CP 160 - Web Programming and Design

Lab 10: Media and Interactivity

Hands-On Practice 1: Use MP3 file

In this Hands-On Practice, you will create a web page similar to **Figure 1** that contains an h1 element and a hyperlink to an MP3 file. The web page will also provide a hyperlink to a text transcript of that file to provide for accessibility. It can also be useful to your web page visitors if you indicate the type of file (such as an MP3) and, optionally, the size of the file to be accessed.

Copy the **podcast.mp3** and **podcast.txt** files from the starters folder in the student files and save them to a folder named **podcast**. Use the **template.html** file as a starting point and create a web page containing a page title of Podcast, an h1 element with the text Web Design Podcast, a hyperlink to the MP3 file, and a hyperlink to the text transcript. Save your page as podcast.html. Display the file in a browser. Try to test your page in different browsers and browser versions. When you click on the MP3 hyperlink, an audio player (whichever player or plug-in is configured for the browser) will launch to play the file. When you click on the hyperlink for the text transcript, the text will display in the browser.

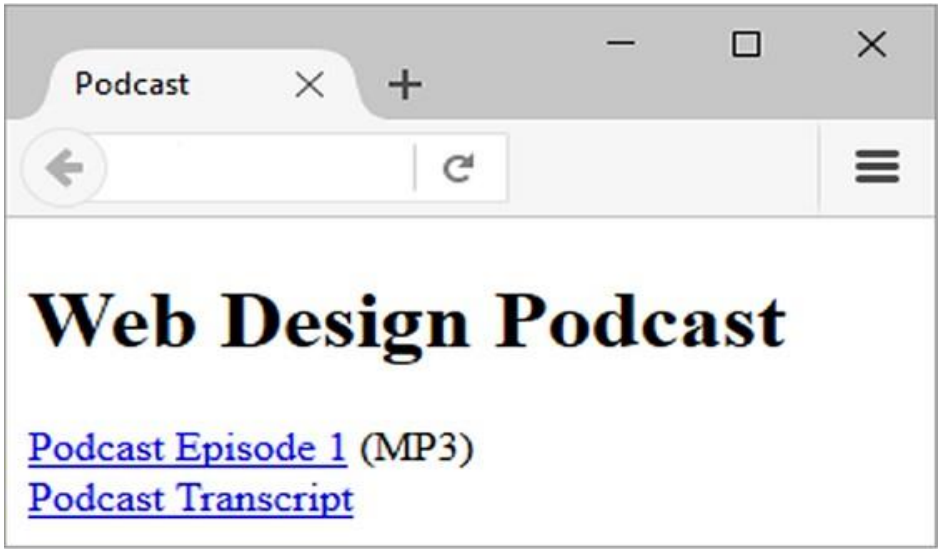

Figure 1 The default MP3 player will launch in the browser when the visitor clicks on Podcast Episode 1.

Hint: the code follows:

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Podcast</title>
<meta charset="utf-8">
</head>
<body>
<h1>Web Design Podcast</h1>
<div><a href="podcast.mp3" title="Web Design Podcast &amp;
Episode 1">Podcast Episode 1</a> (MP3)</div>
<div><a href="podcast.txt">Podcast Transcript</a></div>
</body>
</html>
```

Hands-On Practice 2: Display a Flash slideshow of photographs

In this Hands-On Practice, you will launch a text editor and create a web page that displays a Flash slideshow of photographs. Your page will look like the one shown in **Figure 2**.

Create a folder called **embed**. Copy the **lighthouse.swf** file from the starters folder to your embed folder.

Use the **template.html** file as a starting point and create a web page containing a page title and an h1 element with the text "Door County Lighthouse Cruise" and an `<embed>` tag to display a Flash file named lighthouse.swf that is 320 pixels wide and 240 pixels high. A sample embed tag follows:

```
<embed type="application/x-shockwave-flash"
              src="lighthouse.swf" quality="high"
              width="320" height="240"
              title="Door County Lighthouse Cruise">
```

Notice the value of the `title` attribute in the code sample. The descriptive text could be accessed by assistive technologies such as a screen reader application.

Save your page as index.html in the embed folder. Test it in a browser.

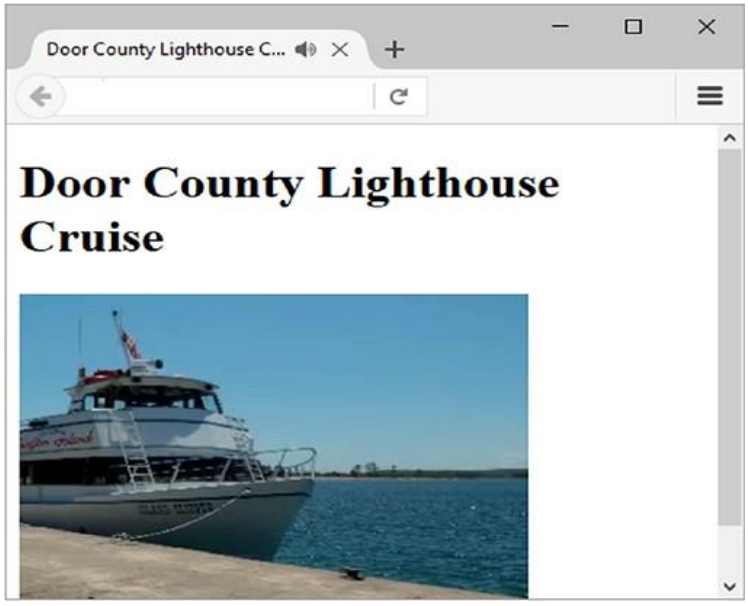

Figure 2 Flash slideshow of images configured with the embed element.

Hint: the **code** follows. If your browser does not support flash, move to the next Practice.

```
<!DOCTYPE html>
<html lang="en">
<head>
<title>Door County Lighthouse Cruise</title>
<meta charset="utf-8">
</head>
<body>
<h1>Door County Lighthouse Cruise</h1>
<embed type="application/x-shockwave-flash"
       src="lighthouse.swf"
       width="320" height="240"
       title="Door County Lighthouse Cruise">
</body>
</html>
```

Hands-on Practice 3: Display a YouTube video

In this Hands-On Practice, you will launch a text editor and create a web page that displays a YouTube video within an iframe element. This example embeds the video found at https://youtu.be/zL080xt6Ems. You can choose to embed this video or select a different video. The process is to display the YouTube page for the video and copy the video identifier, which is the text after the "=" in the URL. In this example, the video identifier is **zL080xt6Ems**.

Use the template.html file as a starting point and configure a web page containing a page title and an h1 element with the text "YouTube Video" and an iframe element. Code the `src` attribute with **https:// www.youtube.com/embed/** followed by the video identifier.

In this example, set the `src` attribute to the value **https://www.youtube.com/embed/zL080xt6Ems** . Configure a hyperlink to the YouTube video page as fallback content. The code to display the video shown in **Figure 3** follows:

```
<h1>Iframe Example</h1>
<iframe src=https://www.youtube.com/embed/zL080xt6Ems
  width="640" height="385">
</iframe>
```

Save your page as **youtubevideo.html** and display it in a browser. Compare your work with **Figure 3.**

Figure 3 The Iframe element in action – youtube video

Hands-On Practice 4: Configure rotate and scale transforms

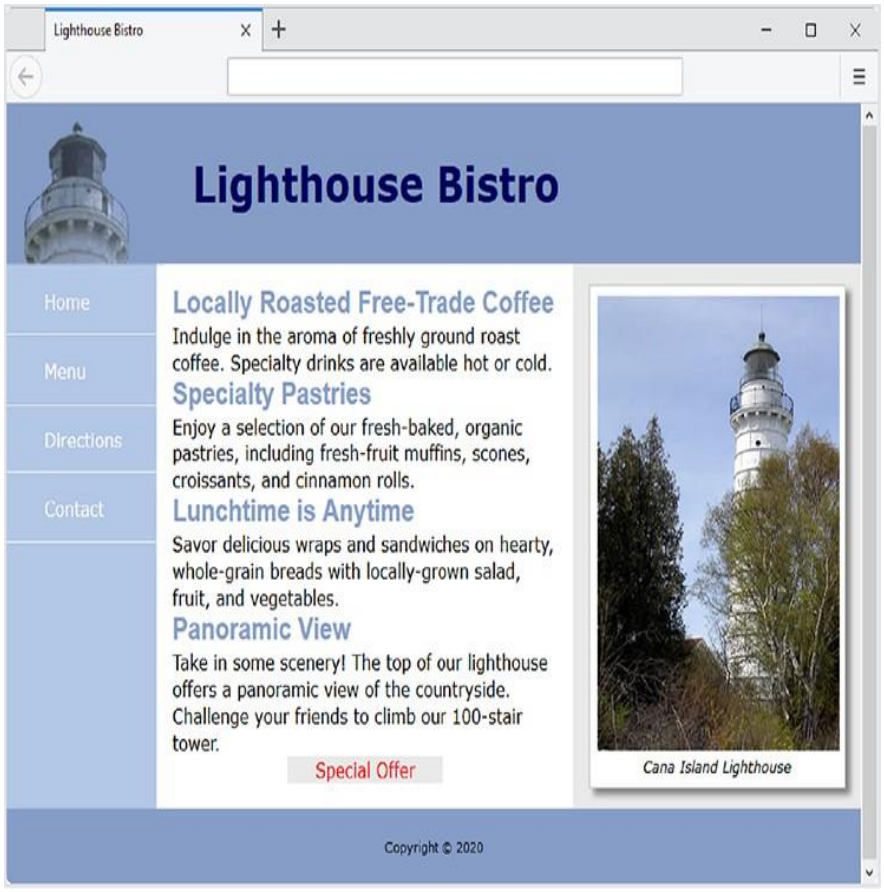

Figure 4 The transform property in action.

In this Hands-On Practice, you will configure the rotate and scale transforms shown in **Figure 4**. Create a new folder named **transform**. Copy the light.gif and lighthouse.jpg images from the starters folder to your transform folder. Launch a text editor and open the starter.html file. Save the file as index.html in your transform folder. Launch the file in a browser, and it will look similar to **Figure 5**.

Figure 5 Before the transform property.

Open index.html in a text editor and view the embedded CSS.

1. Locate the figure element selector. You will add a style declaration to the figure element selector that will configure a three-degree rotation transform. The new CSS is shown in blue.

```
figure { margin: auto; padding: 8px; width: 265px;
         background-color: #FFF; border: 1px solid #CCC;
         box-shadow: 5px 5px 5px #828282;
         transform: rotate(3deg);  }
```

2. Locate the #offer selector. This configures the “Special Offer” div displayed above the page footer. You will add a style declaration to the #offer selector that configures the browser to display the element two times larger. The new CSS is shown in blue.

```
#offer { background-color: #EAEAEA;
         width: 10em;
         margin: 2em auto 0 auto;
         text-align: center;
         transform: scale(2);  }
```

Save the file and display it in a browser. You should see two changes: the figure displayed on a slight angle and larger “Special Offer” text. Compare your work to **Figure 4**.

Hands-On Practice 5: Configure a navigation menu

In this Hands-On Practice, you will configure a navigation menu that is interactive and displays a drop-down menu. **Figure 6** displays a site map for the website. Notice how the Cuisine page has three subpages: Breakfast, Lunch, and Dinner. You will configure a drop-down menu that displays when a visitor hovers over the Cuisine navigation hyperlink as shown in **Figure 7**.

Create a folder named **mybistro**. Copy the files from the **bistro** folder into your mybistro folder. Notice the main menu has hyperlinks for Home, Coffee, Cuisine, Directions, and Contact. You will edit the CSS and edit each page to configure a Cuisine submenu that provides hyperlinks to three pages (Breakfast, Lunch, and Dinner).

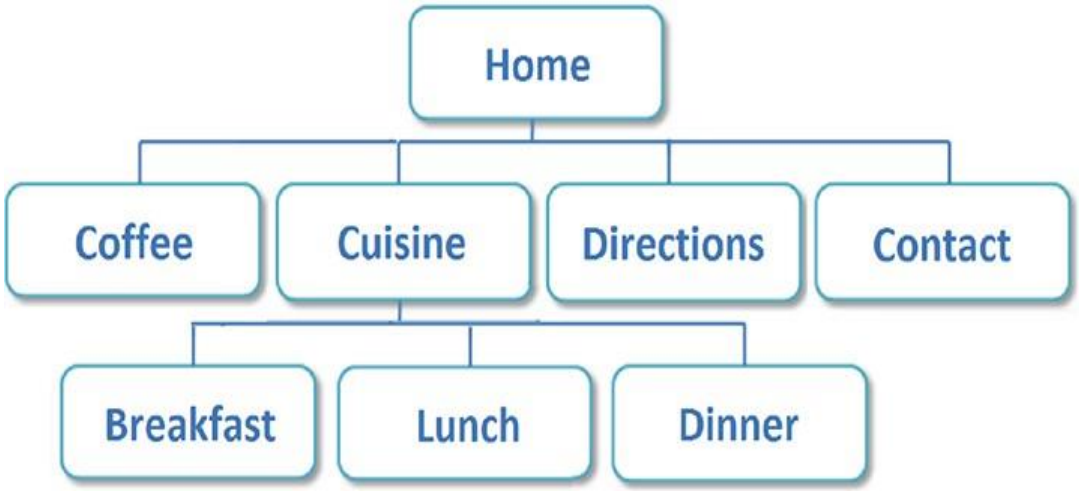

Figure 6 Site map.

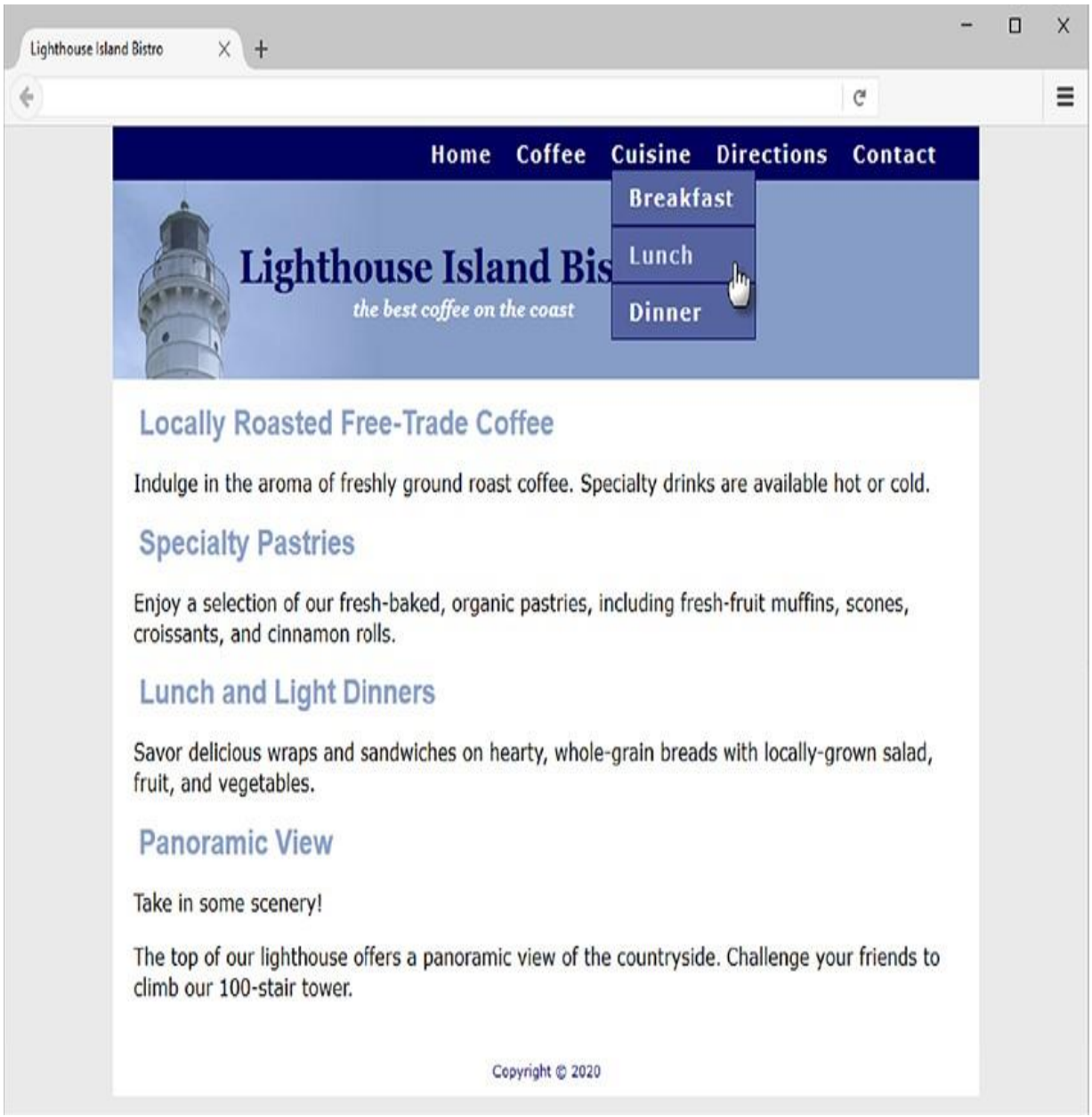

Figure 7 The drop-down menu displays.

Task 1: Configure the HTML

Launch a text editor and open the index.html file. You will modify the **nav** area to contain a new unordered list with hyperlinks to the Breakfast, Lunch, and Dinner pages. You will configure a new **ul** element that is contained *within* the Cuisine **li** element. The new **ul** element will contain an **li** element for each room.

```
<nav>
<ul>
  <li><a href="index.html">Home</a></li>
  <li><a href="coffee.html">Coffee</a></li>
  <li><a href="cuisine.html">Cuisine</a>
    <ul>
      <li><a href="breakfast.html">Breakfast</a></li>
      <li><a href="lunch.html">Lunch</a></li>
      <li><a href="dinner.html">Dinner</a></li>
    </ul>
  </li>
  <li><a href="directions.html">Directions</a></li>
  <li><a href="contact.html">Contact</a></li>
</ul>
</nav>
```

Save the file and display it in a browser. Don't worry if the navigation area seems a bit garbled—you'll configure the submenu CSS in Task/Step 2. Next, edit the **nav** area in each page (coffee.html, cuisine.html, breakfast.html, lunch.html, dinner.html, directions.html, and contact.html) as you did in the index.html file.

Task 2: Configure the CSS

Launch a text editor and open the bistro.css file.

a. Configure the submenu with absolute positioning. Recall from **Lecture 7** that absolute positioning precisely specifies the location of an element outside of normal flow in relation to its first parent non static element. The **nav** element's position is static by default so add the following declaration to the styles for the nav element selector: `position: relative;`

b. The submenu that displays the hyperlinks for the Breakfast, Lunch, and Dinner pages is configured using a new ul element that is contained within the existing ul element in the nav area. Configure a descendent `nav ul ul` selector and code style declarations to use absolute positioning, #5564A0 background color, 0 padding, left text alignment, and display set to none. The CSS follows:

```
nav ul ul { position: absolute; background-color: #5564A0;
            padding: 0; text-align: left; display: none;
}
```

c. To style each li element within the submenu, use a descendent `nav ul ul li` selector and configure the li elements in the submenu with a border, block display, 8em width, 1em left padding, and 0 left margin. The CSS follows:

```
nav ul ul li { border: 1px solid #00005D;
               display: block;
               width: 8em;
               padding-left: 1em;
               margin-left: 0;
}
```

d. Configure the submenu ul to display when the :hover is triggered for the li elements in the nav area. The CSS follows:

```
nav li:hover ul { display: block; }
```

Test your pages in a browser. The drop-down menu should look similar to **Figure 7**.

Lab Completion / Submission:

Complete all the lab practices. Take the **screenshots** of your completed webpages; put them into a **single** word file and submit it to **Blackboard -> CP160 -> Assessments -> Lab10 / Assignment 10**; due date: today

After-lab Assignment:

1. Create a one-paragraph conclusion of what you have learned during the lab today.
2. Write the HTML for a hyperlink to a video called sparky.mov on a web page.
3. Write the HTML to embed an audio file called soundloop.mp3 on a web page that can be controlled by the visitor.
4. Optional – click the gallery.html in the gallery_transition folder in the lab files package lab10.zip; hover on the images in the gallery and see how the larger images are displayed. Read the gallery.html code to understand how this was implemented. Notice the position, opacity, and transition properties used in the html file.

Submit to **Blackboard -> CP160 -> Assessments -> Lab10 / Assignment 10;** due date: 1 week from today.

More on Project Requirements:

1. Due date: **Dec. 18**
2. To finalize your project, you need to **code** to implement the website. The techniques that you learned after the proposal submission should also be used in your project development. You will **push** the code to GitHub to host your web site there (will be instructed/practiced in the next lab). You will need to complete the **documentation** describing your website and project development.

3. What to be submitted by the due date (Dec. 18)?
    a. **A Word or pdf document**
        i. Include the **GitHub link**(s) into the document (from the links your code and site are accessible; will be instructed in the next week)
        ii. Functions/features/pages descriptions
        iii. Technologies/skills applied
        iv. Novelty and any other specialties worth mentioning
        v. Development procedure
        vi. etc.
    b. A **zipped file** including HTML/CSS code and other associated files (in case GitHub is not accessible)

4. Where to submit?
   BB -> CP160 -> Assessments -> Project

5. The project evaluation will be performed mainly from the following aspects: documentation writing/organization (clarity, easiness to read, formatting, etc.), richness of features implemented in the website, technologies/skills applied in the website development, novelty, amount of coding, etc.. You document should well explain the implemented features, techniques applied, and so on.