cursor

- * Show Highlight Rectangle (optional)

2. Open the Flash application in Internet Explorer or Firefox.

3. Gain focus on the objects that should be hidden.

OR

Activate screens, windows or alert boxes that are designed to obscure objects.

4. Ensure that Object Inspector cannot identify any of the accessibility properties (i.e. name, descriptions, state) of the hidden or obscured content.

Using a Screen Reader

To use a screen reader to ensure that specific objects are hidden from AT:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.

2. Activate the Flash application.

3. Gain focus on the objects that should be hidden.

OR

Activate screens, windows or alert boxes that are designed to obscure objects.

4. Ensure that hidden or obscured content cannot be read by assistive technology in any of the following modes:

- * Virtual cursor mode (JAWS)

- * Browse mode (Window-Eyes)

- * Forms mode (JAWS)

- * MSAA mode off (Window-Eyes)

How to Hide Child Objects from AT

You can use the Accessibility panel or ActionScript to designate objects as child objects in order to hide them from AT, while still providing an accessible name to the overall application. The procedures that follow do not preclude you from later making an individual child object accessible.

Using the Accessibility Panel

To use the Accessibility panel to hide child objects from AT:

1. Open the Accessibility panel for the Movie.
2. Select the Make movie accessible checkbox.
3. Deselect the Make child objects accessible checkbox.
4. Deselect Auto label.
5. Enter an accessible name in the Name field.
 - a. The accessible name should be a brief description of the purpose of the parent object, with the most important information first.
 - b. Aim for no more than 50 characters whenever possible.
 - c. Do not use quotation marks.
6. If it is necessary to enter supplementary information, beyond the purpose of the parent object, you may do so in the Description field. Screen readers may read both fields together, so make sure they flow without repetition.

Using ActionScript

To use ActionScript instead of the Accessibility panel to hide child objects from AT, use the following code:

```
if (!mc_name.accessibilityProperties)
mc_name.accessibilityProperties = new
AccessibilityProperties();
```

```
mc_name.accessibilityProperties.silent = false;
```

```
mc_name.accessibilityProperties.forceSimple = true;
Accessibility.updateProperties();
```

Checking Your Work

This technique is often used to make brief but informative animations accessible. The VHA Section 508 checkpoints related to accessible animations are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Animation

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

h.1

If animated objects exist, does the information conveyed by the animated object exist in another non-animated method?

* *

*

h.1.a

Is there a non-animated method to step through or control animation?

* *

*

h.1.b

Is animation content sufficiently described in audio or text?

* *

*

h.2

Do alternatives to animation provide the equivalent functionality?

* *

*

h.3

Does screen transition animation settle within 5 seconds?

* *

*

How to Limit Looping

You can control looping from within Flash (using the Publish Settings dialog), with ActionScript or in the HTML code. It is a good practice to prevent animations from looping or to limit the number of loops to three.

Preventing Looping from within Flash

To prevent looping from within Flash:

1. Select Publish Settings from the File menu.
2. Select the HTML tab on the Publish Settings dialog box.
3. Under Playback options, deselect the Loop checkbox.
4. Select OK.

Controlling Looping with ActionScript

Following is an example of ActionScript code that you can insert in the first and last frames to control looping. This code limits the number of loops to three:

* First frame:

```
* var stopNum = 0;
* function looper(loopLimit) {
* if (stopNum >= loopLimit) {
* stop();
* } else {
* gotoAndPlay(2);
* }
* this.stopNum++;
```

} * Last frame:

```
looper(3);
```

Controlling Looping in HTML

You can control looping in the rendered HTML code by setting the value of the Loop parameter with a line that reads:

```
<PARAM NAME="Loop" VALUE="true">
```

Setting the value to zero prevents looping. The resulting HTML would appear as follows:

```
<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"
codebase="http://download.macromedia.com/pub/shockwave/cabs/
flash/swflash.cab#version=10,0,0,0"
width="550"
height="400"
id="loop_test"
align="middle">
<param name="allowScriptAccess" value="sameDomain" />
<param name="allowFullScreen" value="false" />
<param name="movie" value="loop_test.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
```

```
<PARAM NAME="Loop" VALUE="true">
<embed src="loop_test.swf"
quality="high"
bgcolor="#ffffff"
width="550"
height="400"
name="loop_test"
align="middle"
allowScriptAccess="sameDomain"
allowFullScreen="false"
type="application/x-shockwave-flash"
pluginspage="http://www.adobe.com/go/getflashplayer" />
</object>
```

Checking Your Work

To check your work, simply open the Flash application in Internet Explorer or Firefox and count how many times an image loops before it stops. It should cycle through no more than three times.

Technical Knowledge Checks

Read the following statement and select True or False. Use the Answer button to check your selection.

Top of Form

Flash objects hidden from sighted users by moving the object and its registration point off the screen are hidden from assistive technology, too.

- A. True
- B. False

Bottom of Form

Top of Form

Which ActionScript property can you use to hide individual Flash elements from AT?

Use the Answer button to check your selection.

- A. Opaque
- B. silent
- C. Window
- D. Loop

Bottom of Form

Top of Form

For which of these tasks must you use ActionScript instead of the Flash Accessibility panel?

- A. Hiding a child object
- B. Hiding an entire Flash application from AT
- C. Hiding a Flash element that is off-stage until the last frame
- D. Hiding a Flash element that should not appear unless the user passes a test

Bottom of Form

Working with OS/AT Accessibility Features

Introduction

Users with disabilities rely on the accessibility features of operating systems (OS) and assistive technologies (AT) to access electronic information. One of the most basic Section 508 requirements for software applications is that they not disrupt, disable or otherwise interfere with these important features. The principle underlying this requirement is known as interoperability.

Accessibility hinges on interoperability and this course includes many specific techniques for ensuring that Flash works well with AT. In this lesson, you will learn how the overall principle of interoperability relates to Windows, Flash and AT. Flash does not fully support interoperability with all of Windows' accessibility features, but you will learn how to avoid interfering with the features that are supported in Flash and how to work around the ones that are not.

Windows OS Built-in Accessibility Features

The Windows OS includes several built-in accessibility features. Users with disabilities choose which of these features to activate. Under Section 508, other applications, such as Flash, must not interfere with or disable the OS accessibility features a user has activated.

You can explore Windows' built-in accessibility features by choosing Accessibility Options from the Control Panel. The accessibility features are provided on the following tabs in the Accessibility Options panel:

- * Keyboard
- * Sound
- * Display
- * Mouse

Ideally, these features should function the same way, regardless of the application that is running on the OS. In practice, however, some of these features are supported in Flash and some are not. Where Flash applications do not fully support Windows OS accessibility features, it is important to provide the same level of accessibility from within Flash.

Keyboard Accessibility Options

Windows' keyboard accessibility options include StickyKeys, FilterKeys and ToggleKeys.

- * The StickyKeys option allows the user to issue keyboard commands with sequential, rather than simultaneous, key presses. For example, StickyKeys would allow a user to press the Control, Shift and A keys one at a time to turn on the audio in an application that had Control+Shift+A as a shortcut for this task. This makes it easier for users with dexterity impairments to issue keyboard commands.
- * The FilterKeys option causes the application to ignore short or repeated keystrokes. This feature is useful for people with certain motor impairments.
- * The ToggleKeys option provides a sound when the Caps Lock, Num Lock or Scroll Lock is turned on or off. This is a useful alert for users with motor and visual impairments.

In Flash, default components and movie clips support the StickyKeys and FilterKeys options, but if you use custom keyboard event handling in your application, you will need to evaluate your application with these features on to ensure that they work as expected. In some cases, such as in simulations that monitor keystroke speed, you may need to provide alternative input methods to comply with Section 508. The ToggleKeys feature also works by default in Flash, but if you control these toggleable keys in the application, you should be sure to test it. The Show extra keyboard help in programs checkbox does not apply in Flash.

Sound Accessibility Options

Windows' sound accessibility options, SoundSentry and ShowSounds, are designed for users with hearing impairments:

- * The SoundSentry option creates a visual warning when the system makes a sound.
- * The ShowSounds option enables displaying of captions.

Although Flash does not support either of these Windows accessibility features at this time, you can create Flash applications that are accessible to users with hearing impairments:

- * Make sure your applications do not rely on sound alone to convey information.
- * Provide captioning from within the application.

Display Accessibility Options

Windows' display accessibility options, High Contrast and Cursor Options, are designed for users with low vision and color blindness.

- * The High Contrast feature allows the user to select from a variety of color, contrast and font schemes.
- * The Cursor Options feature allows the user to set the width of the cursor and the rate at which the cursor blinks.

Since Flash does not currently honor user selections made under these options, it is very important that you consider the needs of users with visual impairments when you are developing Flash. You will learn more about how to make good color and contrast selections in the Using Color lesson of this course.

Mouse Accessibility Option

The Windows mouse accessibility option, MouseKeys, allows the user to use the numeric keypad to control the mouse pointer. This is useful for people with certain motor impairments, but it is not a substitute for keyboard accessible applications. By default, Flash will support this.

Windows OS Accessibility Utilities

In addition to the accessibility features available through the Control Panel, Windows provides three basic accessibility utilities. To find these utilities:

XP: From the Start menu, select Control Panel > Accessibility Options.

Vista and Windows 7: From the Start menu, select Control Panel > Ease of Access > Ease of Access Center.

These utilities provide limited accessibility; most users with disabilities rely on other types of AT rather than these utilities. Although you may wish to familiarize yourself with these utilities, we recommend that you test your applications against the AT that users are more likely to be using.

On-Screen Keyboard

The On-Screen Keyboard utility provides very basic alternative input functionality for users with severe motor impairments. These users are increasingly taking advantage of more sophisticated speech recognition software, such as Dragon NaturallySpeaking.

By default, Flash applications should work properly with the On-Screen Keyboard utility, but if you are using custom keyboard handling, you should test your application with this utility running to make sure it works correctly.

Narrator

Narrator is a limited screen-reading utility. We recommend that you test your applications with JAWS and/or Window-Eyes, which are higher functioning screen readers, and not with this utility.

Magnifier

Magnifier is a limited screen magnification utility. We recommend that you test your applications with ZoomText or MAGic, which are higher functioning screen magnifiers, and not with this utility.

Note that some users with low vision use browser settings to change the font size instead of using screen magnification software. Although the goal of accessibility is for applications to be interoperable with such approach, this is not currently possible in Flash.

Interoperability with AT

Assistive technology uses well-documented aspects of the operating system to monitor and, when necessary, intercept system, video, sound and keyboard events. The best way to ensure that your Flash application "plays nicely" with AT is to use standard OS methods or a documented interface, such as MSAA, for writing to the screen and handling sound and keyboard events. Then test your applications with AT to be sure they work as expected and do not override settings used by AT.

In addition to this general guidance, look for more specific techniques for working with AT included throughout this course. To be accessible, Flash must be interoperable with:

- * Screen readers, such as JAWS and Window-Eyes
- * Screen magnifiers, such as ZoomText and MAGic
- * Captioning programs
- * Speech recognition software, such as Dragon NaturallySpeaking
- * On-screen keyboards and other types of alternate input methods

Conceptual Knowledge Checks

Top of Form

Flash applications always honor user settings for Windows OS keyboard accessibility options.

- A. True
- B. False

Bottom of Form

Top of Form

Flash applications do not honor Windows OS sound and display accessibility options.

- A. True
- B. False

Bottom of Form

How to Test Flash for Interoperability with Windows OS Accessibility Features and Utilities

The VHA Section 508 checkpoints related to OS interoperability are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

The best way to test your Flash application for compatibility with OS accessibility features is to try them out yourself. Compare how each feature works in your Flash application with how it works in a standard Windows application such as Windows Explorer.

Testing for Interoperability with StickyKeys

If your Flash application uses custom keyboard event handling, it is especially important to test it with the StickyKeys feature activated.

To activate and test StickyKeys:

1. Select Accessibility Options from the Windows Control Panel.
2. On the Keyboard tab, select the Use StickyKeys checkbox.
3. Select OK.
4. For any key press combinations required in your Flash application, make sure the same result can be achieved by pressing the keys in sequence instead.

Testing for Interoperability with FilterKeys

If your Flash application uses custom keyboard event handling, it is especially important to test it with the FilterKeys feature activated.

To activate and test FilterKeys:

1. Select Accessibility Options from the Windows Control Panel.
2. On the Keyboard tab, select the Use FilterKeys checkbox.
3. Select Settings.
4. Under Filter Options, select Ignore repeated keystrokes (the other option does not work).
5. Select OK.
6. In an input field in your Flash application, hold down a key such as the letter 'a' and make sure it appears only once.

Testing for Interoperability with ToggleKeys

If your Flash application controls or traps the CAPS LOCK or NUM LOCK keys, you should test it with the ToggleKeys feature activated. Testing with ToggleKeys activated is also a good way to make sure your

Flash application has not disabled the sound card function.

To activate and test ToggleKeys:

1. Select Accessibility Options from the Windows Control Panel.
2. On the Keyboard tab, select the Use ToggleKeys checkbox.
3. Press CAPS LOCK, NUM LOCK or SCROLL LOCK and make sure you hear the appropriate system tones.

Testing for Interoperability with Windows OS Sound Accessibility Options

At this time, the Windows OS sound accessibility options are not supported in Flash, so there is no need to test your application for interoperability with the SoundSentry or ShowSounds options. Instead, make sure that your application provides its own visual indicators for sounds and captioning for speech.

Testing for Interoperability with Windows OS Display Accessibility Options

At this time, the Windows OS display accessibility options are not supported in Flash, so there is no need to test your application for interoperability with High Contrast or Cursor Options. Instead, make sure that you choose your color and contrast schemes according to the principles in the Using Color lesson.

Testing for Interoperability with MouseKeys

The MouseKeys option should work with most Flash applications, but remember that this feature is not a substitute for making the application keyboard accessible.

To activate and test MouseKeys:

1. Select Accessibility Options from the Windows Control Panel.
2. Select the Mouse tab.
3. Select the Use MouseKeys checkbox.
4. From within your Flash application, make sure you can emulate mouse movement from the keypad.

Testing for Interoperability with Windows OS Accessibility Utilities

To activate and test the On-Screen Keyboard:

1. From the Windows Start menu, select All Programs > Accessories > Accessibility > On-Screen Keyboard (or type OSK at the Run dialog).
2. Perform all necessary actions in your Flash application from the On-Screen Keyboard and make sure they function as expected. Be sure to use the Tab key and arrow keys on the On-Screen Keyboard and test text input functions if they exist in your application.

Do not use Narrator or Magnifier to test your Flash applications. We recommend that you test with JAWS and/or Windows-Eyes and with ZoomText or MAGic instead.

How to Test Flash for Interoperability with AT

The VHA Section 508 checkpoints related to AT interoperability are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Operating System Interoperability

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(b)

Applications shall not disrupt or disable activated features of other products that are identified as accessibility features, where those features are developed and documented according to industry standards. Applications also shall not disrupt or disable activated features of any operating system that are identified as

accessibility features where the application programming interface for those accessibility features has been documented by the manufacturer of the operating system and is available to the product developer.

b.1

Can all of the documented accessibility options that are available via the operating system be activated and do they work correctly?

* *

*

The best way to test your Flash application for compatibility with AT is to try it out yourself. Remember that interoperability with AT is a basic requirement and does not eliminate the need to meet other Section 508 accessibility requirements.

Using a Screen Reader

To test your Flash application for interoperability with a screen reader:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. Proceed through and inspect content and activate user interface elements, making sure that:
 - a. Neither the application nor the screen reader crashes or locks up.
 - b. There is no substantial slowdown or drain on system resources.
 - c. There is no screen corruption.
 - d. Text, component and images are readable.
 - e. Screen reader speech is clear and uninterrupted.
 - f. Audio is not played automatically.
4. Make sure the following keys, required to access screen reader functionality, are not disabled or controlled by the Flash application:
 - a. Numpad 0
 - b. Num Lock
 - c. Caps Lock
 - d. Ctrl
 - e. Alt

Using a Screen Magnifier

Screen magnifiers use the right-click zoom menu feature to zoom in and out of Flash content, so make sure you do not disable it. You can use the following code to hide all other functions except for the zoom features:

```
var customMenu:ContextMenu = new ContextMenu();
customMenu.builtInItems.forwardAndBack = false;
customMenu.builtInItems.loop = false;
customMenu.builtInItems.play = false;
customMenu.builtInItems.print = false;
customMenu.builtInItems.quality = false;
customMenu.builtInItems.rewind = false;
```

```
customMenu.builtInItems.save = false;  
customMenu.builtInItems.zoom = true;  
contextMenu = customMenu;
```

To test your Flash application for interoperability with a screen magnifier:

1. Activate a screen magnifier, such as ZoomText or MAGic.
2. Activate the Flash application.
3. Proceed through and inspect content, activate user interface elements, scroll and tab through the screen, making sure that:
 - a. There is no screen corruption.
 - b. There are no added text or visual effects.
 - c. Text, component and images are readable.
 - d. The magnification, color and contrast settings provided by the screen magnifier program can be used while interacting with the application.
 - e. The magnified area scrolls to the proper location when you tab to user interface elements.

Designing for Accessibility

Introduction

Poorly designed applications are problematic for all users, but there are some design issues that are especially relevant to accessibility. This lesson covers accessible design issues best addressed in the early stages of application development, before programming.

In this lesson, you will learn about creating accessible link text, images, instructions, character sets and screen transitions. Since most of this accessibility work is conceptual and must be accomplished during the design stage, the technical part of this lesson focuses on testing for compliance.

User Perspective: Inaccessible Link Text

Here is a screen from the New Employee Orientation you saw earlier in this course.

When sighted users encounter a screen in a presentation like this, they usually scan the page first. In this case, they see a list of policies and procedures, with links to access more information about each. They probably ignore the header that appears at the top of every page and most users don't read the whole list — at least not initially.

Screen reader users also scan for important information, such as actionable items. They may use the Tab key to jump from button to button, or they may work from a list of buttons that the screen reader produces for them. To experience this screen from a screen reader user's perspective, select the Example link below. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

In the presentation you just saw, the Flash developer had appropriately identified the link text as buttons, so the screen reader recognized and announced the links. But the link text was not meaningful when it was taken out of context by the screen reader. Although a screen reader user could change screen reader modes and hear all of the text on the screen, the user would not be able to scan the page quickly that way. For a screen reader user to have an equivalent experience to that of a sighted user, the link text must be meaningful even when it is taken out of context by assistive technology (AT).

Provide Meaningful Link Text

Section 508 includes criteria for text that links a user to another location. Meaningful link text is important

to all users, but it is especially important for screen reader users and for users with some learning impairments. To be meaningful to users with disabilities, link text must:

- * Have a clear purpose and unique target
- * Make sense in and out of context

To make meaningful link text into an accessible link in Flash, you will:

1. Make the link text into a button.
2. Enable accessibility for the button.
3. Assign the button an accessible name.

If you have written good, meaningful link text, then you can use that text as your accessible name.

Have a Clear Purpose and Unique Target

Have you ever gotten so lost following links on a web page that you couldn't find your way back to where you started? Returning from a link to a previous location can be especially difficult for assistive technology users. This is one reason why it is so important for developers to write link text that has a clear purpose. Users should know where a link will take them before they select it. If a user has to select a link to determine what the link is for, then you have created an unnecessary hardship for users of AT. If, for example, you have a link to additional information about your company's time sheet policy, your link text should say something like "Read more about time sheet policy," as opposed to just "Read more" or "Time sheets", both of which are ambiguous.

It can be confusing for AT users when the same link text is used for multiple targets. You saw this in the example, where one page had a series of links, all of which said "Continue". It is easy to avoid this problem if your link text has a clear purpose, but you should still be careful to avoid inadvertently writing identical link text for different targets.

Make Sense In and Out of Context

As you have seen, screen reader users often scan a page by using the Tab key, or they may take advantage of the screen reader's ability to assemble lists of buttons or links on a page to help them navigate more quickly. Users with learning impairments may also use AT that provides this functionality. For this reason, it is important that link text make sense not only within the context of the surrounding text, but also out of context. Here is the screen reader's list of buttons for the example you saw earlier:

When you see or hear a list like this, it is obvious why you need to change the link text. The four Continue links have no clear purpose, the same link text is used to take the user to four different targets, and the link text is meaningless when taken out of context.

You will learn more about the procedures for turning meaningful link text into accessible links later in this lesson.

Provide Accessible Instructions

Instructional text can also create accessibility issues that are best addressed during the initial design of an application. You should make sure that instructions do not require the user to rely on any one sense. This page includes examples of inaccessible instructions that rely on sensory characteristics, along with suggestions for simple changes that make them accessible.

These examples include instructions that rely on the following common sensory characteristics: color and text formatting, location, orientation, size, shape, sound.

Color and Text Formatting

The inaccessible example below relies on color to indicate which fields are required. This example is inaccessible to screen reader users and to users who cannot distinguish color.

Notice that the accessible example, which follows, also uses color, but by adding an asterisk to the required fields (and to the instructions) it provides another way for users to know which fields are required. You will learn more about the appropriate use of color in Flash in the Using Color lesson of this course.

Instructions that rely on text formatting are similar to those that rely on color to convey information. In the previous examples, putting the required fields in bold text would not make them accessible.

Location

The following inaccessible instructions require the user to be able to see the location of the button:

To move forward, press the button on the right.

To make these instructions more accessible, you should add a description of the button and/or use the button's accessible name. In this case, the forward arrow button has the accessible name Next. Here's how you could rewrite the instructions to be more accessible:

To move forward, press the forward arrow (Next) button.

Size

The following inaccessible example requires the user to distinguish size differences. These distinctions might be difficult for users with low vision.

For each conviction, select the large gavel if it is a felony; select the small gavel if it is a misdemeanor.

Conviction: Mishandling classified information

You can correct an inaccessible design like this by using different images (for example an image of a jail for a felony and an image of money for a misdemeanor) or by labeling the images, as shown below:

Shape

Similarly, instructions that require a user to distinguish among shapes are inaccessible. Here is an example: For each action described, decide if it is (a) illegal, (b) legal but unethical, or (c) legal and ethical. Select the octagon if it is illegal, the triangle if it is legal but unethical and the circle if it is both legal and ethical.

Action: Sharing a patient's protected medical information with another doctor involved in the patient's care

There are several ways you can make these instructions accessible. The best way is to include labels and accessible names and to reword the instructions to describe the images better, such as:

Select the STOP sign if it is illegal, the YIELD sign if it is legal but unethical or the GO light if it is both legal and ethical.

Orientation

In the inaccessible example below, the user is instructed to "read the side bar". These instructions are not accessible to screen reader users, for whom the side bar may not be identified as such.

In the accessible example that follows, you see that the instructions are more explicit about what the user should read and do not require the user to orient himself to the layout of the page in order to get the necessary information.

Sound

Instructions are not accessible if they rely solely on the user's sense of hearing, either. Users who are deaf would not be able to follow these instructions:

Please wait.

The system is tallying your results.

When you hear the System Ready chime, press ENTER to proceed to the next test.

To make these instructions accessible, you must add a visual indicator when the System Ready chime sounds.

There are no specific technical procedures for writing accessible instructions. The VHA Section 508 checkpoints related to accessible instructions are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Accessible Instructions

§1194.31 Functional Performance Criteria
1194.31
Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.19

Are instructions that rely on sensory characteristics (such as size or location) avoided?

* *

*

Use Images Consistently

Using images consistently is a fundamental principle of good design. It is even more important when you are designing for accessibility. Your Flash applications will meet the criteria for consistent use if all of the following are true:

- * Each image has only one function.
- * Each function is represented by only one image.
- * Accessible names for images are consistent.

One Function Per Image

You should not use the same image to represent more than one concept or function. In the example that follows, the X in a red box represents two different functions.

You can often identify inconsistencies like this one by thinking about how you would provide accessible names. If you need completely different accessible names for the same images, then your inconsistent usage is likely to create a problem for users with low vision or cognitive impairments. This particular example may be even more confusing because it is not clear whether the Xs on the checklist indicate required, not required or completed modules.

One Image Per Function

Conversely, you should not use two different images to represent the same concept, function or user action. In the following example, a developer used a printer icon throughout the presentation to indicate that the user could print, but on the last screen, introduced a different print image for printing the completion certificate.

This type of inconsistent usage often happens on projects where several developers are working simultaneously. The best way to avoid this problem is to coordinate the use of images among developers and to use a common image library.

Consistent Accessible Names

All users rely on their familiarity with consistently-used functions to speed their way through a screen. As sighted users become familiar with consistently-used images, they don't stop to read and process the meaning of each image. Screen reader users also become familiar with the functions that appear repeatedly in an application — by hearing consistent accessible names.

Consistent accessible names are as important to screen reader users as the consistent use of images is to other users. If, for example, you provide a Search function in your application, make sure the button that activates the function has a consistent accessible name throughout the application. It doesn't matter if the accessible name is Search, Find, Go or Fetch, as long as it is the same every time the function appears. In some cases, accessible names are considered consistent even when they are not identical. Here is an example where two identical print icons appear on the same page:

One icon allows the user to print a completion certificate and the other allows the user to print course

materials. In this case, although the same image is used for one function, you would not want both images to have the same accessible name. By naming these icons Print Certificate and Print Course Materials, your naming is consistent, informative and accessible, even though it is not identical.

You must provide an accessible name for each use of an image, but you cannot store accessible names in the image library in Flash. So even if you are using an image library to maintain image consistency, you must be proactive in providing consistent accessible names. You will learn more about the procedures for ensuring that your images are consistently used and named later in this lesson.

Use Standard Character Sets

Some developers use symbolic fonts, such as Wingdings, to convey information visually. Common examples include emoticons, the Wingding character for scissors to indicate where one should cut a form, and the Wingding telephone character, which is often used to flag a telephone number.

Screen readers and other types of assistive technology cannot read these symbolic or non-standard character fonts, so you should use only ASCII (American Standard Code for Information Interchange) or Unicode characters in your Flash applications. If you would like to convey the information indicated by a Wingding character, use a graphic image instead and provide an appropriate accessible name.

The best way to test to be sure that your application does not include character sets that are unreadable by AT is to test your application with AT running. The VHA Section 508 checkpoints related to character sets are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Standard Character Sets

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.14

Are only standard character sets used?

* *

*

Settle Screen Transitions Quickly

Animated screen transitions that take too long to settle or animations that disappear too soon can be problematic for screen reader users. Screen readers begin to read a screen very shortly after it loads. If an animated screen transition has not finished running when the screen reader begins to read, the user is likely to miss out on some content. If you use animated screen transitions in Flash, make sure they settle

within five seconds.

If your application has animated screen transitions, time them. All transition content should be fully rendered and settled on the screen within five seconds. The VHA Section 508 checkpoints related to screen transitions are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Screen Transitions

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

h.3

Does screen transition animation settle within 5 seconds?

* *

*

Conceptual Knowledge Checks

Check your knowledge of some of the principles of accessible design. Use the Answer buttons to check your selections.

Read the example text; then determine whether the link text is accessible.

Top of Form

Example Text: Three men were apprehended behind a local bar last night. Read more.

- A. Link text is accessible
- B. Link text is not accessible

Bottom of Form

Top of Form

Example Text: You must complete the correct form to get credit for your coursework.

Find the form that is appropriate for you.

- A. Link text is accessible
- B. Link text is not accessible

Bottom of Form

Top of Form

Why is it important for link text to be meaningful when taken out of context?

- A. Because screen reader users cannot find link text in context
- B. Because screen magnifiers cannot capture link text in the right context

- C. Because that is the only way to ensure that link text has a clear purpose and a unique target
- D. Because screen reader users often read link text out of context

Bottom of Form

Top of Form

Which of the following is NOT one of the accessibility criteria for consistent use of images?

- A. If you use an image multiple times in one application, it should always have the same accessible name.
- B. If you use an image multiple times in one application, it should always represent the same function.
- C. If the same function appears repeatedly in your application, you should always represent it with the same image.

Bottom of Form

How to Create Accessible Links

Earlier in this lesson, you learned about the importance of writing link text with a clear purpose and a unique target, link text that makes sense both in and out of context. If your link text follows these principles, then you should be able to follow these easy steps to make accessible links in Flash:

1. Make the meaningful link text into a button.
 2. Enable accessibility for the button.
 3. Provide an accessible name property for the button.
 - a. If you have written meaningful link text, that text will serve as an accessible name.
 - b. If it is really impossible to make your link text meet the criteria outlined earlier in this lesson, then you can create an accessible name property that makes it meaningful for screen reader users.
- You can use either the Accessibility panel or ActionScript to enable accessibility and provide an accessible name for your link button. Remember to check your work.

Using the Accessibility Panel

To use the Accessibility panel to enable accessibility and provide an accessible name for a button:

1. Open the Accessibility panel for the button.
2. Select the Make object accessible checkbox.
3. Enter the link text in the Name field.
 - a. Aim for no more than 50 characters whenever possible.
 - b. Do not use quotation marks.
4. If it is necessary to enter supplementary information, you may do so in the Description field. Screen readers may read both fields together, so make sure they flow without repetition.

Using ActionScript

In Flash, it is a good practice to create a layer entitled Actions in your first frame. Then, to enable accessibility and provide an accessible name for link text with ActionScript 3 instead of the Accessibility panel, you can use the following code in the Actions panel:

```
if (!myTimeSheetButton.accessibilityProperties)
myTimeSheetButton.accessibilityProperties = new
AccessibilityProperties();
```

```
myTimeSheetButton.accessibilityProperties.name =
"Time Sheet Policy"
```

```
myTimeSheetButton.accessibilityProperties.silent = false;
```

```
myTimeSheetButton.accessibilityProperties.forceSimple = true;  
Accessibility.updateProperties();
```

Checking Your Work

The VHA Section 508 checkpoints related to link text are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

You can use the Object Inspector or a screen reader to check that your links are meaningful and accessible.

Using the Object Inspector

To use the Inspect32 tool to check your link text:

1. On the Inspect32 Options menu, select the following:

- * On Top
- * SPI_SCREENREADER flag
- * Watch Focus
- * Watch Cursor
- * Show Highlight Rectangle (optional)

2. Open the Flash application in Internet Explorer or Firefox.

3. Navigate to a link.

4. View the accessibility information in Object Inspector for the link by tabbing to the link with both windows open.

5. Verify that the accessible name property is concise and meaningful even when you read it out of the context of the surrounding page.

6. Repeat steps 3-5 for subsequent links, verifying that accessible name properties are not repeated.

Using a Screen Reader

To use a screen reader to check your link text:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.

2. Activate the Flash application.

3. With the screen reader running, Tab through the application, listening to how the screen reader announces the links and buttons.

4. Verify that the announced text:

- * Is concise
- * Provides a clear purpose
- * Is not repeated
- * Is meaningful when taken out of context

Select Next to learn how to ensure that your font is readable by AT.

How to Ensure Consistent Use of Images

One simple step you can take to ensure that images are used consistently in your Flash applications is to draw from a single image library for each application. This is especially important when more than one developer is working on a project. Remember, however, that the Flash image library does not store accessibility properties. You should, therefore, establish standards and naming conventions for your project so that different instances of an image receive consistent accessible names.

You can use either the Accessibility panel or ActionScript to enable accessibility and provide an accessible name for your images. These tools will not ensure that you are using images consistently. You must remain vigilant and remember to check your work.

Using the Accessibility Panel

To use the Accessibility panel to enable accessibility and provide an accessible name for an image:

1. Open the Accessibility panel for the image.

2. Select the Make object accessible checkbox.

3. Enter text in the Name field conveying the meaning or function as briefly as possible.

- a. Aim for no more than 50 characters whenever possible, with the most important information first.
- b. Do not use quotation marks.

4. If it is necessary to enter supplementary information, you may do so in the Description field. Screen

readers may read both fields together, so make sure they flow without repetition.

Using ActionScript

If you have created an Actions layer in your first frame, you can enable accessibility and provide an accessible name for an image with ActionScript 3 instead of the Accessibility panel, by using the following code in the Actions panel:

```
if (!myCertificateButton.accessibilityProperties)
myCertificateButton.accessibilityProperties = new
AccessibilityProperties();
```

```
myCertificateButton.accessibilityProperties.name =
"Print Certificate";
```

```
myCertificateButton.accessibilityProperties.silent = false;
```

```
myCertificateButton.accessibilityProperties.forceSimple = true;
Accessibility.updateProperties();
```

Checking Your Work

The VHA Section 508 checkpoints related to consistent use of images are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

You can use the Object Inspector or a screen reader to check your work.

Using the Object Inspector

To use the Inspect32 tool to check your link text:

1. On the Inspect32 Options menu, select the following:

- * On Top

- * SPI_SCREENREADER flag

- * Watch Focus

- * Watch Cursor

- * Show Highlight Rectangle (optional)

2. Open the Flash application in Internet Explorer or Firefox.

3. Navigate through the Flash application, bringing each actionable image into focus by tabbing to it. For non-actionable images, hover over the image with the mouse to view the accessibility properties.

4. Verify that each image has an accessible name property.

5. If an image appears more than once:

- a. Verify that it performs the same function.

- b. Verify that all instances are named consistently.

6. Verify that the same image is always used for any one function.

7. Verify that no image is used to represent more than one function.

Using a Screen Reader

To use a screen reader to check your images for consistency:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.

2. Activate the Flash application.

3. Navigate to each graphic:

- a. Tab to actionable graphics.

- b. Use the screen reader command to navigate to non-actionable graphics. This command depends on the screen reader you are using. In JAWS, you would press "g".

4. Verify that each image is announced with appropriate text.

5. If an image appears more than once:

- a. Verify that it performs the same function.

- b. Verify that all instances are named consistently.

6. Verify that the same image is always used for any one function.

7. Verify that no image is used to represent more than one function.

Technical Knowledge Checks

Check your knowledge of the procedures for creating accessible links and ensuring that your images are used consistently. Use the Answer buttons to check your selections.

Top of Form

If link text has a clear purpose and a unique target and makes sense both in and out of context, the link will be accessible in Flash.

- A. True
- B. False

Bottom of Form

Top of Form

You do not need the Object Inspector or a screen reader to determine whether a Flash application uses images consistently to represent functions — one function per image and one image per function. But you must use these tools to test for additional consistent use criteria. What are these criteria?

- A. When the same image is used more than once, it must be identical in size, shape and orientation and it must have the same accessible name property.
- B. Each image must have an accessible name and when an image is used more than once, the accessible names must be consistent.
- C. Each image must be stored in the Flash image library.
- D. When the same image is used more than once, each instance must have an identical accessible name.

Bottom of Form

Avoiding Flicker

Introduction

Flash makes animation easy. This allows skilled developers to create rich, multimedia presentations that convey content in unique and valuable ways. But it also allows less experienced developers to make almost any application blink, flash or flicker.

For most users, these blinking applications are annoying. For users with low vision, they can also be distracting. And for users with photosensitive epilepsy, they can be dangerous. Content that flashes, blinks or flickers within certain ranges can induce deadly seizures in these users. In this lesson, you will learn how to ensure that your Flash applications are safe for all users, including those with photosensitive epilepsy.

Flicker Requirements

Flicker, blink, flash, pulse, strobe...these terms are used interchangeably here because Section 508 sets clear, measureable standards to ensure the safety of users with photosensitive epilepsy. Examples of flickering content include:

- * Flashing text
- * Rapid changes in background color
- * Pulsing lights, strobe effects
- * Movie scenes that include gun fire or lightning
- * Timers or buttons that refresh frequently

It is best to avoid this type of flickering content whenever possible. Providing users with an option to disable flashing content does not provide adequate protection because of the possibility that users will not notice the option in time. Seizures can occur faster than most users could disable the flashing.

If you must include flickering or flashing content in your application, you can ensure user safety by using a blink rate that is either very slow or very fast. Section 508 prohibits content that flashes at a rate between 2 and 55 hertz (Hz). (A Hz is one flash per second.) This means that flickering content must blink fewer than two times per second or more than 55 times per second. If you comply with the Section 508 flicker rate, your content will be safe, regardless of the size, color or duration of your blinking content. You will learn more about the procedures for testing the flicker rate later in this lesson.

Conceptual Knowledge Checks

Check your knowledge of flicker. Use the Answer button to check your selection.

Top of Form

You are supervising the development of a course that all personnel will be required to take. The current design is for a Flash course that displays a flickering flag whenever material that the student will be tested on is presented. What guidance should you give the course designers?

- A. Make sure the flag flickers very slowly — fewer than two times per second.
- B. Make sure the flag flickers very fast — more than 55 times per second.
- C. Provide an option at the beginning of the course that allows students to turn off the flickering flag.
- D. Remove the flickering flag; design another method to alert the student to important material.

Bottom of Form

How to Ensure Safe Flicker Rates

The VHA Section 508 checkpoints related to flicker are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Flicker

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(k)

Software shall not use flashing or blinking text, objects, or other elements having a flash or blink frequency greater than 2 Hz and lower than 55 Hz.

k.1

Is the flicker so fast that it is unnoticeable?

* *

*

k.2

Is the flicker slower than twice per second?

* *

*

It is best to avoid flickering content whenever possible. If your Flash application must include content that blinks or flickers, make sure that it flickers in a range that will not cause seizures. You can manually test your content or you can use a tool designed for this purpose.

Testing

To test the flicker rate for a blinking element in Flash:

1. Display the blinking element in Internet Explorer or Firefox.
2. Determine whether the flicker is noticeable.
 - a. If the flicker is so fast that it is unnoticeable, it is probably faster than 55 Hz. You can confirm this with the tool described below.
 - b. If the flicker is noticeable, continue to the next step or use the tool described below to determine the flicker rate.
3. Set a timer or have someone assist you with timing.
4. Count the number of times the element blinks within 10 seconds.
5. Divide the number of blinks by 10 (the number of seconds).
6. If the flicker rate is between two and 55 blinks per second, then you must either slow it down to fewer than two or speed it up to more than 55.

Using the Photosensitive Epilepsy Analysis Tool

You cannot use assistive technology (AT) to measure flicker. You can, however use the Photosensitive Epilepsy Analysis Tool (PEAT) for this purpose. This tool, available as a free download from the Trace Center at the University of Wisconsin-Madison, measures the time intervals between blinking or flickering content.

Technical Knowledge Check

Check your knowledge of flicker rates. Use the Answer button to check your selection.

Top of Form

Which of the following blinking content is safe for users with photosensitive epilepsy?

- A. An element that blinks slowly enough that you can manually count and time the blinks
- B. An element that blinks twice per second
- C. An element that blinks 55 times per second
- D. An element that blinks so fast you cannot perceive the flicker with your naked eye

Bottom of Form

Using Color

Introduction

The colors you use in your applications affect accessibility for users with low vision and for users with a color deficiency such as color blindness. Section 508 includes several requirements and functional criteria related to color and contrast. Currently, Flash cannot meet some of these requirements, so it is very important that you follow the others to make your applications as accessible as possible.

In this lesson, you will learn about the importance of sufficient color contrast and of not using color alone to convey information. You will learn how Section 508 interoperability requirements apply to color and contrast in Flash, and you will learn about providing users with color and contrast options. The technical part of this lesson covers procedures for testing contrast, for providing user options and for ensuring the appropriate use of color in Flash.

User Perspective: Insufficient Contrast

Tom Rankin, the developer of the New Employee Orientation package presented earlier in this course, decided to change the presentation's color scheme so it would look more like a popular Internet application. Here's the new look:

Like many developers who think of accessibility only in terms of screen reader users, Tom did not consider users with low vision or color blindness when he made this change. Here's how it looks now to some of the new employees who need to take it:

Insufficient contrast between the foreground and background colors has made this screen inaccessible to users with low vision.

When you are choosing colors, you should also be aware that not all users see colors the same way. If you would like to see how users with some types of color blindness would see Tom's new color scheme, select the Examples link below.

Provide Sufficient Contrast

As you have just seen, users with low vision or color blindness may have difficulty reading your presentation if there is insufficient contrast between foreground and background colors. Since these users may not be working with assistive technology (AT), you cannot assume that a screen reader will read the screen for them. Section 508 includes functional criteria to help you choose colors with enough contrast to support these users.

In Flash, the best time to think about color and contrast is in the early stages of development, when you are establishing a look and feel and choosing a color scheme. Be sure to consider all the elements that convey information, including:

- * Text
- * Graphs, charts
- * Animation
- * The focus rectangle (the solid or dotted border drawn around the control that has keyboard focus)

You do not need to worry about color and contrast for elements that do not convey information; these include:

- * Decorative elements
- * Logos and brand names
- * Inactive user elements
- * Invisible elements

Ensuring sufficient contrast between your foreground and background colors is straightforward. Each color is defined by name or as a six-character hexadecimal code, and VHA Section 508 checkpoints provide ratios that define sufficient contrast between foreground and background colors. In addition, there are several free online color and contrast analyzers available that you can use to check and adjust your colors against

the ratios.

Here are some examples of accessible color schemes, along with their color codes:

Here are some examples of inaccessible color schemes, with their color codes:

The VHA Section 508 checkpoints that include contrast ratios are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Sufficient Contrast

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(b)

At least one mode of operation and information retrieval that does not require visual acuity greater than 20/70 shall be provided in audio and enlarged print output working together or independently, or support for assistive technology used by people who are visually impaired shall be provided.

b.1

Does text provide sufficient color contrast?

* *

*

b.1.a

Does normal text less than 18 pt use a contrast ratio of at least 4.5:1?

* *

*

b.1.b

Does normal text of at least 18 pt and bolded text of at least 14 pt use a contrast ratio of at least 3:1?

* *

*

b.2

Do icons, images of text, and diagrams and charts use appropriate contrast levels (as described above for text)?

* *

*

Links to color and contrast analysis tools are provided in the Resources section of this course. Procedures for using one of these tools are provided later in this lesson.

Honor User-Selected Color and Contrast Settings

Users with low vision and users who perceive color differently often use operating system (OS) features to set their own foreground and background colors to make them easier to read and follow. Some users choose high contrast, some prefer low contrast, and others choose different colors based on how they perceive color. To see examples of some of the color and contrast options a user might choose with the Windows OS High Contrast feature, select the Examples link below.

As you learned in the Working with OS/AT Accessibility Features lesson, one of the most basic principles of accessibility is interoperability. This principle requires that software applications not disrupt, disable or otherwise interfere with OS accessibility features. Based on this important principle, Section 508 specifically requires applications to honor user-selected color and contrast settings. The VHA Section 508 checkpoints related to interoperability and user-selected color and contrast settings are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

You will not be able to fully comply with these requirements in Flash because Flash doesn't support the Windows OS display accessibility options designed for users with low vision and color blindness. One alternative that is recommended to meet this requirement is to provide users with color and contrast options from within your application.

Provide Options for Color and Contrast

Since Flash does not support the color and contrast settings selected by users in Windows, a recommended alternative is to provide a variety of color and contrast options from within your application to make it accessible to users with low vision and to users with a color deficiency. Your options should allow users to choose from a full range of color and contrast settings, including at least two high contrast options—one with dark text on a light background and one with light text on a dark background — and at least two low contrast options.

Providing these options in Flash requires planning and foresight. You must store the color options as variables that can be changed, instead of hard-coding them as you go along. You do not have to provide options for every element in Flash, but you should provide them for any elements that convey information. You will get the greatest benefit from this effort early in the development process, especially when you are developing reusable templates. Procedures for providing color and contrast options are provided later in this lesson.

User Perspective: Color Coding

It's Wednesday, July 4. You live in California and you need a SmartPhone right away. Check out the SmartPhone availability chart below to find a store with phones available.

If you can distinguish red from green, you probably didn't even read the legend on this chart. Most likely, you scanned the right column for green circles and identified the stores in San Luis Obispo and Santa Clara as the ones that have SmartPhones available.

If, however, you were among the seven to 10 percent of the population with red-green color blindness, here is what you would have seen:

This example illustrates the importance of not relying on color alone to convey information.

Avoid Color Coding

Color can be an effective way to draw a user's attention to important information or features, but when you rely on color alone to convey information, you create a product that is inaccessible to users who are blind or who have a color deficiency such as color blindness. In the Designing for Accessibility lesson, you saw how this concept applies to instructional text. In addition, you must avoid using only color—a practice known as color coding—for any of the following purposes:

- * Conveying information
- * Indicating action
- * Prompting a response
- * Distinguishing visual elements
- * Communicating selection
- * Communicating error

This requirement should not discourage you from using color. Rather, your goal should be to provide additional, alternative methods of conveying the same information whenever you use color. In most cases, you need two alternatives:

- * An on-screen indicator for color blind users not using assistive technology
- * A programmatic indicator or textual equivalent that is exposed to AT for users who are blind

On-Screen Alternatives to Color Coding

Most users who are color blind do not use AT and cannot easily access text equivalents designed for screen reader users. These users need an on-screen alternative that is physically close to the color-coded material on the screen. The alternative must also appear automatically. It should not require the user to perform a task, such as mousing over the material, to expose it, since a color blind user may not even realize when material is color coded.

One on-screen approach to making color-coded material accessible to color blind users is to add words near the color-coded material, as in the following image of the SmartPhone chart presented earlier:

This example allows users who see color to obtain information via the colored symbols, while users with color blindness obtain the same information from the words on the screen. Examples of other ways you can use words as on-screen alternatives to color coding include:

- * Adding the word ERROR to the beginning of an error message, instead of relying on the fact that the message is red to indicate that the user has made a mistake
- * Labeling buttons Submit and Cancel instead of relying on the fact that the buttons are green and red to convey their purpose

If you cannot or do not want to add words on the screen, you can sometimes add formatting or symbols to your color-coded content. Examples of this approach include:

- * Adding larger fonts or boldface to colored text to highlight important content
- * Adding asterisks to colored text to indicate required fields on a form
- * Adding hatch marks, variable borders, shading or other symbols to color-coded diagrams, pie charts and graphs

Programmatic Alternatives to Color Coding

On-screen alternatives to color coding help to make color-coded content accessible to users with low vision or color deficiencies, but they may not help screen reader users. To support AT users, you must also provide programmatic alternatives in the form of text equivalents or other programmatic information that is conveyed through the MSAA. Of particular relevance in Flash is the use of color to indicate state, such as when inactive elements are grayed out. In these cases, you must remember to set the states programmatically, to "unavailable" or "selected", for example, in addition to using color.

You will learn more about text equivalents and setting states programmatically in the Providing Text Equivalents and Providing Accessible User Interface Controls lessons of this course.

Conceptual Knowledge Checks

Top of Form

Since Windows provides accessibility features that allow users to set their own

foreground and background colors, you can use any color scheme to create accessible Flash.

- A. True
- B. False

Bottom of Form

Top of Form

If your color scheme provides sufficient contrast, as defined by the ratios in the VHA Section 508 checkpoints, it is not necessary to provide additional color and contrast options for the user.

- A. True
- B. False

Bottom of Form

Top of Form

For which Flash elements are you NOT required to provide sufficient contrast or contrast options?

- A. Focus rectangles
- B. Logos
- C. Animations
- D. None of the above. You must provide sufficient contrast or contrast options for all elements in a Flash application.

Bottom of Form

Top of Form

Why is it necessary to provide two alternatives to color-coded content in Flash?

- A. Because redundancy is built into Section 508 requirements to ensure accessibility for all users
- B. Because you need one alternative for users who are blind and one for users who are deaf
- C. Because users with different learning styles need different alternatives, such as words for verbal learners and symbols for visual learners
- D. Because color blind users may not be able to access alternatives designed for assistive technology users, and AT users may not be able to see on-screen alternatives designed for color-blind users

Bottom of Form

How to Ensure Sufficient Contrast

Flash makes it easy to create custom colors and to apply colors from other applications, but when you create and alter colors as you go along, it can become difficult to manage contrast levels. It is wise, therefore, to establish color combinations with sufficient contrast at the beginning of a project and save your palette for use throughout the project. This not only creates a uniform look and feel that all users can appreciate, but also ensures accessibility for users with low vision and color blindness.

The process of determining whether two colors have sufficient contrast for accessibility is simple with some free online tools. Basically, you plug your foreground and background colors into a color analyzer, and the analyzer uses an algorithm to determine the contrast ratio between the colors. You can then compare this ratio to the functional criteria in the VHA Section 508 checklist. You will find links to several free, online color and contrast analyzers in the Resources section of this course.

Example

The best way to understand this process is to try it yourself. Let's walk through an example using the Colour Contrast Analyser available from The Paciello Group, shown below. The basic steps for using this tool to determine the contrast ratio between two colors are as follows:

1. Select or enter a foreground color.
2. Select or enter a background color.
3. Select Luminosity under Algorithm. This algorithm provides results that you can compare to the ratios in the VHA Section 508 checklist.
4. Check the resulting contrast ratio.

Let's consider these steps in more detail.

Selecting and Entering Colors

You can select or enter your foreground and background colors, using one of three methods:

- * Select a color from the palette available from the dropdown next to the Colour select field.
- * Enter a hexadecimal code in the Hex field.
- * Use the eyedropper tool to grab a sample of an existing color.

When you select a color, either from the palette or with the eyedropper tool, the hexadecimal code for that color appears in the Hex field. Alternatively, you can enter a code in the Hex field and the color associated with that code will appear in the Colour select field. In the example above, the foreground color selected is black and the background color selected is white; the corresponding hexadecimal codes that appear in the Hex field are #000000 and #FFFFFF.

To use the eyedropper tool to select a color that you have created, either in Flash or in another application:

1. Select the eyedropper; a magnifier will allow you to navigate to the precise location of the color you wish to choose.

2. Drag the eyedropper tool until the crosshairs are over the color you wish to select.

3. Select the color. The color will appear in the Colour select field and the hexadecimal code for that color will appear in the Hex field.

In the example shown below, the pool of paint on the eyedropper icon was selected. You can see the turquoise color in the Foreground Colour select field and the corresponding code in the Hex field.

Notice that the results for this combination of colors (turquoise foreground on a white background) now say Fail instead of Pass. Let's look more closely at the results provided by this tool.

Reading Contrast Ratio Results

To understand the results of the Colour Contrast Analyser in context, you may wish to open the VHA Section 508 checkpoints related to sufficient contrast. They are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Sufficient Contrast

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(b)

At least one mode of operation and information retrieval that does not require visual acuity greater than 20/70 shall be provided in audio and enlarged print output working together or independently, or support for assistive technology used by people who are visually impaired shall be provided.

b.1

Does text provide sufficient color contrast?

* *

*

b.1.a

Does normal text less than 18 pt use a contrast ratio of at least 4.5:1?

* *

*

b.1.b

Does normal text of at least 18 pt and bolded text of at least 14 pt use a contrast ratio of at least 3:1?

* *

*

b.2

Do icons, images of text, and diagrams and charts use appropriate contrast levels (as described above for text)?

* *

*

When you have entered foreground and background colors and selected Luminosity as your algorithm, the Colour Contrast Analyser displays results in a ratio that you can compare to the functional criteria in the VHA Section 508 checkpoints. In the examples above:

* Black text on a white background resulted in a contrast ratio of 21.0 to 1.

* Turquoise text on a white background resulted in a contrast ratio of 2.3 to 1.

Compare these ratios to the criteria outlined in the VHA Section 508 checklist. Notice that the checkpoint criteria include different ratios for text, depending on size and boldness:

* Text less than 18 pt, images, diagrams and charts should have a contrast ratio of at least 4.5 to 1.

* Text that is 18 pt or more and bolded text that is at least 14 pt or more should have a contrast ratio of at least 3 to 1.

Although you can use the procedures outlined above to test for sufficient contrast in an existing application after it is developed, you will save considerable time and effort if you apply them early in the development of a new application, when you are choosing your color scheme.

How to Provide Color and Contrast Options

Since Flash does not honor the color and contrast selections made by users in the operating system, an alternative is to provide users with options from within your Flash application. Such options should represent a full range of color and contrast.

There are several ways to provide users with color and contrast options in Flash. One approach is to provide buttons in a dropdown menu, as shown below. In this example, users are provided with four different contrast settings on the first screen of the presentation. Each option includes preset color values for the foreground and background. The instance names for the buttons are shown to the right in the

image below.

For this approach, you can define your components and text fields in arrays. This allows the application to change the foreground and background colors of all the elements together, using a switch statement. When the user selects a contrast option from the dropdown menu, an update function is called that sets the foreground and background colors of all the components and text fields that were defined in the array. The following sample ActionScript 3 code uses the `TextFormat` class and `opaqueBackground` property to set the colors for components; it uses the `backgroundColor`, `textColor` and `background` properties to set the colors for text fields:

```
import fl.core.UIComponent;

//All buttons or components can be added to this array
var componentItems:Array = new Array(bt1);
//All text objects can be added to this array
var textItems: Array = new Array(dt1);

// each item in the color choice is represented by a button
// a mouse event is attached to each button
// (enter also triggers this event)
btYellowOnBlack.addEventListener(MouseEvent.CLICK,changeColors);
btBlackOnWhite.addEventListener(MouseEvent.CLICK,changeColors);
btBlackOnGray.addEventListener(MouseEvent.CLICK,changeColors);
btGrayOnBlue.addEventListener(MouseEvent.CLICK,changeColors);

function changeColors(e:MouseEvent): void
{ switch (e.target)
{
case (btYellowOnBlack) :
updateComponentColors(componentItems,0xffff00,0x000000);
updateTextColors(textItems,0xffff00,0x000000);
break;

case (btBlackOnWhite) :
updateComponentColors(componentItems,0x000000,0xffffffff);
updateTextColors(textItems,0x000000,0xffffffff);
break;

case (btBlackOnGray) :
updateComponentColors(componentItems,0x000000,0xcccccc);
updateTextColors(textItems,0x000000,0xcccccc);
break;

case (btBlueOnGray) :
updateComponentColors(componentItems,0x0000ff,0xcccccc);
updateTextColors(textItems,0x0000ff,0xcccccc);
break;

default :
updateComponentColors(componentItems,0xff00ff,0xffff00);
updateTextColors(textItems,0xff00ff,0xffff00);
break;
}
}

// called from the changeColors function
function updateComponentColors(componentItems:Array,
newForegroundColor:uint, newBackgroundColor:uint)
```

```

{ for each (var i:UIComponent in componentItems)
{
var myTextFormat:TextFormat = new TextFormat();
myTextFormat.color = newForegroundColor;
i.opaqueBackground = newBackgroundColor;
i.setStyle("textFormat", myTextFormat);
}
}

```

```

// called from the changeColors function
function updateTextColors(textItems:Array, newForegroundColor,
newBackgroundColor) {
for each (var i:TextField in textItems)
{
i.textColor = newForegroundColor;
i.background = true;
i.backgroundColor = newBackgroundColor;
}
}

```

} Checking Your Work

The VHA Section 508 checkpoints related to providing color and contrast options are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

To test your application for compliance with these checkpoints:

1. Make sure you have provided a variety of color and contrast options, including at least two high contrast options and at least two low contrast options.
2. Select each option and proceed through the application, ensuring that all user interface elements follow the color and contrast settings associated with the selected option. Check all of the following:
 - a. Window titles, borders, text and backgrounds
 - b. RadioButtons, CheckBoxes, buttons, ListBoxes and ListViews, TreeViews, ComboBoxes, TextInput fields
 - c. User selections
 - d. Icons, toolbars and menus
 - e. Message boxes and ScrollBars

How to Ensure There Are Alternatives for Color-Coded Content

Earlier in this lesson, you learned about the Section 508 requirement to avoid color coding and the requirement to provide two alternatives to color-coded content. For developers without a visual disability, the trickiest part of making color-coded content accessible can be recognizing its existence in an application. Once recognized, providing alternatives is relatively straightforward.

There are many ways to provide on-screen alternatives to color-coded content, some of which appear earlier in this lesson. The procedures for providing programmatic alternatives to color-coded content are the same as those for making any other content accessible to AT. They include enabling accessibility, providing accessible names and text equivalents and creating accessible interface controls. You will find the procedures for performing these tasks elsewhere in this course.

Check Your Work

The VHA Section 508 checkpoints related to color coding are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

To ensure that any color-coded content in your application is accessible, you must first identify it and then check for both on-screen and programmatic alternatives.

Identifying Color-Coded Content without Assistive Technology

Without assistive technology, the most effective way to identify color-coded content is to use black and white contrast settings, either on the computer screen or on a printer. Using only black and white, proceed

through an application, performing all necessary functions, including:

- * Reviewing and activating user interface controls
- * Reading instructions, answering questions, reading error messages
- * Selecting, entering and editing text
- * Finding information in the Help files
- * Printing, if applicable

If you find that you cannot understand some content or perform some tasks without color, then you have identified color-coded content. It is helpful to have someone other than the application's developers perform this task, since developers may be so familiar with the material that they "see" the colors they created even when the application is in black and white.

Ensuring On-Screen Alternatives for Color-Coded Content

Once you have identified color-coded content, you must ensure that there are on-screen alternatives for users with low vision or color blindness. If any content relies on color to convey meaning, you must add on-screen indicators, such as text or symbols, to convey the same meaning. This includes making sure that:

- * Important cues, such as required fields or emphasized content, are clear
- * Text and image meaning is clear
- * Graphs and charts (including legends) convey meaning
- * Error messages are identifiable
- * User selections are clear
- * Instructions are clear
- * Controls, including control state (e.g., active, inactive, selected), are clear and identifiable

Identifying Color-Coded Content with a Screen Reader

To identify color-coded content with a screen reader, turn off the computer display and proceed through the application, performing all the necessary functions, as indicated above. Listen carefully to ensure that error messages, interface controls, instructions and required fields are clear and identifiable.

Ensuring Programmatic Alternatives for Color-Coded Content

You must also ensure that AT users can access color-coded content through programmatic alternatives. If any content relies on color to convey meaning, you must add programmatic alternatives or text equivalents that are exposed through MSAA to AT. This includes making sure that the screen reader announces, describes or otherwise identifies:

- * Important cues, such as required fields or emphasized content
- * Meaning conveyed by image color
- * Meaning conveyed by color in graphs and charts
- * Errors
- * User selections
- * Instructions
- * Controls, including control state (e.g., active, inactive, selected)

Technical Knowledge Checks

Check your knowledge of the procedures for ensuring accessible color and contrast in Flash. Use the Answer buttons to check your selections.

Two color combinations have been placed in the Colour Contrast Analyser and the results are shown below. For each image, assume that the foreground colors selected are for 12 pt Arial text. Based on Section 508 functional criteria for users with low vision, determine whether the color contrast is sufficient or insufficient. The Section 508 criteria are available in Resources.

Top of Form

The color contrast shown in the tool above is:

- A. Sufficient
- B. Insufficient

Bottom of Form

Top of Form

The color contrast in the tool above is:

- A. Sufficient
- B. Insufficient

Bottom of Form

Top of Form

How does defining all elements that require good color contrast in a dynamic ActionScript function help you meet color and contrast accessibility requirements in Flash?

- A. By allowing the application to change the colors of all the components together based on user-selected options in the application
- B. By ensuring that your foreground and background colors have sufficient contrast
- C. By helping your application follow user color and contrast settings from the operating system
- D. By helping you identify color-coded content

Bottom of Form

Providing Controls for Audio, Video and automatically Updating Content

Introduction

You can use Flash's multimedia capabilities to create applications appropriate for a variety of abilities and learning styles, but without user control over the audio and video, your applications may conflict with assistive technology (AT), rendering your content inaccessible. To make exciting presentations that include scrolling text, moving characters, slide shows, animations and full-fledged multimedia accessible, you must provide AT users with the ability to control the audio and video.

In this lesson, you will learn why Section 508 requires you to provide audio and video controls, including the importance of allowing users to control animations and automatically updating content, and why you should avoid audio that plays automatically. The technical part of this lesson introduces some approaches to providing accessible audio and video controls. To ensure that the controls you provide are truly accessible, you will need to incorporate additional information provided later in this course, in the following lessons:

- * Ensuring Keyboard Accessibility
- * Controlling Reading Order and Tab Order
- * Maintaining Focus
- * Providing Accessible User Interface Controls

User Perspective: No Video Controls

Tom Rankin assigned his assistant Betty to create the New Employee Orientation chapter on Parking Policy. Select the Example link below to see part of what she has created. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

This example illustrates the importance of providing all users with the ability to control the timing of video

presentations. Most users would have preferred to stop this presentation at various spots to note phone numbers or study the maps. If they really needed the information, they would probably have replayed the presentation multiple times until they got it all. A frustrating experience that pales in comparison to that of the screen reader users in the audience!

Screen readers interact with this type of content in various and unpredictable ways. In this case, the screen reader began reading each new page when it appeared, even when it had not read all the content on the previous page. In other cases, the screen reader might get stuck reading text that had already disappeared from the screen. Either way screen reader users would not hear some of the content.

Provide Video Controls

In Flash, there are different types of visual content for which you should consider providing controls; the most accessible approach depends on your application:

- * For applications that include multimedia video presentations, provide standard multimedia controls.
- * For animated content, including slide shows, provide step-through controls.
- * For content that updates automatically, provide separate on-screen controls.

Multimedia

All users, not just users with disabilities, should be able to control multimedia applications. Whenever possible, include full video controls with these applications: Rewind, Play/Pause, Reverse, Forward and Fast Forward. Sometimes, you may need to omit certain controls, such as the Fast Forward control for mandatory training material. If full controls are not practical, such as for very short videos or when you are streaming live multimedia, you should at least provide users with the ability to pause and re-start the presentation. Here are some examples of controls for multimedia:

The default player controls available in Flash are not always accessible without additional configuration. More accessible tools for creating multimedia controls for Flash are the CCforFlash component and ccPlayer, developed by the National Center for Accessible Media (NCAM). A link to these tools is provided in the Resources section of this course. Additional information about providing accessible video controls is provided later in this lesson.

Animation

Animated content includes scrolling text, moving characters, arrows or other figures that move around to describe work flow and slides with text or images that appear and then disappear. Screen reader users often find this type of content challenging. While the screen reader is reading one section, the application may be moving on to the next, interrupting the screen reader and preventing users from hearing all the content. Users with cognitive disabilities may also have difficulty absorbing and following content that moves forward on its own.

To provide all users with direct, equivalent access to this type of content, you should allow users to step through the content at their own pace. The most effective way to do this is by inserting pause points in your animation and including Pause and Next buttons. This allows all users to hear all the content in one section before they move on to the next, and it provides additional time for users with cognitive disabilities to understand the content or perform a task. Planning for and designing these controls in the early stages of development, giving careful thought to appropriate pause points, will save you time and effort and result in an accessible product.

In most cases, alternatives to animated content do not provide equivalent access. If an alternative is truly necessary, you will come closest to creating an equivalent experience for AT users by using the markup and interactive capabilities of HTML.

Reminders about Accessible Animation

Previous lessons in this course included additional caveats about making animation accessible, including the following:

- * See the Avoiding Flicker lesson for requirements to avoid images that blink or flicker.
- * See the Hiding Flash lesson for limitations on looping animations.
- * See the Designing for Accessibility lesson for the requirement that screen transition animations settle within five seconds on the screen.

Auto-Updating Content

Content that updates automatically, such as stock tickers, scoreboards, clocks and real-time information

boards, creates its own set of challenges. Constant changes can disrupt screen readers and distract sighted users. More often in Flash, automatically updating content is ignored completely by AT, and AT users are denied access to the information it contains.

To make automatically updating content accessible, you should provide a separate control, near the content on the screen, for starting and stopping the flow of information. Give careful thought to what happens when a user stops and restarts the content. There are three approaches, depending on the content:

- * You should start the content where the user stopped it if the presentation is sequential and includes information that should not be skipped. If, for example, a user pauses a series of network connection diagnostics, he or she should be able to restart at the same place.

- * You should start a program at the current time, regardless of where the user stopped it, for real time content such as a stock ticker.

- * You may choose to start the content at the beginning for short, cyclical presentations.

Procedures for providing controls for automatically updating content are provided later in this lesson.

User Perspective: No Audio Controls

Recognizing that many new employees don't like to read, Tom Rankin decided to add an audio track to his New Employee Orientation package.

Select the Example link below to see how it works for new employees who are using a screen reader. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Provide Audio Controls

Screen reader users cannot listen to an audio track and the screen reader at the same time. When audio plays while a screen reader is voicing, the content becomes inaccessible. Audio that plays automatically when an application is loaded is especially disruptive. So, whether your audio is a narration, dialog or background music, you should avoid interference with AT in the following ways:

- * Avoid audio that plays automatically.

- * Provide the user with audio controls as early in the application as possible.

It is best to ensure that your audio does not play automatically. If it must, do not allow it to play for more than a few seconds. Provide the option to turn off the audio as the first element in your Flash application so that users can silence disruptive audio quickly.

Procedures for providing audio controls are similar to those for providing video controls. If you are creating a multimedia application and have included multimedia controls, users will already be able to play and pause the audio with the video. As with video controls, you should include appropriate keyboard shortcuts so that users can access them easily. You will learn more about keyboard shortcuts in the Ensuring Keyboard Accessibility lesson of this course.

In addition to Play and Pause, accessible audio controls should include a Mute control that allows the user to silence the audio without stopping it and a volume control. These additional controls allow screen reader users to temporarily listen to the screen reader when necessary and to play some less critical audio, such as background music, more quietly.

Conceptual Knowledge Checks

Check your knowledge of Section 508 requirements for providing audio and video controls. Use the Answer buttons to check your selections.

Top of Form

Which of the following is the best approach to making animated content accessible in Flash?

- A. Provide complete multimedia controls, including Rewind, Play/Pause, Reverse, Forward and Fast Forward, that allow users to proceed at their own pace.

- B. Provide step-through controls, including Pause and Next, that allow users to proceed at their own pace.
- C. Use the interactive and markup capabilities of HTML to provide an accessible alternative.
- D. Provide a text alternative, as Flash animation cannot be made accessible any other way.

Bottom of Form

Top of Form

Which of the following is the most accessible way to handle important content that updates automatically in Flash?

- A. Hide it from assistive technology to avoid disruption.
- B. Limit the number of times it loops.
- C. Turn off the updates and provide a text alternative that includes the same content.
- D. Provide a control that allows the user to start and stop it.

Bottom of Form

Top of Form

Why is it important for accessibility to avoid audio that plays automatically?

- A. Because the audio is likely to be playing at the same time that screen readers are reading other content on the screen
- B. Because the audio may start out too loud and damage users' hearing
- C. Because users may not be ready to listen to it when it starts
- D. Because audio that plays automatically cannot be captioned

Bottom of Form

How to Provide Accessible Multimedia Controls

The VHA Section 508 checkpoints related to the provision of multimedia controls are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

Tools for Creating Accessible Controls

Two recommended ways to create accessible controls, using free tools available from the National Center for Accessible Media (NCAM) are:

- * Add the CCforFlash component to your Flash application, and use it in combination with a standard Flash media player, such as FlvPlaybackControl.
- * Use ccPlayer.

You can download the CCforFlash component or ccPlayer from the NCAM web site. You will find a link to these tools in the Resources section of this course.

With ccPlayer, use the ccVideoAutoStart parameter in calling the movie to play:

- * If you embed the movie, put the parameter in the embed tag's src attribute, using the following code:

```
* src="ccPlayer.swf?ccVideoAutoStart=false&ccVideoBuffer  
* Time=0&ccCaptSourceType=external&ccVideoName=myVideo.f  
* lv&ccCaptionSource=myVideo.dfxp.xml&ccCaptionLanguage=  
* en&ccCaptionAutoHide=false&ccOverrideFileStyle=false&c  
cDisplayRollup=false"
```

- * If you use Flash's version detector program AC_OETags.js, put the parameter in the movie parameter in your call to AC_FL_RunContent: "movie", as follows:

```
* "movie", "ccPlayer.swf?ccVideoAutoStart=false&ccVideoB  
* ufferTime=0&ccCaptSourceType=external&ccVideoName=myVi  
* deo.flv&ccCaptionSource=myVideo.dfxp.xml&ccCaptionLang
```

* uage=en&ccCaptionAutoHide=false&ccOverrideFileStyle=false&ccDisplayRollup=false",

Additional Accessibility Work Required

This lesson has focused on helping you to understand the need for audio and video controls. To ensure that your audio and video controls are accessible, you will need to understand additional concepts and procedures covered later in this course.

Accessible Name, Role, State, Value

All controls, including audio and video controls, should have an accessible name, role and state; some controls should also have an accessible value. These accessibility properties must change, as appropriate, based on user action. You learned about accessible names in the Enabling Accessibility lesson of this course. You will learn more about the other accessibility properties of controls in the Providing Accessible User Interface Controls lesson of this course.

Keyboard Accessibility and Shortcuts

All controls, including audio and video controls, must be keyboard accessible. When there are more than five controls, you should also provide keystroke shortcuts so that users can access them easily without excessive tabbing. Examples of multimedia keyboard shortcuts include Alt+Shift+P for Play/Pause and Ctrl+Shift+F for Fast Forward. You will learn more about keyboard accessibility and shortcuts in the Ensuring Keyboard Accessibility lesson of this course.

Reading Order and Focus

All controls, including audio and video controls should appear in the proper reading order and maintain focus when activated. You will learn more about these concepts in the Controlling Reading Order and Tab Order and Maintaining Focus lessons of this course.

How to Provide Step-Through Controls for Animation

Animated content should include controls, such as Prior, Pause and Next, that allow users to "step through" at their own pace. It is also helpful to provide users with information about the current step relative to the total number of steps in an animation, for example, 2 of 5.

To create step-through controls for a Flash animation, you must plan, early in the development process, where to establish your "pause points." These are the frames to which the application will advance when a user selects Next. Each time the user selects Next, the application advances to the next frame that you have chosen as a pause point and then stops until the user chooses to move on.

One way to think about pause points is to decide where you would advance to the next slide if you were converting your animation to a slide show. It is usually good to pause where content disappears and new content appears. You can allow new content to appear after a pause point, as long as it appears within five seconds. (The requirement that screen transition animations settle within five seconds is explained in the Designing for Accessibility lesson.) You should not allow content to disappear between pause points.

Checking Your Work

The VHA Section 508 checkpoints related to animation are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Animation

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

h.1

If animated objects exist, does the information conveyed by the animated object exist in another non-animated method?

* *

*

h.1.a

Is there a non-animated method to step through or control animation?

* *

*

h.1.b

Is animation content sufficiently described in audio or text?

* *

*

h.2

Do alternatives to animation provide the equivalent functionality?

* *

*

h.3

Does screen transition animation settle within 5 seconds?

* *

*

Testing your step-through animation controls in Flash includes ensuring that the appropriate controls exist and function appropriately and ensuring that the controls are accessible. To test an animation for appropriately functioning step-through controls:

1. Make sure there are controls that pause and restart the animation.
2. Make sure there are controls that take the user to the prior and next step in the animation.
3. Step through the animation (forward and backward) using the Next and Prior controls.
 - a. Each time one of these controls is activated, make sure the application displays only the next "step" and then stops until a control is activated.
4. Make sure the Play button allows the animation to play without stopping.

Ensuring that your step-through controls are accessible is similar to ensuring that any other controls are accessible. You will need to understand additional concepts and procedures covered later in this course, including accessible name, role, state and value; keyboard accessibility and shortcuts; reading order and focus.

How to Provide Controls for Auto-Updating Content

Provide a separate control near any content that updates automatically to allow the user to stop all dynamic content updates except emergency announcements. This control may be one button that toggles between Pause and Play or two separate buttons. Consider how and when the content will restart after the user has stopped it. Following are some examples of ActionScript 3 code you can use:

* To pause:

```
mc1.stop();
```

* To restart content where it was stopped:

```
mc1.play();
```

* To jump to current content:

```
// tickerData is an XML object
```

```
// myLoader contains the location of the current XML
```

```
// data file
```

```
var tickerData: XML;
```

```
var myLoader:URLLoader = new URLLoader();
```

```
var timeout: int;
```

```
var btnPause: Button = new Button();
```

```
// this function should be called at an interval to update
```

```
// the data except when paused
```

```
// is called by a setTimeout unless the pause button is
```

```
// pressed
```

```
function loadData(): void
```

```
{ myLoader.load(new URLRequest("ticker_data.xml"));
```

```
}
```

```
myLoader.addEventListener(Event.COMPLETE, processXML);
```

```
// populate the tickerData XML variable with XML data from
```

```
// the file
```

```
function processXML(e:Event): void
```

```
{ tickerData = new XML(e.target.data);
```

```
// get appropriate nodes from XML to display with
```

```
// real time data and display " not shown
```

```
}
```

```
// attached enter and mouse click event to pause button
```

```
btnPause.addEventListener(MouseEvent.CLICK, togglePause);
```

```
function togglePause(): void
```

```
{ // if not paused
```

```
timeout = setTimeout(loadData,200);
```

```
// if paused
```

```
clearTimeout(timeout);
```

```
} * To restart auto-updating content at the beginning:
```

```
mc1.gotoAndPlay(1);
```

Checking Your Work

The VHA Section 508 checkpoints related to automatically updating content are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Auto-Updating Content

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.11

Can auto-updating content be stopped, paused or hidden?

* *

*

Testing controls for automatically updating content in Flash includes ensuring that the controls exist and function appropriately and ensuring that the controls are accessible. To test for appropriately functioning controls for auto-updating content:

1. Make sure there are controls that allow the user to pause and restart the content.
2. Test the controls to ensure that the content re-starts in the appropriate place for the type of content.
 - a. If the user needs all of the content, make sure it restarts where the user stopped it.
 - b. If content becomes irrelevant during a pause, make sure the presentation jumps to current, relevant content.

Ensuring that these controls are accessible is similar to ensuring that any other controls are accessible. You will need to understand additional concepts and procedures covered later in this course, including accessible name, role, state and value; keyboard accessibility and shortcuts; reading order and focus.

How to Provide Accessible Audio Controls

To ensure that audio does not interfere with screen readers, avoid audio that plays automatically and provide an audio on/off button near the top of the page to allow the user to quickly turn off the audio. When the button is activated, the audio should stop and the button should indicate visually and in an

accessible manner that audio is turned off. Here is an example of the type of control that should be scripted to stop and start audio:

Checking Your Work

The VHA Section 508 checkpoints related to audio controls are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Audio Controls

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.9
Does embedded audio require user action to start playing?

* *

*

a.10
Can embedded multi-media (including audio, video and animations) be controlled from the page on which they occur?

* *

*

To ensure that your application has appropriate audio controls:

1. Run the application; the audio should not begin automatically when the application loads.
 - a. If the audio must play on load, verify that it stops within three seconds.
2. Make sure a button is provided to start and stop the audio.
 - a. If the audio must play automatically for more than three seconds, make sure the button is near the top of the page and first in the tab order.
 - b. Make sure the button has an accessible name and a visual indication of its purpose.
 - c. Toggle the button to make sure it works to turn the audio on as well as off.

Technical Knowledge Checks

Check your knowledge of the procedures for providing controls. Select True or False for each of the following statements. Use the Answer buttons to check your selections.

Top of Form

If you use standard media playback controls, you can be assured that your multimedia controls will be accessible without additional configuration or testing.

- A. True
- B. False

Bottom of Form

Top of Form

A full set of standard multimedia controls provides an accessible solution for animated content in Flash.

- A. True
- B. False

Bottom of Form

Top of Form

Pause points in a Flash animation should be selected by spacing them evenly on the timeline.

- A. True
- B. False

Bottom of Form

Top of Form

When you provide controls for automatically updating content, you should always make sure the content restarts where the user stopped it.

- A. True
- B. False

Bottom of Form

Providing Captions and Visual Indicators for Sound Cues

Introduction

You can use Flash's multimedia capabilities to create powerful presentations that include both audio and video. If your application uses audio to convey information, you must provide the same information visually to make it accessible to users who are deaf or have hearing impairments. The best way to do this is with synchronized captioning. Captioning also benefits visual learners without hearing impairments.

In this lesson, you will learn about the Section 508 requirements for making Flash applications accessible to users who are deaf or have hearing impairments. You will be introduced to several approaches to captioning in Flash, as well as to some of the tools and services that are available to help you.

User Perspective: No Audio

Select the Example link below to experience a demonstration from the perspective of a user who is deaf. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Were you able to follow the demonstration? You might be surprised to learn what the presentation really demonstrated.

Select Next to experience the same demonstration with a very common, but not particularly effective, solution to this accessibility issue.

User Perspective: Transcript Only

Some developers try to meet Section 508 requirements for users who are deaf by providing a transcript. Select the example link below to experience the demonstration with an audio transcript. The presentation will open in a new window, which will include a link for the audio transcript. Select the transcript link to open the transcript before watching the video again.

Was the transcript helpful? Without the benefit of the audio track, most users learn something from the transcript that they didn't get from the non-transcript version. In this case, the transcript probably helped you figure out that the demonstrator was using voice commands to operate the Outlook application and that the computer was responding out loud.

But accessibility is not achieved by providing a solution that is "better than nothing". Equivalent access provides disabled users with content that is similar or identical to that provided to non-disabled users, in a form that produces a similar user experience. You have just experienced some of the issues that arise for users who are deaf when a transcript is the only solution provided for multimedia applications.

The Role of Transcripts

A well-written transcript may be all that is required to make audio-only presentations accessible to users who are deaf or have hearing impairments. For most Flash applications, however, transcripts do not, by themselves, provide equivalent access. For multimedia applications, transcripts are most useful as an intermediate tool for creating synchronized captions.

It may also be useful to provide the transcript in addition to synchronized captions. For example, a transcript may allow users to search for a piece of text or review on a device that does not support Flash, such as a smart phone.

Audio-Only Transcript Requirements

A transcript is sufficient by itself for users who are deaf only when your Flash application plays an audio recording with nothing to watch. In such cases, your transcript must:

- * Provide the full text of what is actually spoken in the recording
- * Include a description of all important audio cues, such as a door bell ringing
- * Be fully accessible via a nearby control

Why Transcripts are Insufficient for Multimedia

When a presentation combines video with audio, as in many multimedia Flash applications, transcripts do not provide equivalent access because they cannot be synchronized to the video presentation. In multimedia, this synchronization is a critical aspect of the user experience. If a user does not read the transcript at the same speed as the audio track, the visual presentation will not have the same effect.

Consider these examples:

- * A video demonstration of how to perform a complicated task, in which the speaker repeatedly says, "Now watch carefully while I perform the next step." Users who don't read the transcript at the same speed as the speaker won't be watching at the appropriate time.
- * A movie that includes extensive video sequences without words, such as battle scenes. Even if the transcript describes the action on the screen, users will find it difficult to know when to move on and will begin to experience the words in the transcript out of step with the visuals on the screen.
- * A movie that combines rapidly changing visual images with a lot of spoken words, such as when many people are talking at the same time. Users will find it difficult to follow along in real time if their attention is split between the transcript and the video.

Another drawback of using a transcript with video is its lack of proximity to the video presentation. In the example you saw earlier, the transcript opened in a separate window. More often, transcripts are printed,

and users must split their attention between the video on a screen and a paper transcript.

Using Transcripts as a Tool

Although transcripts alone do not provide equivalent access to multimedia applications, the creation of a good transcript is often an important and challenging step in the captioning process. Voice recognition technology is not advanced enough to accurately convert speech to text, so a member of the development team must manually create a transcript of the audio content that can then be broken down into synchronized captions. Having a transcript ready before recording speeds the process, but it is important to then adjust the transcript as necessary to reflect what was actually recorded. The requirements for writing good transcripts are similar to those for writing good captions.

Captioning Requirements

For captions to provide equivalent access, they must provide the same content as that provided in the audio track, including the identity of the speaker and all important sounds. They must be synchronized with the video, and they should allow the user to turn them on or off. Additional information about conventions used in captioning is available in the Resources section of this course.

Full, Identical Content

Captions, like transcripts, should provide content that is identical to what is in the audio track. This means that the words in the captions should match the words that were actually recorded — whether or not they match the original script.

Speaker Identified

When there is more than one speaker in a presentation, the speaker should be identified. The traditional approach to this requirement is to use the speaker's name or an abbreviation, followed by a colon, as in this example:

Another approach that may be used with more stylized or animated applications is the speech bubble, as in this example from Adobe:

Sound Effects

Captions must also include all important sound effects, such as doorbells or phones ringing, doors slamming, music playing, dogs barking. These are often set off in italics or brackets.

Synchronization

To deliver an equivalent experience, captions must be synchronized with the visual presentation. How you accomplish this depends on the method you use for captioning:

- * Embed superimposed open captions in the video itself during video post-production (recommended only for video also used in playback devices lacking the user control of Flash applications).
- * Use a shared Flash timeline for slideshows, with captions, soundtrack and visuals appearing in corresponding frames.
- * Use cue points — or markers — in the video file to trigger captions at specified points in the video, and associate a caption display component to the FLV file.
- * Create a timed xml file of the caption text, and associate the file with a caption display component that starts when the video begins playing.

User Control

Although open captions are permanently embedded in a video and cannot, therefore, be turned off, in most cases you must provide a control to allow users to turn captions on or off. Many developers use the universal CC button as a toggle. Flash has several built-in components that include a similar button.

By making a button like this readily available, you allow users with hearing impairments and visual learners to turn the captions on at the outset of a presentation. Auditory learners and others who may find the captions distracting can turn the captions off.

Approaches to Captioning

There are several Section 508-compliant approaches to captioning in Flash.

Making Captions a Permanent Part of a Video (Open Captioning)

You may sometimes need to work with video in which captions are already an integral, permanent part of the video itself. Although this type of captioning is Section 508-compliant and, if done correctly, is synchronized with the video, it is not very flexible. Captions created in this way are superimposed over the picture, so you need video editing software to edit them, and users are not able to control them.

Creating Captions as Text Objects on the Stage

You can create captions as text objects and place them directly on the Flash stage, using the timeline to indicate the frames in which they will appear. This approach is your best option for captioning non-video visuals with synchronized audio, such as timeline-controlled slideshows and animations or brief audio responses to a user action, such as during a quiz or simulation. It allows you to ensure precise timing with the visual content. It also provides flexibility in terms of positioning the caption boxes on the screen to indicate who is speaking, which the other approaches do not.

For captioning a video of more than a few seconds, it is tedious, time-consuming and labor intensive compared to the other methods available. It may be the only option, though, for non-video synchronized captions, and it allows display of a good deal more text at one time for animations or slide shows with audio only loosely synchronized with visual changes.

To make captions created this way easier to modify later, create a reusable movieclip in Flash to display the captions. Load the synchronized text for each keyframe from a separate, easily edited file that ActionScript can read while compiling or playing the movie. You can write such a file in ActionScript, specifying the captions as text values for elements in an array, and read the file in with an `#include` command while compiling. Alternatively, you can write it in XML (Extensible Markup Language) and load it into an XML object in ActionScript while playing the movie. Either one lets you easily edit the text displayed as the timeline reaches each keyframe.

Embedding Cue Points for Captions in a Video

You can embed captions in FLV video files using tools such as Manitu Group's Captionate or the Adobe Media Encoder software from Adobe. FLV is the standard Flash video file format used for most Flash videos. Like the first two approaches, this one allows precise timing of the captions for synchronization, but it provides less flexibility in where the captions display, as you must choose one display location for all the captions.

Cue point captions are embedded in the video file with their start times and optional end times, background color and background transparency. In your Flash file, the standard Flash video playback control must be associated with a caption display component, and the caption display component must be set to pull the captions from the FLV video files cue points. When this occurs and the user has the Closed Caption button activated in the standard Flash media playback component (known as an FLVPlayback Component), captions from embedded cue points will be displayed. The caption display component can be displayed anywhere in the Flash movie, including overlaid on the Flash video itself.

Saving a text copy of the cue points for later revisions or for adding the same captions to different versions of the same movie is a good idea and easily done from both Captionate and Adobe Media Encoder. Note: there is also a way to add cue points at runtime with ActionScript if software to embed the cue points directly in the FLV file is not available.

This, however, is not the typically recommended method.

Displaying Captions from a Timed Text File

You can also create video captions in a separate XML or text file to be read by a captioning component at runtime. Check the component for allowable formats. Most contain start times. Some contain end times or durations. Some allow formatting (italics, background color, transparency, etc.). Since the captions are maintained in a separate file, it is easy to modify them. This approach is similar to the Embedded Cue Points method, as the video playback component is associated with a caption display component, but the caption display component is associated with the timed text file of captions rather than cue points embedded in an FLV file.

A possible disadvantage of this method is that the captioning display relies on elapsed time since the video began playing, rather than on cue points inserted into specific frames in the video. This approach is usually reliable and flexible, but synchronization may suffer slightly if a video does not play at the expected speed.

Overview of One Approach to Captioning:

Embedding FLV Cue Points

To use the "cue point" approach to captioning Flash video, you need to be able to perform the following tasks:

- * Create captions for your video
- * Add caption cue points with these captions to your video file
- * Include a caption display component in your Flash application

Creating Captions

You can create the caption text for the cue points from an audio recording or from a written transcript. To work from an audio recording, you need speech recognition technology or the ability to start and stop the audio and type what you just heard. Although there are times when speech recognition is useful, the technology is not very sophisticated, and the resulting file is likely to require significant editing.

If you have a transcript, you can use any of a variety of tools available to "translate" or "export" the transcript into a text that can then be entered into the Cue Point creation tool to add the captions to the video. But you will still have to do significant work to break the transcript up into screen-size chunks appropriate for captions. The easiest way to perform this task is to use a tool that helps you edit and synchronize your file at the same time.

Adding Caption Cue Points

A cue point is an object in Flash that includes a cue point name, the time in milliseconds (HH:MM:SS.mmm) when the caption should display, a cue point type (event for captions) and some optional parameters:

- * HTML-formatted text for the caption
- * An end time for displaying the caption
- * A background color for the text field
- * A toggle to reverse the background color as needed
- * A word wrap toggle

There are tools available to help you embed cue points into your Flash files. Later in this lesson, you will be introduced to one of them, Captionate.

Including a Caption Display Component

To make all the pieces work together, you need to include in your Flash file a component that can read in and appropriately display captions from the cue points embedded in the video file. The primary player that supports cue points for captioning is the FLVPlayback control provided with Flash. It must be associated with the caption display component, such as the FLVPlaybackCaptioning component or CCforFlash. The FLVPlaybackCaptioning component must then be set to display captions based on Cue Points embedded in the Flash video file (FLV file).

Overview of Another Approach to Captioning: Using a Timed Text File of Captions

To use the "timed text file of captions" approach to captioning Flash video, you need to be able to perform the following tasks:

- * Create a timed text file of captions
- * Include a caption display component in your Flash application

Creating a Timed Text File of Captions

A timed text file of captions includes the time when the caption should display, the text for the caption, the duration of the caption and information about the font style and color. There are tools available to help you create timed text files of captions. Later in this lesson, you will be introduced to one of them, MAGpie.

Creating Captions

You can create captions from an audio recording or from a written transcript. To work from an audio recording, you need speech recognition technology or the ability to start and stop the audio and type what was just heard. Although there are times when speech recognition is useful, the technology is not very

sophisticated and the resulting file is likely to require significant editing.

If you have a transcript, you can use any of a variety of tools available to "translate" or "export" the transcript into a format readable as captions in Flash. But you will still have to do significant work to break the transcript up into screen-size chunks appropriate for captions. The easiest way to perform this task is to use a tool that helps you edit and synchronize your file at the same time.

Synchronizing Captions

There are several good tools, including a free one called MAGpie (see the Resources page for more information), to help you synchronize your captions with your video file. These tools provide an interface for you to watch your video and edit your caption file at the same time. You can use this interface to break your transcript into caption-size chunks and synchronize it to your visual content. You will be introduced to additional tools for editing and synchronizing captions later in this lesson.

Including a Captioning Component

To make all the pieces work together, you need to include in your Flash file a component that can read in and appropriately display captions from the caption file. There are many such components available. Some are free; you can also buy one or custom build your own. Different captioning components and associated media player support different types of timed text files of captions. Flash CS4 includes the FLVPlaybackCaptioning component, which supports the DFXP XML format, a WCAG Candidate Recommendation (see link below). Some of the organizations that provide other captioning tools and services also offer captioning components. The example you saw earlier in this lesson was built using CCforFlash, a free component available from the National Center for Accessible Media (NCAM). The Flash video playback component used to display the video must be associated with the caption display component such as the FLVPlaybackCaptioning component. The FLVPlaybackCaptioning component in this example must then be set to display captions based on a timed text file of captions in DFXP format.

*External Link Disclaimer:

All links marked with an asterisk (*) are External links. By clicking on these links, you will leave the Department of Veterans Affairs Website. VA does not endorse and is not responsible for the content of the linked website.

*DFXP XML format, a WCAG Candidate Recommendation (<http://www.w3.org/TR/ttaf1-dfxp/>)

Captioning Tools and Services

There are a number of tools available to help you perform the tasks required to create synchronized captions in Flash without learning the coding behind the tasks. Or, if you prefer, you can contract your captioning tasks out to a captioning service. Some of the tools and services that are available are introduced here. Links are provided in the Resources section of this course.

Captioning Tools

Captioning tools can help you:

1. Break transcripts into screen-size snippets appropriate for captions
2. Synchronize captions with visual content (let you view the video, type or edit the caption text and press a key to record the time code for the place in the video the caption should start or stop)
3. Embed cue point captions in Flash video files
4. Create timed text files of captions in the proper formats for use during playback

Captionate [2, 3]

Manitu Group's Captionate software helps you embed cue point captions in Flash video (FLV) files. You will need the transcript broken into a list of captions before you use it. It lets you play and navigate your video and type a key to record the time when the next caption in your list should display. It warns where your caption display rate exceeds the words per minute limit you have set. If you have multiple versions of a video (different sizes or frames per second), it automates adding the same captions to all versions. Because it creates a new version of your FLV file, it helps add video metadata and other types of cue points, too.

MAGpie [1, 2, 4]

Media Access Generator (MAGpie), a free tool from the Carl and Ruth Shapiro Family at WGBH National Center for Accessible Media (NCAM), helps you create your captions manually (with spell check) or break up your transcript into captions. It lets you play and navigate your video and type a key to add the time when the next caption in your list should display. It aids in formatting, and it can export properly formatted timed text files for several playback components. A step-by-step tutorial on the use of MAGpie to create captions

in Flash is available from The Georgia Tech Center for Assistive Technology and Environmental Access; you will find a link to this demonstration in the Resources section of this course.

Hi-Caption Studio [1, 2, 4]

Hi-Caption Studio from HiSoftware will help you convert a transcript to captions or captions to a transcript. Like the others, it also helps you synchronize captions to your video by letting you play or navigate the video to add time codes. It helps with formatting and exports several timed text files of captions for several players.

Subtitle-horse [2, 4]

Subtitle-horse is a free application for creating and editing timed text files of translation subtitles or open captions. It exports files in formats for several Flash video players. You can create configuration files to simplify working with foreign alphabets or special characters.

Swift Create [2, 4]

Softel's Swift Create is a professional captioning workstation solution used by many major caption service providers, especially for live captioning and editing of existing captions.

Caption Wrap [2, 4]

Caption Wrap is a post-production caption editing tool that allows you to edit, adjust the timing for and export timed text files of captions created by a speech recognition captioning system called Caption Mic.

Captioning Services

If you prefer to outsource your captioning, the following companies offer captioning services. Links to their websites are provided in the Resources section of this course:

- * Media Access Group at WGBH
- * Automatic Sync Technologies (AST), CaptionSync
- * Caption Colorado
- * National Captioning Institute

Providing Visual Alternatives to Sound Cues

By far, the bulk of the accessibility work for users who are deaf or have hearing impairments is accomplished by providing captioning for multimedia applications and transcripts for audio-only presentations. There are, however, other situations in which Flash applications, such as e-forms and non-multimedia training, convey information with sound.

Sound cues may indicate that an action has occurred or that action is required; they may be used to attract the user's attention or just to add interest to an application. These cues are helpful to users with visual impairments and are sometimes required to make an application accessible. (This requirement is covered in the Using Audio for Visual Content lesson of this course.)

For users who are deaf or have hearing impairments, you must ensure that sound is not the only way that information is conveyed. If, for example, you use a bell to indicate that a form has been completed successfully, or a buzzer to indicate that there is an error in a form field, you must make sure that you also provide a visual indicator of that information. When sounds are used in addition to the visual content, such as when a security alert appears on the screen with a pop sound, the application is accessible to all users.

Conceptual Knowledge Check

Check your knowledge of captioning. Use the Answer button to check your selection.

Top of Form

What is the most important aspect of captioning that cannot be captured in a transcript?

- A. Identification of the speaker
- B. Content that accurately reflects what was actually recorded
- C. Synchronization
- D. User control

Bottom of Form

How to Use the FLVPlaybackCaptioning Component

The VHA Section 508 checkpoints related to captioning are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Captioning

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(c)

At least one mode of operation and information retrieval that does not require user hearing shall be provided, or support for assistive technology used by people who are deaf or hard of hearing shall be provided.

c.1

Are equivalent alternatives provided for deaf individuals to access and interpret content? See 1194.22 (b).

* *

*

VHA Section 508 Checkpoints: Captioning

§1194.22 Web-based Internet Information and Applications

1194.22

Checkpoint

Yes

No

N/A

Comments

(b)

Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

b.1

Is synchronized captioning provided for audio content in videos?

* *

*

VHA Section 508 Checkpoints: Captioning

§1194.24 Video and Multimedia Products

1194.24

Checkpoint

Yes

No

N/A

Comments

(c)

All training and informational video and multimedia productions which support the agency's mission, regardless of format, that contain speech or other audio information necessary for the comprehension of the content, shall be open or closed captioned.

c.1

Is captioning for audio content provided and synchronized with audible content in the multimedia?

* *

*

c.2

Are captions appropriately timed and accurate for multimedia?

* *

*

c.3

Is the current speaker indicated when appropriate for multimedia?

* *

*

c.4

Are important sounds indicated in the captioning?

* *

*

c.5

Is a text transcript provided for audio only presentations?

* *

*

(e)

Display or presentation of alternate text presentation or audio descriptions shall be user-selectable unless permanent.

e.1

Can closed captioning in video or multimedia be turned on or off by the user?

* *

*

One way to add captions in Flash is to use the FLVPlaybackCaptioning component. To use this component, you must add it to the FLVPlayback component and set the properties for each instance of the component, using the Property

Inspector or the Component Inspector.

Add the Component

You can use the Components panel or ActionScript 3 to add the FLVPlaybackCaptioning component to the FLVPlayback component.

Using the Components Panel

To use the Components panel to add the FLVPlaybackCaptioning component to an FLVPlayback component:

1. Select Window > Components to open the Components panel.
2. Open the Video category in the Components panel.
3. Drag the FLVPlaybackCaptioning component from the Video folder to the stage that has the FLVPlayback component on it. If more than one, add its name to the captioning component.
4. Use a skin for the FLVPlayback component that includes a caption button, likeSkinUnderPlayStopSeekCaptionVol.swf. Alternatively, drag a captions text box and a CaptionButton component to the same stage.

Alternatively, Add the Component Using ActionScript 3

To use ActionScript instead of the Components panel to add the FLVPlaybackCaptioning component to an FLVPlayback component:

1. Select Window > Components to open the Components panel.
2. Open the Video category in the Components panel.
3. Drag the FLVPlayback and FLVPlaybackCaptioning components into the Flash file's library.
4. Copy the skin you want to use into the same folder as your Flash file.
5. Add the following code to the Actions panel on Frame 1 of the timeline, changing the highlighted code to indicate the drive on which you installed Flash and the location of the Skins folder for your installation.

```
import fl.video.*;
var my_FLVPlybk = new FLVPlayback();
my_FLVPlybk.x = 100;
my_FLVPlybk.y = 100;
addChild(my_FLVPlybk);
my_FLVPlybk.skin = "SkinUnderPlaySeekCaption.swf";
my_FLVPlybk.source = "http://www.helpexamples.com/flash/video/caption_video.flv";
var my_FLVPlybkcap = new FLVPlaybackCaptioning();
addChild (my_FLVPlybkcap);
my_FLVPlybkcap.source = "http://www.helpexamples.com/flash/video/caption_video.xml";
my_FLVPlybkcap.showCaptions = true;
```

Note: If you create an FLVPlayback instance with ActionScript, you must also dynamically assign to it a skin (a file that, like a stylesheet, determines how something visually appears) by setting the skin property with ActionScript. When you apply a skin with ActionScript, the skin file is not automatically published with the SWF file (two swf files exist, one for the Flash application and one for the skin). Copy the skin's SWF file and the application SWF file to your web server, or the skin SWF file will not be available when a user views the Flash.

Set Component Parameters

To use the Component Inspector to set parameters for the FLVPlaybackCaptioning component on the stage:

1. If the Property Inspector is not open, select Window > Properties, or press Ctrl+F3 (Windows) or Command+F3 (Mac) to open it.
2. Select the FLVPlaybackCaptioning component to see the properties for the component.
3. Select the Parameters tab in the Property Inspector window.
4. Set the FLVPlaybackCaptioning parameters, as appropriate and desired. Auto means the captioning control will check for TextFields, MovieClips and FLVPlayback controls that share its parent and choose one. Enter the instance name only if there is more than one candidate. The important parameter is source, the URL of your timed text file of captions.

* autoLayout

o Determines whether the component automatically moves and resizes the TextField used for captions

- o Default is true
- * captionTargetName
- o Identifies the instance name of the TextField (or MovieClip with a TextField) to display the captions
- o Default is auto
- * flvPlaybackName
- o Identifies the FLVPlayback instance name you want to caption
- o Default is auto
- * showCaptions
- o Determines if captions display initially or require user action
- o Default is true
- * simpleFormatting
- o Limits formatting instructions from the caption file.
- o Default is false
- * source
- o Identifies the source of the caption file
- o Enter the URL path directly into the Source cell

*External Link Disclaimer:

All links marked with an asterisk (*) are External links. By clicking on these links, you will leave the Department of Veterans Affairs Website. VA does not endorse and is not responsible for the content of the linked website.

*Adobe help page for FLVPlaybackCaptioning

parameters(http://help.adobe.com/en_US/AS3LCR/Flash_10.0/fl/video/FLVPlaybackCaptioning.html)

How to Provide Visual Alternatives to Sound Cues

The VHA Section 508 checkpoints related to providing visual alternatives to sound cues are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

One method of providing a visual cue as an alternative to an audio cue is a timeline-based solution that will provide a visual warning as well as an audio warning. In the example below, a "ding" is heard when typing continues after ten characters have been entered for the Employee ID Number. A visual alternative is needed for those who cannot hear the sound. There are two steps.

1. Create a visual warning, such as the tool tip displaying "Ten Character Limit" in the screen shot below.

2. Use ActionScript code to display it when the ten character limit has been reached in the text box txEmployeeID, the Employee ID Number.

```
3. import flash.events.TextEvent;
4. // watch for input changes
5. txEmployeeID.addEventListener(flash.events.TextEvent.
6. TEXT_INPUT, myListener);
7. // callback function called when input changes
8. // in the TextInput
9. function myListener(evt: flash.events.TextEvent): void
10. { if (txEmployeeID.length > 9)
11. {
12. // moves to frame 2, which plays a sound and
13. // displays a text box
14. gotoAndPlay(2);
15. }
16. }
17. // stop on frame 1
stop();
```

This screen shot shows how the timeline looks for this Flash movie. An audio clip appears at key frame 2 and plays through frame 5 on the audio layer. A text box appears at key frame 2 and goes through frame 5 on the Text layer.

Technical Knowledge Checks

Check your knowledge of the procedures for providing captions in Flash. Use the Answer button to check your selection.

Top of Form

The FLVPlayback component does not require any other component (custom or standard) to display closed captions as long as a timed text file of captions is associated with this component or embedded cue points are provided.

- A. True
- B. False

Bottom of Form

Top of Form

Which of the following is NOT a method to provide closed captions?

- A. Associate a timed text file of captions with a caption display component associated with a video playback component that provides a button to display closed captions.
- B. Embed cue points in a Flash video file and provide a caption display component associated with a video playback component that provides a button to display closed captions.
- C. Provide captions as call-out text boxes in appropriate frames throughout an animated presentation of visuals plus an audio track.
- D. Display a timed text file of captions to the user so he/she can determine which times certain text appears alongside of the Flash video.

Bottom of Form

Using Audio for Visual Information

Introduction

Flash lets you add action with videos, slide shows, animations and moving game elements. Add a soundtrack and you get an attention-grabbing multimedia presentation. Without the visuals, though, a lot of information gets lost. If your application conveys any information or functionality through images or movement, you must provide alternatives for users with visual impairments or blindness. One way to do this is with audio. Another is with text.

In this lesson, you will learn when Section 508 requires that you include audio describing the visuals, when it gives you a choice and how to make the best choice for the blind or visually impaired user. You will be introduced to several approaches to providing audio equivalents for Flash video, for timeline-driven visual presentations and for user interaction-driven visuals.

User Perspective: No Visuals

Select the Example link below to experience an instructional video from the perspective of a user who is blind. The presentation will open in a new window; if you are using a screen reader to take this training,

press Enter when you hear the prompt for the second Play button.

Do you think you could do what was demonstrated from just the audio portion of these instructions? Those who cannot see what is happening need to hear it described.

User Perspective: No Response

What if you do as you were instructed and nothing happens? If the only feedback is visual, then it is as if nothing happens for blind users. Select the example link below to experience this. The example will open in a new window. Tab to the Employee ID Number field, close your eyes, and type 12345123456. Open your eyes and delete the 6.

What did you miss when you could not see what you were typing? The experience of the form is not equivalent for sighted and blind users without some sort of audio alert to accompany the visual one. You can create an experience equivalent to a visual experience two ways, with audio or with text that will be read by a screen reader at the right time.

Choosing Audio or Text for Describing Visual Information

There are three things to keep in mind when choosing whether the non-visual equivalent for visual content should be created using audio or text. The first is Section 508, which requires audio in some cases. The second is the type of content; if the visuals show text that can be read by sighted users, a text equivalent is important. The third is consistency, minimizing the need to stop the screen reader for audio and start it for text.

Visuals with Synchronized Audio Require Audio Description

Section 508 requires audio description (also called video description) for any presentation of changing visuals and a synchronized soundtrack, including these:

- * Videos with a soundtrack
- * Recorded web conferences, including Adobe® Connect™ events
- * Animated characters that talk as they move around the Flash stage
- * Slide shows where the visuals and audio track are tightly synchronized
- * Animations and games with tightly synchronized movements and audio

Audio description explains who or what is shown in the visuals and what actions occur. It can be implicit, with a presenter describing what she shows or demonstrates or characters describing their own or each other's actions. Unless the audio is recorded before you begin the Flash project, this is often the easiest approach. Otherwise, audio description must be explicit, with an optional extra audio track or an alternate version of the video or animation. A narrator describes the important visual information during gaps in the original soundtrack.

Although audio description is required, you are not limited to it. When the video or images show important words (for example, a stress management scale) or punctuation (as in programming languages), it helps to provide a screen reader accessible copy of the text in addition to the required audio description. If you choose implicit audio description, the closed captions for the soundtrack offer a text option.

Choose Text or Audio Equivalents for Visuals with Loosely Synchronized Audio

Videos with a soundtrack always require audio description, but slide shows and animations may not. If the audio stops before each visual change or the images are not closely related to what is said, the visuals may be described with text, audio or both.

To choose, consider the user with a screen reader. The screen reader will read everything readable, very quickly, until the user stops it. Once he or she stops to listen to the Flash file's audio, all the text, buttons and icons a sighted user sees while listening are not available without sight until the audio stops and the user resumes using the screen reader's commands to review the screen. Reading while audio plays results in two conflicting audio streams and a challenge to listen to the correct one.

Read this lesson and the next to learn how to describe visual information with audio and text. Then choose a consistent approach throughout your Flash application. If it shows a single animation, the user will know from the web page what to expect and will be able to play and replay it easily. If it appears along with video clips, forms to fill out or controls for a complex application, a different approach for the same animation may be better.

Remember that including implicit audio description in your soundtrack makes it easy to display text that works as both captioning and a text equivalent. Explicit audio description, non-synchronized audio equivalents or lengthy text equivalents for complex animations or images may require extra controls. Choose Text Equivalents for Most Visuals without a Soundtrack

A screen reader can read text equivalents much faster than a narrator can provide the same information. For visuals without a soundtrack, text equivalents generally work much better than audio description.

The one exception is kiosk applications, those intended for public computers where users will not have access to their screen readers. With no assistive technology (AT) to speak text equivalents, you must add audio to all the controls and text to create a self-voicing application.

Audio Description Requirements for Visuals with Synchronized Audio

Whether you include audio description in your script, ask presenters to include implicit audio description in their presentations or add explicit audio description to an existing multimedia video, animation or slide show, you should do these four things:

- * Describe the setting, people and key objects. Include any important information revealed through labels, signs, insignia or clothing.
- * Describe all significant actions, including those important to understanding a story or conveying significant emotion.
- * Convey meaning and significant information. For example, if the trend of a chart is important for understanding, describe the chart, what its axes measure and the trend of the chart: upward, downward, etc. While a character is speaking, mention a wink, but ignore an eye blink.
- * Tailor the description to the purpose for including it. In one context, a group of Veterans standing in a medical center lobby may need no more description than this. In another, it may be important to name the Veterans or the medical center or to describe their age range, who is shaking whose hand or which one walked away.

For explicit audio description, two more requirements apply:

- * Add the description to the soundtrack heard by other users. Do not replace the soundtrack.
- * Synchronize the description to the soundtrack.

The VHA Section 508 checkpoints related to using audio to describe visual elements when a soundtrack is tightly synchronized with them are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Audio Description for Animation

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation

mode at the option of the user.

h.1

If animated objects exist, does the information conveyed by the animated object exist in another non-animated method?

* *

*

h.1.b

Is animation content sufficiently described in audio or text?

* *

*

h.2

Do alternatives to animation provide the equivalent functionality?

* *

*

VHA Section 508 Checkpoints: Audio Description for Web-Based Video and Animations

§1194.22 Web-based Internet Information and Applications

1194.22

Checkpoint

Yes

No

N/A

Comments

(a)

A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

a.3

Are text equivalents provided for background, animated and interactive content?

* *

*

a.3.g

Do video files have audio equivalents or full-text descriptions?

* *

*

(b)

Equivalent alternatives for any multimedia presentation shall be synchronized with the presentation.

b.2

Are clear and meaningful audio descriptions of visual content provided and synchronized in the video?

* *

*

VHA Section 508 Checkpoints: Audio Description for Video and Multimedia

§1194.24 Video and Multimedia Products

1194.24

Checkpoint

Yes

No

N/A

Comments

(d)

All training and informational video and multimedia productions which support the agency's mission, regardless of format, that contain visual information necessary for the comprehension of the content, shall be audio described.

d.1

Is audio description provided and synchronized with visual content in the multimedia?

* *

*

d.2

Is visual multimedia content sufficiently described in the audio/audio description portion of the multimedia?

* *

*

Note that there is no exemption for Flash files playing videos previously recorded without implicit audio description. The only exemption is for multimedia created for some purpose other than supporting the agency's mission, such as a video of an employee's birthday party.

Approaches to Audio Description for Visuals with Synchronized Audio

Approach One: Implicit Audio Description

The easiest way to add required audio description is often with implicit audio description. If a speaker, narrator or character adequately describes the visual content in your main soundtrack, nothing extra is required. Users even get the bonus of a text equivalent in the captions.

Examples of implicit description:

- * "Click on the radio button for your size choice" instead of "click on one of these radio buttons."
- * "The probe must enter at a 45 degree angle" instead of "this is the right way to insert the probe."
- * "Hello, I am Director Jane Doe and..." instead of superimposing a label reading "Jane Doe, Director" as she begins speaking.

Approach Two: Explicit Audio Description Added to a Video

Edit the soundtrack to include a narrator describing the visuals during natural pauses in the audio stream.

Offer this described version in place of the original, or add a control to offer it as a user option.

Use any video player with closed captions, such as ccForFlash or the FLVPlaybackCaptioning control. If you offer both versions, captioning is optional but recommended for the audio-described version.

Approach Three: Explicit Audio Description with an Optional Audio Track

Use a Flash playback component that supports captions and an optional second audio track (JW Player, a modified FLVPlayback control or custom-built) to control a second audio track or a series of audio clips.

Three possible methods are:

- * Provide a secondary audio track that can be played instead of the standard audio track.
- * Play additional audio clips for visuals during breaks in speech in the main audio, using FLV (Flash Video) event cue points for a video, the timeline for an animation or slide show or a timer for either of these.
- * Pause or stop the presentation where needed to create time for sufficient description, and play an audio clip. This is sometimes referred to as "extended audio description." Most players do not support this method. Creating a custom approach for this method could cause timing issues. This lesson does not cover this approach.

Audio Equivalent Requirements for Purely Visual Elements

When a soundtrack is not included or not tightly synchronized with the visual elements or actions, the descriptions of the visual elements need not be synchronized with a soundtrack. Examples include:

- * Silent videos, slide shows or animations
- * Animations with narration that describes the action before it begins or after it ends and not while it takes place
- * Slide shows with a musical soundtrack
- * Animations that display changing formatted text to deliver a message while attention-holding music plays
- * Slide shows that tell one story (e.g., your employee benefits) and show a related one (employees enjoying the benefits of their health care, educational benefits and retirement savings)

Audio is not Required

You have a choice of audio or text equivalents for such elements. You may also provide both. See the Choosing Audio or Text for Describing Visual Information topic for help choosing. If a Flash application includes several of these elements, choose the same approach for all of them.

For Animation, Audio Must Describe the Animation Sufficiently

The audio equivalent version, if you include one, must provide the same information and functionality as the animated version.

For Silent Video, Audio Must Describe the Video Sufficiently

If you use an audio equivalent, it must be as complete as audio description for a video with a soundtrack. See the Audio Description Requirements for Visuals with Synchronized Audio topic.

Provide User Controls

If you offer audio equivalents for these elements, user control of the audio is required. User control of a choice between audio and text equivalents is highly desirable, but audio may be designed not to play automatically based on detection of screen reader use.

The VHA Section 508 checkpoints related to using audio to describe visual elements when a soundtrack is not tightly synchronized with the visual elements are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Audio Equivalents for Animation

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

h.1

If animated objects exist, does the information conveyed by the animated object exist in another non-animated method?

* *

*

h.1.b

Is animation content sufficiently described in audio or text?

* *

*

h.2

Do alternatives to animation provide the equivalent functionality?

* *

*

VHA Section 508 Checkpoints: Audio Equivalents for Animations and Silent Videos
§1194.22 Web-based Internet Information and Applications

Checkpoint

Yes

No

N/A

Comments

(a)

A text equivalent for every non-text element shall be provided (e.g., via "alt", "longdesc", or in element content).

a.3

Are text equivalents provided for background, animated and interactive content?

* *

*

a.3.g

Do video files have audio equivalents or full-text descriptions?

* *

*

(n)

When electronic forms are designed to be completed on-line, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

n.8

Are there audio and visual alternatives provided for security measures like CAPTCHA?

* *

*

VHA Section 508 Checkpoints: : Audio Description for Silent Video

§1194.24 Video and Multimedia Products

1194.24

Checkpoint24

Yes

No

N/A

Comments

(d)

All training and informational video and multimedia productions which support the agency's mission, regardless of format, that contain visual information necessary for the comprehension of the content, shall be audio described.

d.3

Is a text transcript or audio track of video-only presentations provided?

* *

*

Approaches to Audio Equivalents for Purely Visual Elements

Approach One: Implicit Audio Description

When visual elements have no soundtrack or when the soundtrack can be separated from the visuals without affecting the meaning of the presentation, the approaches available to you are simpler than those for tightly synchronized multimedia.

Approach One: Timeline Audio

Attach an audio file describing an animation or slide to the same frames in the timeline in Flash that display the images. This will make the audio available to play when the animation plays or the slide is displayed.

Add user controls to enable or disable playing all such audios in the file and to start and rewind each animation or slide display. For a longer animation, step-through controls may be needed.

If the timeline already includes audio, be careful to fit the description audio into quiet spots in the primary audio.

This works well when the Flash file content is timeline-driven. When it is driven by user selections or actions, you will need the next approach.

Approach Two: Event Handlers

Event handlers tell Flash what to do when an element receives focus or when the user activates it (clicks on it or presses the Enter key while it has the focus). One of the things it can do is play an audio file with an audio description.

A simple example of this is a "self-voicing" application used in a kiosk, where a screen reader is not available. When any button, volume control or animation receives focus, it plays a short audio file announcing what it is and what to do next. Text boxes would play an audio reading the text. And when activated, a button might play audio description of whatever it activates.

When a screen reader will be available, it can handle identifying the controls, but the same event handlers can launch the audio descriptions along with whatever they describe.

If the descriptions are brief, you may not need other audio controls. Just start the audio when the control is activated and stop it when it loses focus.

Conceptual Knowledge Checks

Check your knowledge of audio description. Use the Answer button to check your selection.

Top of Form

Which of these requires audio description when it goes on the intranet?

- A. An animation with no soundtrack to explain it
- B. An audio clip and photo of a Vet talking about returning home
- C. A slide show of rashes with an audio track naming each one
- D. A video clip of students in an emergency procedures class

Bottom of Form

Top of Form

If a video has sufficient pauses in the soundtrack to add audio description but you cannot edit its soundtrack, which method is an option?

- A. Add a text transcript
- B. Play audio clips during the pauses
- C. Replace the video with something accessible
- D. Add captions to the video

Bottom of Form

Top of Form

In what situation should self-voicing be used?

- A. The controls cannot be made accessible by exposing text alternatives
- B. The developer does not have a screen reader installed and thus is not sure if the controls can be made accessible with textual alternatives
- C. The developer does not know how to make the controls accessible with text alternatives
- D. The application is all buttons, like a calculator

Bottom of Form

How to Add Explicit Audio Description to a Video

The VHA Section 508 checkpoints related to audio description for a video are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

Implicit audio description requires scripting skills. Explicit audio description for a video requires a component that supports it, such as the JW Player from Longtail Video, or some ActionScript programming.

Earlier in this lesson, you learned that one way to handle explicit audio description is with cue points for your Flash video (FLV). Cue points are silent, unseen indicators added to the video file. There are five steps to add audio description with FLV cue points:

- * Record the audio description sound files.
- * Add cue points to the video.
- * Add an audio description toggle button and a listener for changes in its state.
- * Create a listener in ActionScript to listen for cue points during video playback.
- * Launch the matching audio (or not, if toggled off) as the video reaches each cue point.

Record the Audio Description Sound Files

Script sufficient description of the visuals to gain the full benefit of the video. Edit the script to fit it into the length of the pauses in the video's soundtrack. Test it by reading it to someone who has not seen the video as they listen to the soundtrack with eyes closed. Edit as needed.

Record each segment as an mp3 file, using a narrator with a clear voice. To avoid extraneous noises in the recording when played in the Flash Player, record at 11, 22 or 44 kHz.

Add Event-Type Cue Points to the Video

Start with a video already converted to FLV format. Add cue points. One way is to add event-type cue points to the FLV file itself. The other is to add ActionScript-type cue points at runtime with ActionScript. Special software is needed to add them to the video. Manitu Group's Captionate Captionate, mentioned earlier for creating captions, will also embed cue points in FLV files. The Adobe Media Encoder (in Edit > Export Settings...) does, too. For other tools, do an internet search for FLV cue point editor. Cue points added this way travel with the video, available for use in every Flash file playing the video.

The sample script below uses the other method, in which ActionScript adds temporary cue points to the FLVPlayback component every time the Flash file displays.

Add an Audio Description Toggle Button and a Listener for Changes in Its State

The FLVPlayback component has no button to toggle audio description on and off. You must add a button and, in ActionScript, add a listener for changes in its on/off state.

Create a Listener in ActionScript to Listen for Cue Points during Video Playback

Whether you add cue points in the video file or with ActionScript, include a listener in ActionScript to know when the video playback reaches each one.

Launch the Matching Audio as the Video Reaches Each Cue Point

When the listener hears a cue point, check the status of the audio description toggle button. If on, use the cue point's name to choose and play an mp3 audio file describing the visual portion of the video and any recent actions.

Step by step

1. Create a package container to reside in an .as file

a. Import all needed classes.

b. Define a new class (e.g., VHAAudioCues) to control audio cues in a particular instance of the FLVPlayback control.

c. Add a constructor to

* Set cue points in the player (e.g., Clip1 at 8.00 seconds and Clip2 at 22.20 seconds)

* Enable the audio description (making enabled the default state)

* Add a listener for the audio description toggle button (the button named enable_VHAEmpAD_btn in the sample code below) to change its label and launch a function to enable or disable audio description.

d. Add a function (e.g., enable) to add a listener for cue points (e.g., VHAEmpCuePoint_listener) while audio description is enabled, using a Metadata event listener.

e. Add a function (e.g., disable) to remove the cue points event listener.

f. Add a function to the cue points listener to play the correct sound when a cue point is reached, using a switch statement and a case for each cue point name.

2. In the Flash application, create an instance of the custom class (e.g., VHAAudioCues), passing in the FLVPlayback control instance to use.

Sample Code for the Audio Description Class Package (cues.as)

```
package {
import fl.video. *;
import flash.events. *;
import flash.media.Sound;
import flash.media.SoundChannel;
import flash.net.URLRequest;
import flash.display.Sprite;

public class VHAAudioCues extends Sprite {
private var channel: SoundChannel = new SoundChannel;
private var _enabled: Boolean = true;
private var _ADOnOffBtn: Button;
private var Dingsnd: Sound = new Sound();

public function VHAAudioCues(VAEmpPlayer: FLVPlayback) {

VAEmpPlayer.addASCuePoint(8.00, "Clip1");
VAEmpPlayer.addASCuePoint(22.20, "Clip2");

enable();
```

```

enable_VHAEmpAD_btn.addEventListener(MouseEvent.CLICK,
VHAEmpBtnClk);
}

private function VHAEmpBtnClk(e) {
_enabled = !_enabled;
if (!_enabled) {
disable();
enable_VHAEmpAD_btn.label = "Enable Audio Descriptions";
} else {
enable();
enable_VHAEmpAD_btn.label = "Disable Audio Descriptions";
}
}

public function enable() {

VAEmpPlayer.addEventListener(MetadataEvent.CUE_POINT,
VHAEmpCuePoint_listener);
}

public function disable() {

VAEmpPlayer.removeEventListener(MetadataEvent.CUE_POINT,
VHAEmpCuePoint_listener);
}

private function VHAEmpCuePoint_listener(eventObject:
MetadataEvent): void {
Secondsnd = new Sound();

switch (eventObject.info.name) {
case "Clip1":
Secondsnd.load(new URLRequest("map_description.mp3"));
channel = Dingsnd.play();

break;
case "Clip2":
Secondsnd.load(new URLRequest("chart_description.mp3"));
channel = Dingsnd.play();
break;
}
}
} Sample Code for Using the Package in a Flash Application (MyApplication fla)
include "cues.as";
var myVHAAudioCues: VHAAudioCues;
myVHAAudioCues = new VHAAudioCues(my_FLVPlybk);

```

How to Add Audio Equivalents to the Timeline

The VHA Section 508 checkpoints related to audio describing an animation are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Audio Equivalents for Animation

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

h.1

If animated objects exist, does the information conveyed by the animated object exist in another non-animated method?

* *

*

h.1.b

Is animation content sufficiently described in audio or text?

* *

*

h.2

Do alternatives to animation provide the equivalent functionality?

* *

*

Animations and slide shows often use a timeline to time visual displays and movements. Audio equivalents can be added to play as the animations or slide shows play. If the slide show or animation has its own soundtrack, you can see both waveforms in the timeline to synchronize them. If it is difficult to synchronize them, consider adding user controls, covered in the next topic.

Sample Flash Application with Slide Show

Here is a screen capture of a loosely synchronized slide show in Flash. A Text Off/Text On toggle button controls whether text displays below the slides. An Audio Off/Audio On toggle button controls whether audio plays or not. The audio will be either a soundtrack that matches the displayed text or an audio-described version of this soundtrack, depending on a selection made on a previous screen.

This gives the user many options to help with cognitive, hearing and vision impairments when the Play button is pressed:

- * Text + audio + slide
- * Text + slide
- * Audio + slide
- * Text + audio + slide + audio describing the slide
- * Text + slide + audio describing the slide
- * Audio + slide + audio describing the slide

Add Audio to a Timeline

1. Create an audio on/off button that toggles audio on and off. Open the Library panel, locate and right-click the button and choose properties. In the Linkage section, check the "Export for ActionScript" and "Export in first frame" checkboxes. Use ActionScript to change the icon displayed on this button to indicate its on or off state.

2. Import your sound file that includes implicit audio description into your movie library.

3. Open your Library panel, locate and right-click on your sound file and select properties. In the identifier text box enter a name (e.g., Mix1) and check the "Export for ActionScript" and "Export in first frame" checkboxes under Linkage.

4. Click the first key frame and put the following ActionScript into the actions layer of the actions panel:

```
5. //create an instance of the audio object
6. var m1:Mix1 = new Mix1();
7. // set state of audio button to off, don't play audio
8. var audioState:int = 0;
9. // display audio off icon/text not shown
10. // play the animation and audio automatically
11. // if a screen reader is not detected
12. if (!Accessibility.active)
13. {
14. m1.play(0,1);
15. audioState = 1;
16. // display audio on icon/text not shown
17. play();
18. }
19. else
20. {
21. stop();
```

} 22. Make your audio on/off button functional using ActionScript; place the following code in the actions layer of the first frame.

```
23. btnAudioOnOff.addEventListener(MouseEvent.CLICK,
24. toggleAudio)
25. function toggleAudio(e:MouseEvent): void
26. {
27. if (m1.isPlaying)
28. {
29. m1.stop();
30. audioState = 0;
31. // display audio off icon/text not shown
32. }
33. else
34. {
35. m1.play(0,1);
```

```

36. audioState = 1;
37. // display audio on icon/text not shown
38. }
} 39. In the 1st frame actions layer, put the following ActionScript code into the Actions panel.
40. btnPlay.addEventListener(MouseEvent.CLICK,
41. playAnimation);
42. function playAnimation(): void
43. {
44. //If audio button is on and audio is not currently
45. // playing, play audio when animate on is played
46. if (audioState == 1 && !m1.isPlaying)
47. {
48. m1.play(0,1); //play the audio
49. }
50. play(); // play the animation
} 51. In the 1st frame put the following ActionScript code into the Actions panel of the actions layer.
52. btnStop.addEventListener(MouseEvent.CLICK, stopAnimation);
53. function stopAnimation(): void
54. {
55. if (m1.isPlaying)
56. {
57. m1.stop(); // stop the audio
58. }
59. stop(); // stop the animation
}

```

How to Add Audio Equivalents to a User Interaction

Sometimes, the display or movement of a visual object does not begin or end according to the timeline. It may result from a user interaction, like these:

- * Typing in a text field
- * Activating a Next button
- * Selecting a RadioButton
- * Activating a button that reaches a winning score
- * Clicking or pressing Enter while the focus is on a visual object

The method for this is the same one shown for adding a visual cue to an audio cue that typing in a text field had exceeded a character count.

Here is the screen shot. When more than ten characters are entered into the Employee ID field, a yellow bubble appears with the warning "Ten Character Limit." At the same time, because screen reader users need to keep focus in the text box and cannot read a text equivalent, an audio sound effect like a ding is perfect.

Use ActionScript code to display the visual cue and play the audio cue when the ten-character limit has been reached in the text box txEmployeeID, the Employee ID Number. The audio portion is highlighted.

```

import flash.events.TextEvent;
import flash.net.URLRequest;
import flash.media.Sound;
import flash.media.SoundChannel;

```

```

// hide help bubble
mcBubble.visible = false;

```

```

txEmployeeID.addEventListener(flash.events.TextEvent.TEXT_INPUT,

```

```
myListener);
```

```
function myListener(evt: flash.events.TextEvent): void
{ if (txEmployeeID.length > 9)
{
// display help bubble
mcBubble.visible = true;
// prep and play sound
var channel: SoundChannel = new SoundChannel();
var req: URLRequest = new URLRequest("ding.mp3");
var snd: Sound = new Sound();
snd.load(req);
channel = snd.play();
}
else
{
// hide help bubble
mcBubble.visible = false;
}
```

} Use a similar approach to add other sound effects on successes and errors or to announce a score. Keep the audio brief and easily repeatable, because it will be competing with a screen reader.

This user interaction-driven approach also gives you a way to create extended audio description where you need it. Break up the animation, slide show or video at logical break points and add step-through buttons. When the audio-described option is off, hide the buttons, and play the segments back-to-back. When it is on, play a segment, wait for the user to activate the step-through button, play the audio description segment, then continue with the next segment of the multimedia presentation.

Technical Knowledge Checks

Check your knowledge of the procedures for adding audio description with cue points, putting audio equivalents in the timeline and adding audio equivalents to user interactions. Use the Answer button to check your selection.

Top of Form

When must an audio cue be used in addition to a visual cue for a user interaction?

- A. When color is used to convey meaning
- B. When it indicates a changed status based on user input
- C. Never, because audio cues interfere with screen readers
- D. Only when captions can be provided for the audio event

Bottom of Form

Top of Form

As you add cue points to a Flash video (FLV), you must provide both a time and a file name for the audio description file for each one.

- A. True
- B. False

Bottom of Form

Providing Text Equivalents

Introduction

All still and moving images in a Flash file require an equivalent version for those who cannot view them. In the last lesson, you learned about one type of equivalent, the audio recording. In this lesson, you will learn about the other, text equivalents.

Flash is a visual medium. One way to create powerful text equivalents is to imagine going through your Flash file on the phone with someone when his or her computer monitor suddenly stops working. How would you convey the visual excitement and the depth of information in the visual components over the phone? There are places in Flash for this description to be available to everyone who needs it.

User Perspective: Missing Information

Select the Example link below to experience a Flash application with a common problem: critical information conveyed only to those who can see it. The presentation will open in a new window.

Could you see the stars? Could you decide what to do without them? Users depending on screen readers run into this all the time. They need a text or audio equivalent for icons that convey information.

Choose Text or Audio for Describing Visual Information

Many blind and visually impaired users need another way to get information contained in the rich visual environment of Flash. There are two ways to provide equivalents for visual information: text and audio. The Using Audio for Visual Information lesson tells how to use Audio description and other audio equivalents. An equivalent is always required for visual content, whether form fields, bouncing balls, printer icons, checkboxes, photographs, charts or even tables. In a few instances, the equivalent is required to be audio. Everywhere else, choose what works best for all your audiences: screen reader users, magnification users, those with color blindness and those who may benefit from receiving the same information two or three ways if the combination is not an overwhelming or distracting one.

When to Choose Text Equivalents

- * Choose text if typing is required.
- * Choose text for most still images and icons.
- * Choose text for most charts, graphs and other data-rich images.
- * Choose text for most user interface controls, but add sound effects where appropriate to draw attention to an error or a success while typing or activating controls.
- * Choose text for content intended solely for screen reader users; screen readers read faster than narrators.

When to Use Audio

- * Choose audio where required: for describing a video with a soundtrack or any other presentation where a soundtrack is closely timed with the display or movement of images. Consider adding text, too, for any textual information or numbers contained in the images.
- * Choose audio, even for controls, to avoid requiring the screen reader user to switch modes repeatedly in an application that includes many audio segments. Avoid requiring typing in such applications.
- * Choose audio for visual content, like medical or forensic photographs, likely to require a lot of close scrutiny for those with visual impairments, and consider pairing it with text for the screen reader user.
- * Choose audio for self-voicing applications used in places like kiosks, where a user's AT is not available, and minimize or eliminate the need to type responses.

Two Types of Text Equivalents to Choose From

There are two types of text equivalents: those available in the text readable by all and those available to assistive technology (AT) through MSA. These two options are similar to the choice of implicit or explicit

audio description: if you make the descriptions available to everyone, you do not need to make it available another way for Section 508.

With the exception of text equivalents for user interface controls (see the lesson on Providing Accessible User Controls), you have the option to describe any photograph, chart, icon or animation in nearby text seen by everyone, rather than using the techniques described in this lesson for making it available to screen readers.

Identify Content Requiring Equivalents

The lesson on Using Audio for Visual Information focuses on equivalents for Flash applications with soundtracks and/or moving visuals. However, all visuals except those that are purely decorative or intended only to draw attention require some sort of equivalent, whether text or audio.

Pictures

Photographs, graphics, images, symbols and icons all require equivalents.

Things that Move

Equivalents are also required for movies, animations, progress bars and movement designed to attract user attention to important information or to convey information.

Data Tables

Tables organize information. Even though the text can be read, the organization is lost to a screen reader.

User Interface Controls

Checkboxes, buttons, sliders and other user interface controls need equivalents, too. Standard Flash components, when made accessible, convey what sort of control they are, but not what they do. You must add this. For custom controls, both are required. The Providing Accessible User Interface Controls lesson provides more information than this lesson can.

Text that Conveys Additional Information through Font Size or Color

A text cloud showing frequency of word use through font size requires a text equivalent, an HTML table with the same data, for example. When the color of the text conveys more than the text itself does, you can provide a text equivalent or rewrite the text. See the lesson on Using Color for more information.

Text that is Not Actually Text

For AT dealing with Flash, text means:

- * Standard fonts (not those where you type a Q and see an airplane or a checkbox or some other symbol)
- * In dynamic text fields (not static text fields)
- * Or in Accessible Name or Description fields

All other things that look like text require text (or audio) equivalents, unless the text is irrelevant or a duplication of accessible text:

- * Images of letters or words
- * Scanned documents
- * Logos
- * Screen captures that contain or look like text
- * Video that shows text
- * Scrolling text
- * Fading text
- * Animated text
- * Bullet points or indentations in text conveying structure or hierarchy
- * Some window and dialog box titles (or use text, as shown here, instead of graphics)

Text in Flash's static text fields is readable with AT, but you cannot control its place in the reading order. Controlling reading order is necessary and is covered in the Controlling Reading Order and Tab Order lesson. Anything in a static text field can and should be displayed in a dynamic text field.

Create Good Text Equivalents

The VHA Section 508 checkpoints that define good text equivalents are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Text Equivalents

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(d)

Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

d.4

Is hierarchy indicated for components including text?

* *

*

(f)

Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.

f.1

Is all text presented in the application readable by assistive technologies?

* *

*

f.4

Do window and dialog titles properly render to text?

* *

*

(h)

When animation is displayed, the information shall be displayable in at least one non-animated presentation mode at the option of the user.

h.1

If animated objects exist, does the information conveyed by the animated object exist in another non-animated method?

* *

*

h.1.b

Is animation content sufficiently described in audio or text?

* *

*

h.2

Do alternatives to animation provide the equivalent functionality?

* *

*

(I)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

I.13

When a timed response is required, is a warning provided to the user that is available to assistive technology?

* *

*

I.14

Is the user given sufficient time to interact and/or request more time?

* *

*

VHA Section 508 Checkpoints: Tables, Hierarchies, and State Changes

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.2

Are tables coded in such a way that they make sense to a user of screen reading technology?

* *

*

a.2.a

Are nested data tables avoided?

* *

*

a.2.b

Do layout tables present information in the intended order to screen readers?

* *

*

a.2.c

In data tables, are means other than empty rows or columns used for presentation purposes?

* *

*

a.2.d

Do layout tables avoid structural markup (e.g., <th>)?

* *
*
a.3
* *
*

Are means provided to accurately convey heading levels, list structures, and other visual indicators of hierarchy, order and/or frequency?

a.3.a
Are implicit headings avoided?

* *
*

a.3.b
Are headings provided for Glossary and Index navigation?

* *
*

a.3.c
Is the use of characters with non-list meanings avoided for list bullets, (e.g., ©, o)?

* *
*

a.3.d
Are lists and nested lists structured properly?

* *
*

a.3.e
Is a validly structured and linked table of contents provided for documents 20 pages or longer?

* *
*

a.3.f
Are means (such as markup tags and/or textual indicators) besides shape and location used to indicate popularity or frequency, as with elements like data clouds?

* *
*

a.20
When an element's state changes, does related alternative text update accordingly?

* *
*

These can be summarized as follows.

Convey Meaning, Not Description

The same photograph might be described as "Captain Mary Stillwell, retired, salutes General Jose S. Ortiz in front of the West Orange, NJ VAMC" or as "female Vet with guide dog salutes a general as VA doctors and nurses look on." Which one is better depends entirely on why it appears in the Flash application. As mentioned earlier, the best way to decide what is relevant is to imagine describing it over the phone.

The same is true for moving images. Sometimes, the movement is important. Other times, the path or the destination is. Instead of describing details, convey the story they tell.

A text equivalent for a chart or graph, to convey its meaning, should report what is being measured, over what time period or other data range and, if there is a definite trend, what it is. If data points can be read from the chart or graph and there are too many for a simple description, include a link to an HTML data table.

If an image conveys no meaning, disable accessibility for it.

The best person to describe the meaning of a photo or animation is usually not the Flash developer, but the person who chose it.

When describing user interface controls, the color is not important, but what the color signifies may be. The image on a button, whether a printer or a right-pointing triangle, is not important; what it does (print, play) is important.

Components from Flash's Component Library, as well as buttons, movie clips, text input and text area fields and dynamic text fields, will automatically identify what type of control they are. Thus, there is no need to use the words "button" or "enter text" or "input" when describing them. Even movie clips acting as a button will announce themselves as buttons. So, for a button with an image of a printer on it, the equivalent text might read just "Print," conveying meaning instead of description.

Group Related Items and Make the Child Objects Inaccessible

Movie clips are perfect for grouping objects or text fields to provide better text equivalents. "Tic-tac-toe game" beats "O, vertical line, vertical line, X, horizontal line, X, horizontal line, O."

Put the component parts in a movie clip, and add a single accessible name to the movie clip. Be sure to make all the child objects (the X's, O's and lines) inaccessible, so screen readers announce only the overall description.

Convey Hierarchy

Hierarchy is often conveyed visually, as with bubbles and lines in an organizational chart, multiple indents in an outline or a pyramid shape set behind stacked text. Equivalence requires you to convey the important relationships between items.

Grouping is one good way to convey relationships along with the individual bits of information. You might group the pyramid shape with all of its text items and then convey the meaning in a single text equivalent. For an organizational chart, you might group all seven people who report up to a single person and describe their reporting relationships.

Take note that Flash allows list item tags () in HTML text, but the hierarchical nature of text in these lists is not conveyed to MSAA or the screen reader. If you use them, remember to add a text equivalent that conveys the relationships, too.

Include All Labels Associated with an Interactive Element

When creating a text equivalent for a text input field or a radio button, check around for all of its labels. Include them all in the accessible name for the control.

- * The field's associated label (e.g., "Name (required):" or "IV Infusion Rate:" for a text input, "Male" and "Female" for two radio buttons)
- * The field's group label (e.g., "Child's gender: Female" and "Child's gender: Male")
- * The form section's label (e.g., "Child – Name (required):" and "Parent or Guardian – Name (required):")
- * Any units of measure label following the control (e.g., "IV Infusion Rate in milliliters:" or "IV Infusion Rate (ml):")
- * Any tips or explanations following the control (e.g., "Date (mm/dd/yyyy format):" or "IV Infusion Rate in milliliters – round to the nearest whole number:")
- * Any help link following the control (e.g., "Parent or Guardian – Name (required) – Help link follows"

Remember to make all nearby labels inaccessible, to avoid repetition. Help links and section headings should remain accessible.

Group Moving or Changing Items or Add User Controls

Sighted users have no problem noticing a marquee is changing slides or text clips every few seconds. Screen readers ignore these changes or start reading from the top again when each change occurs. One way to make all the content accessible is to put a single description of all the items in the movie clip's Accessible Name and make the moving content inaccessible. Another is to add Pause and Play controls. Group moving or changing objects to give them just one text equivalent, or place them under user control. For example, these tumbling dice might come to a halt, and then update their movie clip's Accessible Name to "You rolled a 17" or "You rolled 4, 1, 3, 6, 3."

Remember that an Accessible Name is not available to a low vision user unless using a screen reader. For larger or more complex images that low vision users cannot view in the time between changes, user controls may be required.

Update Text Equivalents When Visual Content Changes

The visual content of a single object may change several times, as it might if you choose user controls for a marquee or slideshow displaying a series of images. The text equivalent must change whenever the visual content does.

Please note you cannot do this in the timeline, by changing the Accessible Name in the Accessibility Panel. You can set the initial text equivalent there, but you must make all subsequent changes with ActionScript.

Updates are very common in user controls, too, as when a Play button becomes a Pause button when clicked or a radio button is selected. When you create a custom control or build a toggle button, you need to add this information about the current state of the control to its Accessible Name. See the lesson on Providing Accessible User Controls for more information.

[Link to an HTML Version of Any Data Table](#)

Unless you can describe the entire table in a few words (e.g., "Team A scored 4 points in round 1 and 5 points in round 2, and Team B scored 3 points, then 6 points"), link to an HTML text equivalent outside the Flash application.

For tables in which the top or side headings are important to understanding what appears in a cell, disable accessibility for the table and add a link to a separate HTML version of the table outside the Flash file. For guidance, see the two lessons on Data Tables in our course on Testing HTML for 508 Compliance.

While Flash does contain a DataGrid control, this control does not allow for the flexibility and ease of reading down columns or across rows that an HTML based data table allows for users of screen readers. The DataGrid control is interactive, designed for viewing and selecting rows and cells. When tables are for review only, with no user interaction, as in most training applications, use a read-only data table in HTML.

Conceptual Knowledge Check

Check your knowledge of text equivalents. Use the Answer button to check your selection.

Top of Form

A chart shows dates of new patient room cleaning techniques and decreasing recurrence of illness outcomes over three years. "Chart showing the implementation of new patient room cleaning techniques starting in 2005" is a good Accessible Name for this chart.

- A. True
- B. False

Bottom of Form

Top of Form

"Left" and "Right" standard radio buttons appear in a "Dominant Hand" panel of a form. Which is the best Accessible Name for the "Left" radio button?

- A. Left
- B. Dominant hand left
- C. Dominant hand left radio button
- D. Dominant hand left radio button selected

Bottom of Form

How to Provide Text Equivalents for Graphic Symbols

Flash uses three types of symbols: Graphics, Movie Clips and Buttons. You cannot attach a text equivalent to a Graphic symbol.

Why not?

- * Graphic symbols cannot be given instance names.
- * The Accessibility Properties object must be associated with an instance name.
- * The Accessibility Panel for a Graphic symbol reads "Current selection cannot have accessibility applied to it."

This gives you two choices for providing a text equivalent:

1. Describe the Graphic symbol in a Dynamic Text field near it.

2. Convert the Graphic symbol into a Movie Clip, so you can add Accessibility Properties.

Here are instructions for the second option.

Convert a Graphic Symbol to a Movie Clip

1. Create a graphic on the Flash stage.
2. Right-click (Windows) or Ctrl-click (Mac) on the graphic to open the context menu.
3. Select Convert to Symbol.
4. In the Convert to Symbol dialog box that appears, give the object a name (not necessary but good practice) and select Movie Clip in the Type combo box.

Add Accessibility Properties

When you convert a Graphic symbol to a Movie Clip, it looks the same and behaves the same. Now, though, you can add an Accessibility Properties object, which lets you:

1. Add an Accessible Name, a text equivalent for screen reader users.
2. Include it in the tab order, so it can be included in the screen reader's reading order (see the Controlling Reading Order and Tab Order lesson) and can be scrolled into view by a keyboard user with the Tab key.

You can also add other movie clip methods to it, but these are outside the scope of this course.

For information on adding an Accessibility Properties object, see the Enabling Accessibility lesson.

How to Include Hierarchy Information in Text Equivalents

How you display text often conveys additional information. With HTML, heading level and nested list tags make hierarchical relationships available to screen reader users as well as sighted users.

Although Flash allows some HTML tags in dynamic text fields, heading levels, ordered lists and unordered lists are not among them. The tag will add a bullet before an item, but it will not create an HTML list. Relationships are more often shown with graphics or indented text. These are not accessible or Section 508 compliant without text equivalents.

Example of Hierarchical Information

Here is an example, a two-level list that might be presented in Flash with eye-catching graphic backgrounds or more simply in a text field, using tags for bullets and spaces to indent the second level items:

Contact the Appropriate Liaison or Technical Support Person in Your Department

* Department 19F

o Liaison – John Smith

o Tech Support – Bernice Jadza

* Department 20F

o Liaison – Raj Reddy

o Tech Support – Ted Smart

* Department 22

o Liaison – Jennifer Morgan

o Tech Support – Raul Torres

Sample Text Equivalents

Here is one way to describe the hierarchical nature of the information.

1. Group the graphics and text containing the entire contact list in one movie clip. For a longer list, consider a MovieClip for each top-level item in the list.
2. Make the movie clip accessible.
3. `if (!mcContactList.accessibilityProperties)`
4. `mcContactList.accessibilityProperties =`
5. `new AccessibilityProperties();`
6. `mcContactList.accessibilityProperties.silent = false;`
7. `mcContactList.tabIndex = 5;`
7. Add the following ActionScript, giving the movie clip an Accessible name that conveys both the information and the hierarchy of the list:
8. `mcContactList.accessibilityProperties.name =`
9. "Department 19F Liaison, John Smith. Department 19F Tech

10. Support, Bernice Jadza. Department 20F Liaison, Raj
11. Reddy. Department 20F Tech Support, Ted Smart. Department
12. 22 Liaison, Jennifer Morgan. Department 22 Tech Support, Raul Torres.";
13. If the movie clip includes any dynamic text fields or movie clips, don't forget to set forceSimple to true, so those child objects are made not accessible.
mcContactList.accessibilityProperties.forceSimple = true;
14. Make the new information available through MSA.
15. // Make the changes active
Accessibility.updateProperties();

These four steps can be done with ActionScript or through the Accessibility Panel. Select the movie clip, open the Accessibility Panel, check "Make object accessible," and add the Tab index in step 2. Add the Name in step 3. In step 4, uncheck "Make child objects accessible," which is the equivalent of forceSimple=true.

When there are more than two levels in a list, as in an organizational chart, use the word "level" as you traverse the diagram down each of its legs.

```
mcOrgChart1.accessibilityProperties.name = "Level 1 – Chief
Information Security Officer. Level 2 – Deputy information
Security Officer. Level 3 – Intern Assistant Security
Officer. Level 2 – Security Operations Manager. Level 3 –
Deputy Security Operations Manager. Level 1 – Strategic
Security Initiative Program Director. Level 2 – Security
Officer Level II";
```

How to Provide Text Equivalents for Animated Content

Animated content is one of the big reasons for using Flash, and it can be made accessible for those using screen readers.

Add a Text Equivalent to a Brief Animation with the Accessibility Panel

For a brief animation, like an employee ID card being swiped through a reader:

1. Use a movie clip for the animated object. If it includes text fields that should not be read or other movie clips that should not be named, make those child objects not accessible.

2. Create your animation.

3. Open the Accessibility Panel for the movie clip and add a Name and an optional Description.

Remember that the Name should succinctly convey information equivalent to seeing the animation. The Description can provide additional information. However, remember that it may be read immediately following the Name, on request or not at all by various screen reader programs.

Add a Text Equivalent to a Brief Animation with ActionScript

To do the same thing with ActionScript:

```
// Make the movie clip accessible
if (!empIDCard.accessibilityProperties)
empIDCard.accessibilityProperties = new
AccessibilityProperties();
empIDCard.accessibilityProperties.silent = false;
empIDCard.tabIndex = 4;
```

```
// Make any child objects not accessible
empIDCard.accessibilityProperties.forceSimple = true;
```

```
// Add the name and description
empIDCard.accessibilityProperties.name = "Employee ID Card";
```

```
empIDCard.accessibilityProperties.description =
"A graphic image of an Employee ID card moving through a
Card Swipe";
```

```
// Make the changes active
```

```
Accessibility.updateProperties();
```

```
Add a Text Equivalent to a Longer Animation
```

Animations of more than five seconds require user controls to step through them, as explained in the Providing Audio and Video Controls lesson. Use ActionScript to change the Accessible Name with each step. Then add code to update the Accessibility Properties so the change takes effect:

```
Accessibility.updateProperties();
```

How to Provide Text Equivalents for Progress Bars

A progress bar conveys important information, for which a text equivalent is usually required. An equivalent is not required for an initial loading progress bar if loading completes within five seconds. It is required for all other uses.

The Flash progressBar component does not provide accessibility support automatically. We present three possible solutions.

Approach One: Provide an On-screen Text Equivalent

This solution may be used in addition to the progressBar or in place of it.

1. Create a MovieClip on the stage, because this text must be placed in the tab order, to allow users of screen readers to manually inspect the changing values.
2. Put the MovieClip in the tab order with the Accessibility Panel or ActionScript.
3. Add a dynamic text field to the MovieClip with an initial value reflecting the initial state of the progress bar: 0%, 0 out of 5, etc.
4. Set `tabEnabled=true` for the MovieClip.
5. Attach a focus listener to the MovieClip that sets focus to the TextField when the dummy MovieClip receives focus. To ensure Shift+Tab works, too, set the dummy MovieClip's `tabEnabled` property to false until focus leaves the TextField and then reset the dummy MovieClip's `tabEnabled` property to true. For example:

```
6. var mc2: MovieClip = new MovieClip();
```

```
7. addChild(mc2);
```

```
8. mc2.addChild(tx1);
```

```
9. mc2.tabChildren = true;
```

```
10. mc2.tabEnabled = true;
```

```
11. mc2.addEventListener(FocusEvent.FOCUS_IN,updateFocus);
```

```
12. tx1.addEventListener(FocusEvent.FOCUS_OUT,updateFocus2);
```

```
13. tx1.tabIndex=10;
```

```
14. function updateFocus(e:Event): void
```

```
15. {
```

```
16. stage.focus = tx1;
```

```
17. mc2.tabEnabled = false;
```

```
18. }
```

```
19. function updateFocus2(e:Event): void
```

```
20. {
```

```
21. mc2.tabEnabled = true;
```

```
} 22. As the value changes, update the on-screen textual equivalent: 50%, 2 out of 5, etc. through ActionScript.
```

Approach Two: Update the Accessible Name of the ProgressBar

1. Assign an Accessible Name to the progressBar, including the name of the progressBar and the value: 0% downloaded; question 1 of 5, etc.
2. Set `tabEnabled=true` for the progressBar, so users of screen readers can manually inspect the changing

values.

3. As the value changes, update the Accessible Name: 20% downloaded; question 5 of 5, etc.

4. Call `Accessibility.updateProperties()` after each update, but not more often than once a second.

Approach Three: Create an Accessibility Implementation for the ProgressBar

A better version of the progressBar would take Approach Two a bit farther, so the update frequency and format of the Accessible Name could be specified as parameters to an accessible progressBar. This approach is beyond the scope of the course.

How to Provide Text Equivalents for Window & Dialog Titles

Flash makes it easy to provide text Equivalents for Window and dialog titles.

When Graphics are Used for Window and Dialog Titles

1. Convert Graphic symbol to a Movie Clip symbol. See the topic How to Provide Text Equivalents for Graphic Symbols in this lesson.

2. Add an Accessible Name, for example, "Dialog box: Not Eligible."

3. Place the Movie Clip in the tab order.

When Text is Used for Window and Dialog Titles

* Put the text in the tab order and reading order. See Approach One in the topic How to Provide Text Equivalents for Progress Bars in this lesson.

Technical Knowledge Checks

Check your knowledge of the procedures for providing captions in Flash. Use the Answer button to check your selection.

Top of Form

To be accessible, Graphics must be sometimes converted to Movie Clips. Three of these are reasons for converting them. Which one is not?

- A. An instance name is needed to add Accessibility Properties.
- B. Graphics cannot be made keyboard accessible.
- C. Graphics cannot have an Accessibility object associated with them.
- D. Movie Clips are automatically assigned an Accessible Description.

Bottom of Form

Top of Form

When providing a text equivalent for a progressBar that all can see, why do you need to enclose the text in a dummy Movie Clip?

- A. A Movie Clip will announce text updates to the screen reader.
- B. An Accessible Name can be added to a Movie Clip.
- C. Users of screen readers must be able to tab to the text.
- D. Putting text in a Movie Clip makes it dynamic.

Bottom of Form

Ensuring Keyboard Accessibility

Introduction

Imagine you can see all those fascinating things to click on in a Flash application. What if you cannot operate a mouse? What if your screen reader tells you they are there, but you cannot see them to point at them with a cursor? To avoid frustrating your non-mouse-using audience, every Flash file must also be keyboard accessible.

User Perspective: Mouse Required

Select the Example link below to experience a Flash application with a common problem: one or more controls that work only with a mouse. For this one, pretend your mouse is broken. Use the Tab key to check out all the controls. The presentation will open in a new window.

Imagine you could never use your mouse, but had to rely on your keyboard or on puffing into a straw to work an onscreen keyboard. This Flash application would not be usable by you. Many users who cannot operate a mouse or see the screen depend on keyboard accessibility but also need access to everything in your Flash application.

User Perspective: Need for Shortcuts

Select the Example link below to experience a common problem: the elements in the application are keyboard accessible, but they require too much effort to use as easily as with a mouse. Try it. Press the Tab key multiple times to reach the Next button and press the Enter key to activate the Next button — a "Well Done" message will appear. Screen reader users should use Ctrl+Enter to activate the Next button. The presentation will open in a new window.

You had to press Tab many times to reach the Next button. Users of the keyboard such as those with mobility impairments or those using screen readers often experience this issue. A shortcut keystroke is needed. Open the example again and this time tab into the Flash file and press the shortcut assigned to the Next button, Control+Shift+Period.

Provide Keyboard Accessibility

Flash makes it easy to give users a choice between the mouse and the keyboard for interacting with Flash content. However, sometimes keyboard access is not automatic. You must be sure to add keyboard access when it is not automatically provided. If you are a mouse user, read this introduction to keyboard accessibility.

Tab, Space, Enter and Arrow Keys

Keyboard users expect:

- * The Tab key will move to the next actionable control.
- * Shift+Tab will move to the previous actionable control.
- * Space will activate a button, check a checkbox, manipulate a control or type a space in a text field or text area.
- * Enter will activate a button or start a new line in a text area.
- * Arrow keys (up and down) will move among the choices in a menu, a list component, a combo box component or a group of radio buttons, selecting them without activating them, and move among the lines in a text area.

There is no need to listen for these key presses in Flash. Pressing space or Enter triggers the standard Flash event handler for mouse clicks. Standard Flash components handle arrow keys appropriately. And the Tab key and Shift+Tab keystroke are handled through properties assigned to the element, as explained in the Controlling Reading Order and Tab Order lesson.

In fact, manually listening for Enter and space key presses would not work correctly for screen readers. When screen readers are in virtual cursor/browse mode, for reading the screen quickly instead of

interacting with it, they trap keys such as Enter and space and will not send them through to the Flash application.

Instead, ensure that all actionable items are tab-enabled by setting the appropriate properties, use standard components or correctly implement custom ones, and monitor the proper Flash event that covers activation by a mouse click, Enter or space.

Ensuring keyboard access is straightforward for developers, however, it can be a challenge for the layout designer and the person writing support documentation and on-screen text. For example, the instructive terms "choose," "select" and "activate" are more appropriate for keyboard and mouse users than the terms "click on" or "drag," which are more mouse dependent. Planning the tab order from the start will produce layouts that are easier to write keyboard-based instructions for and to code.

Shortcuts and Mnemonics

As you experienced on the previous page, tabbing gets tiring when you know where you want to go and you must press the Tab key repeatedly to get there. For users who have mobility impairments, it may be even more tiring. A shortcut or mnemonic key makes getting there a lot easier and provides a more equivalent experience for keyboard-only users.

Mnemonics are shortcuts that use a letter from the name of the control or list item. The letter is then underlined in that name. For example, Ctrl+P is a standard mnemonic, in many applications, for a Print button or Print menu item. F1 is a common non-mnemonic shortcut for Help. Because operating systems, web browsers and screen readers use many shortcuts, we include a topic in this lesson on choosing good shortcuts for your Flash application.

While specific technical requirements do not exist for when to include shortcuts, a general functional rule of thumb is to add a shortcut or mnemonic in the following types of controls:

- * Requires pressing the Tab key more than 5 times to reach
- * Controls media playback
- * Needed quickly, such as to silence audio or stop repeated updates of content

Using shortcuts in these situations will address two of the checkpoints on the VHA Functional Checklist. Shortcuts and mnemonics always require adding ActionScript code to recognize when the shortcut is typed. This is true whether the shortcut is communicated to the screen reader using the shortcut property in the Accessibility Panel or ActionScript. Refer to the How-To section of this document for instructions on how to create working keyboard shortcuts.

Keyboard Accessibility of Dynamic Text

Users of screen readers like JAWS and Window-Eyes also rely on the keyboard for access. In addition to including all actionable controls in the tab order, these text items belong in the tab order or should have a shortcut key assigned to them:

- * Text that changes for any reason
- * Error messages and other very important text
- * License text
- * Alerts text

See the Controlling Reading Order and Tab Order lesson for details on making sure users of screen readers can access all relevant Flash content in the right order.

Objects that Should Not be in the Tab Order

In Flash, it is easy to hide an object temporarily just by layering another one on top or moving it slightly off-stage. This hides it from sight, but not from the keyboard. The Flash Player places these tab-enabled items in the tab order in this case. For example, the Flash Player may place seemingly disabled objects, like a Back button that does not work at the beginning of a set of slides or pages, in the tab order.

When you hide an object or disable it, check that you can no longer tab to it. If you can tab to it, make one of these changes:

- * For a movie clip, set the tab-enabled property to false.
- * For a component within a movie clip, set the Tab Children property to false on the parent movie clip.
- * For other components, set the Focus Enabled property to false.

In the rest of this topic are a few exceptions in which a hidden or disabled object should be tab-enabled or in the tab order. However, most should not be keyboard accessible or in the tab order until they are visible and enabled again.

Objects that Must Be Keyboard Accessible

The VHA Section 508 checklist specifies several checkpoints that relate to keyboard accessibility. These checkpoints are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Objects Requiring Keyboard Access for Animation

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(a)

When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.

a.2

Are applications accessible when used with the keyboard only?

* *

*

a.2.a

Are all inactive or hidden interface elements that are not relevant to the user ignored when using the keyboard only?

* *

*

a.2.b

Is dynamically changing text accessible with the keyboard?

* *

*

(d)

Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

d.2

Are obscured layered components and content made inactive by pop-up panels or page tabs unavailable to assistive technologies?

* *

*

(f)

Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.

f.1

Is all text presented in the application readable by assistive technologies?

* *

*

f.4

Do window and dialog titles properly render to text?

* *

*

(l)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

l.1

Can all areas of the form be completed, and can the form be submitted, using only the keyboard?

* *

*

VHA Section 508 Checkpoints: Objects Requiring Keyboard Access

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.6

Is off-screen and hidden content that is not relevant to the user ignored by assistive technology?

* *

*

(f)

At least one mode of operation and information retrieval that does not require fine motor control or simultaneous actions and that is operable with limited reach and strength shall be provided.

f.5

Is license text accessible via the keyboard?

* *

*

The lesson on Maintaining Focus also contains content related to this section. Objects that must be keyboard accessible can be summarized as follows.

Controls

All controls (play, stop and pause buttons, volume sliders, help buttons, choice buttons, toggle buttons, etc.) must be keyboard accessible (able to be operated from the keyboard) unless one or more of these is true:

- * They are disabled.
- * Their function is duplicated by a second, accessible control.
- * Their function is one that cannot be implemented without a mouse, drawing pad or other handheld device, including such things as freehand or pressure sensitive drawing but not drag-and-drop interfaces whose function could be accomplished with keyboard accessible buttons.

Remember to disable Accessibility for any control that is actionable but cannot be made keyboard accessible or that activates a function that cannot be made keyboard accessible. Remember alternative controls must exist unless the functionality cannot be achieved with a keyboard, as in the example above creating a freehand drawing based on the pressure of a pen device.

Media playback controls should have keyboard shortcuts, as should any controls not reachable with the first five Tab key presses.

Form Elements but Not Their Labels

It must be possible to use every radio button, checkbox, slider, text input field, choice button, submit button, list box, combo box or data grid from the keyboard.

As explained in detail in the Providing Accessible User Interface Controls lesson, controls in a form should be keyboard accessible, but labels for the controls should not be. Include the label information in the form element's Accessible Name instead.

Movie Clips

Movie clips the user can interact with serve as controls and must be keyboard accessible according to the guidelines above. This is also true for any movie clips that represent error messages, alerts, license text or dynamically changing values. Any purely decorative movie clips should not have Accessible Names and should not be tab-enabled.

Menus

Menus must be keyboard accessible. If any portion cannot be made keyboard accessible, there must be another way to execute the same function from the keyboard. The use of mnemonics and other shortcuts can help.

Dynamically Changing and Critical Text

Be sure all dynamically changing or critical text appears in the tab order, so screen readers can read it. This means dynamic text within a movie clip or input text must be used as you cannot place Flash static text objects in the tab order.

- * Dynamic text – should have a tab index that places it in the correct place in the tab order and should be tab-enabled. Making dynamic text tab-able is discussed in the topic How to Make Changing Dynamic Text Keyboard Accessible in this lesson.
- * Input text – should have a tab index that puts it in the correct place in the tab order and be tab-enabled, so the user can tab to it to enter text.
- * Window and dialog box titles – should be dynamic text or a movie clip with an Accessible Name. They should be given a tab index that places them in the correct place in the tab order.
- * Error text – should be dynamic text at the top of the form with a tab index that places it at the start of the tab order when it is on-screen.

- * License text – must be reachable via the tab order and readable from the keyboard. If license text is too long to read without scrolling, the field must be scrollable via the keyboard. See Scrollable Content, below.

* Read-only input text – should be reachable via the Tab key and have an appropriate tab index if it contains a visible value sighted users can read.

Scrollable Content

When possible, developers should ensure that scrolling is not required to access content. If scrolling is required, developers must ensure that the scrolling mechanism is keyboard accessible. This is typically achieved by the Flash Scroller object, which provides for keyboard access via up/down and page up/down.

Functionality that Must Be Keyboard Accessible

The VHA Section 508 checkpoints that specify functionality that must be keyboard accessible are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Functionality Requiring Keyboard Access

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(a)

When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.

a.1

Can all actions or functionality be performed from the keyboard?

* *

*

a.1.a

Do context menus have keyboard or redundant methods of access?

* *

*

a.1.b

If contiguous and noncontiguous text and data can be selected, copied, and/or pasted via the mouse, can these tasks be accomplished with the keyboard?

* *

*

a.1.c

Can objects/windows be moved, resized, and manipulated via the keyboard?

* *

*

a.1.d

Can meaningful rollover content be triggered by keyboard actions?

* *

*

a.1.e

Can drop-down lists be opened properly via the keyboard?

* *

*

a.1.f

Can users navigate between application panes via the keyboard?

* *

*

a.1.g

Does tab order proceed logically and reflect the normal flow of use?

* *

*

a.1.h

Is toolbar functionality keyboard accessible or duplicated in menu structure?

* *

*

a.2

Are applications accessible when used with the keyboard only

* *

*

a.2.c

Are forced focus changes avoided?

* *

*

a.2.c.1

If focus changes cannot be avoided, is the user notified before any unanticipated keyboard focus shift?

* *

*

(c)

A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

c.2

Is keyboard focus indicated visually?

* *

*

(d)

Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

d.3

Are shortcut keys and mnemonics indicated when the user interface element is not in the tab order?

* *

*

(f)

Textual information shall be provided through operating system functions for displaying text. The minimum information that shall be made available is text content, text input caret location, and text attributes.

f.2

Is information provided about text content, text input caret location, and text attributes?

VHA Section 508 Checkpoints: Functionality Requiring Keyboard Access

§1194.31 Functional Performance Criteria

1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.8

Can all read-only edit elements receive keyboard focus?

* *

*

a.11

Can auto-updating content be stopped, paused or hidden?

* *

*

a.12

Do content updates define focus to the proper location?

* *

*

(f)

At least one mode of operation and information retrieval that does not require fine motor control or simultaneous actions and that is operable with limited reach and strength shall be provided.

f.3

Are means provided to ease keyboard access for those who cannot use a mouse?

* *

*

f.3.a

On pages with more than five elements reached by a press of the Tab key, are shortcut keys provided for all important functions including toolbars, navigation controls, and media playback controls?

* *

*

f.3.b

Do accelerator/shortcut keys avoid conflicts with standard browser keystrokes, especially where those browser functions cannot be accessed through menus or other keyboard-accessible means?

* *

*

f.3.c

Is each shortcut or mnemonic key used for a specific function within an application unique?

* *

*

f.3.d

Where they exist, are shortcut or mnemonic keys clearly indicated to the user?

* *

*

These can be summarized as follows.

Ease and Speed of Response

Keyboard accessibility should provide the same ease and speed of response as the mouse does. Add keyboard shortcuts to control media playback and to activate any functionality that would require more than five presses of the Tab key.

Remember to indicate somewhere on the screen what the shortcuts are, for sighted keyboard-only users and for users of screen readers that do not announce the MSAA-provided shortcuts that you can enter through the Accessibility Panel.

You can put the shortcuts list directly on the screen, provide a button for a popup window with them or use underscores in menu items and button names for mnemonics. If this is not possible, provide the shortcuts in user support documentation such as online help.

Visual Indication of Focus

Keyboard users with some sight need to know where their tabbing takes them. Do not turn off the yellow rectangular outline used for this purpose (called the focus rectangle) unless you add a substitute that meets the visual requirements for accessibility such as use of color. See the lesson on Maintaining Focus for details on how to create an acceptable substitute.

Focus Changes

Avoid forcibly moving the focus when the keyboard user did not issue an action that warrants one. This is confusing to all keyboard users including users of screen readers. If an unexpected but not forced focus change is made (e.g., when tabbing out of a text input field), notify the user where focus will be placed. The Maintaining Focus lesson covers focus changes in depth.

One place where a focus change is generally needed to improve keyboard access is when a form's submit button is activated. The focus should move to the top of the next screen or to an error message at the top of the current one. This assists keyboard-only users in quickly interacting with the new content and provides a clear indication of where keyboard focus is.

Keyboard Access to Pause, Stop or Hide Dynamically-Changing Content

When text dynamically changes continually (e.g., a stock ticker), provide keyboard accessible user controls to pause, hide or stop the updates. This topic is covered in detail in the Providing Audio/Visual Controls lesson. Providing keyboard access to these controls allows users with cognitive impairments, visual impairments and other disabilities to access the content at their own pace.

Rollover Content

Meaningful content that appears as the mouse rolls over an element must also be accessible via the keyboard. For buttons with brief rollover content, like those below, you can reveal the popup when focus moves to the button and hide it when focus moves away. To make the content available via the screen reader, treat the popup as you would a label: keep it out of the reading order and add its text to the button's Accessible Name.

More extensive rollover content, such as a panel of details about a location as the user rolls over a map location or an animation launched by a rollover, requires a way to activate or skip the rollover content and to tab in and out of it.

Context Menus

Context menus are those that change depending on the item selected. They open with a right-click (Windows) or a Control-click (Mac) of the mouse. This functionality should also be available from the keyboard. Consider a shortcut key such as Shift+F10 or an alternative accessible from the keyboard, such as a small button next to the item.

Combo Boxes and Dropdown Lists

Dropdown lists should work with arrow keys, as well as with a mouse. Watch that activating a selection occurs only when the user presses the Enter key, presses the space bar or when focus leaves the combo box or dropdown but not as combo box or dropdown items are being selected.

Toolbar Functionality

If a toolbar cannot be made keyboard accessible, add shortcut keys or menu items to perform the same actions. Anytime keyboard shortcuts are used, clearly document the shortcuts for the user in an accessible format. The documentation should be in the Flash file or in the page displaying it but may also be provided in user support documentation.

Scrolling Text

When text can be scrolled with the mouse, it must also be scrollable from the keyboard.

Selectable Text

If the user can select, copy or paste text with the mouse, a method for doing the same from the keyboard must be included. In Flash, when text fields are set to allow copying, this functionality automatically works with either the mouse or keyboard.

Navigation between Components and Non-components

Flash components (radio buttons, checkboxes, data grids, etc.) use a different focus manager than standard Flash objects (movie clips, simple buttons) to manage keyboard focus. Because of this, tabbing between components and other controls (movie clips and simple buttons) sometimes fails. Text Input components generally provide smooth navigation among either components or movie clips. See the topic [How to Make a Movie Clip Button Keyboard Accessible](#) for a solution.

Navigation between Panes or Simulated Page Tabs

Simulated page tabs must provide keyboard access to switch from one tab to the next. For applications with two or more panes, keyboard navigation between panes is required. It can be provided with the Tab key, menu or shortcut keys. An example of a shortcut key for pane navigation is the F6 key in Windows Explorer.

Window Resizing, Moving and Closing

If windows within a Flash application can be resized, moved or closed with the mouse, it must be possible to do the same from the keyboard. Create controls that are keyboard accessible and can be operated with the Tab, Enter and arrow keys. For example, if an object can be moved with the mouse after activating a menu item, ensure that the menu and menu item are keyboard accessible and that the arrow keys can then be used to move the object around.

This Windows context menu is an example of keyboard accessible window manipulation. Once this move menu item is activated, a thick gray border appears around the window indicating that it can be moved. The arrow keys then move this thick gray border representing the window. Pressing Enter completes the move. Escape cancels it.

Creating Good Keyboard Shortcuts

Keyboard shortcuts make Flash applications with many controls a good deal easier to access for users who cannot use the mouse.

When to Use Keyboard Shortcuts

Add keyboard shortcuts for the following situations:

- * To silence audio quickly
- * To pause repeated dynamic content updates quickly
- * To control media playback (start, stop, pause, volume control)
- * To access any control that requires more than five presses of the Tab key to reach

Shortcuts to Avoid

When choosing shortcut keys or key combinations, you should try to avoid the main shortcuts for:

- * Windows (e.g., Ctrl+Alt+Delete, F1, Alt+F4, Ctrl+F4)
 - * Internet Explorer and Firefox (e.g., F11, Alt+D, Ctrl+P, Ctrl+N, Ctrl+T, Ctrl+F, Ctrl+A, Ctrl+X, Ctrl+V, Ctrl+H)
 - * JAWS and Window-Eyes (e.g., Numpad Plus, Ctrl+Shift+A, Ctrl+Home, Ctrl+End, most single letters, Shift+most single letters, Insert+Down Arrow, Ctrl+Shift+R; click on the Resource link below for a list)
- It is unavoidable to have some conflicts, but choose carefully to avoid the most commonly used shortcuts. Single letter shortcuts may be more difficult for screen reader users, because screen readers trap keys in certain modes, but they may be better for users with mobility impairments. Consider using ActionScript to test for the presence of a screen reader (see the How-To section of this document). Or remind screen reader users to turn off or pass keys through Virtual PC cursor mode, which traps single-letter keystrokes.

Conceptual Knowledge Check

Check your knowledge of text equivalents. Use the Answer button to check your selection.

Top of Form

Which of the following must be made possible from the keyboard in a Flash application?

- A. Drawing a diagram
- B. Drawing functions that measure the amount of pressure applied by a pen
- C. Handwriting functions replicating the user's own handwriting on the screen
- D. Scrolling to read a license agreement

Bottom of Form

Top of Form

Which of these is the best shortcut for a Health Actions tab in a Flash application?

- A. Ctrl+7
- B. Ctrl+H
- C. Alt+F4
- D. Ctrl+A

Bottom of Form

How to Make a Movie Clip Button Keyboard Accessible

Flash movies often include MovieClips that function as buttons. Making them keyboard accessible requires ActionScript.

Making the MovieClip Keyboard Accessible

1. Start with a MovieClip with an associated AccessibilityProperties object.
2. Put the control in the tab order. There are two ways to do this.
 - a. If the MovieClip functions like a button and does not need to support child object accessibility, set the `buttonMode` property to `true`.
`mcStartButton.buttonMode = true;`
 - b. Otherwise, set `tabEnabled = true`.
`mcStartButton.tabEnabled = true;`
3. Add a `MouseEvent.CLICK` handler. This changes the role of the MovieClip to a button for screen reader users. It will also listen for Enter or space key presses correctly and accept activation programmatically from assistive technology (AT). Screen readers will trap these keys in Virtual cursor/Browse mode and will not send them through, so use the `MouseEvent.CLICK` event rather than a method associated with a key up or key down event handler.
4. // assign mouse click event handler

```

5. mc_StartButton.addEventListener(MouseEvent.CLICK,startCourse);
6. // callback function associated with mouse click
7. function startCourse(): void
8. {
9. // start course, for example
10. gotoAndPlay(2);

```

} Ensuring Keyboard Navigation between Actionable MovieClips and Components

Flash components (RadioButtons, CheckBoxes, DataGrids, etc.) use a different focus manager than standard Flash objects (MovieClips, simple buttons) to manage keyboard focus. When mixing MovieClips with components, Tab and Shift+Tab may not always correctly navigate between the two groups. Tabbing may just cycle among the components or among the MovieClips. TextInput controls appear to work well when combined with either group. Test your application. If focus does not move properly between MovieClips and components, use the solution below. It is not needed if focus moves correctly.

The solution is to put components inside a dummy MovieClip with its tabChildren property set to true. Then ensure that all other MovieClips appear at the same depth as the dummy MovieClip: put each group of interlaced components or MovieClips into a dummy MovieClip as needed. For example, if your application contains groups of components followed by MovieClips and then a group of more components, you will need to create a dummy MovieClip for each group of components.

```

// import and enable accessibility
import fl.accessibility.CheckBoxAccImpl;
CheckBoxAccImpl.enableAccessibility();

// create instances for CheckBoxes
var chkBusinessDevelopment:CheckBox = new CheckBox();
chkBusinessDevelopment.label = "Business Development";
var chkFinance:CheckBox = new CheckBox();
chkFinance.label = "Finance";
var chkInformationTechnology:CheckBox = new CheckBox();
chkInformationTechnology.label = "Information Technology";

```

```

// create the dummy MovieClip
var dummy:MovieClip = new MovieClip();
// ensure that items in the dummy clip are tab-able
dummy.tabChildren=true;
// add the dummy to the stage
addChild(dummy);
// add the CheckBoxes to the dummy
dummy.addChild(chkBusinessDevelopment);
dummy.addChild(chkFinance);
dummy.addChild(chkInformationTechnology);
stop();

```

Checking Your Work

With the Keyboard Only or with a Screen Reader

To check keyboard accessibility of your MovieClip button and the ability to navigate to and from it if other components are used:

1. If using a screen reader, such as JAWS or Window-Eyes, activate it and a compatible browser.
2. Activate the Flash application.
3. Tab to each actionable MovieClip button.
4. Activate each actionable MovieClip button using space or Enter. (Testing this step with a screen reader will verify that the event listeners are correctly listening for a MouseEvent.CLICK event, not a KEY_UP or KEY_DOWN event.)
5. Ensure the desired action occurs.
6. Tab through the application to all components and MovieClips by pressing the Tab key.
7. Shift+Tab (back tab) through the application to all components and MovieClips by pressing the Shift+Tab keystroke.
8. Verify that all actionable items can be reached via the keyboard and that the keyboard focus does not

get stuck just in components or just in MovieClips.

How to Add Keyboard Shortcuts

Adding a Shortcut for Activating a Control

To create a shortcut, you must:

1. Add a listener for the key up or key down event to the stage (not to the component or movie clip).
2. Evaluate which key was pressed.
3. Activate the appropriate click event function or call the method that performs the desired action.
4. // assign event listener for key up
5. stage.addEventListener(KeyboardEvent.KEY_UP,
6. handleShortcut);
- 7.
8. // callback function for key up listener
9. function handleShortcut(event: KeyboardEvent): void
10. {
11. // key code 190 is for the period
12. if (event.keyCode == 190 && event.ctrlKey &&
13. event.shiftKey)
14. {
15. // perform action based on shortcut being pressed
16. gotoAndPlay(2);
17. }

} Creating Different Shortcuts for Different Needs

Single letter shortcuts may be more difficult for screen reader users, because screen readers trap keys, but they may be easier for users with mobility impairments. Consider using ActionScript to test for the presence of a screen reader with Accessibility.active, and activate a different set of shortcuts and instructions. Or remind screen reader users to turn off Virtual PC cursor mode, which traps single-letter keystrokes.

// assign event listener for key up

stage.addEventListener(KeyboardEvent.KEY_UP, handleShortcut);

// callback function for key up listener

function handleShortcut(event: KeyboardEvent): void

{ // determine if MSAA is supported on this platform

// and if assistive technology (e.g., speech

// recognition or a screen reader) is active

if (Capabilities.hasAccessibility &&
Accessibility.active)

{

// key code 190 is for the period

if (event.keyCode == 190 && event.ctrlKey
&& event.shiftKey)

{

// perform action based on Ctrl+Shift+Period

// shortcut being pressed

gotoAndPlay(2);

}

}

else

{

// perform action when user presses the letter n.

if (event.keyCode == 78 || event.keyCode == 110)

{

// if any TextInput fields, add a test here

```
// that focus is not currently in one
```

```
// perform action based on shortcut
// being pressed, such as
gotoAndPlay(2);
}
}
} Indicating Keyboard Shortcuts
```

Here are some ways to convey keyboard shortcuts to those who need them:

- * Use underlined letters in button names or menu items, where the shortcut is Ctrl plus the letter, Alt plus the letter or, if there are no TextInput fields, the letter alone.
- * Add tooltips that appear when a user tabs to the control, and make them accessible to the screen reader as well by adding them to the control's Accessible Name.
- * Add an online help window in the Flash application with the shortcuts.
- * Add a link in the Flash application or HTML page housing it to online documentation.
- * Distribute documentation to all users separately.
- * Use the accessibilityProperties.shortcut property in ActionScript or the Shortcut field in the Accessibility Panel to convey the shortcut to the screen reader through MSAA, in addition to one or more of the methods listed above, because keyboard shortcuts also help those who do not use screen readers.

```
* mcStartButton.accessibilityProperties.shortcut =
```

```
"Control+Shift+Period";
```

Checking Your Work

Using the Keyboard Only

To use the keyboard to check your shortcut:

1. Open the Flash application in Internet Explorer or Firefox.
2. Determine if an actionable item contains a keyboard shortcut:
 - a. Review on-screen content.
 - b. Review any on-screen tooltips that are keyboard invoked.
 - c. Review the support documentation.
3. Navigate to a different object within the scope of the object with the shortcut.
4. Press the keyboard shortcut.
5. Verify that the expected outcomes occur:
 - a. Was the result the same as you would see when directly activating the button, menu item, toolbar item or other control?
 - b. If a focus shift should have occurred, did it?

Using a Screen Reader

To test your shortcut with a screen reader:

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. Determine if an actionable item contains a keyboard shortcut:
 - a. Listen to what the screen reader announces when the object with the shortcut is tabbed to or is focused (the shortcut should be announced at the end of the identity information for the current object if the accessibilityProperties.shortcut property was set).
 - b. Review the support documentation.
 - c. Review on-screen content.
 - d. Review any on-screen tooltips that are keyboard invoked.
4. Navigate to a different object within the scope of the object with the shortcut.
5. Press the keyboard shortcut.
6. Verify that the expected outcomes occur:
 - a. Was the result the same as you would see when directly activating the button, menu item, toolbar item or other control?
 - b. If a focus shift should have occurred, did it?

How to Make Read-only Text and Controls

Keyboard Accessible

Form fields meant to be read-only (not editable) must be properly set as read-only, and not disabled. Disabling removes them from the tab order, which means mouse users can review, select and copy the text, but keyboard users cannot.

```
// set the editable property to false
txtPhoneNumber.editable = false;
```

```
// if the enabled property was set to false, fix it
txtPhoneNumber.enabled = true;
```

Checking Your Work

Using the Keyboard with or without a Screen Reader

To check whether your read-only TextInput field is keyboard accessible:

1. If using a screen reader, such as JAWS or Window-Eyes, activate it and a compatible browser.
2. Activate the Flash application.
3. Determine which fields should not be editable but which a user may need to review:
 - a. This does not include controls that are truly disabled and the user does not need to review.
 - b. Fields that should be evaluated are fields such as:
 - * Completed form fields like name, address, phone number
 - * Dynamically changing text fields
 - * Text fields showing numbers and percentages, such as percent complete
 - * Fields in which the user may need to copy content

How to Make Changing Dynamic Text Keyboard Accessible

When dynamic text changes continually (e.g., a stock ticker or scores in a game, a heart rate in a medical simulator), ensure it can be focused with the Tab key, using a shortcut keystroke or both. Keyboard access to this dynamic content assists those with vision impairments, especially those who use screen readers and will likely need a quick way to review the changing content. Here is an example.

The changing heart rate in the example appears in a dynamic text object. Dynamic text cannot directly be placed in the tab order. Setting the tabEnabled property on the dynamic text does not allow a user to tab to it. Manually setting focus to the dynamic text produces odd results when the user attempts to Tab or Shift+Tab from the text.

There are two solutions. One is to use a non-editable TextInput field instead of a dynamic text object. See the previous topic for how to make this keyboard accessible.

The other solution is to place the dynamic text inside of a dummy movie clip. Place the dummy movie clip in the tab order in the desired location, using the tabEnabled property and the tabIndex property. Set the Accessible Name of the movie clip to the dynamic text value. The dynamic text is now in the tab order and has an accessible name. Call Accessibility.updateProperties() no more than once per second to keep updating it.

```
// The heart rate dynamic text field is txtHeartRate
// create a dummy MovieClip to hold it
var dummy:MovieClip = new MovieClip();
```

```
// allow the dummy MovieClip to be tabbed to
dummy.tabEnabled = true;
```

```
// set the tab order
dummy.tabIndex = 4;
```

```
// add the dummy to the stage
addChild(dummy);

// place the heart rate text in the dummy MovieClip
dummy.addChild(txtHeartRate);

// create a new accessibility properties object for
// the dummy MovieClip
dummy.accessibilityProperties =
new AccessibilityProperties();

// repeat the two items below every time the text
// changes, but no more than once per second

// set its accessible name to the dynamic text
dummy.accessibilityProperties.name =
txtHeartRate.text;

// prevent reading this text twice
dummy.accessibilityProperties.forceSimple = true;

// call update properties for name change
Accessibility.updateProperties();
```

Some dynamic text updates move too quickly or include too much text to read in real time. See the Providing Audio/Video Controls lesson for how to provide user controls to pause, stop or hide dynamic updates. Remember to make the dynamically changing content focusable via the keyboard and to make any controls used to stop, pause or hide the content keyboard accessible with the Tab key, a shortcut keystroke or both.

Checking Your Work

Using the Keyboard with or without a Screen Reader

To test a dynamic text field with the keyboard:

1. If using a screen reader, such as JAWS or Window-Eyes, activate it and a compatible browser.
2. Activate the Flash application.
3. Determine which content dynamically changes.
4. Navigate to the content by:
 - a. Tabbing to the contentor
 - b. Pressing the designated keyboard shortcut
5. Verify that the dynamically changing content is focused.

Technical Knowledge Checks

Check your knowledge of the procedures for providing appropriate keyboard access in Flash. Use the Answer button to check your selection.

Top of Form

To take something out of the tab order which of the following properties must be set to false?

- A. tabStop
- B. tabEnabled
- C. tabIndex
- D. tabOrder

Bottom of Form

Top of Form

How can you prevent problems tabbing between RadioButtons and MovieClips?

- A. Convert the RadioButtons to MovieClip symbols.
- B. Use CheckBoxes instead of RadioButtons.
- C. Put the RadioButtons in a dummy MovieClip.
- D. Add them both to the tab order with the same tabIndex.

Bottom of Form

Controlling Reading Order and Tab Order

Introduction

Making Flash content available to screen reader and other keyboard-only users is not, by itself, enough to make your application accessible to those users. You must also take steps to ensure that your content is exposed to assistive technology (AT) in a meaningful sequence. You do this by controlling the reading order. Failing to set the reading order appropriately is one of the most common Section 508 violations in Flash.

In this lesson, you will learn about the concepts of reading order and tab order and how users with disabilities are affected when those are not set appropriately. You will learn about Section 508 requirements for reading order and tab order and how they are controlled in Flash. You will also learn about Section 508 requirements for avoiding repetitive navigation links. The technical part of this lesson explains how to set reading order and tab order and how to allow the user to skip navigation links.

User Perspective: Incorrect Reading Order

When the reading order is not set, Flash exposes elements on the screen according to an algorithm that may or may not result in a logical reading order. In one version of the New Employee Orientation you have seen throughout this course, the developer decided to add menu buttons down the left-hand side of the screen, but did not set the reading order.

Select the Example link below to learn how a screen reader might read this version of the orientation. The presentation will open in a new window. If you can see the screen, you will notice highlights that track what the screen reader is reading. If you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Definitions

To understand how reading order and tab order are controlled in Flash, you must first understand the distinction between the two concepts.

Reading order:

- * Refers to the sequence in which a screen reader reads elements on a page
- * Affects only screen reader users
- * Should include all meaningful elements on a page

Tab order:

- * Refers to the sequence in which a keyboard user accesses, via the Tab key, the keyboard-accessible elements on a page
- * Affects screen reader users when they are navigating with the keyboard; also affects users with mobility or dexterity impairments who rely on the keyboard to interact with the application
- * Should include all meaningful, active elements on a page

There may be more elements in the reading order than in the tab order because screen reader users must

access not only keyboard-accessible controls and form fields, but also text and images that do not require keyboard accessibility. The following example illustrates this concept:

In this example, screen readers would read all 13 elements on the page in the sequence shown. The reading order would be 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13. However, two of the elements in the reading order (the title, number 1; and the text in the middle of the screen, number 8) do not need to be keyboard accessible. For screen reader users tabbing through the screen and for keyboard-only users who can see the screen, the tab order would be 2, 3, 4, 5, 6, 7, 9, 10, 11, 12, 13.

In reading order, the order is more important than the numbers themselves. In the previous example, the screen reader would read the elements in the same order whether they were numbered, 1, 2, 3 or 10, 20, 30. As you will see later in this lesson, it is advantageous to leave spaces between your reading order numbers, as this allows you to add elements to the reading order later. Later in this lesson, you will also learn how to allow the user to avoid having to continuously tab through all of those menu items on every screen.

Requirements for Accessibility

Section 508 requires that reading order:

- * Be comprehensive
- * Not include hidden or invisible elements
- * Be logical
- * Coincide with the tab order

Be Comprehensive

Your reading order should include all necessary elements on a page. Throughout this course, you have seen how important it is to make any content that conveys meaning accessible. A comprehensive reading order includes all the elements for which you are required to enable accessibility and provide accessible names or text equivalents. For more information, see the *Enabling Accessibility and Providing Text Equivalents* lessons of this course.

Don't Include Hidden/Invisible Elements

Remove hidden, temporarily invisible or offstage elements from the reading order during the time they are invisible. This includes rollover content that has not been activated, controls that do not appear on the screen and elements that are hidden behind popup windows. In addition, do not include decorative elements that do not convey information in the reading order. When inactive controls are visible on the screen, you should include them in the reading order but remove them from the tab order by making them inaccessible to the keyboard.

Be Logical and Coincide with Tab Order

Do not rely on Flash's default reading order. The more complex the application, the more likely you are to end up with undesirable results. Screen reader users should be able to hear the contents of a page in a logical, intuitive way — usually the same way a sighted user would read the page. Similarly keyboard-only users should be able to tab through a page without excessive jumping around, following form fields in a way that makes sense.

Control in Flash

In Flash, the same attribute, `tabIndex`, controls both reading order and tab order, so if you set the reading order correctly, elements that you have made keyboard-accessible will be in the correct tab order.

Nonetheless, the dynamic nature of most Flash applications makes controlling reading order a bit more complicated. As users interact with an application, elements appear, disappear or change; controls become active, inactive or change state; form fields are completed; error messages appear; and popup windows and rollover text appear and disappear.

For many applications, the operating system manages how elements are read as they are added to and

removed from the screen, but in Flash, it is the Flash Player that manages these changes. This makes it the Flash developer's responsibility to add elements to the reading order as they appear and remove them when they disappear or become inactive. In general, the best way to do this is to use ActionScript to build a control that manages the reading order for all the current readable elements. Following are examples of situations in which the reading order must change dynamically.

Reading Order for Rollover Content

Rollover content typically appears based on user action, requiring a dynamically-changing reading order. In this example, "rollover actuator" refers to the element a user rolls over with the mouse or selects from the keyboard, and "spawned rollover content" refers to what is displayed when the user does so.

- * Rollover actuator 1 – tabIndex 10
- o Spawned rollover Content 1 – tabIndex 12
- * Rollover actuator 2 – tabIndex 20
- o Spawned rollover Content 2 – tabIndex 22
- * Rollover actuator 3 – tabIndex 30
- o Spawned rollover Content 3 – tabIndex 32

This reading order approach allows the user to work down a list accessing rollover content, rather than sending the user to the top of the list after each rollover. More complete procedures for another approach to controlling reading order for rollovers are provided later in this lesson.

Reading Order for Error Messages

Error messages may also appear based on user input. In the example below, assume that the Name field is assigned a reading order number of 10 and the Address field is assigned number 20. The Name field is designed to accept only alpha characters, with an error message designed to appear if the user enters numbers.

In this case, if the user enters numbers in the Name field, the reading order would be updated (via ActionScript), and the field specific inline error message would receive a reading order number between 10 and 20.

More complete procedures for controlling reading order for error messages are provided later in this lesson.

Reading Order for Child Movie Clips

If you have a series of child movie clips within a parent movie, you must assign a unique reading order value to each element in each child movie, being sure the reading order works for the entire package. If, for example, you have a two child movies, each with three elements, loaded into a parent movie, and you want the contents of one child movie to be read before the contents of the second child movie, you might number the elements 1, 2, 3 for one movie and 4, 5, 6 for the next movie. If you numbered the elements in both child movies 1, 2, 3, a screen reader would probably read the first element of the first movie, followed by the first element of the second movie and so on.

Here is an example:

- * Main movie
- o Button 1 – tabIndex 1
- o Movie 1 – tabIndex 2
- * Button in movie clip – tabIndex 3
- * Button in movie clip – tabIndex 4
- o Movie2 – tabIndex 5
- * Button in movie clip – tabIndex 6
- * Button in movie clip – tabIndex 7

Allow Users to Skip Repetitive Navigation Links or Buttons

Repetitive navigation links are tedious and burdensome for many AT users.

If the menu items in this image appeared on every screen in the New Employee Orientation, non-AT users would quickly learn to ignore them until they needed them. But screen reader users would be forced to

listen to them at the beginning of each new screen, and keyboard-only users would be forced to tab through them on each screen in order to reach the Next button.

There are two ways to remedy this situation without changing the look and feel of your interface: change the reading order or provide a "skip" link or button.

Changing the Reading Order

When you have repetitive navigation links that appear above or to the left of content, one way to make the application accessible is to change the reading order so that screen readers read the content before the navigation links. This allows screen reader users to skip over the links/buttons when they are not needed; it also allows keyboard-only users to access the content without tabbing through the links/buttons on each screen. If there are many other keyboard-accessible elements on the screen, you will also need to provide a shortcut to the links so that keyboard-only users can reach them without excessive tabbing.

Note that you should not use this approach if the user may need to access the links before the content, such as when the links are needed to stop or control the audio. Procedures for changing the reading order to allow the user to skip repetitive navigation links/buttons are provided later in this lesson.

Providing a Skip Mechanism

Another way to allow users to skip repetitive navigation links/buttons is to provide a "Skip Navigation" link or button before the navigation links/buttons, as in the example below. When the user activates the "Skip Navigation" link or button, focus moves to the main content, allowing the user to skip over the repetitive links and continue reading or tabbing from the main content.

Conceptual Knowledge Checks

Check your knowledge of reading order and tab order. Use the Answer buttons to check your selections.

Top of Form

In the image that follows, which element(s), not in the tab order, should be included in the reading order?

- A. Title and yellow text
- B. Title, yellow text and map
- C. Back to Main Menu button
- D. All of the above

Bottom of Form

Top of Form

In the image that follows, which element should not be included in the tab order or the reading order?

- A. The form fields for entering name, address, city, state, phone number
- B. The active NEXT button for advancing to the next page
- C. The inactive SUBMIT button
- D. An error message that is not currently displayed but may appear, based on user input

Bottom of Form

How to Control Reading Order and Tab Order

In Flash, you control both reading order and tab order with the same `tabIndex` attribute. All the elements to which you assign a `tabIndex` value will fall into the reading order, with lower numbers being read first. Among those elements, only the keyboard-accessible ones (buttons, elements for which `tabEnabled` is set to

true and elements for which focus has been set) will appear in the tab order. Note that not all elements can be assigned a tabIndex value in Flash. You should avoid using static text, as it cannot be included in the reading order; you must choose dynamic or input text for any text that conveys meaning.

If your Flash content is set on the screen and does not change, you can use the Accessibility panel to set tabIndex values. If, however, your application includes dynamically changing content, such as elements that change upon user action, then you should use ActionScript. You will also need to use ActionScript if your content includes several Flash movies together.

Using the Accessibility Panel

To use the Accessibility panel to set the tabIndex value for an element in order to control reading order or tab order:

1. Open the Accessibility panel for the object.
2. Make sure the Make object accessible checkbox is selected and that the object has an accessible name. For dynamic text, the accessible name is set automatically to the text that is displayed.
3. In the Tab index field, enter the numerical order in which you wish the object to appear. In the example below, the Accessibility panel is open for the Next button, which is number 4 in the reading order for this screen.

To ensure that all the elements on this screen are read in the proper order, you would open the Accessibility panel for each element and enter numbers in each Tab index field, as follows:

- * Title text, "New Employee Orientation": 1
- * Text, "Welcome to new user training.... through this guide.": 2
- * Pause button: 3
- * Next button: 4
- * Help: 5
- * Employee ID Number (label): 6
- * Employee ID Number input field: 7
- * Table of Contents: 8
- * Close Window: 9

Using ActionScript

To use ActionScript instead of the Accessibility panel to set the tabIndex value for an element, use the following code (substituting the element name and tab index number as appropriate):

```
mcNext.tabIndex = 4;
```

This code is used in conjunction with event handlers, change events and/or conditional statements to handle dynamically changing elements.

Checking Your Work

The VHA Section 508 checkpoints related to reading order and tab order are available in a separate window by selecting the link that follows:

Related Checkpoint(s)

The best way to test an application for appropriate reading order and tab order is with a screen reader, as follows:

1. Open the Flash application in Internet Explorer or Firefox.
2. Verify that the screen reader is in virtual cursor or browse mode.
3. Use the down arrow to review the contents, verifying that content is announced by the screen reader in the same order as it should be read visually on-screen.
4. Tab through the active elements on the page, verifying that the tab order coincides with the visual reading order of page elements.

How to Control Reading Order for Rollovers

Following is an example of an application that includes rollover information. When a mouse user rolls over one of the training courses on this screen, additional course information appears on the screen. Since this application is keyboard accessible, keyboard users can tab to the course name and press Enter to display the rollover content.

In this example, your reading order might be as follows:

- * Changing RollOver Text – tabIndex = 9
- * Security Awareness – tabIndex = 10;
- * Scheduling Policies – tabIndex = 11
- * Filing Procedures – tabIndex = 12
- * Data Storage – tabIndex = 13

Notice that this is a different approach to reading order than that presented earlier in this lesson. Instead of allowing users to work their way down the list, accessing rollovers, this example assumes that users might want to review from the beginning after they have read a rollover. So the changing rollover content is placed before the first item on the list and appears only when it is activated by the user.

You cannot use the Accessibility panel to perform all the steps required to make a screen like this accessible. Instead, you should use ActionScript to perform the following tasks. Sample ActionScript code is provided at the end of this task list:

1. Set the tabIndex property for each element.
2. Assign mouse-click and mouse-over event handlers:
 - a. To ensure that keyboard-only users can access the course description information, you must add a mouse-click event handler to each training course option. In Flash, the mouse-click event responds not only to a mouse click, but also to a user pressing Enter or the spacebar on the keyboard. Do not use keyboard event handlers, as screen readers in virtual cursor mode will trap these keys.
 - b. Each training course option should also have a separate mouse-over event handler to allow mouse users to hover over the option and see the rollover content. Although it is possible to combine the mouse-over event handler with the keyboard focus event handler, it is recommended that you keep them separate so that you do not accidentally shift the focus in response to a mouse-over.
3. Set the function that the mouse-click event handler calls. The mouse-click event handler should update the text in the rollover area and move the focus to the appropriate rollover MovieClip. Although there are many ways for the event handler to determine which rollover clip to display, the easiest way is usually with a switch statement on the activated object (in this case, the course name). The event callback function displays appropriate rollover content based on which switch statement is called.
4. Set accessible names for both the training course options and the rollover MovieClips.
 - a. The accessible name of each training course should include text informing the user that the course name can be activated and that when it is activated, it will result in a focus change, for example, "Security Awareness. Activate to display details about this course and move focus there."
 - b. The accessible name for the rollover MovieClips should be the rollover text itself because you will not be able to set the focus directly to the text field.

Following is ActionScript code to perform the preceding task list:

```
// set the tabIndex property of all the item
mcRollover.tabIndex = 9;
trainingChoice1.tabIndex = 10;
trainingChoice2.tabIndex = 11;
trainingChoice3.tabIndex = 12;
trainingChoice4.tabIndex = 13;

// Initialize the accessibilityProperties object and tell
// it not expose the text inside of it since we will later
// set the accessible name on the movieclip to that of
// the text.
mcRollover.accessibilityProperties =
new AccessibilityProperties();
mcRollover.accessibilityProperties.forceSimple = true;

// assign event handlers for mouse clicks and mouse over
// events
trainingChoice1.addEventListener(MouseEvent.CLICK,
changeRollOverOnActivation);
trainingChoice1.addEventListener(MouseEvent.MOUSE_OVER,
changeRollOverVisually); // function not shown
trainingChoice2.addEventListener(MouseEvent.CLICK,
changeRollOverOnActivation);
```

```

trainingChoice2.addEventListener(MouseEvent.CLICK,
changeRollOverVisually); // function not shown
trainingChoice3.addEventListener(MouseEvent.CLICK,
changeRollOverOnActivation);
trainingChoice3.addEventListener(MouseEvent.CLICK,
changeRollOverVisually); // function not shown
trainingChoice4.addEventListener(MouseEvent.CLICK,
changeRollOverOnActivation);
trainingChoice4.addEventListener(MouseEvent.CLICK,
changeRollOverVisually); // function not shown

```

```

//The changeRollOverOnActivation function that is called
// whenever the enter/space key is pressed or the training
// course item is clicked.

```

```

function changeRollOverOnActivation(e:MouseEvent): void
{ // update rollOver text where textInside is a
// dynamic text object inside the mcRollover movieclip
switch (e.target)
{
case trainingChoice1:
{
mcRollover.textInside.text =
"This security training requires A1 clearance";
//set focus
stage.focus = mcRollover;
// set the new accessible name of the rollover
// movieclip to that of the text displayed inside
// of it. This is because we can't directly set
// focus to the text.
mcRollover.accessibilityProperties.name =
mcRollover.textInside.text + " - activate to "
+ "display details about this course above and "
+ "move focus there";
// tell assistive technology that we have changed
// an accessibility property
Accessibility.updateProperties();
// break out of the switch statement
break;
}
// case trainingChoice 2 ...
// case trainingChoice 3 ...
// case trainingChoice 4 ...
}
}

```

How to Control Reading Order for Error Messages

If your application includes error messages that are triggered by user input, you must dynamically update the reading order to include the error messages when they occur. The following instruction provides two approaches to error messages.

Error Messages that Hide Screen Content

One way to include an error message triggered by user input is to place the error message so that it hides the rest of the screen content when it appears, as in this example:

In this case, the error message would be placed first in the reading order and designed to hide the other content on the screen when it appears. You hide the other content on the screen by setting its `accessibilityProperties.silent` property to "true". When the user closes the error message, you allow the content to reappear by setting the `accessibilityProperties.silent` property to "false". To ensure that the content reappears accessibly, you must also make sure the user's focus is returned to where it was before the error message appeared. You accomplish this by assigning the `stage.focus` property to the last-focused control. You will learn more about focus in the Maintaining Focus lesson of this course.

Field-Specific Error Messages

Another approach is to display an error message at the top of the form and then display an error message in front of each field that contains an error, as in the following example:

In this case, when the form is submitted, you would use the `stage.focus` property to set the focus on the MovieClip containing the error message at the top of the form. You would use the `tabIndex` property to set the reading order with the inline error message before the form field with the error.

To control the reading order this way, use the following ActionScript 3 code:

```
// Create the error messages
var mcErrorMessage: MovieClip = new MovieClip();
var mcFieldSpecificErrorMessage: MovieClip = new MovieClip();
mcFieldSpecificErrorMessage.visible = false;
addChild(mcErrorMessage);
addChild(mcFieldSpecificErrorMessage);
var textInside:TextField = new TextField();
mcErrorMessage.addChild(textInside);
textInside.text = "An error has occurred, instructions on how "
+ "to correct each error appear next to each field in error";
var textInsideFieldError:TextField = new TextField();
mcFieldSpecificErrorMessage.addChild(textInsideFieldError);

// set the tabIndex of the error movieclip and training course
// items
mcErrorMessage.tabIndex = 8;
trainingCourse1.tabIndex = 10;
trainingCourse1.tabIndex = 12;
trainingCourse1.tabIndex = 14;
trainingCourse1.tabIndex = 16;

// initialize accessibilityProperties object for the error
// message MovieClips and set it to not expose accessibility
// for items inside of it
mcErrorMessage.accessibilityProperties = new
AccessibilityProperties();
mcErrorMessage.accessibilityProperties.forceSimple = true;
mcFieldSpecificErrorMessage.accessibilityProperties = new
AccessibilityProperties();
mcFieldSpecificErrorMessage.accessibilityProperties.forceSimple =
true;

// initially set the accessibilityProperties object to silent
mcErrorMessage.accessibilityProperties.silent = true;
mcFieldSpecificErrorMessage.accessibilityProperties.silent =
true;

// attach the event for the mouse click, enter and space
btnNext.addEventListener(MouseEvent.CLICK, displayError);

// the event function that is called
function displayError(e: MouseEvent): void
```

```
{ // display the movieclip
mcErrorMessage.visible = true;
// enable accessibility for the error movieclip
mcErrorMessage.accessibilityProperties.silent = false;

// update the accessible name of the movieclip to be the
// form error message
mcErrorMessage.accessibilityProperties.name = textInside.text;

// tell screen readers that the accessibility properties have
// been updated
Accessibility.updateProperties();
// set focus to the movieclip
stage.focus = mcErrorMessage;

// where fieldInError is the object representing the field
// in error
switch (fieldInError)
{
case trainingCourse1:
{
// set the tabIndex for the field specific error message
mcFieldSpecificError.tabIndex = 9
// set the text to the appropriate message (not shown)
// set location of field specific error (not shown)
// display field specific error
mcFieldSpecificErrorMessage.visible = true;
// set the accessible name of the field specific error
// message to that of the text inside of it.
mcFieldSpecificErrorMessage.accessibilityProperties.name =
textInsideFieldError.text;
// enable accessibility for the field specific error message
mcFieldSpecificErrorMessage.silent = false;
// tell assistive technology that accessibility properties
// have changed
Accessibility.updateProperties();
}
case trainingCourse2:
{
...
}
case trainingCourse3:
{
...
}
case trainingCourse4:
{
...
}
}
}
```

How to Allow Users to Skip Repetitive Navigation Links or Buttons

The VHA Section 508 checkpoints related to skipping repetitive navigation links are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Skipping Repetitive Navigation Links

§1194.22 Web-based Internet Information and Applications

1194.22

Checkpoint

Yes

No

N/A

Comments

(0)

A method shall be provided that permits users to skip repetitive navigation links.

0.1

If a group of links is repeated over more than one page, is there a way to skip over those links?

* *

*

0.2

Is a valid skip link target present and located before the unique content of the page?

* *

*

To allow users to skip repetitive navigation links or buttons, you can change the reading order, or you can provide a Skip Navigation link or button.

Changing the Reading Order

The image below shows the reading and tab order that was presented earlier in the lesson. In this example, the menu items are read just after the title; they are the first items the user tabs to on the screen.

To allow the user to get to the content without having to tab through the menu items on every screen, you would give the lowest tab index value, usually 1, to the first content object you want read, and set all other content to incrementally higher numbers, in the order you want it to be read. After all of the content has been given a tab index value, assign values to the repetitive navigation links or buttons. Following is ActionScript code you might use:

```
// The first text we want read is the "New Employee Orientation"  
// page header text, so we give it a tabIndex of one  
mcNewEmployeeOrientation.tabIndex = 1;  
//The second piece of text we want read is the main
```

```
// welcome content in the movie so we assign it a tab
// index of two
mcWelcomeContent.tabIndex = 2;
//The remaining buttons on the screen we assign order
// to according to where we want them to appear in the
// tab and reading order of the page
mcPause.tabIndex = 3;
mcNext.tabIndex = 4;
mcMainMenu.tabIndex = 5;
mcPolicies.tabIndex = 6;
mcBenefits.tabIndex = 7;
mcParking.tabIndex = 8;
mcSecurity.tabIndex = 9;
mcGeneralInfo.tabIndex = 10;
mcHelp.tabIndex = 11;
mcTableOfContents.tabIndex = 12;
mcClose.tabIndex = 13;
The new reading order/tab order would look like this:
```

Providing a Skip Navigation Link/Button

A Skip Navigation link or button is usually one of the first items on the page. It may not be visible, but it must be keyboard accessible. If the skip navigation link/button is not visible by default, it should become visible when focused via the keyboard.

1. Create the Skip Navigation object in the top left hand corner.
2. Convert the object to a MovieClip.
3. Set the tabIndex for the Skip Navigation MovieClip object to a value that is lower than the items you want to skip. In this example, you might set the Skip Navigation link to be the first item in the tab order, as follows:
4. `mcSkipButton.tabEnabled = true;`
`mcSkipButton.tabIndex = 1;`
5. Assign an accessible name, as in the following example:
6. `if (!mcSkipButton.accessibilityProperties)`
7. `mcSkipButton.accessibilityProperties =`
8. `new AccessibilityProperties();`
9. `mcSkipButton.accessibilityProperties.name =`
`"Skip to main content";`
10. Create an anchor point MovieClip to set focus to just before the main content objects.
11. Provide an instance name for the anchor. For this example we will use the name `mcContent` and assign it an accessible name and tab index value that is greater than those of the repetitive navigation links but lower than the content objects, as follows:
12. `var mcContent: MovieClip = new MovieClip();`
13. `addChild(mcContent);`
14. `if (!mcContent.accessibilityProperties)`
15. `mcContent.accessibilityProperties =`
16. `new AccessibilityProperties();`
17. `mcContent.accessibilityProperties.name =`
18. `"Page Content";`
`mcContent.tabIndex = 9;`
19. Add an event handler to the Skip Navigation MovieClip object and create a function that sets focus to the content anchor when the link/button is activated, as in the following ActionScript 3 code:
20. `mcSkipButton.addEventListener(MouseEvent.CLICK,`
21. `SetFocusToContent);`
22. `function SetFocusToContent (evt:MouseEvent):void`
23. `{`
24. `// set focus to the a movie`
25. `// clip just prior to the main page content`

```
26. stage.focus = mcContent;  
}
```

Technical Knowledge Checks

Check your knowledge of the procedures for controlling reading order and tab order in Flash. Use the Answer buttons to check your selections.

Top of Form

Which attribute should you set to control the reading order of elements on a page?

- A. tabStop
- B. tabEnabled
- C. tabIndex
- D. tabOrder

Bottom of Form

Top of Form

Which property or method should you use to set focus to a MovieClip after a Skip Navigation link/button has been activated or an error message appears?

- A. stage.setFocus()
- B. stage.focus
- C. focusManager.focus
- D. the MovieClip name followed by the setFocus() method

Bottom of Form

Maintaining Focus

Introduction

Focus refers to the place in the user interface that is ready to receive input — the point at which user interaction takes place. Accessible Flash includes three types of focus: keyboard focus, visual focus and programmatic focus. In most cases, your focus will be correct if you follow keyboard accessibility and tab order procedures. There are, however, situations in which you may be required to set the focus manually or make other changes to ensure that focus is maintained accessibly for keyboard-only and screen reader users.

In this lesson, you will learn how keyboard-only and screen reader users experience some common focus violations. You will learn about the three types of focus and about Section 508 requirements for controlling focus in Flash. The technical part of this lesson explains how to set focus explicitly, including how to set initial focus; how to ensure user control over focus, including how to return focus to an open Flash application; how to ensure that inactive, disabled or hidden elements do not receive focus; and how to change the default visual focus indicator.

User Perspective: No Visual Focus

To understand the concept of focus, select the Example link below and try to complete the form without using the mouse. The form will open in a new window. When you get to the In State checkbox, use the spacebar to check it. This sample form will not actually submit the content you enter.

This example highlights several problems. The most obvious, for sighted users, is that there is no visual cue

to indicate when the program is ready to accept input in the checkbox or on the buttons. In other words, there is no visual focus. So you don't know when you should press the spacebar to check the checkbox or press Enter to activate the Submit button. These elements are keyboard accessible — you can activate them from the keyboard — but keyboard accessibility without visual focus is not enough! Another problem in this example is that the focus was not set in a logical place when the screen opened, and the tab order was not logical. All these elements — keyboard accessibility, tab order and focus — need to work together to make Flash accessible.

User Perspective: Forced Focus Changes

In the following example, you will learn what happens when an application, rather than the user, is allowed to change the focus. Again, select the Example link below and try to complete the form without using the mouse. The form will open in a new window. When you get to the Group Number box, use the down arrow to select group number 5. Then use the spacebar to check both checkboxes. The sample form will not actually submit the content you enter.

This example demonstrates what is called a forced focus change. When you press the down arrow to select a group number, the focus is shifted out of the Group Number selection box. You are forced to the next field, the Active Member checkbox, before you can select the group number you want.

This problem happens when developers think only from a mouse user's perspective. Assuming that a mouse user would select the down arrow and choose from the selection list with a single mouse click, developers frequently shift the focus automatically to the next field to save the mouse user an extra click.

Unfortunately, this forces keyboard-only users out of the field before they have made a selection. Keyboard users are forced to press Shift+Tab repeatedly or use some other keystrokes to keep going back to the field. For screen reader users, these forced focus shifts are an even bigger problem, since screen reader users do not receive visual cues that the focus has shifted.

Focus in Flash

To understand how to maintain focus in Flash, you must learn to distinguish among three types of focus and have a basic understanding of how focus is controlled in Flash.

Definitions

To create accessible Flash, you must pay attention to three types of focus:

- * Flash elements have keyboard focus when a user can reach and appropriately interact with them from the keyboard, without using the mouse. Keyboard focus is important to all keyboard-only users — users with mobility or dexterity impairments, as well as screen reader users.

- * Visual focus is a visual indicator of where the current (keyboard) focus is. In Flash, visual focus is usually indicated with a blue, yellow or (in earlier versions) green focus rectangle. Visual focus is important to all sighted users, but especially to those who cannot use a mouse.

- * Programmatic focus is a systemic indicator, exposed to assistive technology (AT), of where the current (keyboard) focus is. Screen reader users and other users who cannot see the visual focus indicator rely on programmatic focus.

How Focus is Controlled in Flash

For the most part, Flash automatically provides keyboard, visual and programmatic focus to standard components and movie clips when you follow the procedures for ensuring keyboard accessibility and setting the tab order. These procedures are in the Ensuring Keyboard Accessibility and Controlling Reading Order and Tab Order lessons of this course.

By default, Flash moves focus forward to the next appropriate element when the user presses Tab and back to the previous appropriate element when the user presses Shift+Tab. The component focus manager controls this functionality for components and the Flash Player controls it for movie clips and buttons. To allow keyboard users to navigate between components and movie clips, you must use special keyboard accessibility procedures, which are provided in the Ensuring Keyboard Accessibility lesson. Further discussion of the Flash focus managers and the creation of custom focus managers, which may be needed for complex custom components, are beyond the scope of this course.

There are times when the focus should change in response to user actions other than pressing Tab or Shift+Tab. Since Flash changes focus automatically only in response to the Tab or Shift+Tab keys, these situations require the developer to set the focus explicitly in ActionScript. On the other hand, developers should avoid forcing a focus change when the user does not expect or desire it.

Requirements for Accessibility

To ensure that focus is maintained accessibly in Flash, you may have to perform some or all of the following tasks:

- * Set focus explicitly when it should change in response to something other than the user pressing Tab or Shift+Tab. This includes setting initial focus.
- * Provide user control over focus shifts, avoiding forced focus shifts.
- * Keep track of and restore focus to open applications.
- * Make sure hidden, inactive or disabled elements do not receive focus.
- * Keep Flash's default visual focus or create an accessible alternative.
- * Make sure focused elements remain within view.
- * Provide additional code for custom components.

These topics are covered on the screens that follow.

When to Set Focus Explicitly

If the user would expect the focus to change in response to anything other than a Tab or Shift+Tab key press, then the developer must make this happen explicitly. You must set focus explicitly when a screen initially opens, when an error message appears, when a user activates an element that causes a new field to appear and when a keyboard shortcut is associated with a focus change.

Set Initial Focus

You must explicitly set the initial focus on the first element of each new screen that opens in Flash. This does not happen automatically. Setting initial focus ensures that screen readers read all important content in the intended order. If you do not explicitly set initial focus on a screen, screen readers may retain focus on content that is no longer on the screen, attempt to guess where the focus should be or not announce anything at all when the screen changes.

Procedures for setting initial focus are provided later in this lesson.

Set Focus on Error Messages

When you design error messages into your Flash applications, you must ensure that you set focus explicitly on the error messages when they appear. Otherwise, assistive technology users may not know that an error has occurred.

Set Focus When a User Activates a New Field

When a user activates an element that should move focus to another area of the screen, you must set the focus explicitly. Flash's focus managers will not handle this situation automatically. In the following example, an Add Email button is designed to allow a user to enter more than one email address on the Contact Form.

When the user activates the Add Email button, a new field, Email Address 2, appears, and focus is shifted to the new field. The developer must explicitly make this focus shift.

Set Focus in Response to Some Keyboard Shortcuts

Flash's focus managers do not automatically shift focus in response to keyboard shortcuts. When your application includes keyboard shortcuts associated with controls that require a focus change (such as input fields), then you must explicitly set the focus to the desired field. Not all keyboard shortcuts trigger focus changes. Shortcuts such as those that toggle audio on and off do not require you to change the focus. Additional information about keyboard shortcuts is provided in the Ensuring Keyboard Accessibility lesson of this course.

Procedures for setting focus explicitly are provided later in this lesson.

Provide User Control: Avoid Forced Focus Changes

The flip side of setting focus explicitly is making sure your application does not forcibly shift a user's focus unexpectedly. When you forcibly shift focus, keyboard-only and other AT users may make unintended selections, or they may have to tab excessively to return to a control from which they were forced away. Unnecessary or forced focus shifting typically occurs in applications requiring user input, especially when developers have designed applications in which multiple functions are performed in response to a single mouse click.

If you prefer single mouse-click operations, you do not have to redesign your interface for accessibility. Instead, you should create dual functionality, providing mouse-click functionality for mouse users and avoiding forced focus changes for keyboard users. In most cases, this dual functionality looks the same on the screen and does not require additional controls. In some situations, you may have to provide an extra button to allow keyboard-only users to confirm a selection.

Following are some examples of situations in which a keyboard user's focus is forcibly changed, along with a keyboard-accessible solution for each.

Combo Boxes

One of the user perspective examples earlier in this lesson included a selection box, known as a combo box, with a forced focus change. When you pressed the down arrow to select a group number, you were forcibly shifted to the next field, the Active Member checkbox.

You can make this application accessible programmatically without changing the way it looks. All you need to do is make sure the focus change doesn't happen until the user tabs out of the combo box to indicate he has made his selection and is ready to move on to the next element.

Procedures for ensuring that combo boxes do not forcibly shift focus are provided later in this lesson.

Radio Buttons

Many developers design radio buttons that accept a user selection and then move focus to the next element or even, sometimes, to the next screen. When focus changes are triggered this way, the radio buttons become inaccessible to keyboard-only users.

Here is an example. The following image is part of a quiz that has been programmed to advance automatically from one question to the next when a radio button is selected. It works great for mouse users who, with one click of the mouse, make a selection and move on to the next question.

Unfortunately, keyboard users who try to use the arrow keys to make a choice find their choices are made for them at the first keystroke! Before they have even made a choice, the next question appears! This kind of programming can be even more problematic when radio buttons trigger lengthy processes such as database searches.

You can avoid this type of forced focus change programmatically, keeping the focus on a radio button until the user opts to move on. You can trigger the focus change to occur when the user tabs out of the radio button group or you can add an action button, like the Next Question button shown below.

Form Validation

Flash applications that include forms are often designed with internal validation checks that respond to user input in real time. These applications prompt the user with an immediate error message if the user has violated a data input rule, such as, for example entering letters in a field that required numerical input. They also move the user along to the next input field if the user has entered data consistent with the rules. Although this approach is designed to help the user enter data faster, this, too, is an inaccessible forced focus change.

Consider a common example of this problem. Many applications that ask users for a telephone or Social Security number forcibly shift the user's focus when the prescribed number of digits has been entered. In the application shown below, the Phone Number field has been divided into separate input fields for the user's area code, exchange and extension.

In the example above, the user has accidentally typed 702 in the area code field; his area code is actually 703. Since the form validation programming considers only the number of digits and not their accuracy, the user's focus is moved to the exchange field. The user cannot use the keyboard to return to the area code field to correct the error because there are already three digits there and he is repeatedly forced to the next field!

Accessibility requires that keyboard users be allowed to remain in a field until they actively tab to the next one. Procedures for ensuring user control over focus shifts like this one are provided later in this lesson.

Return Focus to Open Applications

Many users toggle among open applications with the keyboard. Because screen readers can interact with only one window at a time, AT users tend to switch in and out of applications more often than other users. By default, the Flash Player removes the focus when a user leaves a Flash application; when the user returns, no item has focus and the user must tab or use some other method to navigate to the place he or she left. This can be time consuming and makes the application inaccessible.

Allowing users to maintain control over focus includes keeping track of where they left off in an open application and returning them there when they return. You must ensure that users who leave your Flash application to toggle to another application can return to the last focused element.

Procedures for returning focus to open applications are provided later in this lesson.

When Not to Provide Focus

Elements that are not keyboard accessible should not receive focus. These include:

- * Form labels: Users should be able to tab to and enter data into form fields. The labels for those fields should not be keyboard accessible nor should they receive focus.
 - * Invisible elements: Elements that are offstage or hidden behind other elements should not receive focus. Note that this does not include focused controls that are out of sight in a scrollable area (see the note below).
 - * Inactive or disabled elements: Controls or other elements that are inactive or disabled should not receive focus unless it is important to convey that the control exists in its inactive or disabled state.
 - * Decorative elements: Elements that convey no meaningful information should not receive focus.
- If you follow appropriate procedures for keyboard accessibility, these elements do not usually receive focus as a user tabs through an application. However, Flash's focus manager sometimes places elements in the tab order when they should not be. This happens when developers do not code properly and when developers hide elements behind each other. So it is important for developers to remember the following rules about elements that should not receive focus:
- * Don't make them keyboard accessible.
 - * Don't explicitly set focus on them.
 - * Set properties correctly (e.g., setting visible property to false when elements are supposed to be invisible).
 - * Move off-screen elements fully off screen.

Specific technical procedures for ensuring that elements like these do not receive focus are provided later in this lesson.

Keep Visual Focus

By default, Flash provides the following visual focus indicators:

- * Blue rectangle for standard AS3 components
- * Green rectangle for standard AS2 components
- * Yellow rectangle for MovieClips and simple buttons (not button components)

You should not turn visual focus off. As you saw earlier in this lesson, this creates a product that is inaccessible to keyboard-only users who rely on the visual focus rectangle to know where they are on a

screen.

Sometimes developers hide or remove the Flash focus rectangles for aesthetic reasons. If you hide the standard focus rectangles or create custom components that do not support them, you must create another form of visual focus. Your alternative must also meet the Section 508 requirements for color and contrast, covered in the Using Color lesson of this course. Procedures for how to create a custom visual focus indicator are provided later in this lesson.

Note that active, enabled components that require focus must be visible on the screen. If you design a focused component out of sight in a scrollable area, Flash will not automatically scroll the component into view. If your application includes focusable components, you should design it so that scrolling areas are not needed. The VHA Section 508 checkpoint related to focused controls in scrollable areas is available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Focused Controls in Scrollable Areas

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(c)

A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

c.3

Is the focused control scrolled into view when focused via the keyboard?

* *

*

Focus for Custom Components

Flash may not automatically provide focus to custom components. If you create your own custom or simulated controls or components, you may need to take extra steps to provide keyboard, visual and programmatic focus. The procedures for providing focus to custom components are not within the scope of this course, but you may find the following hint helpful in diagnosing a problem:

Many developers create custom or simulated components by altering standard components or MovieClips. In these cases, focus violations typically occur within the component, where the developer has made alterations, rather than on the component, which was accessible from the start. For example, if a developer uses a standard ListBox component and then customizes the functioning of the list items within it, the ListBox itself will likely receive keyboard, visual and programmatic focus but the items within it may not.

Conceptual Knowledge Checks

Check your knowledge of focus. Use the Answer buttons to check your selections.

Top of Form

Why is it necessary to maintain visual focus for elements that have keyboard and programmatic focus?

- A. So that keyboard-only users can access elements
- B. So that users who can see the screen know which element has focus
- C. To ensure that the current focus is exposed to assistive technology
- D. None of the above; visual focus is not necessary for elements that have keyboard and programmatic focus

Bottom of Form

Top of Form

By default, the focus managers in Flash move focus forward in response to which of the following keystrokes?

- A. Enter
- B. Space bar
- C. Tab
- D. Shift+Tab

Bottom of Form

Top of Form

Why is it important to avoid forced focus shifts?

- A. Because mouse-click operations are not keyboard-accessible
- B. Because users who toggle between applications need to be able to return to the last focused element in a Flash application
- C. Because forced focus shifts require the developer to set the focus explicitly
- D. Because forced focus shifts may force a user to make unintended selections or to tab excessively

Bottom of Form

Top of Form

Which of the following must be set explicitly, rather than relying on Flash's focus managers to do so automatically?

- A. Focus on an error message that appears in response to user input
- B. Focus on a control reached in response to a keyboard shortcut
- C. Focus on the first element in a new screen
- D. All of the above

Bottom of Form

How to Set Focus Explicitly

Earlier in this lesson, you learned about many situations in which you must set focus explicitly, including setting initial focus when a screen changes and new actionable items appear. How you set focus depends on whether you are setting it to a component, a MovieClip or a custom element.

Explicitly Setting Focus, Including Initial Focus, to a Component

To set focus explicitly to a component, use the `setFocus()` method.

In the following example, when the user selects the Add Email button, a new email address field appears; you must set the focus explicitly to the new field.

You can do so with the following ActionScript code:

```
// add an event listener to the add email button that
```

```
// creates and moves focus to the second email address
// field using the setFocus() method.
```

```
package btnAddEmail.addEventListener(MouseEvent.CLICK, addEmail);
function addEmail(): void
{ var txtEmailAddress2 = new TextInput();
// positioning of field not shown
addChild(txtEmailAddress2);
txtEmailAddress2.setFocus();
} Explicitly Setting Focus, Including Initial Focus, to a MovieClip
```

To set focus explicitly to a MovieClip, use the `stage.focus` property, assigning the property the name of the MovieClip that should receive focus. For example, when a Flash application containing a media player appears, you can set focus on the Play button with the following ActionScript code:

```
stage.focus = mcPlay;
```

You can do this in the first frame of the actions layer in the ActionScript panel, shown here.

To set the focus on MovieClips on other screens of the Flash application:

1. Select the appropriate key frame in the actions layer.
2. Open the Actions panel.
3. Insert the ActionScript code setting focus to the desired MovieClip.

Explicitly Setting Focus to a Custom or Simulated Component

You can use standard Flash methods to set visual and keyboard focus to custom or simulated components, but setting programmatic focus for use by assistive technology may require additional efforts. You may need to indicate that focus is on an item within a simulated or custom control and then send a focus event to let AT know that the focus change has occurred. Although simulated and custom controls are beyond the scope of this course, and the `sendEvent` method is not well documented, advanced developers may find the following approach useful:

To send a programmatic focus change notification event to AT, use the `Accessibility.sendEvent` method, which requires three parameters:

- * The name of the MovieClip
- * The child ID of the item with focus
- * The event to fire (in this case the focus change event)

This function does not actually set the focus; it sends an event telling AT which control (or child control) should be focused. The second parameter is usually 0 unless focus is being set on a child element within the control. For example, if focus is on the first list item in a custom ListBox then the child id would be 1. Here is an example of how the call might look:

```
static const EVENT_OBJECT_FOCUS:uint = 0x8005;
Accessibility.sendEvent(MovieClipName,0,EVENT_OBJECT_FOCUS);
```

Checking Your Work

The VHA Section 508 checkpoints related to setting focus explicitly are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Setting Focus Explicitly

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(c)

A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

c.1
Do all interactive controls expose programmatic focus?

* *

*

c.4
Is the focus set appropriately after user action?

* *

*

VHA Section 508 Checkpoints: Setting Focus Explicitly
§1194.31 Functional Performance Criteria
1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.8
Can all read-only edit elements receive keyboard focus?

* *

*

a.12
Do content updates define focus to the proper location?

* *

*

Ensuring that focus is set properly includes checking for visual, keyboard and programmatic focus. You can check for programmatic focus using a screen reader or the Object Inspector.

Checking for Visual Focus

Make sure that a rectangle, caret or other visual indication of focus appears on the appropriate element when a new screen appears and follows keyboard focus throughout the screen.

Checking for Keyboard Focus

Make sure that the keyboard position starts at the first actionable element on a page and that you can tab (forward) and Shift+Tab (backward) through each element in a logical order. Make sure you can interact appropriately via the keyboard with actionable controls, including typing into input fields, activating buttons and toggling CheckBoxes and RadioButtons. Make sure you can tab to and away from all controls.

Checking for Programmatic Focus with a Screen Reader

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. Make sure the screen reader announces initial focus appropriately.
4. Activate each control that should move focus.
5. Make sure the screen reader announces the control, including name, role, state and value, that should have focus.

Checking for Programmatic Focus with the Object Inspector

1. Launch the Object Inspector.
2. Open the Flash application.
3. Confirm via the Object Inspector that the first logical field for user entry or review has focus on entry.
4. Activate each control that should move focus.
5. Make sure the Object Inspector shows the accessibility information (name, role, state and value) for the control that should have focus.

In the example you saw earlier, user selection of the Add Email button was supposed to move focus to a new email address field. If you explicitly set this focus correctly, the Object Inspector window would show the following information:

6. Navigate to any additional screens, including pop-ups, within the application, repeating steps 3-5 above.

How to Avoid Forced Focus Changes

Earlier in this lesson, you saw how forced focus changes make Flash applications inaccessible to keyboard users. The following examples show how incorrectly coded ComboBoxes, RadioButtons and form validations can be corrected to make them accessible.

Avoiding Forced Focus Changes in Combo Boxes

Here again is the forced focus change ComboBox example you saw earlier in this lesson. In this example, you were forcibly shifted to the Active Member CheckBox when you pressed the down arrow in the Group Number ComboBox.

Following is an example of the incorrect ActionScript code that might result in this forced focus shift:

```
// call changeOccurred when data is changed in ComboBox
cbGroupNumber.addEventListener(Event.CHANGE, changeOccurred);
```

```
function changeOccurred(event:Event):void
{ // focus desired ComboBox
stage.focus = chkActiveMember;
// update the visual focus as this does not come
// automatically for some components.
chkActiveMember.drawFocus(true);
}
```

To make sure ComboBoxes do not forcibly shift focus, you must use ActionScript code to perform the following tasks:

- * Add a mouse-click event listener to check for mouse clicks. This is to ensure that the application works for both mouse and keyboard users.
- * Update change events to check for a prior mouse click before moving focus.
- * Add an event listener for focus leaving the ComboBox.
- * Make sure that focus is moved only when the focus leaves the ComboBox.

The following ActionScript code allows the ComboBox to function accessibly, without a forced focus shift:

```
// create a Boolean variable to indicate whether the
// mouse was clicked or not. Ideally this would be set to
// false when the control gains focus using the
```

```

// FocusEvent.FOCUS_IN event.
var mClick:Boolean = false;
// set function to be called when focus leaves ComboBox
cbGroupNumber.addEventListener(FocusEvent.FOCUS_OUT, moveFocus);
// set function to be called when data is changed in ComboBox
cbGroupNumber.addEventListener(Event.CHANGE, changeOccurred);

// assign anonymous function indicating that the mouse has been
// clicked to the Mouse Click event listener
cbGroupNumber.addEventListener(MouseEvent.CLICK,
function(): void { mClick=true;});

// moveFocus function called when the ComboBox loses focus
function moveFocus(event:Event):void
{ stage.focus = chkActiveMember;
// update the visual focus as this does not come automatically
// with some components.
chkActiveMember.drawFocus(true);
}

// function called when a change occurs in the ComboBox
function changeOccurred(event:Event):void
{ // only move focus if the mouse was click previously
if (mClick)
{
stage.focus = chkActiveMember;
// set visual focus as this doesn't come automatically with
// stage.focus for some components
chkActiveMember.drawFocus(true);
// reset the mouse clicked variable
mClick = false;
}
}
}

```

} Avoiding Forced Focus Changes in RadioButtons

By default, when RadioButtons are focused via the arrow keys, they are also selected. Thus, using the arrow keys to navigate to or review the RadioButtons triggers a selection and a change event. Here is an example you saw earlier in this lesson; you saw how problematic this can be for keyboard users attempting to take a quiz.

Following is an example of the incorrect ActionScript code that might result in this forced focus shift:

```

question1Choice1.addEventListener(Event.CHANGE, moveFocus);
question1Choice2.addEventListener(Event.CHANGE, moveFocus);
question1Choice3.addEventListener(Event.CHANGE, moveFocus);
question1Choice4.addEventListener(Event.CHANGE, moveFocus);
function moveFocus(event: Event): void
{ if (event.currentTarget.selected)
// advanced to next question
stage.focus = question2Choice1;
}

```

To make sure RadioButtons do not forcibly shift focus, you must use ActionScript code to add a button, such as the Next Question button shown below, and require it to be activated to proceed to the next step or question. Even though it requires mouse users to click both the RadioButton and the Next Question button, this is the recommended approach.

The following ActionScript code allows the RadioButtons to function accessibly, without a forced focus shift:

```

// add a next button and an event listener
btnNext.addEventListener(MouseEvent.CLICK, moveFocus);
function moveFocus(event: Event): void
{ // advanced to next question

```

```
stage.focus = question2Choice1;
} Avoiding Forced Focus Changes with Form Validation
```

Earlier in this lesson, you saw what happens when form field validation forcibly shifts a user to another form field based on the number of characters entered, as in this telephone number form field example:

Following is an example of the incorrect ActionScript code that might result in this forced focus shift:

```
// call the moveFocus event when text is entered
txtPhoneAreaCode.addEventListener(TextEvent.TEXT_INPUT,
moveFocus);
```

```
function moveFocus(e:TextEvent): void
{ // if there are at least 3 characters in the
// field move to the exchange field
if (txtPhone.text.length >= 3)
txtPhoneExchange.setFocus();
} You should avoid immediate form field validation that is based on the number of characters entered and
remove any events that trigger movement based on input length. If field data needs to be processed,
process it only when focus leaves the field. You must also be sure not to trap the keyboard user in a field if
the field is in error. Instead, warn the user through an error mechanism such as sound and a visual
indicator.
```

The following ActionScript code allows the field validation to function accessibly, without a forced focus shift:

```
// call the checkField event when focus leaves the field
txtPhoneAreaCode.addEventListener(FocusEvent.FOCUS_OUT,
checkField);
```

```
function checkField(): void
{ if (txtPhone.text.length < 3)
// warn the user but do not trap the user in
// the field. For example, indicate an error
// visually and through sound but do not trap
// the keyboard in the area code field
// ...
} Checking Your Work
```

The VHA Section 508 checkpoints related to forced focus changes are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Avoiding Forced Focus Changes

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(a)

When software is designed to run on a system that has a keyboard, product functions shall be executable from a keyboard where the function itself or the result of performing a function can be discerned textually.

a.2.c

Are forced focus changes avoided?

* *

*

a.2.c.1

If focus changes cannot be avoided, is the user notified before any unanticipated keyboard focus shift?

* *

*

a.2.c.2

If focus changes occur, is context maintained?

* *

*

To ensure that you have avoided forced focus shifts:

1. Navigate to each actionable control.
2. Use the arrow keys to navigate within controls such as ComboBoxes, ListBoxes and DataGrids. Enter data into controls, such as edit fields, that accept input.
3. Verify that focus does not move away from the control when you are navigating within it or entering data.

How to Return Focus to an Open Application

By default, the Flash Player sets the `stage.focus` property to null (meaning no item has focus) when users leave a Flash application. Then, when the users return to the application, they must tab back to where they left off. To be accessible, Flash applications must store the last place where users had focus and return focus to that location. In some situations, such as in simple applications with a limited number of actionable items on the screen, this may not be necessary.

To ensure that focus is returned to the right place in an open application, store the current focused item in a variable when the deactivate event is sent. Then, in the activate event associated with the movie, set the `stage.focus` property to the last item that had focus. Following is sample ActionScript code for this task:

```
// create a variable to store focus
var focusObject: InteractiveObject;
// attach events for watching for when focus is lost and gained
this.addEventListener(Event.DEACTIVATE, losingFocus);
this.addEventListener(Event.ACTIVATE, gainingFocus);
// called when focus is leaving the Flash Player
function losingFocus(e:Event):void
{ focusObject = stage.focus;
} // called when focus is returning to the Flash Player
function gainingFocus(e:Event):void
{ if (focusObject)
stage.focus = focusObject;
else
// set focus to most appropriate element
}
}
Checking Your Work
```

The VHA Section 508 checkpoints related to setting and returning focus are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Returning Focus

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(c)

A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

c.1

Do all interactive controls expose programmatic focus?

* *

*

c.4

Is the focus set appropriately after user action?

* *

*

(l)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

l.6

Is focus returned to the last focused element when returning to an open module?

* *

*

To ensure that focus is returned to the last-focused element in an open Flash application:

1. Open the Flash application and any other application.
2. Identify screens where it is likely the user will be toggling among open applications and important that focus return to the right place. Examples include:
 - a. Screens that require the user to enter data from another source
 - b. Screens that require the user to review content from another application, such as a PDF document or Help page
 - c. Training course screens that are likely to be open while a user is performing other duties
3. Set focus to a component or MovieClip (but not the first one) on each identified screen.
4. Switch applications by press Alt+Tab.

5. Switch back to the Flash application by pressing Alt+Tab.
 6. Verify that focus has returned to the component or MovieClip where you left it.
- Select Next to learn how to ensure that inactive and disabled elements not receive focus.

How to Ensure that Inactive/Disabled Elements Do Not Receive Focus

Earlier in this lesson, you learned about the types of elements that should not receive focus. One example is elements that are obscured. The following example shows a pop-up that obscures other elements. In this example, the pop-up warns the user that the session is about to end and the user is prompted to request more time if necessary. When this pop-up appears, only two buttons should be able to gain focus, the Yes button and the No button. Since this is a simulated pop-up window and not a real modal window, the Flash focus manager may try to maintain focus on the fields and buttons obscured by the pop-up.

When the focus manager does not remove an inactive or disabled element from the tab order, it is usually because the element's visible property is not set correctly, the object is obscured or the item is not properly offstage. Keep the following general rules in mind:

- * Don't make elements that should not receive focus keyboard accessible.
- * Don't explicitly set focus to elements that should not receive focus.
- * Set the visible property to false when elements are supposed to be invisible.
- * Move off-screen elements fully off screen.

If it is still necessary to remove elements from the tab order and thus prevent focus, you can perform the following tasks:

- * Set the tabEnabled property to false for all the MovieClips and simple buttons
- * Set the focusEnabled property to false for all the components, such as input fields and buttons.

In the example above, you would perform the following tasks:

1. Place all the components and MovieClips other than the pop-up in a container MovieClip.
2. Set the tabChildren property to false on the newly-created MovieClip.

These actions will prevent keyboard focus from landing on any child MovieClips and components and keep focus to the Yes and No buttons in the popup. When the pop-up disappears, the container holding the simple buttons and components should once again allow keyboard access and focus. To perform these tasks, you could use the following ActionScript code:

```
var mcContainer: MovieClip = new MovieClip();
addChild(mcContainer);
// when pop-up appears
mcContainer.addChild(txtEmailAddress1);
mcContainer.addChild(txtEmailAddress2);
mcContainer.addChild(btnAddEmail);
mcContainer.addChild(btnSubmit);
mcContainer.addChild(chkDoNotContact);
mcContainer.addChild(mcHelp);
mcContainer.addChild(mcTOC);
mcContainer.addChild(mcClose);
```

```
function onSessionTimeout(): void
{ // display pop-up and Yes and No buttons not shown
// do not allow the Tab key to go to children
mcContainer.tabChildren = false;
// when the Yes button is clicked close the pop-up
btnYes.addEventListener(MouseEvent.CLICK,
closePopUp);
}
```

```
// ...
```

```
// when pop-up disappears because Yes button was
pressed
function closePopUp(): void
{ // give the user more time -- not shown
// allow the Tab key to go to children
mcContainer.tabChildren = true;
}
```

How to Change the Default Indicator of Visual Focus

Some developers and design teams do not like the default focus rectangle color provided in Flash. Turning this focus rectangle off without providing a replacement is a violation of the VHA 508 checklist requirements. The procedures for creating your own visual focus indicator are different for components and for MovieClips.

Changing the Visual Focus Indicator for Components

To change the visual focus indicator for components, you must modify the focusRectSkin associated with each component used in the component library. This is required for each type of component, not for each instance of a component. If, for example, you have two CheckBoxes, you need to change the focusRectSkin for the CheckBox component, not for each individual CheckBox. You can modify the focusRectSkin for a component as follows:

1. Select the component, using one of the following methods:
 - a. Double-click the component
 - OR
 - b. Right-click the component and choose Edit Selected from the context menu, as shown below:

A MovieClip showing the skin associated with each state of the component will appear, as shown below.

2. Select focusRectSkin using one of the following methods:
 - a. Double-click focusRectSkin
 - OR
 - b. Right-click focusRectSkin and choose Edit Selected from the context menu.
3. Use the color picker in the Flash Tools panel to change the color of the focus rectangle or create another indicator of visual focus such as an underline, brackets or arrow.
4. Return to the main scene of the Flash movie.
5. Save and publish the Flash application.
6. Tab to a component with an updated skin to review the new focus indicator.

Changing the Visual Focus Indicator for MovieClips

The visual focus indicator for a MovieClip appears when a MovieClip is rolled over with the mouse or when the MovieClip is focused with the Tab key or the Shift+Tab keystroke. If you disable the default yellow focus rectangle by setting the focusRect property to false, then you must create a custom visual focus indicator for the Over state of the MovieClip, as follows:

1. Set up your MovieClip symbol as a Button in the Properties panel, as shown here:

When you set your MovieClip as a Button, the frames in the timeline for the MovieClip change from numbers into editable MovieClip states, such as Up, Over, Down and Hit, as shown here:

2. Insert a key frame on the Over frame.
3. Using tools in the Flash Tools panel, draw your own custom visual focus indicator.

In the following example, a custom red rectangle is drawn around the MovieClip in the Over state. This red rectangle will be displayed when the MovieClip is focused via the Tab key, the Shift+Tab keystroke or the stage.focus property.

Please note that setting the focusRect property to false also stops event handler calls to the MouseEvent.CLICK event for Enter and space. You must create handlers for them.

Checking Your Work

The VHA Section 508 checkpoint related to visual focus is available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Visual Focus

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(c)

A well-defined on-screen indication of the current focus shall be provided that moves among interactive interface elements as the input focus changes. The focus shall be programmatically exposed so that assistive technology can track focus and focus changes.

c.2

Is keyboard focus indicated visually?

* *

*

To ensure that keyboard focus is indicated visually:

1. Open the Flash application.
2. Tab to each actionable element.
3. Verify that each element has a visual indication of focus and that the visual indication is not just color; it should also include another indicator, such as a rectangle, bolding, a caret or an arrow.
4. If actionable elements have child elements, tab to elements and verify their visual indicators, too.

Technical Knowledge Check

Check your knowledge of the procedures for maintaining focus. Use the Answer button to check your selection.

Top of Form

To explicitly set focus to a MovieClip called mcNext, which of the following code snippets should you use?

- A. mcNext.setFocus();
- B. mcNext.focus;
- C. stage.focus = mcNext;
- D. focusManager.setFocus(mcNext);

Bottom of Form

Providing Accessible User Interface Controls

Introduction

You have learned a good bit about user interface controls already. You have learned how to provide audio or text equivalents, make them keyboard accessible, get the tab order and reading order correct and maintain focus. Time to pull these all together and add a few more specifics about good user interface controls. This lesson covers how to ensure forms are accessible, including grouping controls, error messages, instructions, field constraints and other topics related to how user interface controls should work together.

Select Next to experience a demonstration of a screen reader encountering a poorly coded form.

User Perspective: Unseen Requests and Double-Talk

Select the Example link below to experience a Flash application through a screen reader. Listen to what information is provided and when as the screen reader reads through the entire form. The presentation will open in a new window; if you are using a screen reader to take this training, press Enter when you hear the prompt for the second Play button.

Listen to this next example to hear the screen reader while tabbing through the same file.

Completing a form you cannot see is much easier when all the instructions you need come from the control's automatically announced role (e.g., radio button) and its Accessible Name and Description, not by making labels keyboard accessible, repeating the role in the name or placing information after the control.

User Perspective: Improperly Grouped Radio Buttons

Select the Example link below to see what happens for keyboard users when radio buttons are not grouped. The presentation will open in a new window. Try it using the Tab key to move to the first radio button and the up and down arrow keys to move through and select your choice. Then tab to the Next button.

Frustrating, no? They work fine if you select one with a mouse, although as you may notice unselecting them with the mouse does not produce expected results. If you try to select one with the arrow keys, you cannot, because these radio buttons are grouped incorrectly.

Make Interface and Form Controls Accessible

In this topic, we pull together what is required to make your Flash user interface controls and the controls in your forms fully accessible, including the relevant audio and text equivalents, keyboard access, reading order, tab order and focus handling discussed in earlier lessons.

Enable Accessibility for Controls

Flash does not enable accessibility for all objects and components by default. Accessibility requires some effort on the part of the developer:

- * Graphic symbols and static text fields cannot be made accessible; unless irrelevant to the content, they must be converted.
- * Dynamic text fields, simple buttons and movie clips (including the Flash movie itself) can be made accessible through the Accessibility Panel or ActionScript.
- * Other standard Flash components can be made accessible only through ActionScript. The Accessibility Panel may be used to set some properties, however, ActionScript is required to properly enable accessibility for most components.
- * Custom Flash components must be specially built for accessibility and properly configured.

Detailed instructions appear later in this lesson.

Place Instructions and Labels Where Keyboard and Magnification Users Will See Them

Remember that some users will not see as much of the screen at one glance as you do. To make sure they can fill out forms correctly and use any animation, multimedia or game controls easily, follow these conventions:

- * Place all instructions or questions above or to the left of controls and their labels, never below.
- * Place labels for buttons inside the buttons if possible.
- * Place labels for radio buttons and checkboxes, including small buttons operating as radio buttons or checkboxes, to the right of the controls.
- * Place all other labels above or to the left of their controls.

Field specific instructions should generally be placed prior to the form field and not after it. For example:

- * Field formats: "Date Hired (mm/dd/yyyy):" is preferable to placing the "(mm/dd/yyyy)" format instruction after the date field.
- * Units of measure for a dropdown list: "Length of Bandage (in inches):" with choices 1 through 10 or "Length of Bandage:" with choices "1 inch" through "10 inches" is preferable to placing "inches" after the dropdown list.
- * Number to check for checkboxes: "Certifications you currently hold (check all that apply):" or "Options to compare (check up to three):" is preferable to placing "Check all that apply" or "Check up to three" below the set of checkboxes.

While keeping instructions with labels is very helpful to those using screen magnification, it is not required for the visual presentation of the form. It is required for the control's accessible name, as explained below.

Keep Labels Out of the Reading Order

When a control's Accessible Name and Description are created correctly, the information in the field labels for form fields such as text input, text area, combo box, list and data grid, will be redundant. Developers should prevent the labels from appearing to the screen reader in the reading order by hiding accessibility for these labels. See the lesson on Controlling Reading Order and Tab Order for how to do this.

Create a Logical Tab Order for Completing a Form

What is a logical tab order for a form? It is the same order a mouse user would follow. It goes from left to right and top to bottom, but if it encounters a set of related fields one on top of the other (e.g., name/address, user ID/password, a set of checkboxes), it goes down from the top one of those fields to the bottom one before moving on to whatever lies to the right.

See the How to Properly Form, Group and Label Controls topic later in this lesson for how to do this.

Group Radio Buttons Correctly

As you saw in the User Perspective example, you must group radio buttons to let users make a selection from the keyboard. You should be able to select one with the arrow keys and leave it selected by pressing the Tab key to move on. For how to group radio buttons, see the How to Properly Form, Group and Label Controls topic later in this lesson.

Ensure Proper Focus Handling by the Control

If you create your own custom or simulated controls or components, you may need to take extra steps to provide keyboard, visual and programmatic focus. See the Maintaining Focus lesson for some helpful tips.

Provide and Explain Keyboard Equivalents for Non-Standard Operation

Unless a control is a standard Flash component, button, text field, movie clip or a custom component that properly implements accessibility:

- * Provide an alternate control; disable accessibility and tab access

OR

- * Provide and explain a way to operate it from the keyboard

For example, if mouse users get two different results by clicking and double-clicking a control, you need two ways to operate the control from the keyboard. One can be the control's standard space, Enter or arrow key operation. The other, a shortcut key combination for example, will be non-standard and requires explanation.

For another example, if mouse users drag a pointer around a circle or drag an icon across the screen, tell keyboard users what keystrokes, such as numeric or cut-and-paste shortcuts, let them accomplish the same results.

Add a Help screen, a downloadable instructional manual or text on the screen for those who can see the control. For those who use screen readers, put the information in the Accessible Description or Accessible shortcut; see below for more about this.

Include a Captioning Component for any Video Player Component

Video with a soundtrack requires captioning. See the Using Audio for Visual Information lesson for caption display options.

Make Accessible Names Unique within a Form

Users of screen readers rely solely on a control's Accessible Name, rather than its position, shape or location on the screen, to use a control. Give each control a unique name. Include all the hints a user looking at the form might see visually. For example, if you have sections for Veterans to enter information for themselves and a guest, accessible names that do not include section or context will not be helpful:

The accessible names below do not provide sufficient context.

Your Information

- * First Name

- * Last Name

- * Email

Your Guest's Information

- * First Name

- * Last Name

- * Email

Instead, a good accessible name would be something like the following:

Your Information

- * Your First Name

- * Your Last Name

- * Your Email

Your Guest's Information

- * Guest First Name

- * Guest Last Name

- * Guest Email

All form field accessible names on the same screen must be unique, not just text input fields. For example, the use of several "Go" buttons or "More Information" buttons will be confusing for users of screen readers using the list of buttons feature to find the one they want to use. Unlike visual scanning of links, the surrounding text is not available for context with this screen reader tool.

See the topic How to Provide an Accessible Name, Description, Role, State or Value later in this lesson for more information.

Put Identity in the Accessible Name

Section 508 requires that a text equivalent be provided for every control to reveal its identity, state and operation. Other accessibility guidelines use the terms name, role, state and value. Accessibility standards require very specific information. What is this control? What does it do? If it can change states (on/off, pause/play) or its value (choices in a dropdown, radio button selections, text entered in a text input, slide from 0 to 10), what is its current state or value?

For standard Flash components, identity information other than the accessible name is provided by the standard accessibility implementation. For example, in the prior example, the accessible name "Your First Name" reveals only the identity (name) for a text input field. This identity information is sufficient, because the standard text input field in Flash reveals its operation (role), content (value) and active state automatically.

There are several key pieces of identity information that are important to include in an accessible name:

- * All levels of labels applying to it — the control's label, the name of the section it appears in, the word "required" from an instruction that all items in this section are required, etc.

- * Form field constraints – any maximum number of characters or required formatting

- * Text equivalent of images – "Print" or "Save," not "printer icon" or "floppy disk icon"
- * Text equivalent of color-coding – "Available," not "green," and "Stop," not "red"
- * For simulated components only, such as a button that looks like a single checkbox, the element type (role) – button, checkbox, etc.

Do not include the element type for any standard Flash component, button, text field or movie clip. The Flash Player can recognize these and will pass the information to MSAA automatically. Including it in the accessible name will cause it to be announced twice to users of screen readers. For example, a user might hear "Next Button button".

Place Information Regarding Non-Standard Operation in the Accessible Description, with a Specific Keystroke in the Accessible Shortcut as Needed

As mentioned earlier, controls that require non-standard operation often need different keystrokes other than space, Enter or the arrow keys. Include this how-to information in the Accessible Description as well as on-screen. For example, a developer has created a custom dropdown list that is opened via the Ctrl+down arrow keystroke. The text "Control+down arrow" should be placed in the accessible shortcut property or "Use Control+down arrow to Open and Ctrl+down arrow to close" in the accessible description of the custom dropdown list. Generally, the accessible shortcut and accessible description properties are announced after the control name, role, value, and state.

Keep the accessible properties (name, description, and shortcut) as succinct as possible, but include enough to remind an experienced user of the application how to use this control. If more description is needed the first time the control is encountered, place it in the accessible description. If a shortcut keystroke is required, put it in the accessible shortcut property. In writing these descriptions, remember that they may be read back-to-back.

Place the Role of a Simulated Control in the Accessible Name of the Control

When a simulated control is used rather than a standard component — for example, when a movie clip is used in place of a checkbox component — the purpose (role) of the control should be placed in the accessible name property of the control. This is not the ideal situation; use standard components or create custom components with custom accessibility implementation whenever possible. For example, the accessible name for a quiz question's simulated checkbox may be "Choice A. 23 inches checkbox."

Put Initial State in the Accessible Name of a Simulated Control (if Changeable)

If the state or value of a simulated control can change, include the initial state in the Accessible Name. Examples include a slider that is set initially to 0% or 50% or 100%, a simulated checkbox already checked, a Mute/Unmute button initially set to function as a Mute button, etc.

Skip this step if the initial value is null or empty (such as an empty Name text input field) and for standard Flash components or other controls that automatically report the state or value through MSAA.

Remember to update the accessible name of the simulated control to include the new state or value when it changes. The change will not automatically be announced to users of screen readers, but they will be able to review the change on demand.

Ensure Updates to Controls and Instructions are Accessible

It is not enough to simply make controls accessible at the start when the Flash application loads. Flash content is interactive, and accessibility requires updating as the user interacts with content and interface controls.

Indicate State Changes in the Accessible Name and Elsewhere as Needed

Consider the following actions:

- * When the user clicks on a mute button and it becomes an unmute button
- * When the user enters an address in a text input field, and its value is no longer null
- * When the user toggles a simulated checkbox and it becomes a checked checkbox
- * When the user moves a slider to 25% and the Accessible Name "Percent complete: 20%" is no longer valid

These are state changes. In addition to the visual indicators, these state changes must be indicated to assistive technology (AT) in a textual/programmatically way.

For simulated or custom controls that fail to report state changes, you must change the accessible name to include the new state or value each time it changes. In addition, an accessible state change event should be sent. Any time an accessible name changes, an event must be sent informing assistive technology such as screen readers of the change.

You do not need to manually indicate state changes when you use standard components; the Flash Player will report state changes through MSAA automatically.

If a state change is not expected, report it elsewhere, too. An example is a text input field that automatically changes its value when a user clicks a radio button. While visually the change is apparent immediately, to a screen reader, it is not. Mention the automatic change in the instructions for the radio button or the radio button's accessible name.

Put Feedback Graphics and Text in the Accessible Name

If making a selection results in marking one or all of the choices, add a text equivalent of this feedback to the control's Accessible Name and indicate textually either on-screen or in the accessible name/description where the feedback will appear. Some examples include:

- * Selecting an answer to a quiz question marks the correct answer with a checkmark or the selected answer with either an X or a star.
- * Making a choice in a poll displays the current percentages beside each option.
- * Marking a fourth checkbox displays "3 max" beside it.

Add New Questions or Instructions after the Control that Adds Them

If activating a control adds new questions, controls or text to the screen, place it after the control in the reading order. Again, the change is immediately apparent visually, but will not be noticed with a screen reader until the focus reaches it.

Indicate Within-Control Errors both Visually and with Sound

When an error occurs as a user interacts with a control, such as typing more than the maximum number of characters, use two indicators of the error, one for those who are looking at the control and another for those listening with a screen reader. See the two earlier lessons, *Providing Captions and Visual Indicators for Sound Cues* and *Using Audio for Visual Information*, for more information and examples.

Make Form-Submission Error Information Accessible

When the user clicks on an OK, Done, Submit, Register, Update or other button to complete a form, feedback about errors must be accessible:

- * Indicate at the top of the form or in a popup window that an error occurred and the form was not accepted. It is strongly recommended to put this notice in the tab order and move focus to it upon submission of the form.
- * Indicate the error(s) one or both ways:
 - o List the elements with errors at top of the form.
 - o Mark each control with an error using both a visual indicator and an Accessible Name change that indicates both error status and the nature of the error.

Do not rely on color alone to indicate error status for a control.

Ensure Users Have Time to Complete a Form or Use a Control

On timed interactions, allow users to adjust the time allowed or to request more time. Timed interactions include these:

- * Automatic logout after a period of inactivity
- * Automatic feedback if a response is delayed too long
- * Loss of points or the chance to continue in a learning game when the clock runs out
- * Timers used to force faster thinking or writing

For those using keyboard or voice input, sip and puff devices, screen readers or for those with cognitive impairments, the time allowed for other users may not be enough.

The VHA Section 508 checkpoints about timed interactions are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Timed Responses

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(I)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

I.13

When a timed response is required, is a warning provided to the user that is available to Assistive Technology?

* *

*

I.14

Is the user given sufficient time to interact and/or request more time?

* *

*

Best Practice

The recommended best practice is to alert the user with a popup message 30 seconds before time runs out and to allow at least 20 seconds to request a time extension.

An Exception

Time limits are allowed for quizzes or games that test skills. You may also want to offer a practice version with no time limit that will not count toward the test score.

Select Next to check your knowledge of accessible user interface controls.

Conceptual Knowledge Checks

Check your knowledge of text equivalents. Use the Answer button to check your selection.

Top of Form

Checkbox labels belong to the right of the checkbox to make them tab accessible to a screen reader.

A. True

B. False

Bottom of Form

Top of Form

Which of these is the best Accessible Name for a text field for a date in mm/dd/yy format, in which the user has already typed 12/11/1983?

- A. Date Injury Occurred (mm/dd/yy)
- B. Date Injury Occurred (mm/dd/yy) – Error: too many characters
- C. Date Injury Occurred (Contains error, bad format)
- D. Date Injury Occurred (Error: shorten 1983 to 83)

Bottom of Form

Select Next to learn how to enable accessibility for user interface controls.

If you would like to skip the technical part of this lesson, you have completed the course. Congratulations! If you are developing e-learning products for VA or products to be used in VHA web communications, and you have questions — in any stage from planning through evaluation — about how to make your software applications or operating system Section 508 compliant, feel free to contact our VHA OHI Section 508 Office for assistance.

Our accessibility experts are available to answer your questions about assistive technology and to work with you to help you make accessible products.

How to Enable Accessibility for User Interface Controls

All Flash controls are not automatically accessible. To expose all the required information about them (identity/name, operation/role, state/value), the first step is to enable accessibility for each control. Here is how to turn on accessibility for each type of Flash element.

Graphics and Text

Graphic symbols and standard (non-dynamic) text fields cannot be made accessible; unless irrelevant to the content, they must be converted to Movie Clip symbols and dynamic text fields.

Simple Controls

To make dynamic text fields, simple buttons and MovieClips (including the Flash movie itself) accessible, use the Accessibility Panel or attach an AccessibilityProperties object to each one with ActionScript.

```
if (!btnNext.accessibilityProperties)
btnNext.accessibilityProperties = new
AccessibilityProperties();
btnNext.accessibilityProperties.silent =
false;
btnNext.accessibilityProperties.name =
"Next";
btnNext.tabIndex = 6;
```

Standard Flash Components

To make RadioButtons, CheckBoxes, ListBoxes (including TileLists), ComboBoxes and DataGrids from the Flash components library accessible, you must use ActionScript. Simply adding the component from the components library will not enable accessibility. In addition to what is required for the simple controls, you must import the accessibility support library for each type of component. For example:

```
import fl.accessibility.CheckBoxAccImpl;
CheckBoxAccImpl.enableAccessibility();
```

Enable accessibility once for all components used in the Flash application. The best place to do this is at the top of the first frame in the actions layer of your Flash application. This does not have to be performed for each instance of a component but for each different type of component used.

When using ActionScript to set accessibility properties, you should always check to see if an AccessibilityProperties object exists for an instance before trying to set its properties.

```
// if an accessibility properties object does not
// exist already create one
if (!chkActiveMember.accessibilityProperties)
```

```
chkActiveMember.accessibilityProperties = new
AccessibilityProperties();
// then set the desired accessibility properties such
// as name, description, shortcut, etc.
```

Custom Flash Components

These must be specially built for accessibility and properly configured. Do not use custom controls unless you can enable accessibility for them.

How to Properly Form, Group and Label Controls

Adding a form to your Flash file? In addition to everything covered in earlier lessons about tab order, reading order, text equivalents, focus and keyboard access, you will need to know how to form, group name and label controls.

Define the Form

Unlike HTML, Flash does not have a special component to designate a group of controls as a form. Designers often create the appearance of a form visually, an outlined or colored section of the screen. If there are other elements on the screen, consider a MovieClip container for the form to allow you to more easily set accessibility for the entire form. For example, if a pop-up can appear, `AccessibilityProperties.silent = true` can be set on the MovieClip making all of the elements in the form inaccessible until the pop-up is dismissed.

Group and Label RadioButtons

1. Group RadioButtons to work with arrow keys. Use the `groupName` property in the Parameters tab of the Component Inspector panel to assign each button the same `groupName` (see image below), or create a `RadioButtonGroup` with ActionScript:

```
2. var rbg1:RadioButtonGroup =
3. new RadioButtonGroup("options");
4. rbg1.addEventListener(Event.CHANGE, changeHandler);
5.
```

```
6. var rb1:RadioButton = new RadioButton();
```

```
7. rb1.name = "Arm";
```

```
8. rb1.value = "arm";
```

```
9. rb1.group = rbg1;
```

```
10. rb1.move(10,10);
```

```
11. addChild(rb1);
```

// repeat for remaining buttons

12. Add the number of RadioButtons in the group to each Accessible Name if important to know, because assistive technology does not typically indicate this for Flash applications as it does for HTML:

```
13. rbOpt1.accessibilityProperties.name = "Arm (1 of 6)";
```

```
14. rbOpt2.accessibilityProperties.name = "Leg (2 of 6)";
```

// and so on

15. Add a group name to each Accessible Name if more than one group offers the same or similar options, because Flash has no direct method to provide the group name to MSAA:

```
16. rbPhy1.accessibilityProperties.name =
```

```
17. "Attending physician gender: Female";
```

```
18. rbPhy2.accessibilityProperties.name =
```

```
"Attending physician gender: Male";
```

Label Form Controls Properly

Every control needs both a visible label and an accessible name. The label and accessible name text are often the same. In this section, we refer to the placement of the visible label.

The default label placements in Flash are the correct ones:

- * Instructions or questions above or to the left of controls and their labels
- * Labels for buttons inside the buttons
- * Labels for CheckBoxes and RadioButtons to the right of the controls

* All other labels above or to the left of their controls

To correct a form in which they are incorrectly placed, set labelPlacement=right (or left or top) in the Parameters tab of the Component Inspector panel.

Or, if the placement is set through ActionScript, correct the value assigned to the labelPlacement property:

rb1.labelPlacement =

ButtonLabelPlacement.RIGHT;

Place Instruction Text Properly

Place instruction text on how to complete the form at the top of the form. Include any notation for required fields such as an asterisk here and not later in the form.

Checking Your Work

The VHA Section 508 checkpoints that specify accessibility requirements for form controls are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Forms

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(d)

Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

d.1

Do user interface elements including custom controls and informative graphics provide a textual name, description, role, state, and value (where applicable)?

* *

*

(l)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

* *

*

l.3

Are meaningful accessible names provided for all form elements?

* *

*

l.4

Do elements with multiple labels expose these in their accessible names?

* *

*

l.5

Is there an alternative means of indicating completion or error other than audio information alone?

* *

*

I.7
Are form field constraints clearly indicated?

* *

*

I.8
Are error messages explicitly indicated and easily found by a user?

* *

*

I.9
Is instructive text placed at the beginning of a form or prior to the relevant form field?

* *

*

I.10
Are proper programmatic events fired to provide access to context changes?

* *

*

I.11
Are form control groups properly labeled?

* *

*

I.12
Are radio button groups properly formed?

* *

*

VHA Section 508 Checkpoints: Forms
§1194.31 Functional Performance Criteria
1194.31

Checkpoint

Yes

No

N/A

Comments

(a)

At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.14

Is each element or component on a page read by assistive technology only once?

* *

*

a.22

Are event handlers that automatically trigger navigation or form submission avoided?

* *

*

(b)

At least one mode of operation and information retrieval that does not require visual acuity greater than 20/70 shall be provided in audio and enlarged print output working together or independently, or support for assistive technology used by people who are visually impaired shall be provided.

b.4

Are all checkboxes and radio buttons positioned to the left of their labels?

* *

*

Using a Screen Reader

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. If the application contains a form, make sure any instruction text appears prior to the form.
4. Navigate to all form controls.
 - a. Make sure the form controls appear in a logical tab order.
 - b. Make sure form labels, including labels to the right, field formats, field instructions, constraints or required field information, are announced when navigating to each control.
 - c. If the control is part of a group and the group name is needed to understand its purpose, check that the group name is announced with the field name and that the arrow keys cycle through all RadioButtons in a group. Tabbing into the group should focus the first item in the group and Shift+Tabbing into the group should focus the last RadioButton in the group.
 - d. If the control is a RadioButton or CheckBox or functions as one, make sure its label is placed to the right of the RadioButton or CheckBox icon.
 - e. If not, make sure that its label appears above or to the left of the control or within a button.

With the Keyboard and Object Inspector

1. Launch Object Inspector.
2. Open the Flash application.
3. If the application contains a form, make sure any instruction text appears prior to the form.
4. Navigate to all form controls.
 - a. Make sure the form controls appear in a logical tab order.
 - b. Make sure form labels, including labels to the right, field formats, field instructions, constraints or required field information, display in the Object Inspector window in the accessible name property when navigating to each control.
 - c. If the control is part of a group and the group name is needed to understand its purpose, check that the group name is shown as part of the accessible field name, with the field name, in the Object Inspector. Check that the arrow keys cycle through all RadioButtons in a group. Tabbing into the group should focus the first item in the group and Shift+Tabbing into the group should focus the last RadioButton in the group.
 - d. If the control is a RadioButton or CheckBox or functions as one, make sure its label is placed to the right of the RadioButton or CheckBox icon.
 - e. If not, make sure that its label appears above or to the left of the control or within a button.

How to Provide an Accessible Name, Description, Role, State or Value

It is not enough to provide an accessible name for a control. The accessible name must be unique, meaningful, terse, accurate and updated appropriately. Without a correctly formed accessible name, users of assistive technology such as screen readers may not be able to correctly complete a form, review necessary information or find the appropriate control. There are similar needs for controls to expose proper role, state and value. While users with vision can often determine many of these properties visually, it is not possible for many users with low vision or those who rely on screen readers.

This section describes how to implement properly identifying controls.

* Make Accessible Names unique within a form (e.g., Your First Name, Your Last Name, Your Email, Guest First Name, Guest Last Name, Guest Email).

* Put the control's identity in the Accessible Name (includes all levels of labels, constraints, text equivalent of images and color-coding).

* Put non-standard operation in the Accessible Name and additional details, if needed, in the Accessible

Description (includes non-standard keystrokes).

* For simulated controls only, put the role (type of element) in the Accessible Name; add the initial state if it is changeable and not null or empty.

* For simulated controls only, update the state or value in the Accessible Name when it changes.

Set the Accessible Name

Every control requires a succinct Accessible Name.

1. Check that accessibility has been properly enabled for the Flash movie and for the component type or the individual button, MovieClip, TextField or TextArea. (See previous topic for instructions.)

2. Set the accessible name through the Accessibility Panel by setting the Name property. Or set it in ActionScript through the accessibilityProperties.name property (either ActionScript approach below will work).

3. if (!txtAddress.accessibilityProperties)

4. txtAddress.accessibilityProperties =

5. new AccessibilityProperties();

6. txtAddress.accessibilityProperties.name =

7. "Address:";

8. Accessibility.updateProperties();

9.

10. // OR

11.

12. if (!txtAddress.accessibilityProperties)

13. {

14. var accProps:AccessibilityProperties =

15. new AccessibilityProperties();

16. accProps.name = "Address:";

17. txtAddress.accessibilityProperties = accProps;

18. }

Accessibility.updateProperties();

19. Include all required information for understanding the component: name, operation and state. Standard controls and components will automatically reveal their operation and state. Include these only for simulated controls. For custom controls, check that these are conveyed through the control's accessibility implementation.

20. For TextInput, TextArea and Label, a workaround is required to assign the accessible name to the actual text field. If the workaround is not used, or if the Accessibility Panel is used, the accessible name will appear as a graphic and the actual input field or text field will appear without an accessible name. This can be observed through the Object Inspector or a screen reader. The solution requires that the accessible name be set on the inner TextField and no accessible name be set on the component itself. This issue does not occur in ActionScript 2 and was introduced in ActionScript 3 with changes in controls that use text fields. Developers should also be sure to set the tabIndex on the inner text field of label, TextInput and TextArea components to the next or same tab index as the component itself so the object with the correct accessible properties will appear in the correct reading order — otherwise it will be placed in the reading order automatically by the Flash Player based on its x and y location.

21. if (!txtInputAddress.TextField.accessibilityProperties)

22. txtInputAddress.TextField.accessibilityProperties =

23. new AccessibilityProperties();

24. txtInputAddress.TextField.name = "Address:";

25. // set the tab index

26. txtAddress.tabIndex = 3;

27. // set the same tabIndex on the actual text field

28. // as is on the TextInput

29. txtInputAddress.TextField.tabIndex =

30. txtAddressField.tabIndex;

31. // tell assistive technology the name has been updated

Accessibility.updateProperties();

Update the Accessible Name

You may need to update an Accessible Name to convey a change in state for a simulated control.

1. Update the Accessible Name after each state change. This must done be through the

ActionScript accessibilityProperties.name property, followed byAccessibility.updateProperties().

2. Include all required information for understanding the component: name, operation (role), value and state.

Set the Description

Use the description to convey additional details of non-standard operation.

1. Set the Accessible Description through the Accessibility Panel by setting the Description property. Or set it in ActionScript through the accessibilityProperties.descriptionproperty followed by a call to Accessibility.updateProperties().

2. Include only supplementary information; the Accessible Name must provide all required information whether or not the Accessible Description is read by a screen reader.

3. The description property is a good place to put information such as special instructions on how to interact with the component or control or an extended description that may be useful in understanding the component or control the first time it is used.

Set the Shortcut

When a field requires a shortcut or one is provided for convenience, set the accessibility shortcut property to the keystroke that activates the event listener associated with the shortcut keystroke. Refer to the lesson on Ensuring Keyboard Accessibility for how to set keyboard shortcuts.

Set the Value, Role or State

* It is not possible to directly change the default value, role or state of accessible objects through the Accessibility Panel or through ActionScript. To directly set these values, create a custom accessibility implementation and associate it with the object or class from which the object was created. Otherwise, a screen reader will announce the role of the base component class such as buttons instead of "slider" in the case of a custom slider control based on the button component. The topic of creating accessibility implementations is not covered by this course.

* For simple simulated controls that contain custom values or roles, it may be possible to add text equivalent information for these items into the accessible name of the object. For example, a custom CheckBox that has a role of button may have an accessible name of "Do you want emails: checkbox checked." In this example, the role and state of checked is appended to the Accessible Name of the object. This is not the preferred approach, but you can use it in limited situations to provide some level of accessibility.

Checking Your Work

The VHA Section 508 checkpoints that specify accessibility requirements for revealing a control's name, description, role, state or value through MSAA are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Name, Role, Description, State, Value

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(d)

Sufficient information about a user interface element including the identity, operation and state of the element shall be available to assistive technology. When an image represents a program element, the information conveyed by the image must also be available in text.

d.1

Do user interface elements including custom controls and informative graphics provide a textual name, description, role, state, and value (where applicable)?

* *
*

d.5

Are the proper events fired to provide context changes?

* *
*

(I)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

I.3

Are meaningful accessible names provided for all form elements?

* *
*

I.10

Are proper programmatic events fired to provide access to context changes?

* *
*

Using a Screen Reader

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. Navigate to each control.
 - a. Verify that an accessible name is announced.
 - b. Verify that the accessible name includes all relevant information such as
 - i. All labels that apply to the control
 - ii. Any group label if it is necessary to understand the purpose of the controls in context with other controls
 - c. Verify that any role, value and state information is announced either
 - i. with the accessible name (for simulated/custom controls)
- or
- ii. after the accessible name (for standard components, buttons, text)
- d. Verify that a description is announced if the control requires special instructions relevant to its use.
4. Determine if the name or state changes based on user action.
 - a. If so, perform the action and check the name and state again by shifting focus away and back to the control or by pressing Insert+Tab.

Using the Object Inspector

1. Launch the Object Inspector.
2. Open the Flash application.
3. Navigate to each control.
 - a. Verify that an accessible name is displayed for the current control shown in the Object Inspector.
 - b. Verify that the accessible name includes all relevant information such as
 - i. All labels that apply to the control
 - ii. Any group label if it is necessary to understand the purpose of the controls in context with other controls
 - c. Verify that any role, value and state information is shown either
 - i. with the accessible name (for custom/simulated controls)
- or
- ii. as the accessible role, value or state for standard components, buttons, text, etc.

- d. Verify that a description is shown in the Object Inspector window if the control requires special instructions relevant to its use.
4. Determine if the name or state changes based on user action.
 - a. If so, perform the action and check the name and state again by shifting focus away and back to the control.

How to Indicate Form Field Constraints and Feedback

Form field constraints include any additional information on the screen about the reply expected. Constraints include a maximum number of characters, a required format (two-letter state abbreviation, no hyphens, mm/dd/yy), the designation of an item as required and other notices of how to respond. Feedback refers to additional information added beside a button, RadioButton or CheckBox after a response is made: correct, incorrect, excessive (too many checkboxes selected), poll percentages displayed beside each option and similar text or graphic markings.

Form Field Constraints

Put all constraints in the form field's Accessible Name. For example, if a required field has a name of "Employee ID:", set the accessible name to "*Employee ID:" or "Employee ID (required):". Set the accessible name through the Accessibility Panel or through the `ActionScript.accessibilityProperties.name` property. Remember to call `Accessibility.updateProperties()` after setting the accessible name.

Form Control Feedback

Update the Accessible Name of the button, RadioButton or CheckBox to include any feedback. For example, if the RadioButton labeled "B. Humerus" gets marked with an icon that represents a correct answer, change its accessible name to "Correct answer: B. Humerus." This must be done through the `ActionScript.accessibilityProperties.name` property, followed by a call to `Accessibility.updateProperties()`. As with error messages, it is recommended to focus the field with feedback after a form has been submitted. For example, if the user submits a quiz question and correct or incorrect feedback is provided, it will be helpful to users of assistive technology such as screen readers and screen magnifiers if the field containing the feedback is focused. This will cause the screen reader to read the field name with the feedback and will move the magnified area to the field of users with low vision. Focus should be set using the method discussed in the Maintaining Focus lesson.

```
rbQuestion2B.accessibilityProperties.name =
"Correct answer: B. Humerus.";
```

```
// set focus to RadioButton component
rbQuestion2B.setFocus();
```

Checking Your Work

The VHA Section 508 checkpoints that specify accessibility requirements for constraints and feedback are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Form Field Constraints and Indicators

§1194.21 Software Applications and Operating Systems

1194.21

Checkpoint

Yes

No

N/A

Comments

(I)

When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

1.3

Are meaningful accessible names provided for all form elements?

* *

*

1.4

Do elements with multiple labels expose these in their accessible names?

* *

*

1.7

Are form field constraints clearly indicated?

* *

*

Using a Screen Reader

1. Activate a screen
2. reader, such as JAWS or Window-Eyes, and a compatible browser.
3. Activate the Flash application.
4. Determine if any fields contain constraints.
 - a. If so, navigate to those fields and verify that the announced accessible name contains the constraint information.
5. Determine if any feedback is provided after perform an action such as submitting a form or answering a question.
 - a. If so, verify that the field containing feedback is focused and its name, role, value and state contents are automatically announced.
 - b. Verify that the feedback that is provided is also announced as part of the accessibility identity information announced by the screen reader.

Using Object Inspector

1. Launch Object Inspector.
2. Open the Flash application.
3. Determine if any fields contain constraints.
 - a. If so, navigate to those fields and verify that the accessible name is shown and contains the constraint information.
4. Determine if any feedback is provided after performing an action such as submitting a form or answering a question.
 - a. If so, verify that the field containing feedback is focused and its name, role, value and state contents are shown in the Object Inspector.
 - b. Verify that the feedback provided is also shown as part of the accessibility properties shown in the Object Inspector.

How to Indicate Errors

Errors are a special form of feedback with their own requirements.

Indicate Within-Control Errors both Visually and with Sound

When an error occurs as a user interacts with a control, such as typing more than the maximum number of

characters, Section 508 does not require any immediate feedback. However, if you provide visual or audio feedback, you must provide both. See the sample ActionScript code in How to Add Audio Equivalents to a User Interaction in the Using Audio for Visual Information lesson.

Make Form-Submission Error Information Accessible

If a form is submitted and not valid due to errors:

- * Indicate at the top of the form or in a popup window that an error occurred and the form was not accepted.

- * When known, indicate the error(s) one or both ways:

- o List the elements with errors at the top of the form.

- o Mark each control with an error using both a visual indicator (not just color) and an Accessible Name change that indicates both error status and the nature of the error.

Add an Error Message to the Tab Order at the Top of the Screen

The preferred way to display a form-submission error message is to add it to the top of the screen, put it in the tab order, and move the focus there.

```
// create the text
```

```
var txtErrorText: TextField = new TextField();
```

```
txtErrorText.text = "Error: Please complete the first name field.";
```

```
// wrap in a dummy MovieClip for focus control
```

```
var mcErrorText: MovieClip = new MovieClip();
```

```
addChild(mcErrorText);
```

```
mcErrorText.addChild(txtErrorText);
```

```
mcErrorText.tabEnabled = true;
```

```
// add an accessible name to the MovieClip
```

```
var accObj: AccessibilityProperties =
```

```
new AccessibilityProperties();
```

```
accObj.name = txtErrorText.text;
```

```
accObj.silent = true;
```

```
mcErrorText.accessibilityProperties = accObj;
```

```
Accessibility.updateProperties();
```

```
// on error set focus
```

```
submitButton.addEventListener(MouseEvent.CLICK,
onFormValidation);
```

```
function handleFormValidation(e: Event): void
```

```
{ // can't focus a TextField, so focus the
```

```
// dummy MovieClip
```

```
mcErrorText.accessibilityProperties.silent = false;
```

```
mcErrorText.tabIndex = 3;
```

```
stage.focus = mcErrorText;
```

```
...
```

```
} Add an Error Message to the Accessible Name of Every Control with an Error
```

Adding the error information to the control's Accessible Name makes it much easier to locate and fix errors with a screen reader.

```
// use MovieClip mcErrorText defined in prior example
```

```
// on error set focus
```

```
submitButton.addEventListener(MouseEvent.CLICK,
onFormValidation);
```

```
function handleFormValidation(e: Event): void
```

```
{ // can't focus a TextField, so focus dummy MovieClip
```

```
mcErrorText.accessibilityProperties.silent = false;
```

```
mcErrorText.tabIndex = 3;
```

```
stage.focus = mcErrorText;  
txtSSN.accessibilityProperties.name =  
"Has Error: SSN: (must be 9 characters without dashes)";  
Accessibility.updateProperties();  
} Checking Your Work
```

The VHA Section 508 checkpoints that specify accessibility requirements for indicating and preventing errors in forms are available in a separate window by selecting the link that follows:

VHA Section 508 Checkpoints: Error Handling

§1194.21 Software Applications and Operating Systems

1194.21
Checkpoint
Yes
No
N/A
Comments
(l)
When electronic forms are used, the form shall allow people using assistive technology to access the information, field elements, and functionality required for completion and submission of the form, including all directions and cues.

I.8
Are error messages explicitly indicated and easily found by a user?
* *
*

VHA Section 508 Checkpoints: Error Handling

§1194.31 Functional Performance Criteria

1194.31
Checkpoint
Yes
No
N/A
Comments
(a)
At least one mode of operation and information retrieval that does not require user vision shall be provided, or support for assistive technology used by people who are blind or visually impaired shall be provided.

a.5
Are error and alert mechanisms used consistently?
* *
*

a.12
Is error prevention provided for legal commitments and financial (and medical) data?
* *
*

a.16
Is a nonvisual indication provided to inform users when input text overflows a field?

* *

*

a.19

When an element's state changes, does related alternative text update accordingly?

* *

*

Using a Screen Reader

1. Activate a screen reader, such as JAWS or Window-Eyes, and a compatible browser.
2. Activate the Flash application.
3. Determine if the application contains a form that has error validation.
 - a. If so, trigger an error to appear.
 - b. Verify that an error message appears at the top of the form.
 - i. Navigate to the error message at the top of the form using arrow keys (up or down) with the virtual cursor and verify that the screen reader announces this text.
4. Determine if specific field error text can be obtained.
 - a. If so, verify that this field specific error text appears either in the error message at the top of the form or inline, associated with each field.
 - i. Navigate to the error message at the top of the form using arrow keys (up or down) with the virtual cursor and verify that the screen reader announces this field specific text.
 - ii. Navigate to the field with the error and verify the error text is announced when the field is encountered (for example when tabbing to the field).
5. Determine if the form contains an error message that is in the tab order and focused when the error occurs (recommended but not required).
 - a. Enter content that will trigger the error and submit the form.
 - b. Verify that focus shifts to the error message at the top of the form and the screen reader announces the error message.
 - c. Move focus away from the error message: press Tab or Shift+Tab.
 - d. Move focus back to the error message: press Shift+Tab or Tab, the opposite of the keystroke pressed in the prior step.
 - e. Verify that focus is on the error message and the screen reader announces the error message.

Using the Object Inspector

1. Launch Object Inspector.
2. Open the Flash application.
3. Determine if the application contains a form that has error validation.
 - a. If so, trigger an error to appear.
 - b. Verify that an error message appears at the top of the form.
 - i. Navigate to the error message at the top of the form using the mouse or the toolbar commands in Object Inspector and verify that the Object Inspector shows this text.
4. Determine if specific field error text can be obtained.
 - a. If so, verify that this field specific error text appears either in the error message at the top of the form or inline, associated with each field.
 - i. Navigate to the error message at the top of the form using the mouse or the toolbar commands in Object Inspector and verify that the Object Inspector shows this field specific text.
 - ii. Navigate to the field with the error and verify the error text is displayed in the Object Inspector when the field is encountered (for example when tabbing to the field).
5. Determine if the form contains an error message that is in the tab order and focused when the error occurs (recommended but not required).
 - a. Enter content that will trigger the error and submit the form.
 - b. Verify that focus shifts to the error message at the top of the form and that the Object Inspector shows the error message as the accessible name.
 - c. Move focus away from the error message: press Tab or Shift+Tab.
 - d. Move focus back to the error message: press Shift+Tab or Tab, the opposite of the keystroke pressed in the prior step.

- e. Verify that focus is on the error message and the Object Inspector shows the error message as the accessible name.

Technical Knowledge Checks

Check your knowledge of the procedures for providing accessible user interface controls in Flash. Use the Answer button to check your selection.

Top of Form

Which of the following is not an appropriate method to indicate error messages?

- A. A pop-up window indicating an error has occurred and listing all of the fields with errors
- B. An error message at the top of the form that reads: One or more errors found; fields with errors show "error" next to them
- C. Changing the labels of all fields with errors to red to indicate an error has occurred
- D. A tab-able error message at the top of the form indicating that an error has occurred and listing all of the fields with errors

Bottom of Form

Top of Form

ActionScript is the only way to enable accessibility for which of these controls?

- A. RadioButtons
- B. Form submit buttons
- C. TextInput fields
- D. Error messages

Bottom of Form

Top of Form

RadioButtons work as a set you can navigate with the arrow keys when:

- A. Their Accessible Names all begin with the same group name
- B. Their Component Inspector groupName values match
- C. Their Component Inspector labelPlacement values = right
- D. Their roles are all set to "RadioButton"

Bottom of Form

You have completed the course. Congratulations!

If you are developing e-learning products for VA or products to be used in VHA web communications, and you have questions — in any stage from planning through evaluation — about how to make your software applications or operating system Section 508 compliant, feel free to contact our VHA OHI Section 508 Office for assistance.

Our accessibility experts are available to answer your questions about assistive technology and to work with you to help you make accessible products.