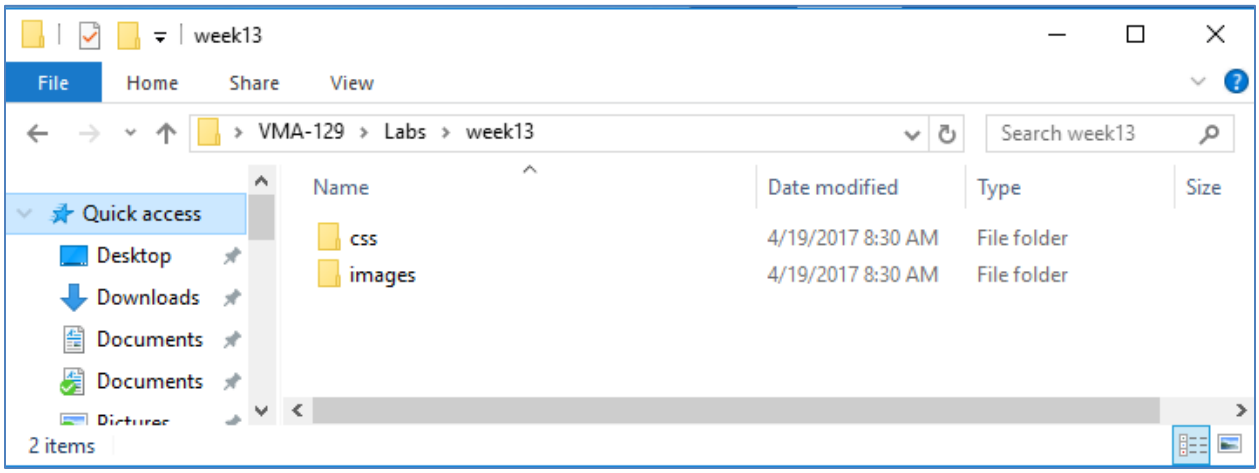
Week 13: Creating a JS Popup Window
VMA-129: VISUAL DESIGN with HTML and CSS

This lab will demonstrate how to create a JavaScript popup window that will appear in the browser once the designated link has been clicked. Create a new site folder named `week13xx` (where xx are your initials). Do not use spaces or capital letters. You will be saving all your work in this folder. Once the lab has been completed you will zip the folder and submit it to Moodle under week 13.

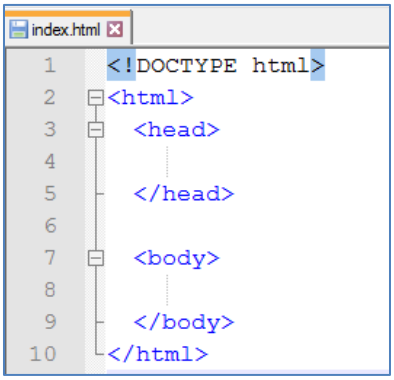
Creating the Lab Site folder:

1. On your storage device create a new site folder and name it `week13xx` (where xx are your initials).
2. Inside of the new site folder create a folder named `css` and a second folder named `images`. Your folder structure should look like the one below.



Creating the `index.html` File:

1. Open a text editor and type in the HTML Document tree.
2. Save the page in the site folder with the name index.html.



3. In the head section add the 4 meta elements that are required for this course.

```
<meta charset="utf-8">
<meta name="keywords" content="Popup Window lab">
<meta name="description" content="Popup Window Lab using JavaScript">
<meta name="author" content="student name">
```

4. Add the title element under the last meta element in the head section.

```
<title>Popup Window Lab</title>
```

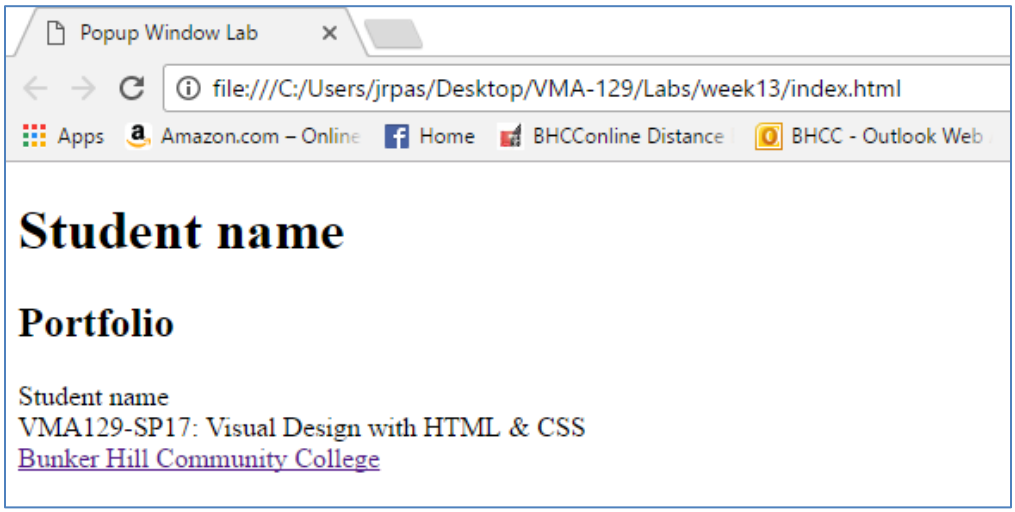
5. In the body section of the document add the following semantic structural elements:

```
<body>
  <header>
    <h1>Student name</h1>
  </header>

  <main>
    <h2>Portfolio</h2>
  </main>

  <footer>
    <p>Student name<br>
      VMA129-SP17: Visual Design with HTML &amp;amp; CSS<br>
      <a href="http://www.bhcc.mass.edu/" title="Bunker Hill
      Community College" target="_blank">Bunker Hill Community College</a>
    </p>
  </footer>
</body>
```

6. Save your work and view it in a browser. It should look like the image below.



Adding An External Style Sheet:

1. Create a style sheet and name it **`week13style.css`** and save the file in the css folder that is nested inside of the site folder. This is crucial for linking the style sheet to the html file with file pathing.
2. Add the following declaration to the style sheet so older browsers that do not support HTML5 will be able to interpret the following HTML5 elements.

```
article, aside, footer, header, nav, main, section {
    display: block;
}
```

3. Save your changes.

4. Add a base font for you site:

```
body {
    font-family: Ariel, Helvetica, sans-serif;
    font-size: 0.875em; /* 14px */
    line-height: 1em; /* 16px */
    color: #9fa8a3;
    background-color: #c5d5cd;
}
```

5. Return to the `index.html` file and link the style sheet to the page.

```
<link rel="stylesheet" type="text/css" href="css/week13style.css">
```

6. Save your changes and test the `index.html` page in a browser to make sure the styles are being applied.
7. Return to `week13style.css` file and add padding for the following structural elements.
8. Create visual separation between structural elements by adding the formatting to the footer element. Adding padding will create space between the edge of the element and the text. The clear property will remove other properties that might interfer with the appearance of the footer.
9. Continue creating visual space by adding the following background color to the footer element.
10. Add a background color the main element to complete defining the visual separation between elements.
11. Have the first `<h2>` element stand out by changing the color and adding a shadow by applying a CSS Pseudo-element to the `week13style.css` file.
12. Save your work and view your changes in a browser. Your work should look like the example below:

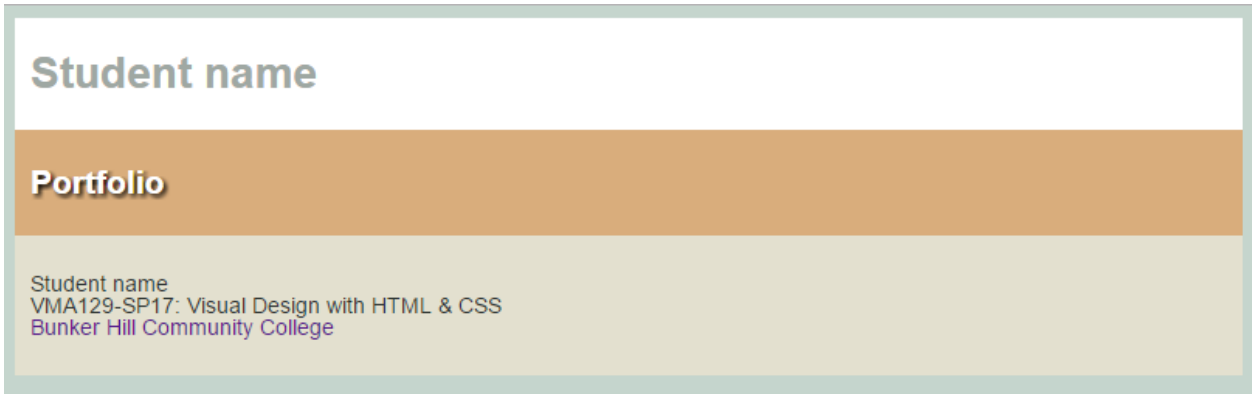
```
footer, header, main {
    padding: 10px;
}

footer {
    background-color: #e3e0cf;
    color: #303633;
    clear: both;
}

header {
    background-color: #ffffff;
}

main {
    background-color: #d9ad7c;
}
```

```
h2::first-line {
    color: #ffffff;
    text-shadow: 2px 2px 4px #000000;
}
```



Adding the Flexbox Layout:

Flexible boxes, or flexbox, is a new layout mode in CSS3. Use of flexbox ensures that elements behave predictably when the page layout must accommodate different screen sizes and different display devices. For many applications, the flexible box model provides an improvement over the block model in that it does not use floats, nor do the flex container's margins collapse with the margins of its contents.

1. Return to the Create `week13style.css` file and at the bottom of the page add the following class to apply the flexbox to any HTML element. This class will define the space where the other elements will be added to the page and keep them together.
2. Add the following class to apply a container to hold the image element and have each element float to the right of the pervious element.
3. Save your work.
4. Return to the `index.html` file and add a blank line under the `<h2>` element with the text Portfolio nested between the open and close tags.
5. Add the `<div>` element with the following class that will define the start and end points of the flexbox that was defined in step 1.
6. Add the `<div>` element with the `portimage` class three times. Your code should look exactly like the code sample to the right.
7. Save your work and view it in a browser. Your page should look like the image below.

```
.portbox {
    display:  -webkit-flex;
    display: flex;
    background-color: #d9ad7c;
    justify-content: center;
}
```

```
.portimage {
    background-color: #f0f4f2;
    width: 260px;
    height: 160px;
    margin: 10px;
    padding: 10px 0px 0px 10px;
}
```

```
<main>
  <h2>Portfolio</h2>

 <div class="portbox">
  <div class="portimage"> </div>
  <div class="portimage"> </div>
  <div class="portimage"> </div>
 </div>
</main>
```

Adding Images:

This page is now ready to hold thumbnail images of the pictures that will be used with the JavaScript Popup window. The final page will have six images total displayed in the Portfolio section. There are two image sizes for each picture. The smaller of the two will be used in the flexbox layout that was just created in the previous section. The larger image will be displayed when a visitor clicks on the thumbnail. To complete this section of the lab you will need to download the `lab_images.zip` folder and move the 12 files into the `images` folder for lab. Now you can add the images to the html page.

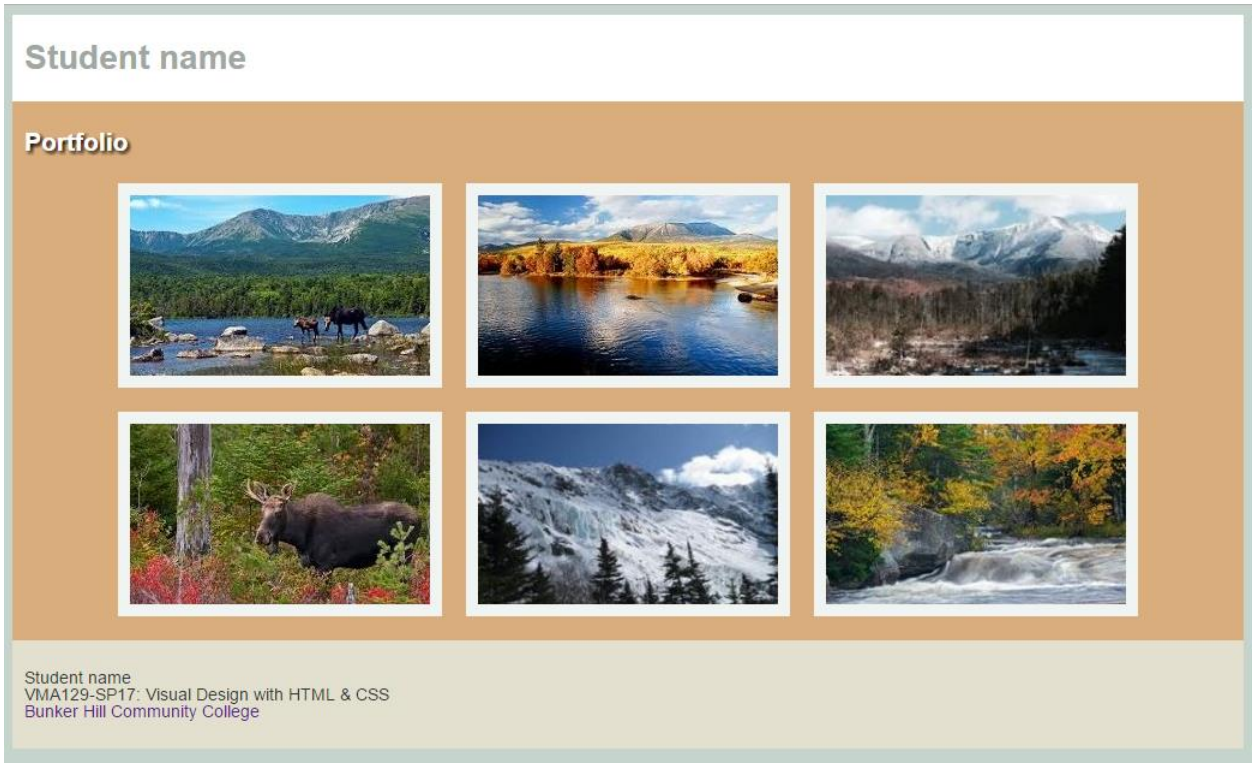
1. Return to the `index.html` file and update each `<div>` element to reflect one of the images in the folder and add the `title` attribute with descriptive text of each image.

```
<div class="portbox">
 <div class="portimage">
   <img src="images/baxter1.jpg" alt="Moose in lake at Mt. Katahdin" title=
   "Click on image for larger image">
 </div>
 <div class="portimage">
   <img src="images/baxter2.jpg" alt="Millinocket Lake with Mt. Katahdin in the
   background" title="Click on image for larger image">
 </div>
 <div class="portimage">
   <img src="images/baxter3.jpg" alt="Mt. Katahdin in Winter" title="Click on
   image for larger image">
  </a>
</div>

<div class="portbox">
 <div class="portimage">
   <img src="images/baxter4.jpg" alt="Moose at Baxter State Park" title="Click
   on image for larger image">
 </div>
 <div class="portimage">
   <img src="images/baxter5.jpg" alt="Mt. Katahdin after the snow" title="Click
   on image for larger image">
 </div>
 <div class="portimage">
   <img src="images/baxter6.jpg" alt="Stream inside Baxter State Park" title=
   "Click on image for larger image">
 </div>
</div>
```

The `alt` attribute is important for Web Accessibility compliance and will display if for some reason the image is not available. The `title` attribute will provide a tip to the user when they hover over the image with their mouse.

2. Save your work and view it in a browser.



Creating the JS Popup Window Feature:

The page is now ready to add some JavaScript to enable a popup window to display each image in a larger window for viewing purposes.

1. Return to the `head` section of the `index.html` file.
2. Add a blank line under the stylesheet link.
3. Add the open `<script>` element followed by a blank line and then close the `</script>` element on its own line.
4. Define the JavaScript function as it is shown below.

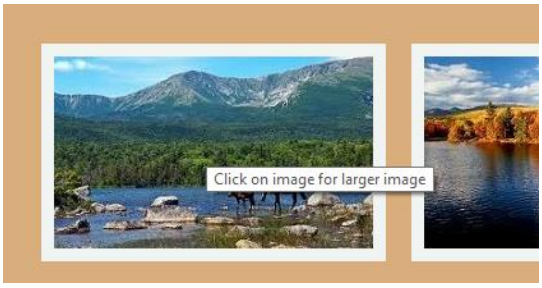
```
<script>
  function showPopup(url) {
  newwindow=window.open(url, 'name','height="400,width=625, top=200, left=300,
  resizable');
  if (window.focus) {newwindow.focus()}
  }
</script>
```

5. Save your work.
6. Locate the first `<img>` element in the body of the `index.html` page and add the following link element.

```
<div class="portimage">
<a href="images/baxter1a.jpg" onClick='showPopup(this.href);return(false);'
title="Moose in lake at Mt. Katahdin">
  <img src="images/baxter1.jpg" alt="Moose in lake at Mt. Katahdin" title=
  "Click on image for larger image">
</a>
</div>
```

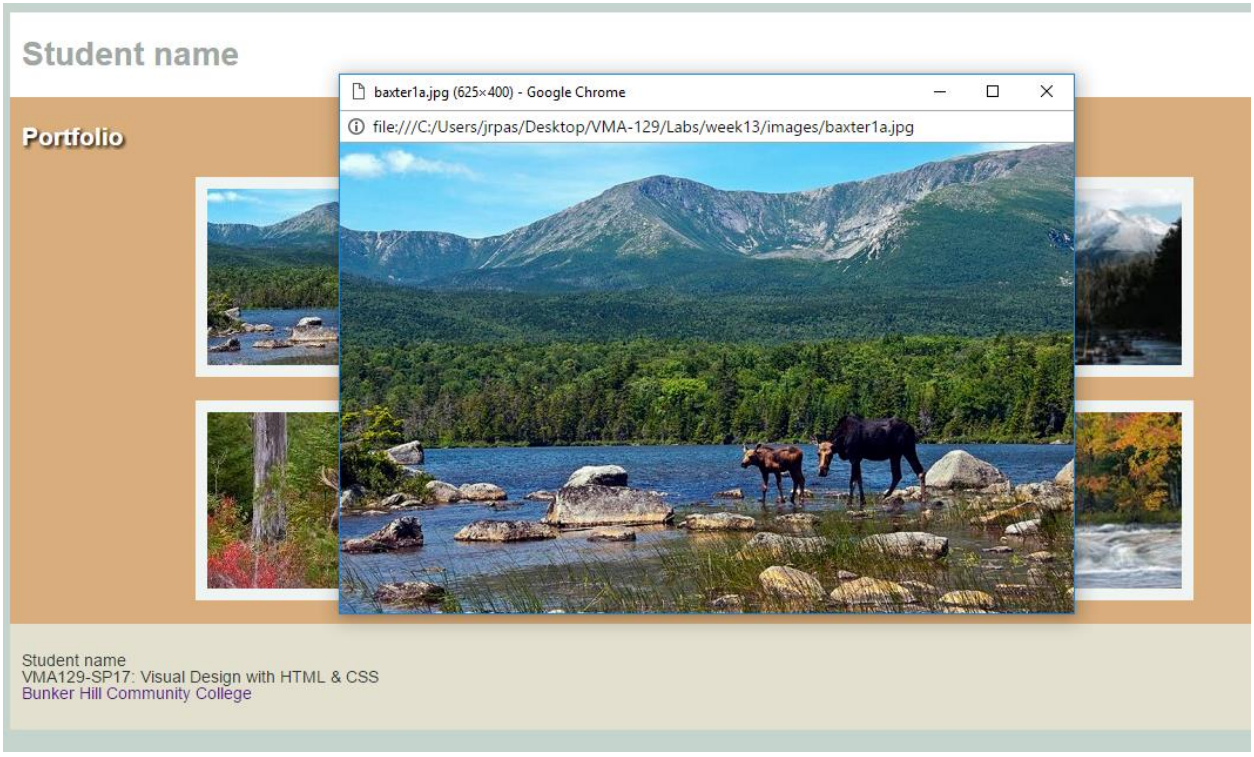
Add Element

7. Save your work and then test the first image a browser.



Note: When you hover over the image the title box is displayed.

8. Click on the image. A popup window should appear over the page displaying the image that was selected as demonstrated in the image below.



9. If the link works then you will need to add it to **each** of the images in the code. Update each image to reflect each picture and then test each image in the browser.

Validate Your Code

Why validate your code? Validation is one of the simplest ways to check whether a page is built in accordance with Web standards, and provides one of the most reliable guarantees that future Web platforms will handle it as designed. As a new coder of HTML and CSS you *will* make many syntax mistakes and errors along the way of learning how to code a webpage. Using a Validator will speedup finding mistakes

https://validator.w3.org/docs/why.html

Now that you have coded the index.html webpage it is now time to validate the code.

1. In a browser type in the web address https://validator.w3.org
2. Click on the **Validate by File Upload** tab
3. In the **Upload a document for validation:** section, select the **Choose File** button and navigate to your storage location for this lab and locate the index.html file and then click on the **Check** button.
4. You should get a report back stating **Document Checking Complete. No errors or warning to show.**

   Or

   You will get a report showing errors in the code you just validated. This program will not correct those errors. You will need to read through the report and go through each line of code in the index.html document that is mentioned and correct those errors yourself. Once you have finished making the necessary corrections upload the file to the validator again. If more issues persist you will need to go back and resolve them.