Criterion Solution Provided

Section 508 Compliance First Pass Audit of Red Hat JON3 Web Application

Prepared by:

Criterion 508 Solutions, Inc.
5012 Prairie Place, Suite 100
Johnston, Iowa 50131
Toll: (888) 508-EXPERTS (3973)
Email: Info@Criterion508.com
Visit: www.Criterion508.com

Date: 12/20/2011 9:47 AM

Table of Contents

Executive Summary

The purpose of this document is to evaluate the **Red Hat JON3 Web Application** for Section 508 compliance.

JAWS v12 Overview

Internet Explorer v 8
This JAWS user found the Dashboard to be very inaccessible. I was able to log in and follow the steps through Step 6 of the instruction sheet. At this point I could not activate the “alert”. I tried numerous JAWS keystrokes, enter, control + enter and the keyboard mouse click. No action could be taken. No buttons were presented. Using the graphics list box, control + insert + g and reviewing the list no “new” graphic could be found. Using other JAWS features such as insert + F5 to open the forms list box and insert + F7 to open the links list box, no buttons or appropriate links could be found.

ZoomText 9.1 Overview

ZoomText is an assistive technology tool used by low-vision individuals. While this program is mainly used for its screen magnification purposes, below I have listed software related issues while using the speech features in ZoomText.

Note: Current released version of ZoomText tested (10.0.344)

General:

- Visual tracking using magnification worked flawlessly in IE8, FireFox did exhibit problems tracking buttons.

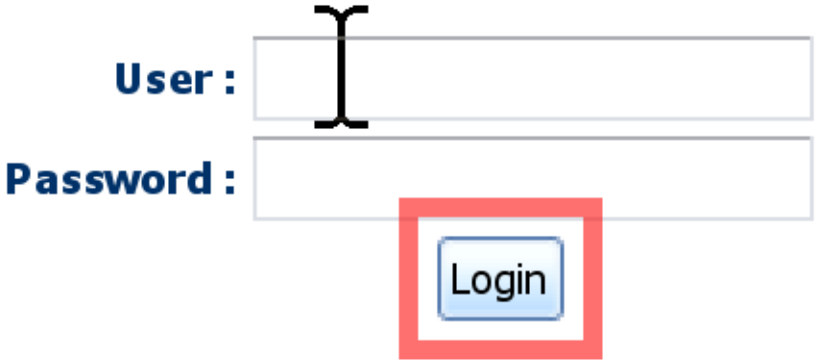
Recommendation: None

- When screen information changes, ZoomText will repeat browser window title. Example, clicking the “New” button will cause ZoomText to read the IE window title or URL

Recommendation: There is likely a focus event triggering this to happen, and unless the web application can suppress it, there may be no way around this.

Login Screen:

- Login button is read as “Group box” even though its table label is correct as “Login”



Recommendation: This is a ZoomText bug and not representative of a problem with the page itself.

Dashboard / Inventory Screens:

- Most Buttons are read as “Group box”, even though their Table labels are the appropriate text

  Recommendation: This is a ZoomText bug and not representative of a problem with the page itself.

- Using reading tools results in MANY images being read as “Graphic” as well as tables being read as “Graphic” almost repeatedly.

  Recommendation: Provide meaningful descriptions of all graphic or images on the page, as well as eliminate redundant tables that are unnamed or labeled “graphic”

Conclusion

The Red Hat JON3 Web Application has been audited and found to be **Not Compliant** with Section 508: Subpart B — Technical Standards: §1194.22.

Section 508 Compliant Overview

Refer to the table below to find provisions in need of repair in order to bring the web application into compliance with Section 508: Subpart B — Technical Standards: § 1194.22.

Please carefully review items highlighted below.

Section 508 Criterion	Compliant
(A) Text	**NO**
(B) Multimedia Presentations	**NA**
(C) Color	**NO**
(D) Readability	**NO**
(E) Server-Side Image Maps	**NA**
(F) Client-Side Image Maps	**NA**
(G) Data Table Headers	**NA**
(H) Data Table Header Associations	**NA**
(I) Frames	**NO**
(J) Flicker Rate	**NA**
(K) Text-Only Alternative	**NA**
(L) Scripts	**NO**
(M) Applets and Plug-Ins	**NA**
(N) Electronic Forms	**NO**
(O) Navigation Links	**NO**
(P) Time Delays	**YES**

How to Use This Document

Document Usage Summary

The sections which follow give further details to help the developer understand the provisions of Section 508: Subpart B—Technical Standards: § 1194.22 and how this site meets or does not meet those provisions.

Each "**Provision**" has been listed along with its "Plain English" interpretation for clarity.

"**Why this is important**" is intended to help the developer understand the accessibility issues as this provision relates to users with disabilities.

"**Section 508 Compliant**" indicates whether the requirements of the provision have been met, or, if not, it details the violations.

"**What must be fixed**" is the minimum required remediation in order to achieve compliance with Section 508.

"**Examples/Screen Captures**" have been used liberally throughout to illustrate a point, or to help the developer identify a problem within context.

"**Recommendations/Repairs**" are given to assist the developer in remediating the violation. This is intended to start the developer in the right direction.

"**How to Test**" instructs the developer in the testing that needs to be performed in order to verify compliance. It is expected that the developer will ensure that all tests provided are passed prior to resubmitting the application for a second audit.

"Usability Issues" are offered as additional **suggested** improvements to make your application more accessible to users with and without disabilities.

Section 508: Subpart B § 1194.22

Web-Based Internet Information and Applications

§ 1194.22 (a) Text Equivalents

Provision

A text equivalent for every non-text element shall be provided (e.g., via "alt", "LongDesc", or in element content).

Plain English

Non-text elements and graphics used to convey functionality and/or information to the user must be accompanied by an appropriate `alt` or `longdesc` attribute containing a meaningful description of the element and/or its purpose.

The element description should focus on the purpose of the element and not the appearance of the element itself.

Where an image is used for appearance of design-related purposes only (in the case of padding or positioning), such images should be labeled with a blank `alt` attribute.

Why this is important.

People using a screen reader, or other user agents such as a text based console, need to have meaningful alternative text assigned to non-text elements so that they can distinguish between important functional elements and visual elements that are simply contributing to the appearance of the page.

Section 508 Compliant?

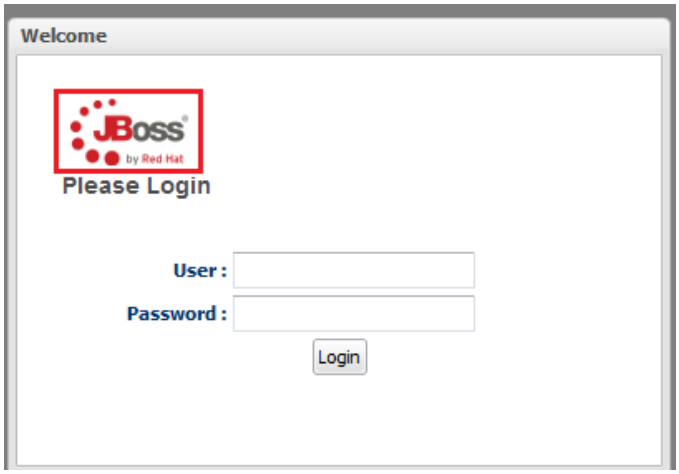
Not Compliant

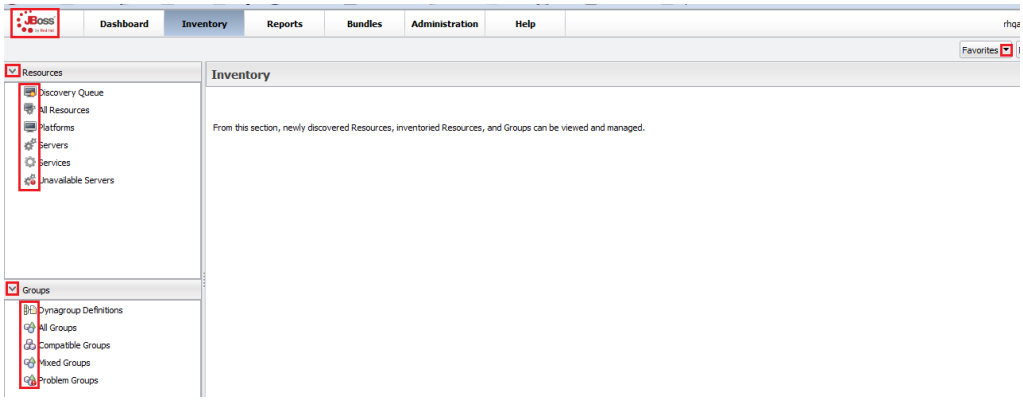
What must be fixed?

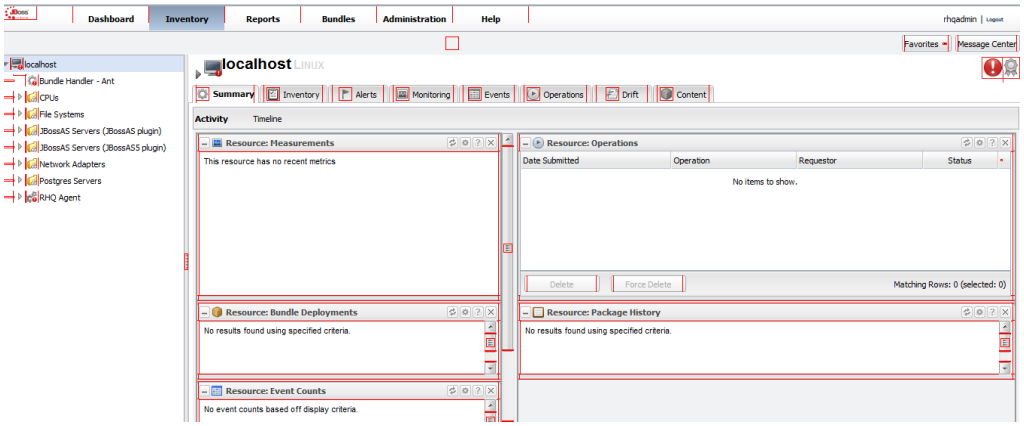
- The website contains non-text elements and graphics used to convey functionality and/or information to the user; these graphics do not contain a meaningful alternate text description.

Examples/Screen Captures

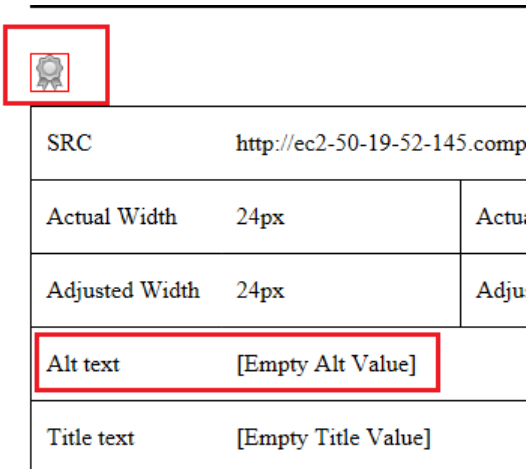
The three screenshots below show graphics (outlined in red) that have Empty Alt Text Values. Users of assistive technology, such as screen readers, rely on the alt text value given to an image to discern that image's value, meaning or role on the page. It is therefore important that an image's alt text strives to provide an equivalent value as would be experienced by a user who could directly perceive the image. These images convey meaning to a user and should be assigned an alternative text value.

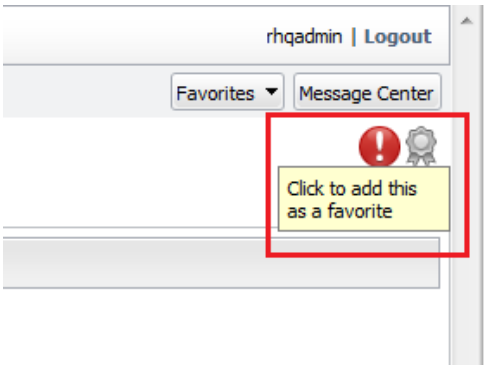


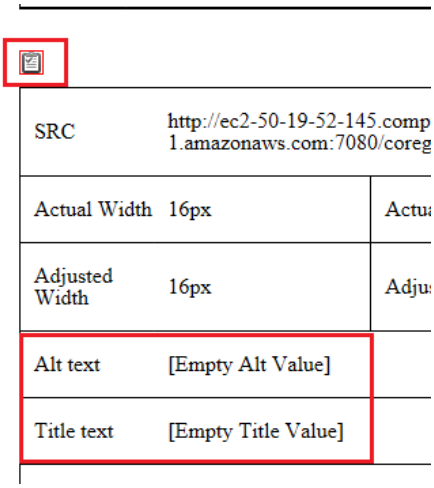




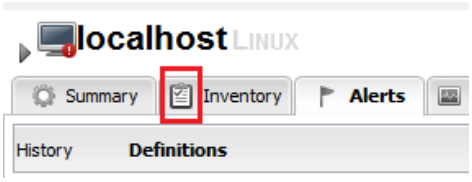
The six screenshots below, show graphics (outlined) that have Empty Alt Values. Users of assistive technology, such as screen readers, rely on the alt text value given to an image to discern that image's value, meaning or role on the page. It is therefore important that an image's alt text strives to provide an equivalent value as would be experienced by a user who could directly perceive the image. These images convey meaning to a user and should be assigned an alternative text value.

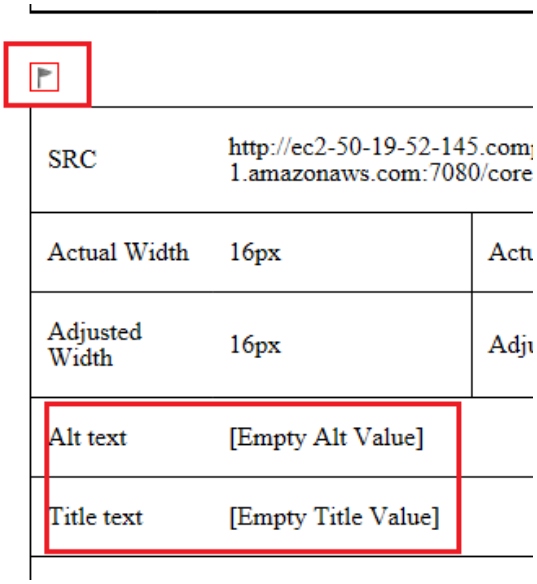




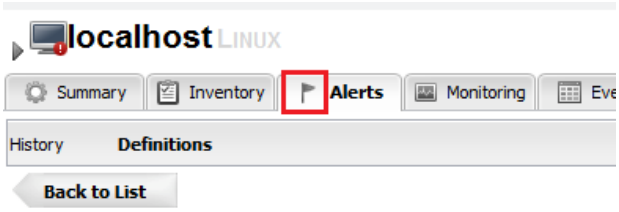


SRC	http://ec2-50-19-52-145.comp 1.amazonaws.com:7080/coreg	
Actual Width	16px	Actu
Adjusted Width	16px	Adju
Alt text	[Empty Alt Value]	
Title text	[Empty Title Value]	





SRC	http://ec2-50-19-52-145.com 1.amazonaws.com:7080/core	
Actual Width	16px	Actu
Adjusted Width	16px	Adju
Alt text	[Empty Alt Value]	
Title text	[Empty Title Value]	



JAWS User 1 Comments

What must be fixed?

All functional elements must have a meaningful description of their purpose.

Examples

Beginning with the Login page:
http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/
many graphics are present that have no meaningful description of their purpose.

Steps To Verify With JAWS:

1. Open Internet Explorer and proceed to the Red Hat logon page:

http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/

2. Move to the top of the screen with the key combination control + home.
3. Begin reviewing the page with the down arrow. The first graphic encountered is labeled 'images/blank'.
4. Press the "g" key to move to the next graphic which is labeled 'header/rhq_logo_40px'.
5. Press the key combination control + insert + g to open the graphics list box. Review the list with the down arrow. Notice that most graphics listed give the user no clear meaning of their purpose. The JAWS screen reader user cannot take advantage of these elements when no description is offered to allow the user to know what action will be taken if activated.

Another example: Dashboards page - this page has buttons and graphics with no meaningful textual description.
http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/#Dashboards

Steps To Verify With JAWS:

1. After logging on the Dashboards page is displayed. Move to the top of the page with the key combination control + home.
2. Open the buttons list box with the key combination control + insert + b. Review the list to find buttons with only a number as their label. No meaningful description accompanies the label.
3. Press escape to close the buttons list box.
4. Press the key combination control + insert + g to open the graphics list box. JAWS announces 'header/rhq_logo_28px 1 of 155'.

5. Review the list with the down arrow. Most of the graphics listed have no clear meaning of their purpose.

Recommendations/Repairs

- Identify all non-text elements (such as image tags) and ensure that appropriate alt or longdesc attributes are associated with the element describing the purpose and function (if applicable). Cosmetic and design related images should be assigned explicitly blank alt attributes: <img src=”spacer.gif” alt=””>

- Provide brief, functional descriptions for graphics used as links. Wordy descriptions for graphical links slow down screen reader users. Alt=”Home” is preferable to Alt=”image of a house” when the graphic is part of a link.

- Screen readers identify graphics and graphical links:
  Graphic Company Logo
  Link Graphic Home
  Image map Link Search
  Wording such as “image of” or picture of” within a text equivalent is repetitive and slows down screen reader users.

How to Test

The ability to efficiently locate accessibility issues for this provision depends on familiarity with the code in question. While the methods described below can help identify and resolve the issues, experience with the code itself is one of the most valuable assets in identifying and addressing relevant accessibility issues.

- Examine the source code.
  - Review rendered page content in a browser. Pay particular attention to embedded and linked content and the information available in a non-visual way.
  - Do audio or video objects have a transcript? Are they fully described in surrounding content.
  - View the content with the images turned off. Alt Text should be used in combination with the LongDesc if the image requires a more detailed description.
  - Images with null Alt Text should have no significant meaning toward the understanding of the page or its contents.
  - Use a non-graphical browser such as Lynx or eLynx. Can you understand the content of the linked or embedded content without seeing them?

- Use Firefox with the Web Developers toolbar from http://chrispederick.com/work/webdeveloper/. Select Tools>Web Developer> Images> Display Alt Attributes.

§ 1194.22 (b) Multimedia Presentations

Provision

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

Plain English

Presentations made using a multimedia technology need to provide an accessible equivalent alternative. For example, closed captions for a video presentation must be synchronized with the presentation.

Why this is important.

A multimedia presentation involving an on-screen speaker, for example, might have an associated text caption for hearing-impaired users. In order that lip movement, gestures, and expressions can be associated with what is presently being said, the displayed text must be synchronized with the presentation

Section 508 Compliant?

Not Applicable

§ 1194.22 (c) Color

Provision

Web pages shall be designed so that all information conveyed with color is also available without color.

Plain English

- Color may be used throughout a site for functional purposes but it cannot be the only means of conveying that function.
- The functional meaning associated with the color must be shown via a different method that does not depend on the use of color.

Why this is important.

People who are blind, color-blind, and/or color-deficient are unable to distinguish color-based visual cues such as those used to portray status or function.

Section 508 Compliant?

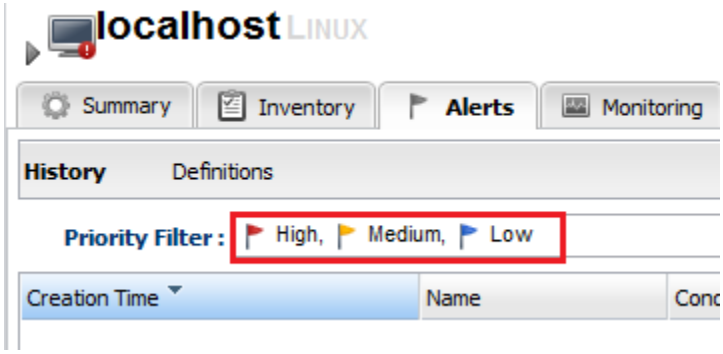
Not Compliant

What must be fixed?

- Color is used as a primary means of communicating meaningful information to the user, alternate means of providing this information must be provided.

Examples/Screen Captures

The screenshot below shows a color scheme used to inform users of priority. Color is being used as a primary means to identify important information. The functional meaning associated with the color must be shown via a different method that does not depend on the use of color. People who are blind, color-blind, and/or color-deficient are unable to distinguish color-based visual cues such as those used to portray status or function.



Recommendations/Repairs

- Provide other means to identify functionality.
- For required form controls, include a textual indication within the label element associated with each required control.
- Font attributes such as bold or italics, when used as the sole means of indicating functionality, create the same barrier as use of color alone to indicate functionality.

How to Test

- Set your browser to ignore color and font styles. Internet Explorer> Tools> Internet Options> General> Accessibility. Check these check boxes: Ignore colors specified on Web pages, and Ignore font styles specified on Web pages. Firefox with the Web Developers toolbar from http://chrispederick.com/work/webdeveloper/: Select Tools>Web Developer> Disable> Disable Page Colors. (Only colors can be tested in this way with Firefox.)
- Are there parts of the content that do not respect the end user colors? Are these elements necessarily confined to their current colors? How might these affect end user access [Red/Green color blindness or poor color contrast for example].
- Have icons or other images been used to denote meaning? Can these icons be distinguished from one another by their appearance without the use of color?

§ 1194.22 (d) Readability

Provision

Documents shall be organized so they are readable without requiring an associated style sheet.

Plain English

Documents must have content logically organized and be prepared in a manner which can be read without the use of an associated style sheet, either as an included separate document or by explicit definition in the document body.

Why this is important.

Some user agents such as screen readers or text only browsers do not parse style sheets. Users may also opt to negate provided style sheets in favor of providing their own to be able to change fonts, increase text sizes, improve contrast, or other attributes. Documents must therefore be readable and flow logically without the use of style sheets or style definitions.

Section 508 Compliant?

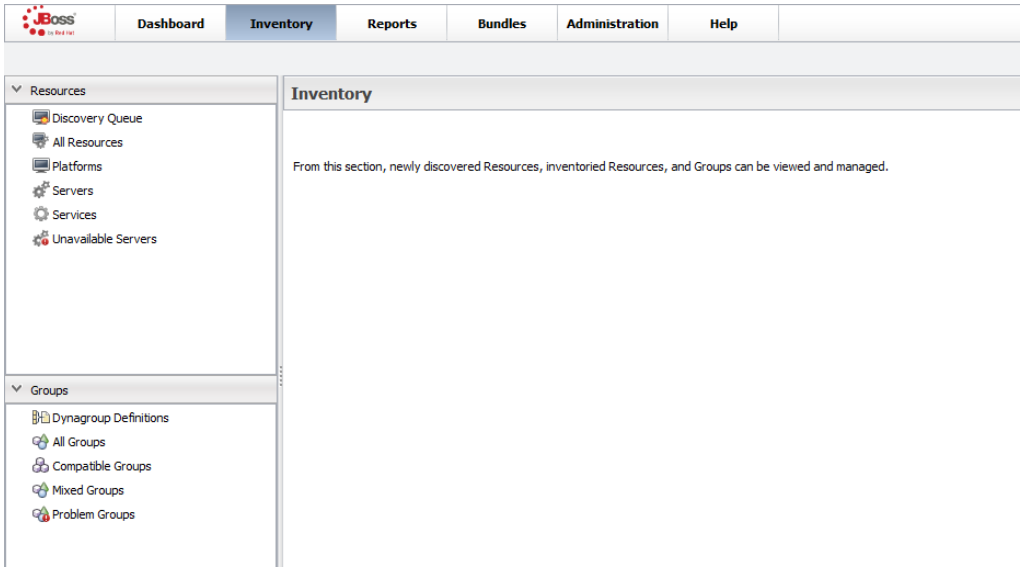
Not Compliant

What must be fixed?

- Documents must be readable and flow logically without the use of style sheets or style definitions.

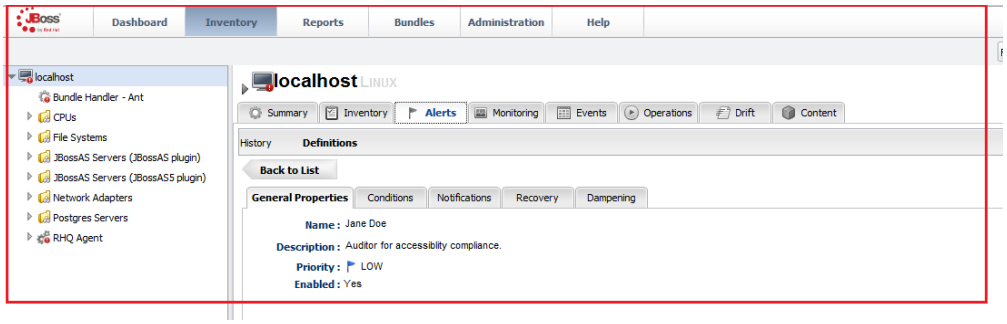
Examples/Screen Captures

The screenshot below illustrates a page as it appears when CSS is enabled.

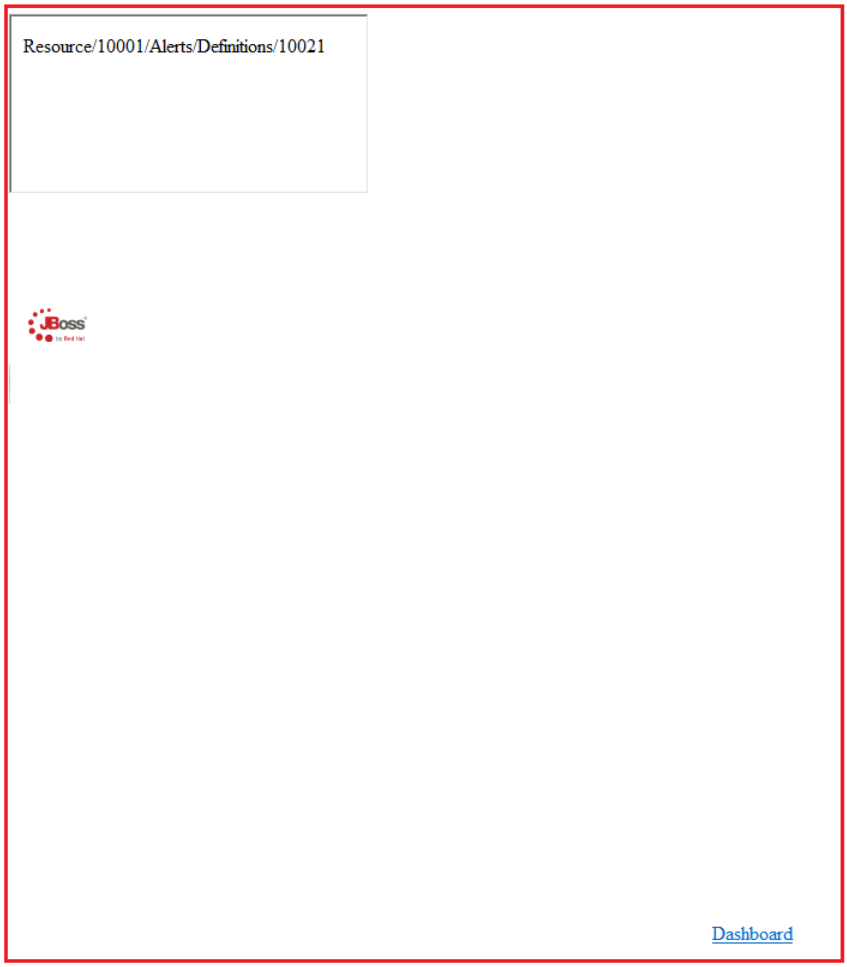


The screenshot below illustrates the same page shown above as it appears when CSS is disabled. Content is now spaced sporadically throughout the page. The flow of the page is severely disrupted.

The screenshot below illustrates a page as it appears when CSS is enabled.



The screenshot below illustrates the same page shown above as it appears when CSS is disabled. Content that was present on the page before is now suddenly not present to the user. This content is only available however, to any user who has disabled styles settings in their user agent (including assistive technology users such as screen readers). This makes it confusing to the user and disrupts the flow of the page.



Recommendations/Repairs

- Do not use style declarations alone to hide content. When style declarations are used to hide content, the content becomes visible again when Style sheets are not in use. Additionally, using CSS to hide content visually often hides it for screen readers too.

- Avoid excessive use of CSS positioning to reorder content. Screen readers navigate page content in the order it is coded. When CSS positioning is over-used, the logical flow of the page designed for the visual user may differ significantly from what the screen reader user experiences, resulting in confusion and increased difficulty using the site.

How to Test

- Use Firefox with the Web Developers toolbar from http://chrispederick.com/work/webdeveloper/. Select Tools>Web Developer> CSS> Disable Styles> All Styles.

- Examine page content with styles turned off:
  - Is everything that should be there present?
  - Is anything that should not be seen by the user suddenly exposed?
  - Is all of the content still legible and easily readable? What if images are replaced with their accessible-text equivalents as well – is the alternate text readable?
  - Is anything cut off from a table, frame, or page because of content repositioning?
  - Has any content become harder or impossible to read because of background colors or images defined without the use of style sheets?
  - Has the ordering of page content altered significantly?

§ 1194.22 (e) Server-Side Image Maps

Provision

Redundant text links shall be provided for each active region of a server-side image map.

Plain English

Web pages that employ server-side image maps (as opposed to client-side) must provide the same functionality as the map through the use of text links.

Why this is important.

Since server-side image maps contain no client-readable information about the available options in the map, they are effectively unusable to many users requiring assistive technologies. Text-only alternative links must therefore be provided for each active region of server-side image maps.

Note: According to provision (f), server-side image maps must only be used if client-side image maps would be impossible or impractical to provide instead.

Section 508 Compliant?

Not Applicable

§ 1194.22 (f) Client-Side Image Maps

Provision

Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

Plain English

Client-side image maps are to be used instead of server-side maps, excepting only occasions where the available geometric shapes of client side image maps would be difficult or impossible to represent the required mappings.

Why this is important

Except in the case of highly complicated image map coordinates (such as a detailed city map) where providing image map information via basic geometric shapes proves either impossible or impractical, a client-side image map must be used instead of a server-side map.

Client-side image maps are preferred because they provide considerably better usability through text alternative attributes in the map itself, particularly for users with vision problems (for whom server-side image maps are difficult or impossible to use).

Section 508 Compliant?

Not Applicable

§ 1194.22 (g) & (h) Data Table Headers & Associations

Provision

(g) Row and column headers shall be identified for data tables.

(h) Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

Plain English

Data tables must be rendered using markup appropriate to each portion of the table.

Each element of a data table must be tagged with the SCOPE, or alternatively the ID and HEADERS, attribute to allow each cell to be identified as to its meaning.

Why this is important

For most users, scanning the headers along the top or side of a table is a trivial reality of reading tables. However, for those with visual disabilities, the table is often not fully displayed on-screen or is not the focus of their efforts to see via their assistive technology. It is therefore essential that each value in a table be explicitly identified by the appropriate markup

Section 508 Compliant?

Not Compliant

JAWS User 1 Comments

What must be fixed?

- Tables must have a summary attribute defining the purpose and content of the table.

Examples

The Login page has a table with no summary attribute.

Steps To Verify With JAWS:

1. Open the Red Hat application:

http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/

2. Move to the top of the page with the key combination control + home.

3. Begin reviewing the page with the down arrow. Notice the announcement ‘table with 2 columns and 3 rows’. With no summary attribute the JAWS user knows she is about to encounter a table but does not know its purpose or content.

Recommendations/Repairs

- Define a summary attribute for tables describing the table's purpose and contents.

- Appropriate markup must be used on each data table to clearly identify column headings and scope for each individual data cell.

- Appropriate markup must be used to associate row data and column headings within individual data cells.

- All table cells containing row or column headers should use the TH element.

- All TH elements in basic tables should include either Scope="col" or scope="row" attributes.
  <th scope="row">Row Header</th>
  <th scope="col">Column Header</th>

- Complex tables – with multiple logical levels – should include ID attributes in each TH element and Headers attributes in each TD element.
  <th ID=r2>Row 2 header</th><td Headers="r2">Cell data</td>

- In tables containing multiple sections, use the Axis attribute to identify the section information.
  <tr><th I="s2" Axis="City">Des Moines</th></tr>
  <tr><td Headers="s2">Appointments</td>…</tr>

How to Test

Provision (g)

- Is the data table element necessary, e.g. can it be replaced by block division elements and style sheets instead?

- Are data tables used for layout purposes unnecessarily nested within one another, or do they contain any needlessly empty cells to create horizontal or vertical spacing between elements?

- Are data tables used inappropriately to render data sets that are better rendered using other structural elements (such as ordered, unordered, or definition lists)?

For table elements used to render appropriate types and amounts of data:

- Use Firefox with the Web Developers toolbar from http://chrispederick.com/work/webdeveloper/. Select Tools>Web Developer> Information> Display Table Information.
    - Is the correct and complete column and/or row header information displayed in each cell that contains data?
    - Is each cell containing row or column header information identified as a header for a column or row?

- Has the appropriate basic appearance of the table itself been included in the markup of the HTML document?

- Has the table been provided an appropriately descriptive summary attribute describing the table's purpose/contents?

- Have tables containing large data sets been assigned appropriate column group and column elements to allow the table to render as it streams down to the user agent?

- Are data table contents divided into appropriate table header, footer, and body sections in the correct order to render properly and accessibly to all users?

- Are data table column and row headers identified using the table heading attribute and assigned the scope of column or row?

Provision (h)

- Visually identify tables with multiple levels of headers by looking at rendered pages.

- Verifying with the Firefox Web Developer toolbar as referenced above, if the table has multiple sections:
    - is each section title identified as containing the correct Axis information?
    - Do data cells within each section contain the correct Axis information?

- Examine page source code for tables with multiple levels of headings. Here is what to look for:
    - Check table structure for table header, footer, and body blocks.

- Verify that column header tags are used within the table header block.
- Examine each table header element—has an ID been assigned to each that is unique to the page?
- In the table body block, perform the same checks as above against row headers.
- Spot check several table cells referring to both the source code and rendered page simultaneously to see whether assigned headers match up with the correct, corresponding headers from the rendered page.

§ 1194.22 (i) Frames

Provision

Frames shall be titled with text that facilitates frame identification and navigation.

Plain English

Frames must be clearly identified so users understand their purpose and contents.

Compliance may be achieved by defining attributes in the tags that generate the frames, or simply by text content in each document visible inside frames.

Why this is important.

Frames are an additional challenge to navigate for users with special needs: They effectively creating multiple separate HTML documents within the same browser window. The purpose or content for each frame must be clearly identified in text for users of assistive technologies to navigate through the frames effectively.

Section 508 Compliant?

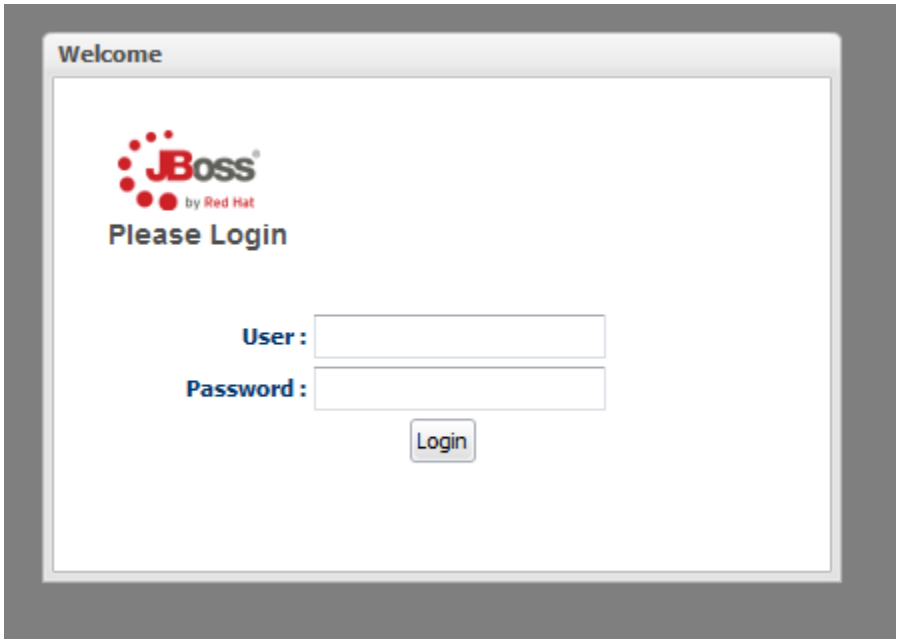
Not Compliant

What must be fixed?

- Frames lack a meaningful and descriptive title that would provide screen reader users with an understanding of each frame’s content.

Examples/Screen Captures

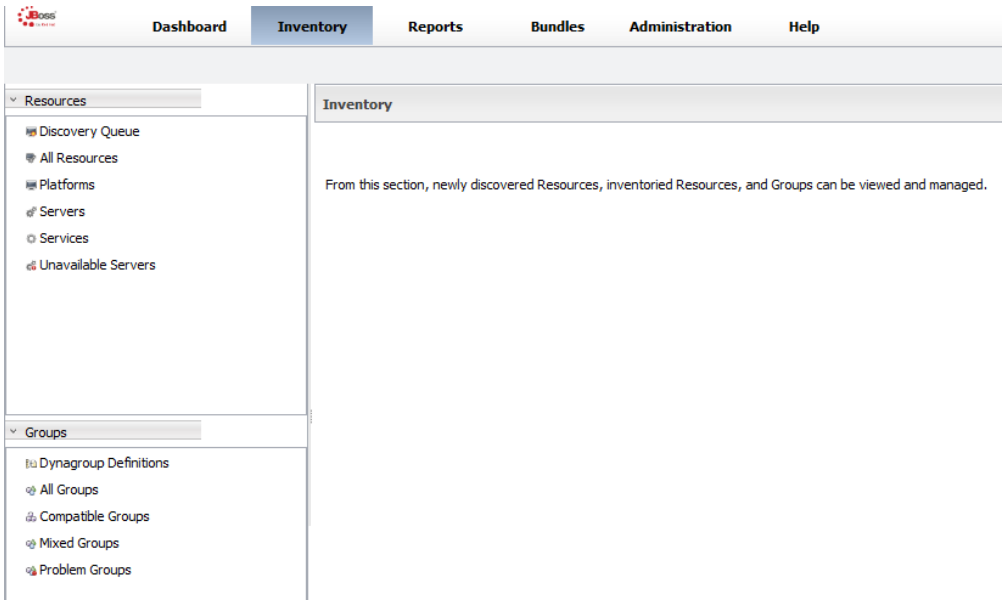
The screenshot below shows a page that contains three iframes. These iframes have not been assigned proper descriptive names or titles or long descriptions. Screen reader technology will attempt to identify and name any iframe found – even when these frames are blank containers. Since the screen reader's technology will first try to identify a frame using the title element, frames and iframes should always be provided appropriate, descriptive titles that allow user of this technology to ascertain the role and function of each frame.

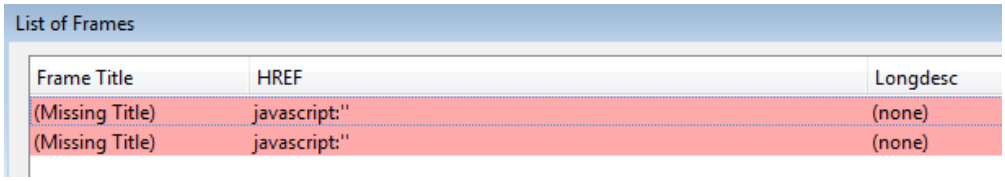


Frame Information

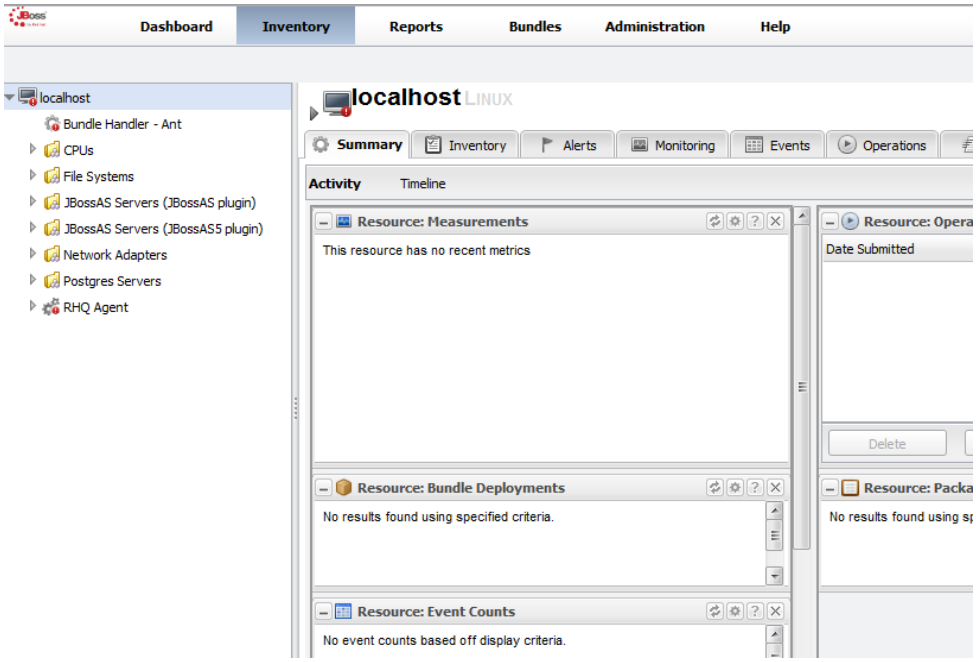
- IFRAME-1
  - Name **not specified**
  - Longdesc **not specified**
  - Title **not specified**
  - Src="javascript:""
- IFRAME-2
  - Name **not specified**
  - Longdesc **not specified**
  - Title **not specified**
  - Src="about:blank"
- IFRAME-3
  - Name **not specified**
  - Longdesc **not specified**
  - Title **not specified**
  - Src="http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/org.rhq.enterprise.gui.coregui.CoreGUI/B230368CEDB26FF3C5C94B61F84887DB.cache.html"

The screenshot below shows a page that contains two iframes. These iframes have not been assigned proper descriptive names or titles or long descriptions. Screen reader technology will attempt to identify and name any iframe found – even when these frames are blank containers. Since the screen reader's technology will first try to identify a frame using the title element, frames and iframes should always be provided appropriate, descriptive titles that allow user of this technology to ascertain the role and function of each frame.

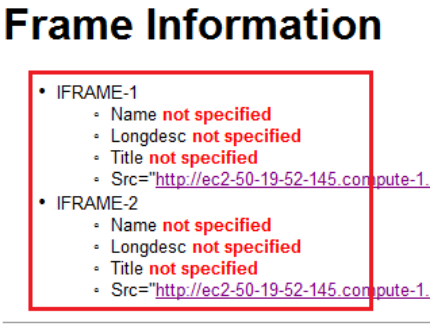




The screenshot below shows a page that contains three iframes. These iframes have not been assigned proper descriptive names or titles or long descriptions. Screen reader technology will attempt to identify and name any iframe found – even when these frames are blank containers. Since the screen reader's technology will first try to identify a frame using the title element, frames and iframes should always be provided appropriate, descriptive titles that allow user of this technology to ascertain the role and function of each frame.

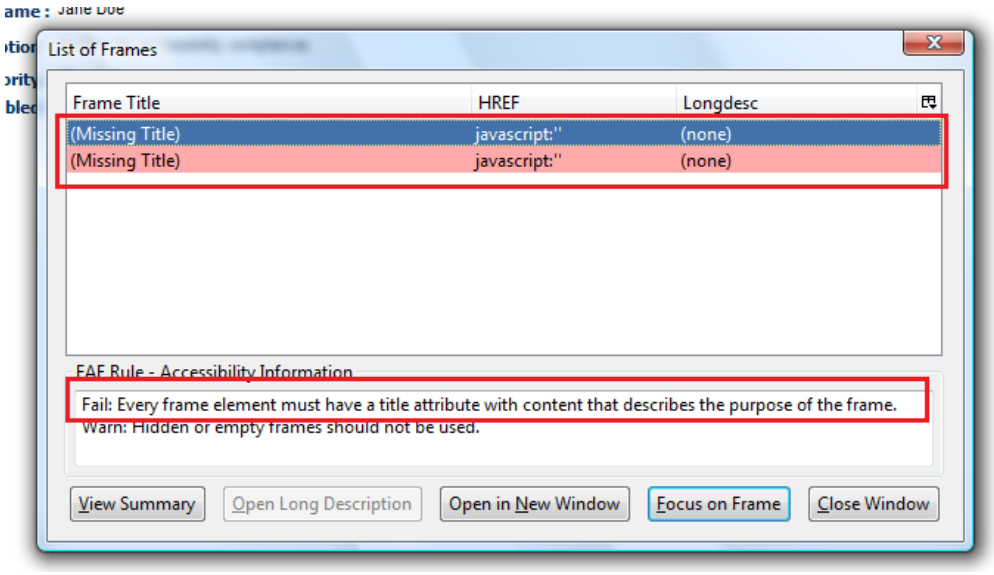


The screenshot below shows the Home page which contains iframes. These iframes have not been assigned descriptive names or titles or long descriptions. Screen reader technology will attempt to identify and name any iframe found – even when these frames are blank containers. Since the screen reader's technology will first try to identify a frame using the title element, frames and iframes should always be provided appropriate, descriptive titles that allow user of this technology to ascertain the role and function of each frame.



The screenshot below shows a page of hidden or empty frames. Frames should be assigned a meaningful title that would allow assistive technology users to understand the meaning or purpose of the frame or aid their navigation.

Fail: Every frame element must have a title attribute with content that describes the purpose of the frame. This is consistent with the website.



JAWS User 1 Comments

What must be fixed?

- Frames must be clearly identified so users understand their purpose and contents.

Examples

The Login page has 3 frames that have no title. The JAWS user was not able to understand the purpose or content of the frames.

Steps To Verify With JAWS:

1. Open the Red Hat application:

http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/

Jaws will announce that the page has 2 frames and no links.

2. Move to the top of the page with the key combination control + home.
3. Press the “m” key to move to the first frame. JAWS announces ‘frame’.
4. Press the ‘m’ key again to move to the next frame. JAWS announces ‘blank frame’.

5. Press the ‘m’ key once more to move to the 3rd frame. JAWS announces ‘frame’. The user has no clear understanding of the purpose or content of the frame. It is also not clear why JAWS announces ‘page has 2 frames and no links’ when the application is first opened since there appear to be 3 frames.

Another example of unlabeled frames - The Dashboard page: http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/#Dashboards

Steps To Verify With JAWS:

1. Open the Red Hat application and log in. The Dashboard page will be displayed.

JAWS does not announce the presence of any frames on this page. 2. Press the key combination insert + F9 to open the frames list box.

3. JAWS announces ‘frames list box 1 of 7’. Review the list with the down arrow. No frames have a label or description of their contents or purpose.

4. Press escape to close the frames list box.

5. Ensure the focus is at the top of the page with the key combination control + home.

6. Press the “m” key repeatedly to move to the frames on the page. When reaching a frame use the down arrow to determine if there is any content within the frame. Most of the time JAWS announces ‘blank frame’ and ‘blank frame end’.

Recommendations/Repairs

- Frames and iframes must be made accessible by providing users with meaningful descriptions of frame content using the name and title attributes.
- Frames require a descriptive title element in their containing document head.
- Avoid the use of frames if possible as they present certain accessibility and usability challenges; favor the use of CSS to create the same effects where possible.
- Avoid blank frame contents; these can be confusing to users. Add basic accessible text to describe the frame's function/purpose to users of assistive technologies.

How to Test

- Examine frame setting documents:
  - Are documents designed to be relatively simple and straightforward, or are they complex or nested frame definitions that may cause usability and accessibility problems?
  - Have readable, meaningful names been assigned to all frames?
  - Have explicit title attributes been assigned to all frames and iframes?
- Examine documents referenced by frame elements:
  - Are all documents called by frames and iframes readable source code?
  - Are document contents clearly and meaningfully described to the user, whether or not they are visibly rendered in the finished design?

§ 1194.22 (j) Flicker Rate

Provision

Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

Plain English

At no time should the whole—or any portion of—the application flash, blink, or visibly strobe on the screen between twice and 55 times per second, whether by intentional or unintentional means.

Why this is important.

Lights or brightly lit objects which strobe within this frequency range can produce seizures in individuals susceptible to certain conditions, such as photosensitive epilepsy.

Section 508 Compliant?

Not Applicable

§ 1194.22 (k) Text-Only Alternative

Provision

A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

Plain English

In the event that a page cannot be made to comply with these provisions using any other means, a text-only version of the same page will be provided at the user's option. This page must contain the same information as its sibling, including updates after initial publication.

"Text-only" in this case refers to the information visible on-screen—meaning graphics and embedded multimedia content are not present. Appropriate structural markup such as lists and tables remain required as the text-only document must comply with all provisions listed in § 1194.22.

Why this is important.

In rare cases where pages cannot be made accessible by other means, a separate presentation of content must be used—a text-only version of the page, with all of the same information provided using a format for which content is assured of being accessible to users with disabilities.

Section 508 Compliant?

Not Applicable

§ 1194.22 (l) Scripts

Provision

When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

Plain English

Pages that employ scripts to affect the display of a page, presence of content, or create or modify interface elements must be identified by accessible text which can be interpreted by assistive technologies such as screen readers.

Why this is important.

Certain scripting functions (commonly JavaScript) used to generate or manipulate user interface elements or content may be incorrectly interpreted or altogether ignored by accessibility technologies; others may result in unexpected or confusing behaviors to people with special needs. In these cases, alternative design practices and alternate functional text must be provided to ensure accessibility to the user.

Section 508 Compliant?

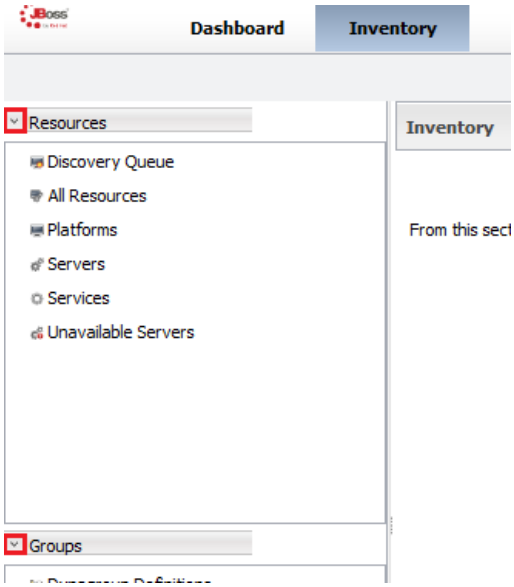
Not Compliant

What must be fixed?

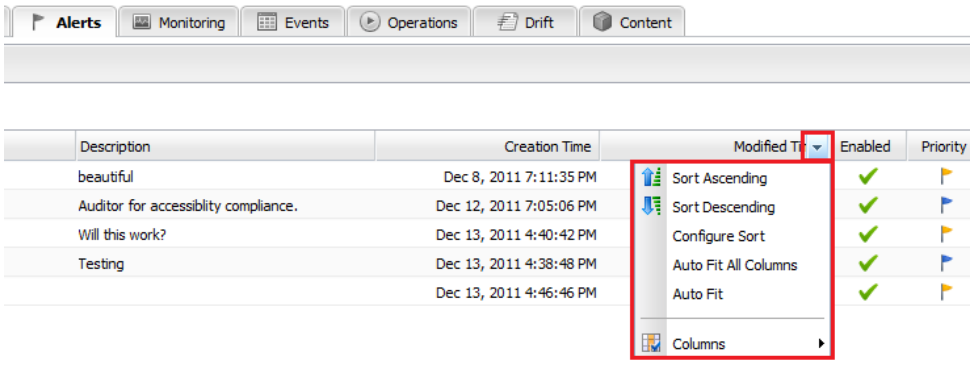
- Some scripted form elements use mouse-only event controllers that are not accessible for keyboard-only and assistive technology users.
- All scripting functions should be tested throughout the entire site.

Examples/Screen Captures

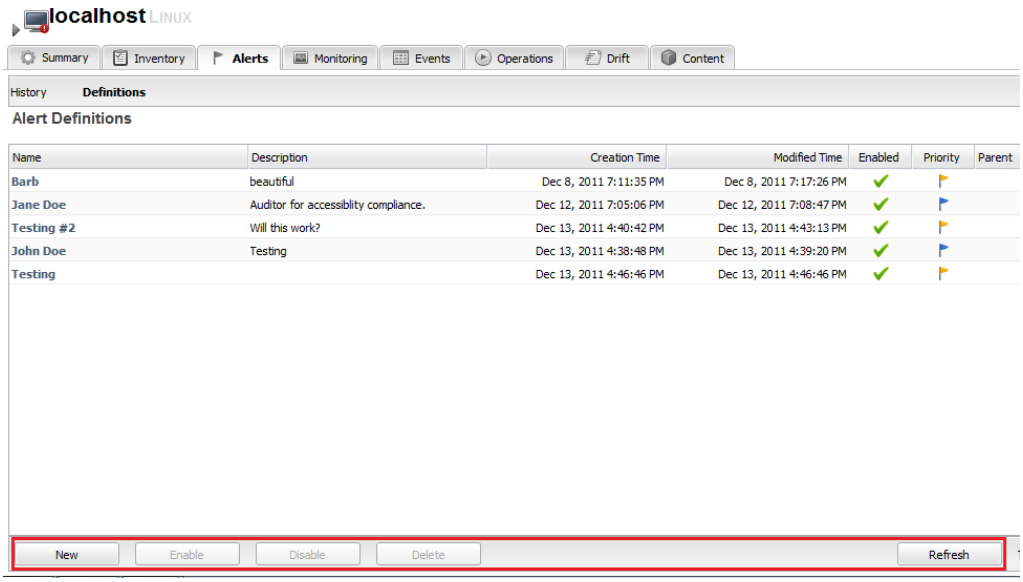
The screenshot below shows expand and collapse buttons (outlined in red) that are not keyboard accessible. The user is unaware of this function until they select the image using the mouse. This function is only accessible via the mouse and not via keyboard functions. Scripts should be made directly accessible whenever possible by employing natively keyboard-accessible markup elements (such as anchors and forms) and generic event handlers (such as onfocus, onblur and onselect).



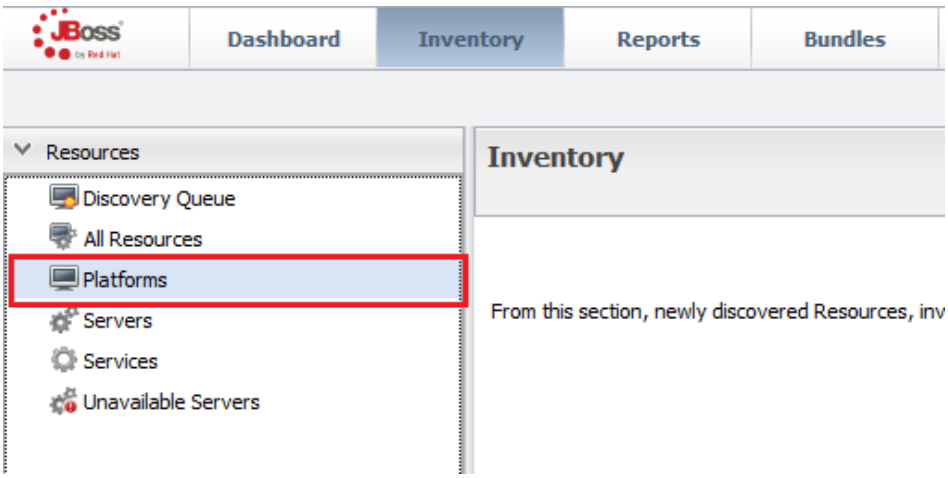
In the screenshot below, the image outlined in red acts as a sorting link. The user is unaware of this function until they select the image using the mouse. This sorting function is on accessible via the mouse and not via keyboard functions. Scripts should be made directly accessible whenever possible by employing natively keyboard-accessible markup elements (such as anchors and forms) and generic event handlers (such as onfocus, onblur and onselect).

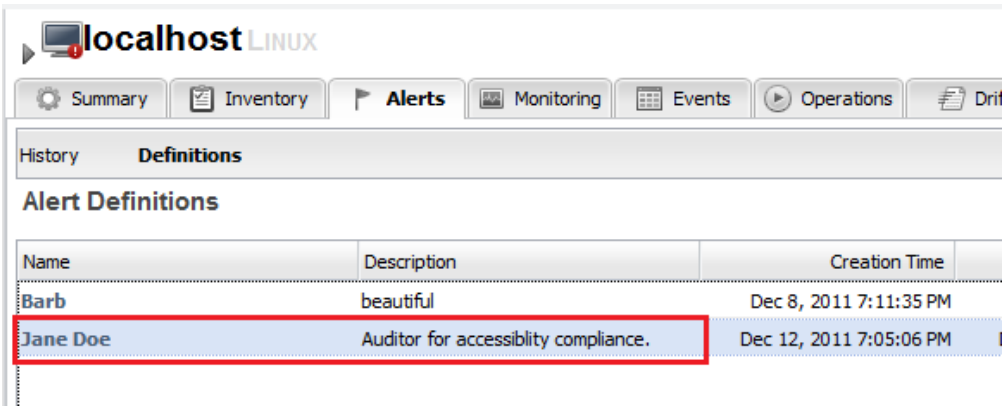


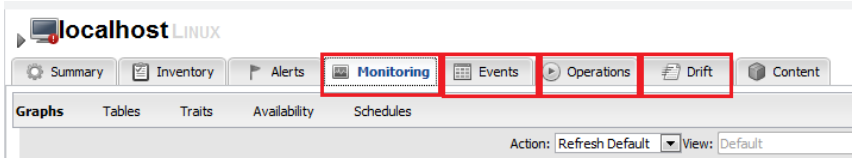
The screenshot below shows a group of navigation links that cannot be accessed using the keyboard. These links use the scripted event handlers onmouseover and onmouseout to open and close these links' submenu items. In this example, the keyboard user can access the main menu item, but their keys cannot open the menu. Without a mouse, the submenu items cannot be seen or accessed. This and similar scripted events should be reviewed to determine if adding other event triggers (such as onfocus and onblur) would make these elements device independent.



The three screenshots below show buttons that work with a mouse but do not work with the keyboard functions. Scripts should be made directly accessible whenever possible by employing natively keyboard-accessible markup elements (such as anchors and forms) and generic event handlers (such as onfocus, onblur and onselect).







Recommendations/Repairs

- Scripts should be made directly accessible whenever possible by employing natively keyboard-accessible markup elements (such as anchors and forms) and generic event handlers (such as onfocus, onblur and onselect).

- Accessible alternatives described within noscript tags will not make a page accessible when scripts and scripted events that cannot be made directly accessible to assistive technologies are used. Content within noscript tags only appears when JavaScript is disabled. Most screen reader users have scripting enabled in their browsers.

- Use server side scripts instead of client side scripts when client side scripts cannot be made accessible. If neither server side or client side scripts can be made accessible, consider a redesign of the page that does not require similar scripting.

- Include noscript tags to denote where JavaScript is required for page functionality.

- If used, scripts based on dropdown select boxes should employ a “Go” button to activate the script once a selection has been confirmed or provide instructions within the form on how to operate the element and navigate the choices without making a selection until one is desired (Alt-Down Arrow lists options, navigate with Up and Down arrows, Enter to make a selection).

- Do not use scripting to automatically refresh page content or redirect to other pages unless you provide clear indication that this will occur and provide a means for the user to stop or delay the reload or redirect.

- Page content generated by scripts must comply with relevant parts of the Section 508 Technical Standards.

- Screen readers will behave in differing ways when scripts write information to a page. Some versions of screen readers are only aware of the additional content if the user refreshes the screen reader’s buffered information about the page, which often results in the user losing their location on the page. Other screen readers will reload their buffer of the entire page each time any new content is written to the page, again resulting in the user losing their location on the page.

- Use alert boxes instead of writing information directly to the page when possible. This is a far more reliable solution for form verification

warnings and notices. Screen readers do not notify users when scripts write new content to a page, so the user does not know to check for changes to the page. Alert boxes gain focus when they open and are automatically read by screen readers.

- With “Rollovers” that use event handlers to display menus of links, use device independent event handlers, use redundant event handlers, or provide redundant links on the page for each link in the “menu.”

- Avoid use of scripting to make non-focusable elements behave like a link or other element that can receive focus. Use the appropriate focusable element, such as links, to ensure that native browser keyboard functionality allows users to navigate to these elements.

- Do not use scripting to modify native browser behaviors.

- Avoid trapping key combinations used for navigation.

- Do not use scripting to provide keyboard navigation that conflicts with native browser keyboard commands.

How to Test

- Use a pointing device to navigate to and operate scripted controls such as links, buttons, and form elements:
    - Can elements be focused (accessed) using the keyboard (e.g. tab key)?
    - Are visuals associated with device-dependent events (such as hovering over an element with the mouse) also available when the item is focused using the keyboard?
    - Do any elements inherently require the use of a pointing device to operate—for example, to allow an element to be “dragged” from one location to another on the screen?
    - Do any events load another page that was not clearly anticipated, or have scripted controls that are difficult to make a selection from before the scripted event is activated (such as a dropdown selection form element) for events that alter or create content on the page?

- Verify that altered or added content, page elements and element state are accessible to screen readers both before and after the change has occurred. In some cases, screen readers will be unable to interpret newly loaded content because their buffers will remain unchanged

from the original, completed page—such pages will require a document reload to force the screen reader to flush the buffer and read in the new content.

- Turn off JavaScript in your user agent. In Firefox with the Web Developers Toolbar from http://chrispederick.com/work/webdeveloper/ select Tools> Web Developer> Disable> Disable JavaScript.

- Ensure that all pages containing client-side scripted events have declarations to that effect using the noscript element to warn users if scripted events are required.

- Test for equivalent functionality to client-side scripted events for essential features, such as form validation.

§ 1194.22 (m) Applets and Plug-Ins

Provision

When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

Plain English

Content or information presented through an applet, plug-in, or other third-party software isn't HTML and therefore doesn't fall under the provisions outlined for web accessibility.

Applets, plug-ins, and other third-party software embedded into (or required by documents linked to within) the html document must comply with the provisions for accessible software applications instead (§1194.21).

Why this is important.

Applets, plug-ins and other client applications are not part of the HTML document itself, nor are they under direct control of the browser or user agent, so they must be tested for compliance against a set of provisions which ensures the accessibility of those applications to persons with special needs.

Section 508 Compliant?

Not Applicable

§ 1194.22 (n) Electronic Forms

Provision

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.

Plain English

- Online electronic forms must be accessible to those using assistive technologies.
- Labels, directions, information, and functionality must all be clear to those same people
- Explicit textual prompts using the LABEL tag should be used whenever possible, along with the ID attribute in form control elements.

Why this is important.

When filling out online forms, an un-impaired user can quite easily discern what they are to enter into a given field by looking for the label that is visually close to the field they are inputting to. However, for users that employ assistive technology, the software that reads the raw HTML may not easily understand which text to read for which form control. This can quickly result in a form that is impossible to understand, or correctly complete, for the impaired user.

Section 508 Compliant?

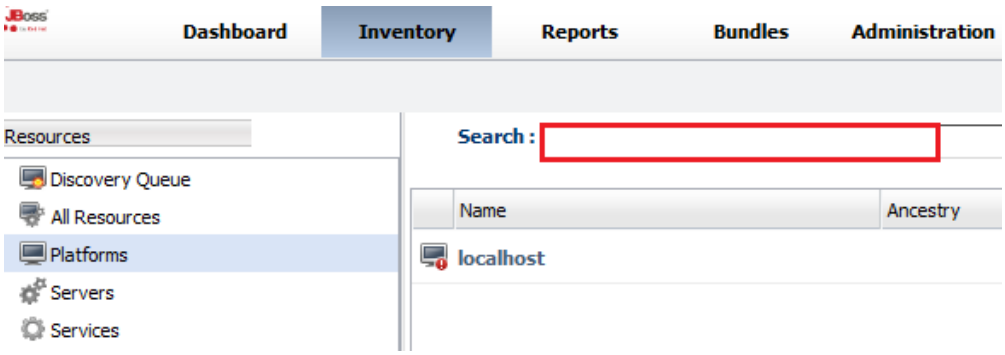
Not Compliant

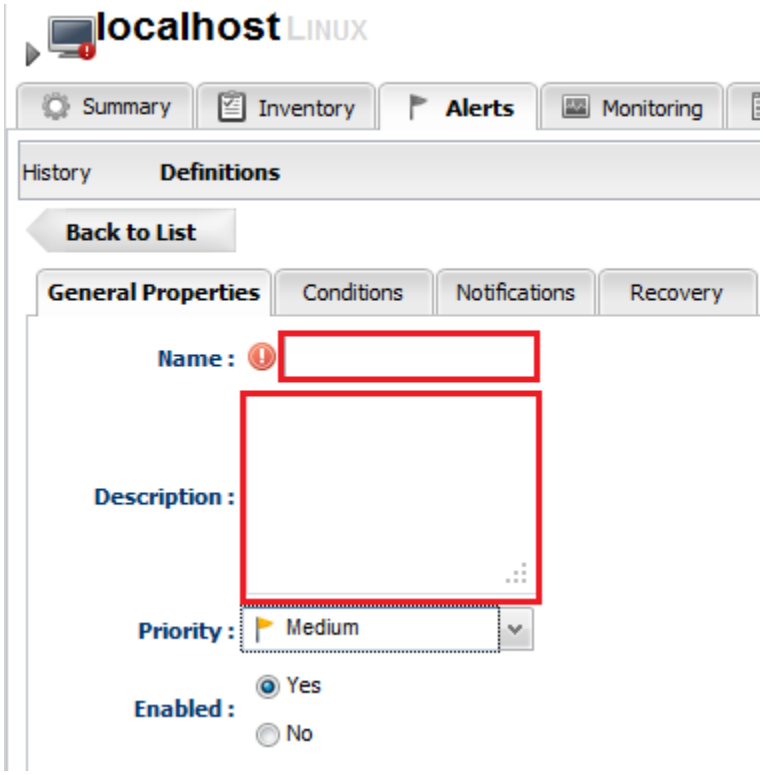
What must be fixed?

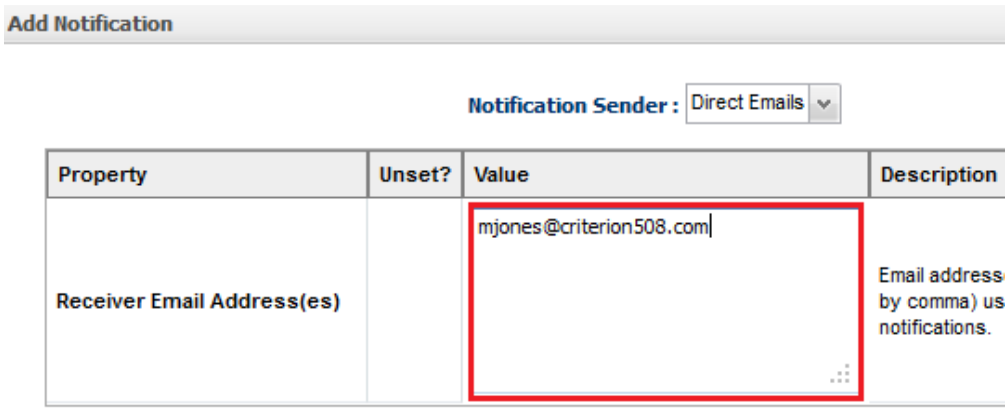
- Explicit textual prompts using the LABEL tag along with the ID attribute must be applied to form control elements on this website.

Examples/Screen Captures

The three screenshots below show form elements (outlined in red) that are missing their explicit label that describes the element. Each element requires an explicit LABEL tag so the value of its content can be programmatically associated with the appropriate form field or control. Some form controls, such as submit buttons, are accessible via their value labels and require no label. But text fields, select menus, text area's, checkboxes and radio buttons all require an explicit label using the LABEL element and associated `for` attribute. Using this markup allows assistive technologies to properly interpret form element values for users.







JAWS User 1 Comments

What must be fixed?

- No link, functional element, or specific instructions are provided on the Login page of this application to allow the user to understand what steps must be taken to login.

Examples

On the login page 2 form fields are present: user and password. No link appears for the user to click to login. After considerable searching on the page, this user pressed the enter key after typing the password and was logged in. Information must be provided to let the user know that pressing enter after the password field has been completed will launch the login.

Steps To Verify With JAWS:

1. Open the Red Hat application:

http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/

2. The focus is immediately taken to the user form field. Enter the user credentials.
3. Press tab to move to the password field and complete it. At this point the JAWS user would press tab to move to the login link. No such link is provided. This user pressed the arrow keys to try to find information as to how to proceed. No information was displayed. Using the key combination insert + F5 to open the form fields list box and insert + F7 to open the links list box did not produce any login information.

4. Move back to the password form field and ensure the password credentials have been added. Then, press enter. The Dashboard page is opened and the user has been logged in.

Recommendations/Repairs

- Ensure that all form elements have explicit label text describing the form element.

- Place text labels in visual proximity to the associated form element to make the relationship between the two clear.

- Use label tags to structurally associate form elements with their associated text labels. Use explicit labels (e.g. employ the for attribute) to ensure accessibility; implicit label tags are not supported by most screen readers.

- Label Example:
  <label for=”search”>Search:</label><input type=”text” ID=”search” name=”search”>

- Cues indicating required form controls should be included within label elements containing each form controls textual prompt to ensure that screen readers read this information automatically.

- When multiple levels of field prompts are present (common with radio button groups), ensure that all relevant textual prompts are associated with form controls. The Fieldset and Legend elements are often used to accomplish this.

- Fieldset/Legend Example:
  <fieldset><legend>Are you ready for the weekend?</legend>
  <input type="radio" name="weekend" id="wy" value="yes">
  <label for="wy">Yes</label>
  <input type="radio" name="weekend" id="wn" value="no">
  <label for="wn">No</label></fieldset>

- Avoid use of Fieldset/Legend elements to provide instructions and cues for entire sections of a form. Screen readers announce the legend each time the user navigates between controls, so instructions could become very verbose and repetitive, making the form difficult to complete.

- Verify that keyboard navigation moves through the form controls in a logical order. Screen readers navigate form controls in the order they appear in the source mark-up.

- If the Tabindex attribute is used to modify the Tab order of form elements, use the Tabindex attribute on all tabbable elements on the

page. Some screen readers will first Tab through all elements with Tabindex attributes, then Tab through elements that do not include Tabindex attributes, which has the potential to make a page very difficult to use.

- Verify that no other barriers are present that could prevent completing the form in a logical order. For example, make sure that submit or reset buttons are the last controls in the form.

How to Test

- Examine all forms as a whole. Are form instructions provided to make clear what a user is required to do to successfully complete the form?
    - Are all queues required to successfully complete the form present in an accessible manner and in the appropriate location in the form?
    - Are forms designed to flow in a logical progression, including the order in which both fields and submit actions are presented?

- Examine individual form elements:
    - Are clear, explicit text labels provided with each form element (whether made invisible via CSS or not)?
    - Are labels for all form elements visually clear?
    - Are appropriate and timely cues provided for each element?
    - Can each element be understood without having to refer to its contextual surroundings?
    - Do forms behave as expected when interacted with using both a pointing device and the keyboard?

§ 1194.22 (o) Navigation Links

Provision

A method shall be provided that permits users to skip repetitive navigation links.

Plain English

When a web site has a listing of navigational links that repeat on each page on the site, the ability to skip those links must be available.

The ability to skip those links must be clearly indicated as to how and when one can use it.

Why this is important.

For users that require a screen reader, it is essential that they not have to have each item on a repeating navigation bar read to them with each step in the navigation of a website. This produces a huge waste of time as the navigation links are re-read redundantly.

Instead, a user should be able to click directly through to the destination information at any time.

Section 508 Compliant?

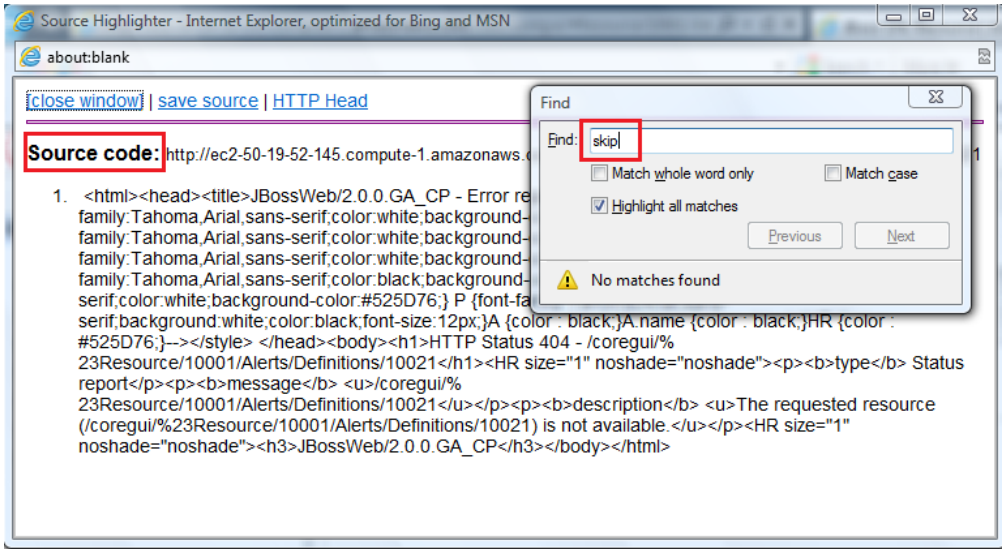
Not Compliant

What must be fixed?

- A method shall be provided that permits users to skip repetitive navigation links.

Examples/Screen Captures

This web site lacks a required skip navigation structure that allows users to bypass repetitive links and progress directly to the unique and essential information offered by each page. To rectify this, a “skip nav” link that places focus directly before the beginning of the unique page content needs to be employed site-wide. All pages within this site are affected.



JAWS User 1 Comments

What must be fixed?

- Skip navigation links must be provided to allow the user to skip past redundant links.

Examples

No skip navigation link is provided on the Dashboards page to allow the user to skip to main content.

Steps To Verify With JAWS:

1. After logging in the Dashboards page is displayed:

http://ec2-50-19-52-145.compute-1.amazonaws.com:7080/coregui/#Dashboards

2. The JAWS screen reader user often moves to the top of a page to determine location and content. Move to the top of the page with the key combination control + home.

3. Begin reviewing the page with the down arrow. No skip navigation link is provided to allow the user to quickly move to the main content.

Recommendations/Repairs

- Ensure that skip navigation links actually provide a means to bypass repetitive navigation links and other repetitive page elements to access page content quickly, if desired.

- Ideally, a skip navigation link will take the user directly the beginning of content unique to that page or particular view.

How to Test

- View the document source. Look for the body tag, and an immediately following anchor that braces either text or an image (with appropriate alternate text) that allows users to skip repetitive page elements.

- Search for the name that the anchor references. Is it present immediately before unique page content begins?

- You also can look for this link in a manner similar to how a screen reader operates by viewing rendered pages on a text-based browser such as eLinks—the link to skip to the content should be the first item displayed on the page.

If the site is frames-based:

- Identify frame divisions in the document – you may need to alter the frame-defining document to show borders, or use style sheets to identify the edge of each body element by drawing a border around them.

- Locate the frame which contains the content. Are there any items within that framed document that are also repetitive from page to page? If so, test that document as a document with no frames (as above).

- View the frame-setting document's source. Are frame names clearly defined to allow users to easily distinguish repetitive elements (e.g. navigation) from content?

§ 1194.22 (p) Time Delays

Provision

When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

Plain English

Any response that is required of the user within a certain amount of time must be adjustable to suit the needs of the user.
That is, the user must be warned of their time limit and that time limit must be adjustable or extendable by the user.

Why this is important.

Some disabilities may prevent users from being able to read or react to a page before a timed event expires: A screen reader, for example, might take many times longer than a fully sighted user to read through a page's contents and locate a desired option.

Section 508 Compliant?

Compliant

Video or Multimedia Products (1194.24)

Multimedia products involve more than one media and include, but are not limited to, video programs, narrated slide production, and computer generated presentations. Provisions address caption decoder circuitry (for any system with a screen larger than 13 inches) and secondary audio channels for television tuners, including tuner cards for use in computers. The standards also require captioning and audio description for certain training and informational multimedia productions developed or procured by Federal agencies. The standards also provide that viewers be able to turn captioning or video description features on or off.

Functional Performance Criteria (Subpart C)

The performance requirements of this section are intended for overall product evaluation and for technologies or components for which there is no specific requirement under the technical standards in Subpart B. These criteria are designed to ensure that the individual accessible components work together to create an accessible product. They cover operation, including input and control functions, operation of mechanical mechanisms, and access to visual and audible information.

These provisions are structured to allow people with sensory or physical disabilities to locate, identify, and operate input, control and mechanical functions and to access the information provided, including text, static or dynamic images, icons, labels, sounds or incidental operating cues. For example, one provision requires that at least one mode allow operation by people with low vision (visual acuity between 20/70 and 20/200) without relying on audio input since many people with low vision may also have a hearing loss.

Information, Documentation, and Support (Subpart D)

The standards also address access to all information, documentation and support provided to end users (e.g., Federal employees) of covered technologies. This includes user guides, installation guides for end-user installable devices, and customer support and technical support communications. Such information must be available in alternate formats upon request at no additional charge. Alternate formats or methods of communication can include Braille, cassette recordings, large print, electronic text, Internet postings, TTY access, and captioning and audio description for video materials.

Appendix A: Section 508: Subpart B § 1194.22

Web-Based Internet Information and Applications

a. A text equivalent for every non-text element shall be provided (e.g., via "alt", "LongDesc", or in element content).

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

c. Web pages shall be designed so that all information conveyed with color is also available without color, for example from context or markup.

d. Documents shall be organized so they are readable without requiring an associated style sheet.

e. Redundant text links shall be provided for each active region of a server-side image map.

f. Client-side image maps shall be provided instead of server-side image maps except where the regions cannot be defined with an available geometric shape.

g. Row and column headers shall be identified for data tables.

h. Markup shall be used to associate data cells and header cells for data tables that have two or more logical levels of row or column headers.

i. Frames shall be titled with text that facilitates frame identification and navigation.

j. Pages shall be designed to avoid causing the screen to flicker with a frequency greater than 2 Hz and lower than 55 Hz.

k. A text-only page, with equivalent information or functionality, shall be provided to make a web site comply with the provisions of this part, when compliance cannot be accomplished in any other way. The content of the text-only page shall be updated whenever the primary page changes.

l. When pages utilize scripting languages to display content, or to create interface elements, the information provided by the script shall be identified with functional text that can be read by assistive technology.

m. When a web page requires that an applet, plug-in or other application be present on the client system to interpret page content, the page must provide a link to a plug-in or applet that complies with §1194.21(a) through (l).

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.

o. A method shall be provided that permits users to skip repetitive navigation links.

p. When a timed response is required, the user shall be alerted and given sufficient time to indicate more time is required.

JavaScript Accessibility

Introduction

Writing accessible JavaScript can be challenging if accessibility-enhancing tools such as screen readers and their various uses are not well-understood by developers.

It is therefore recommended that developers install and familiarize themselves with the operation of screen reader package such as JAWS to better understand the issues described in this document such as JavaScript accessibility.

Developers should use references such as those provided on the web and in this section as guidelines to develop accessible JavaScript. Ultimately, the responsibility for delivering the appropriate, expected accessible behavior for the thousands of lines of JavaScript code in the application lies with developers who understand the challenges presented by developing accessible JavaScript, and how to overcome those challenges.

An excellent example of how to develop accessible JavaScript from The Access Board web site (http://www.access-board.gov/sec508/guide/1194.22.htm#(l)) has been reproduced below for your reference.

Developing Accessible JavaScript

Web developers working with JavaScript frequently use so-called JavaScript URL's as an easy way to invoke JavaScript functions. Typically, this technique is used as part of <a> anchor links. For instance, the following link invokes a JavaScript function called myFunction:

```
<a href="javascript:myFunction();">Start myFunction</a>
```

This technique does not cause accessibility problems for assistive technology. A more difficult problem occurs when developers use images inside of JavaScript URL's without providing meaningful information about the image or the effect of the anchor link. For instance, the following link also invokes the JavaScript function myFunction, but requires the user to click on an image instead of the text "Start myFunction":

```
<a href="javascript:myFunction();"><img src="myFunction.gif"></a>
```

This type of link, as written, presents tremendous accessibility problems, but those problems can easily be remedied. The <img> tag, of course, supports the alt attribute that can also be used to describe the image and

the effect of clicking on the link. Thus, the following revision remedies the accessibility problems created in the previous example:

```
<a href="javascript:myFunction();"><img src="myFunction.gif"
alt="picture link for starting myFunction"></a>
```

Another technique advocated by some developers is to use the title attribute of the <a> tag. For instance, the following example includes a meaningful description in a title attribute:

```
<a title="this link starts myFunction"
href="javascript:myFunction();"><img src="myFunction.gif"></a>
```

This tag is supported by some but not all assistive technologies. Therefore, while it is part of the HTML 4.0 specifications, authors should use the alt tag in the enclosed image.

Finally, the browser's status line (at the bottom of the screen) typically displays the URL of any links that the mouse is currently pointing towards. For instance, if clicking on an anchor link will send the user to http://www.usdoj.gov, that URL will be displayed in the status line if the user's mouse lingers on top of the anchor link. In the case of JavaScript URL's, the status line can become filled with meaningless snips of script. To prevent this effect, some web developers use special "event handlers" such as onmouseover and onmouseout to overwrite the contents of the status line with a custom message. For instance, the following link will replace the content in the status line with a custom message "Nice Choice".

```
<a href="javascript:myFcn();" onmouseover="status='Nice Choice';
return true;" onmouseout="status='';"><img src="pix.gif"></a>
```

This text rewritten into the status line is difficult or impossible to detect with a screen reader. Although rewriting the status line did not interfere with the accessibility or inaccessibility of the JavaScript URL, web developers should ensure that all important information conveyed in the status line also be provided through the alt attribute, as described above.

JavaScript uses so-called "event handlers" as a trigger for certain actions or functions to occur. For instance, a web developer may embed a JavaScript function in a web page that automatically checks the content of a form for completeness or accuracy. An event handler associated with a "submit" button can be used to trigger the function before the form is actually submitted to the server for processing. The advantage for the government agency is that it saves government resources by not requiring the government's server to do the initial checking. The advantage for the computer user is that feedback about errors is almost instantaneous

because the user is told about the error before the information is even submitted over the Internet.

Web developers must exercise some caution when deciding which event handlers to use in their web pages, because different screen readers provide different degrees of support for different event handlers. The following table includes recommendations for using many of the more popular event handlers:

- onClick – The onClick event handler is triggered when the user clicks once on a particular item. It is commonly used on links and button elements and, used in connection with these elements, works well with screen readers. If clicking on the element associated with the onClick event handler triggers a function or performs some other action, developers should ensure that the context makes that fact clear to all users. Do not use the onClick event handlers for form elements that include several options (e.g. select lists, radio buttons, checkboxes) unless absolutely necessary.
- onDblClick – The onDblClick event handler is set off when the user clicks twice rapidly on the same element. In addition to the accessibility problems it creates, it is very confusing to users and should be avoided.
- onMouseDown and onMouseUp – The onMouseDown and onMouseUp event handlers each handle the two halves of clicking a mouse while over an element – the process of (a) clicking down on the mouse button and (b) then releasing the mouse button. Like onDblClick, this tag should be used sparingly, if at all, by web developers because it is quite confusing. In most cases, developers should opt for the onClick event handler instead of onMouseDown.
- onMouseOver and onMouseOut – These two event handlers are very popular on many web sites. For instance, so-called rollover gif's, which swap images on a web page when the mouse passes over an image, typically use both of these event handlers. These event handlers neither can be accessed by the mouse nor interfere with accessibility – a screen reader simply bypasses them entirely. Accordingly, web designers who use these event handlers should be careful to duplicate the information (if any) provided by these event handlers through other means.
- onLoad and onUnload – Both of these event handlers are used frequently to perform certain functions when a web page has either completed loading or when it unloads. Because neither event handler is triggered by any user interaction with an element on the page, they do not present accessibility problems.

- onChange – This event handler is very commonly used for triggering JavaScript functions based on a selection from within a <select> tag. Surprisingly, it presents tremendous accessibility problems for many commonly used screen readers and should be avoided. Instead, web developers should use the onClick event handler (associated with a link or button that is adjacent to a <select> tag) to accomplish the same functions.
- onBlur and onFocus – These event handlers are not commonly used in web pages. While they don't necessarily present accessibility problems, their behavior is confusing enough to a web page visitor that they should be avoided.