Assignment #5 – Forms: Best Practices for Usability & Accessibility (Report – 16 pts.

Peer Review – 6 pts.)

# The files you need to start this lab are at <http://webdevgroupcu.org/atlas4519/forms/>

(or download from the [course site](https://canvas.colorado.edu/courses/68477/pages/html-and-other-files-for-assignments?module_item_id=2174655) (under HTML and other files for assignments)

(Login directions to the server are available under the [*Class Resources & Files* section](https://canvas.colorado.edu/courses/68477/pages/server-login-instructions?module_item_id=1822021) on Canvas. Remember, your home directory will be named: http://webdevgroupcu.org/[your identikey])

This exercise assumes you have read the WebAIM article on Creating Accessible Forms, both page 1 - [General Form Accessibility](http://webaim.org/techniques/forms/) and page 2 - [Accessible Form Controls](http://webaim.org/techniques/forms/controls). If you haven’t already done so, I suggest you read it now.

Some URLs we will use during this exercise:

<http://www.w3.org/WAI/demos/bad/before/survey.html> - this is the before version (this is the inaccessible version).

<http://www.w3.org/WAI/demos/bad/after/survey.html> - before and after demo of a form interface (this is the well designed, accessible version).

## Demonstrating an inaccessible vs. an accessible form

1. Go to [WAI’s “bad” form page](http://www.w3.org/WAI/demos/bad/before/survey.html)
   1. Try clicking on the radio button labels and within the text fields. Does anything happen? (Hint: probably not).
      1. Nothing happens when I try to click on the radio button labels.
   2. Move through the form using **[TAB]**. Are you able to circle your way through the entire form? What do you notice about visual feedback as you move through the form? In other words, can you tell where your keyboard focus is as you tab through the form.
      1. I’m not able to circle through the entire form because in the first form it goes to none and then skips to the next form. I can tell where my keyboard focus is as I tab through the form, but it’s kind of difficult because of how similar it is to background color.
   3. [Now go to the accessible version of the form](http://www.w3.org/WAI/demos/bad/after/survey.html). Try the same operations and observations you conducted for the inaccessible version. What are the differences?
      1. The differences are that you can click on the button labels and it also highlights it red for you, too. Also, when you tab through the different labels, if you tab to an input field, it highlights it red which makes it easier to follow.
   4. Now launch NVDA (for Windows) or VoiceOver (for MacOS) and **[TAB]** through each form, the inaccessible and then the accessible version. What do you notice about what is read by the screenreader? (Note: the screenreader is more functional when the user tabs through form fields – more information is read to the user. Notice the difference when you use the arrow keys instead).
      1. The main differences that I noticed between this is the accessible page has more descriptive alt text for the links and buttons. Another main difference is that you can use the arrow keys to go through each button in the accessible page whereas the inaccessible page only goes to the button that is chosen. Lastly, in the accessible page it lists out the cities options.

## Exercise: Adding Coding for Accessibility

1. Download the file below from the [course site](https://canvas.colorado.edu/courses/68477/pages/html-and-other-files-for-assignments?module_item_id=2174655) (under HTML and other files for assignments) or from the URL below

<http://webdevgroupcu.org/atlas3519/forms/>

***form-exercise1(one).zipx***

This zip\* file contains an html and a css file

\*(Use a free program such as [Winzip](http://www.winzip.com/win/en/downwz.html), [WinRar](https://winrar.en.softonic.com/download) or [7-Zip](http://www.7-zip.org/) for Windows or [iZip for the Mac](http://www.izip.com/) to extract the files below)

***form-exercise1(one).html***

***form.css***

After you extract the files you should place the CSS file in a subfolder called “css” and it should be a child of the folder where your ***form-exercise1(one).html*** file is located, as shown in the screen shot below.

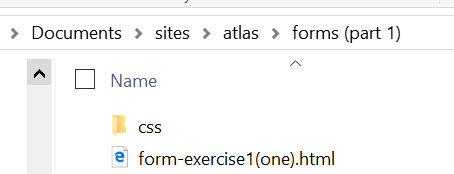


Figure 1 Screen shot of CSS in subdirectory

Remember, it does not matter in which directory you put the form-exercise(one).html file, as long as the css folder is in an immediate subdirectory.

When you upload your files to your server directory (I suggest creating a subdirectory on your server directory “forms” – or select another name of your choice – in order to keep your projects organized). You should mirror the setup on your hard drive by placing your css file in a subdirectory called “css”.

Note: the form-exercise1(one).html file points to the style sheet using a link tag in the <head> area as shown below:

<link rel="stylesheet" href="css/form.css" />

This tells the browser to look for the style sheet in a subdirectory called “css” and with the name of “form.css”.

If you change the name of your style sheet or the directory, you must also match the changes in the above link tag.

From a browser, the folder should appear as follows:



(Note: instead of “atlas4519”, your path will include your identikey name).

Open up the ***form-exercise1(one).html*** file from your local drive. You can use text-wrangler, DreamWeaver, Coda, TextWrangler, or another web editor of choice. (I suggest a wysiwig editor so that you can see a preview of your page as you enter and change coding).

### Testing with a Screenreader

1. As with the before and after page, test the page (***form-exercise1(one).html)*** with a screenreader, NVDA or VoiceOver and notice what is spoken. You will notice that you hear no headings announced and very little to no information about the form fields, especially when tabbing through the form.
2. Also note what type of visual feedback you receive as you tab through the form.

### Creating Structure & Organization with Headings

1. The first thing we will do is add structure to the page by marking up headings appropriately.

If we run WAVE on the page we see that there is no heading structure.

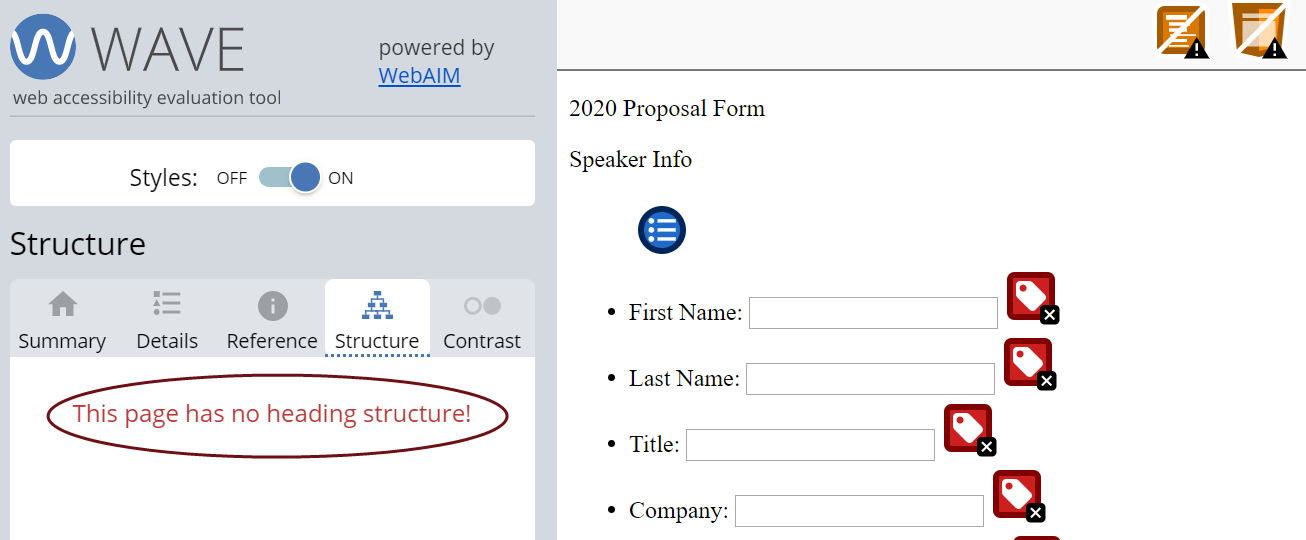


Figure 2 WAVE screenshot - no heading structure

1. If we look at the code and design view of the page (in Dreamweaver in this case – see screen shot below) we can see that there are no heading tags (<h1>, <h2>, etc.) in the code.

[continues on next page]

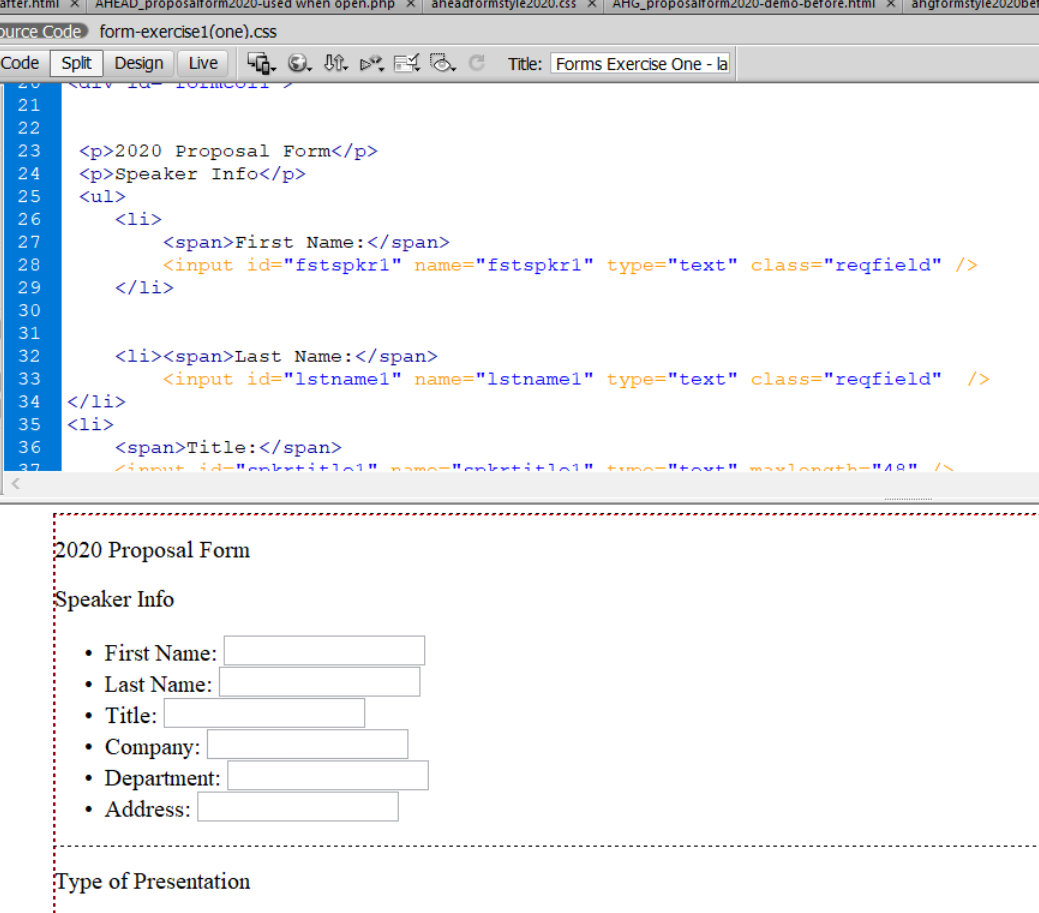


Figure 3 Screen capture from Dreamweaver

1. What do you think the heading 1, heading 2, etc., should be for this form? Heading 1 should be the item that best describes the overall page. In this case, that would be “2020 Proposal Form.” “Speaker Info” and “Type of Presentation” are subsections of the page, thus they normally should be marked up as heading 2s. However, because they are in a form we will later be marking them up as <legend> items which marks them as a section heading for the group of fields below each item. Therefore we will only implement the <h1> for now.

Therefore,

* 1. Change:

<p>2020 Proposal Form</p>

<p>Speaker Info</>

…

…

<p>Type of Presentation<p>

To:

**<h1>2020 Proposal Form</h1>**

<p>Speaker Info</>

…

…

<p>Type of Presentation<p>

### Title text

Check the title text for the page. Note that you can use a screenreader or hover over the browser tab to hear or view the text. You can also reference [WAI Easy Checks](https://www.w3.org/WAI/test-evaluate/preliminary/#title) for tips on finding the title text. You should notice that the title – “Forms” - is not very helpful.

1. You can find the title tag in the <head> of the document, as shown below. Change the text to something that better describes the page.

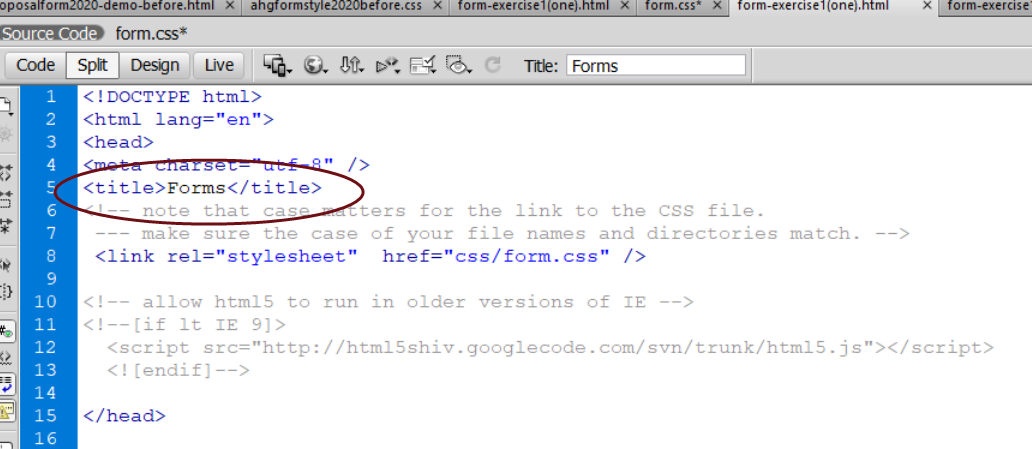


Figure 4 Screen shot of title tag

### Grouping & Labeling Form Segments

1. The WebAIM article discusses how the <fieldset> and <legend> tags are used to group related fields and form elements. Enclosing a set of form elements in *<fieldset>* tells assistive technology, browsers and other code-reading tools that the elements are logically related. The <legend> tag provides a visual cue (by default, a border around the included elements and a bolded heading) that the fields are related. This information is also provided to screenreaders and other Assistive Technology.

Therefore let’s group and label the Speaker Info and content fields using the fieldset and legend tags.

* 1. Change:

<h1>2020 Proposal Form</h2>

<p>Speaker Info</p>

<ul>

<li>…

…

To:

<h2>2020 Proposal Form</h2>

***<fieldset>***

***<legend>Speaker Info</legend>***

<ul>

<li>…

…

* 1. Where do we enter the closing tag for fieldset?

Not sure? - see the 3rd example of code on [page 2 of the WebAIM article](http://webaim.org/techniques/forms/controls).

* + - * You’d put it after the </ul>.

1. Let’s do the same for “Type of Presentation”
   1. Change:

<div id="multchoice">

<p>Type of Presentation</p>

….

To:

<div id="multchoice">

***<fieldset>***

***<legend>Type of Presentation</legend>***

….

Save ***form-exercise1(one).html*** and upload it up to your server directory (or you can save it to your hard drive and view it locally).

Remember to refresh the page.

### Labeling the individual input fields

1. Now let’s add labels to the individual fields, starting with the first two fields on the form.

*(Remember: labels explicitly associate the label with the field to avoid any confusion for those not accessing the form visually – i.e. Assistive Technology users).*

Change:

***<span>First Name:</span>***

<input id="fstspkr1" name="fstspkr1" type="text" class="reqfield" title="Required field" />

To:

***<label for="fstspkr1">First Name:</label>***

<input id="fstspkr1" name="fstspkr1" type="text" class="reqfield" />

</li>

(**Note**: as mentioned in the reading, the label is associated to an input field through the use of *for=”fstspkr1”* to point to the id of the input field).

(**Note**: the ***class="reqfield"*** attribute allows us to later use styling to format the field in a way to indicate that it is a required field – perhaps styling it with a different background color. This attribute alone does not *require* the user to enter a value for the field. We will go over the process for requiring entry when we address validation in a subsequent exercise).

(**Another note**: I’ve used <li> … </li> - a list item – as a structure to encapsulate each label and field. This is not required – it’s simply the way I chose to approach it).

Change:

<li>

**<span>Last Name:</span>**

<input id="lstname1" name="lstname1" type="text" class="reqfield" />

</li>

To:

<li>

***<label for="lstname1">Last Name:</label>***

<input id="lstname1" name="lstname1" type="text" class="reqfield" />

</li>

1. Save and refresh the file in your browser to view the changes.

### Styling

Let’s start styling the form to align the labels and fields and to improve usability.

1. First of all, let’s get rid of the bullet points for each label/field line. We will also set the width of the labels so that the input fields line up at the start of the field.

Add the following style code to your css file (form.css)

label {

margin-right: .3em;

width: 6.3em; /\* I made this deliberately wide – come up with your own optimal width \*/

display: inline-block; /\* width will not work for “inline” elements such as “label”. Therefore we need to set it to “inline-block” \*/

}

To remove the bullet points, we set list style to none:

/\* we place the id formcol1 in front of the li element in order to limit this formatting to the top part of the form \*/

#formcol1 li {

list-style:none;

}

1. Save and refresh the file in your browser to view the changes.

Your form should look something like the screen shot below.

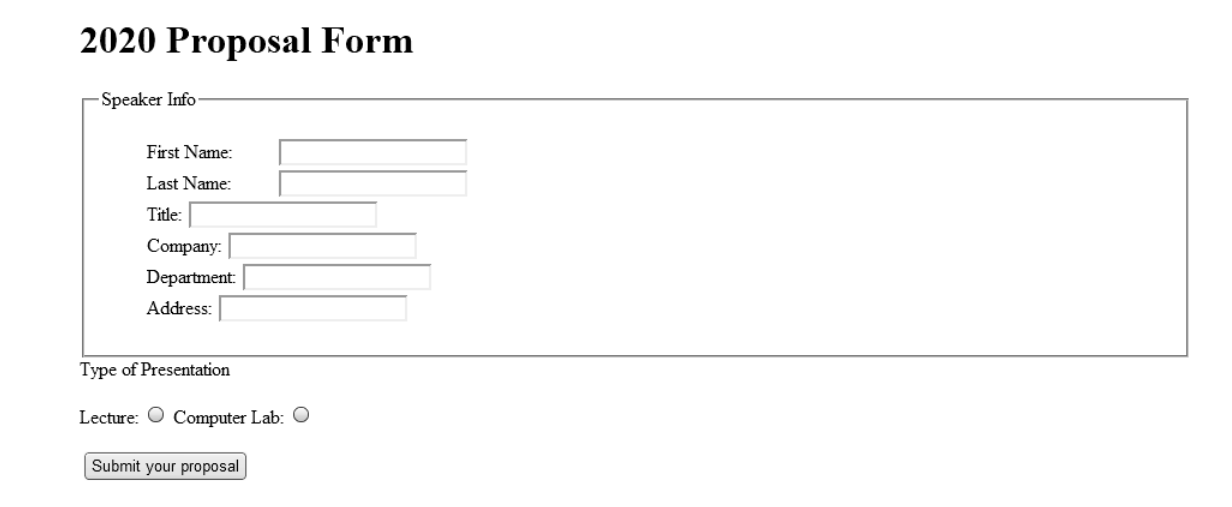
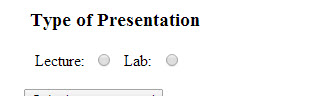


Figure 5 Screen shot after setting margins for label and removing bullet points

1. Before we change the rest of the text input fields, let’s jump down to the last question which has radio button input.



* 1. The labeling for radio buttons and check-boxes is slightly different than for text fields so we’re review it here.
  2. Use the label element for each radio button choice.
  3. Change:

<p>Type of Presentation</p>

<p**><span>Lecture:</span>**<span><input value="1" id="TypeofPres\_lec"

name="p-type" type="radio" /> </span>

**<span>Lab:</span>** <span><input value="2" id="TypeofPres\_lab" name="p-type" type="radio" /></span></p>

To:

<legend>Type of Presentation</legend>

<p><span><input value="1" id="TypeofPres\_lec" name="p-type"

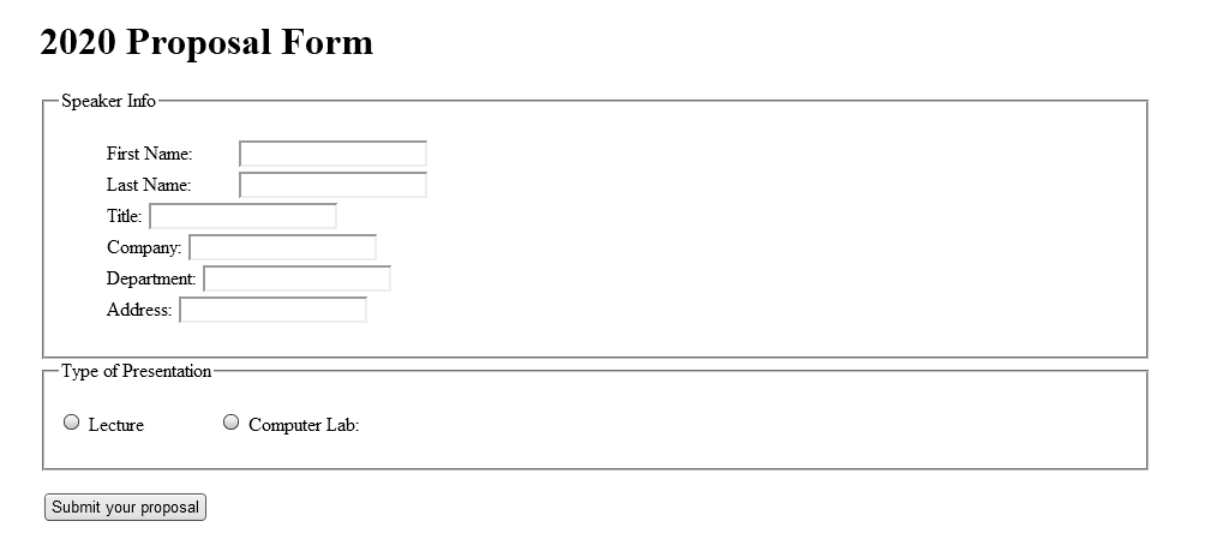
type="radio" /> ***<label for="TypeofPres\_lec">Lecture</label>*** </span>

<span><input value="2" id="TypeofPres\_lab" name="p-type" type="radio" />***<label for="TypeofPres\_lab">Lab</label>***</span></p>

Note: in this case we are putting the input code before the label code because the convention is for labels to follow the radio button control.

1. Save and review your changes.

Your form should look something like the following:



1. Now do the following
   1. Add labels to the rest of the text fields under “Speaker Info.”
   2. Add some instructions to the top of the form (for example – let users know how to determine required fields).

Note: one convention to indicate a required field is to place an asterisk (\*) at the end of the label.

* 1. Change any colors, formatting or fonts you think would improve the form. Note how the radio button items are far apart. Can you bring them closer?
  2. Test your form by clicking on labels to select radio buttons or move focus to an edit field using the keyboard.

### Adding Additional (Semantic) Constraints to the Form

1. Aligning labels along right edge. [Research](https://uxplanet.org/designing-more-efficient-forms-structure-inputs-labels-and-actions-e3a47007114f) seems to support the advantage of aligning labels to the right just before the data entry field.
   1. We will need to change the label styling to align right. The styling code is as follows:

***text-align: right;***

Place this code in the styling segment for “label”.

* 1. What happens to the form styling when we make this change? How do we correct the problem? (Don’t see it, look at the bottom of the form).
     1. Doing this makes the labels and the radio buttons for the second form far apart from each other. I corrected the problem by making a div class just for the top section and in the CSS made a “.topSection”
  2. Is the right alignment of the labels at the top section of the form an improvement?
     1. I think having the right alignment of the labels at the top section of the form is an improvement because no matter how long the word or words are, it always end at the same space

1. Adding highlighting of current keyboard and mouse focus

We have observed keyboard accessibility in the before and after WAI demo pages. On the inaccessible pages we could not move through the entire page using the tab key and received no visual feedback on the location of the location of the cursor (i.e. keyboard focus). On the accessible page these issues were not present.

* 1. Add the following styling segment to provide highlighting of current focus

select:focus, select:active, input:focus, input:active, input:focus + label, input:active + label, label[for]:focus, label[for]:active {

color: #FFF !important;

background-color: #BB3D2F !important;}

* 1. Use your keyboard and mouse to move to different areas of the form. Feel free to change the highlight color I provided.

(some suggested tools for checking and picking accessible color combinations: [TPGi Colour Contrast Analyser](https://www.tpgi.com/color-contrast-checker/) or WCAG Color contrast checker for [Chrome](https://chrome.google.com/webstore/detail/wcag-color-contrast-check/plnahcmalebffmaghcpcmpaciebdhgdf?hl=en)\* or [Firefox](https://addons.mozilla.org/en-US/firefox/addon/wcag-contrast-checker/).)

\*(copy and paste the URL <https://chrome.google.com/webstore/detail/wcag-color-contrast-check/plnahcmalebffmaghcpcmpaciebdhgdf?hl=en> into your browser if you run into problems finding the plugin)

* 1. Using color (in addition to asterisk) to provide cueing of required fields.
     1. Add the following styling code to your style sheet:

#formcol1 input.reqfield {

background:#FF9;

}

## Assignment #5 Deliverables

1. In the ***discussion area*** for assignment #5, enter the URL of the webpage that contains the form you created. Write a couple of paragraphs in the discussion area noting how well these form techniques worked in improving the usability/accessibility of the form. Or you can focus on discussing how you think one or more of these techniques would benefit persons with different types of disabilities.

### Respond to at least 2 other posts

1. Respond to at least 2 other posts, either on the features of the form your classmate posted or on their comments about the changes they implemented or the benefit of the features implemented to persons with various disabilities.

Output:

* Post URL of your form and 2 paragraph report (see details above)
* Respond to 2 other student posts (see instructions above)

Where to post: assignment #5 discussion area.