## In-Class Exercise/Demo

## In-Class Exercise / Demo

- Because this week's demo exercises are also all done on the browser console, you will be tasked to take a screenshot of your console for every exercise you finished. Below are instructions on how to take screenshots on your computer:

PC Windows:

**Win + Alt + Print Screen** – *Captures only the active window*. This command saves an image to *C:Users<user name>Videos>Captures* by default.

MAC:

**Command + Shift + 4** – Marquee the area of the screen you want to screenshot. Saves the screenshot as a PNG file on your desktop.

## In-Class Exercise / Demo

- We will continue with using the browser console to practice today's demo exercises. Go ahead and open a new browser tab in your chrome browser. And then press the following to open the console:
  - **cmd + option + j** (mac)
  - **ctrl + shift + j** (windows)
- The first JavaScript concept we'll practice is loops. As noted in the lecture slides earlier, there are three type of loops. We'll practice all three types and a couple of variations as well.

## In-Class Exercise / Demo

- First up – the **for** loop. For loops are great for counting through a set of finite steps. Enter the following on your console window and then press the enter or return key on your keyboard:

```
for (var i = 0; i < 10; i++) {
    console.log(i);
}
```

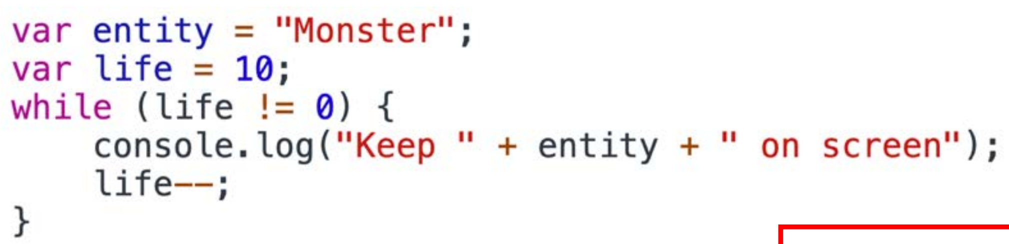
Results printed on console:

```
0
1
2
3
4
5
6
7
8
9
```

**Don't forget to take a screenshot!**

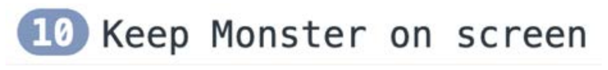
## In-Class Exercise / Demo

- Next – the **while** loop. While loops are useful for repeating a task or set of tasks until the condition is no longer true. Enter the following on your console window and then press the enter or return key on your keyboard:



```
var entity = "Monster";
var life = 10;
while (life != 0) {
    console.log("Keep " + entity + " on screen");
    life--;
}
```

Results printed on console:



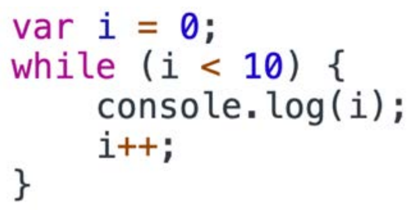
```
10 Keep Monster on screen
```

(print repeated 10 times)

**Don't forget to take a screenshot!**

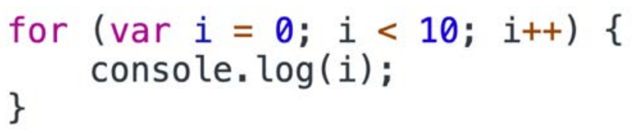
## In-Class Exercise / Demo

- This demo is to demonstrate that a **while** loop can also be used just like a *for* loop. Enter the following on your console window and then press the enter or return key on your keyboard:



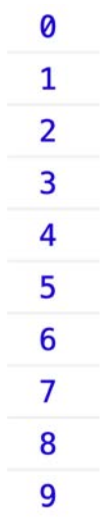
```
var i = 0;
while (i < 10) {
     console.log(i);
     i++;
}
```

for loop example:



```
for (var i = 0; i < 10; i++) {
    console.log(i);
}
```

Results printed on console:



```
0
1
2
3
4
5
6
7
8
9
```

Don't forget to take a screenshot!

## In-Class Exercise / Demo

- The next loop is the **do..while** loop. This loop’s main difference is that it will execute the code at least once even if the loop fails or evaluated to false right when it starts. We will look at this particular example in the next slide. Enter the following on your console window and then press the enter or return key on your keyboard:

```
var entity = "Monster";
var life = 10;

do {
    console.log("Keep " + entity + " on screen ");
    life--;
} while (life != 0);
```

Results printed on console:

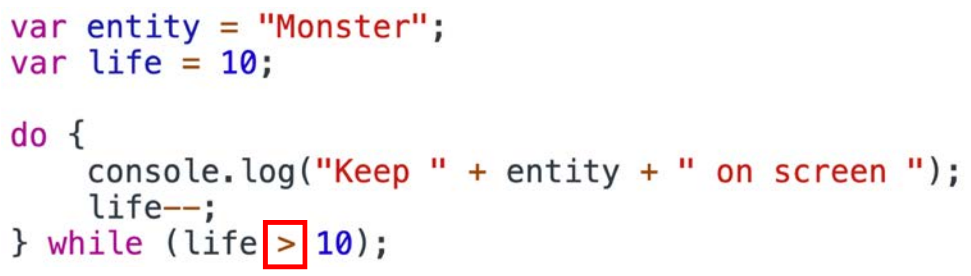
```
10 Keep Monster on screen
```

(print repeated 10 times)

**Don’t forget to take a screenshot!**

## In-Class Exercise / Demo

- This example shows how **do..while** loop will execute the code at least once even if the loop fails or evaluated to false right when it starts. Enter the following on your console window and then press the enter or return key on your keyboard:



```
var entity = "Monster";
var life = 10;

do {
    console.log("Keep " + entity + " on screen ");
    life--;
} while (life > 10);
```

Results printed on console:

```
Keep Monster on screen
```

(prints 1 time)

**Don’t forget to take a screenshot!**

## In-Class Exercise / Demo

- Next – **arrays**. As mentioned in the lecture, arrays are just another type of variable. But a more complex and dynamic variable. Because it can store multi dimensional data, unlike a regular variable. To start, let’s create a standard array that contains a small list of items. Enter the following on your console window and then press the enter or return key on your keyboard:

```
var fruits = ["apples", "oranges", "pears"];
```

Nothing printed after hitting enter or return at this time.

# In-Class Exercise / Demo

- Nothing will happen to the array and its items until you tell it what to do. That “to do” are **methods**. Arrays have many methods and we will practice some examples. With the *fruits* array already created in the console, we can write methods by refencing the array name (*fruits*). The first method we will practice is the **push** method. Either on the existing console window or a  cleared console window, enter the following on your console window and then press the enter or return key on your keyboard:

```
fruits.push("grapes");
4
```

indicates number of items on the list after hitting enter or return

- Then enter *fruits* and hit enter or return to check the updated array list:

```
fruits;
▶ (4) ["apples", "oranges", "pears", "grapes"]
```

grapes is added to the array (at the end of the list)

**Don’t forget to take a screenshot!**

## In-Class Exercise / Demo

- The next method is **pop**. Pop removes the *last* item on the array list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:

```
fruits.pop();
"grapes"
```

indicates item removed from the list after hitting enter or return

- Then enter *fruits* and hit enter or return to check the updated array list:

```
fruits;
▶ (3) ["apples", "oranges", "pears"]
```

grapes was removed from the array (at the end of the list)

**Don’t forget to take a screenshot!**

## In-Class Exercise / Demo

- The next method is **shift**. Shift removes the *first* item on the array list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:

```
fruits.shift();
"apples"
```

indicates item removed from the list after hitting enter or return

- Then enter *fruits* and hit enter or return to check the updated array list:

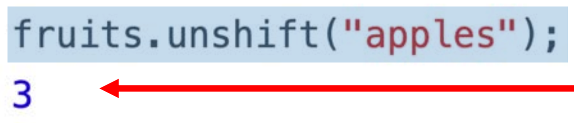
```
fruits;
▶ (2) ["oranges", "pears"]
```

apples was removed from the array (at the front of the list)

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

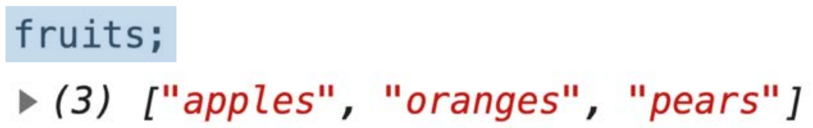
- The next method is **unshift**. Unshift adds an item to the front on the array list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:



```
fruits.unshift("apples");
3
```

indicates number of items on the list after hitting enter or return

- Then enter *fruits* and hit enter or return to check the updated array list:



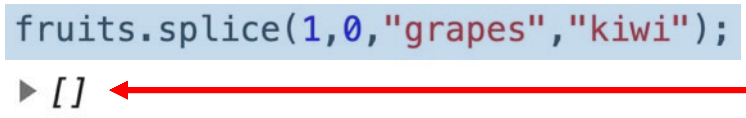
```
fruits;
▶ (3) ["apples", "oranges", "pears"]
```

apples was added to the array (at the front of the list)

**Don't forget to take a screenshot!**

# In-Class Exercise / Demo

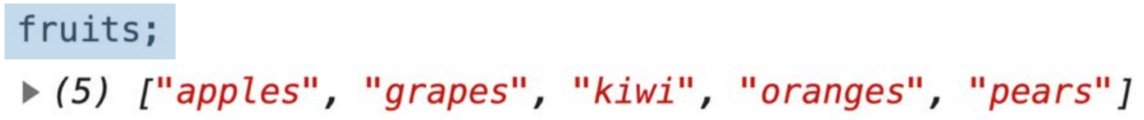
- The next method is **splice**. Splice add or remove items on the array list. The first two integers represents: 1) position on the array items to add or remove 2) number of items to remove (optional). Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:



```
fruits.splice(1,0,"grapes","kiwi");
▸ []
```

indicates array list updated after hitting enter or return

- Then enter *fruits* and hit enter or return to check the updated array list:



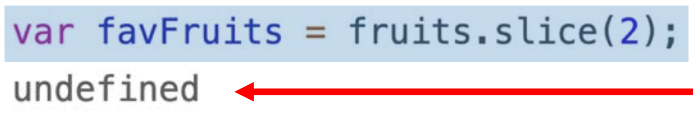
```
fruits;
▸ (5) ["apples", "grapes", "kiwi", "oranges", "pears"]
```

grapes, kiwi was added to the array (at the specified position of the list)

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

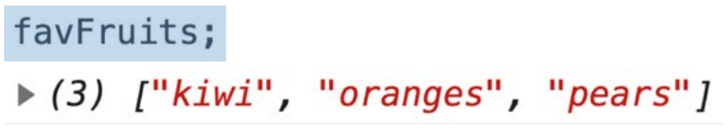
- The next method is **slice**. Slice remove specified number of items at the front of the array list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:



```
var favFruits = fruits.slice(2);
undefined
```

indicates no expected result after hitting enter or return

- Then enter *favFruits* and hit enter or return to check the updated array list:



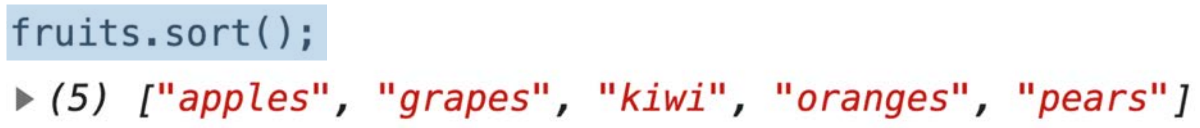
```
favFruits;
▶ (3) ["kiwi", "oranges", "pears"]
```

apples, grapes was removed from the array (at the front of the list)

**Don’t forget to take a screenshot!**

## In-Class Exercise / Demo

- The next method is **sort**. Sort sorts items by alphabetically order (a→ z) of the array list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:



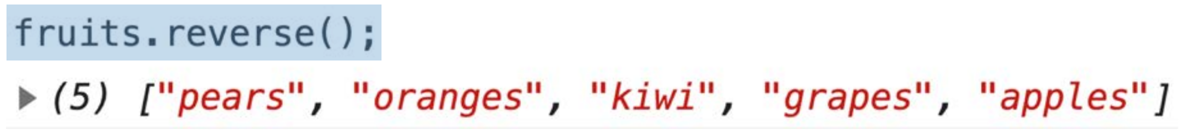
```
fruits.sort();
▶ (5) ["apples", "grapes", "kiwi", "oranges", "pears"]
```

prints sorted array items after hitting enter or return

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

- The next method is **reverse**. Reverse sort items in the reverse order of the array list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:



```
fruits.reverse();
▶ (5) ["pears", "oranges", "kiwi", "grapes", "apples"]
```

prints reverse order of array items after hitting enter or return

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

- The next method is **concat**. Concat combines two array items into one array list. Because it will create a new list, a new array should be created to store this list. Continuing from where you left off on the existing console window, enter the following on your console window and then press the enter or return key on your keyboard:

```
var beenThereList = ["New York City","London","Rome"];
var bucketList = ["Shanghai","Santiago"];

var myList = beenThereList.concat(bucketList);
undefined
```

indicates no expected result after hitting enter or return

- Then enter *myList* and hit enter or return to check the new array list:

```
myList;
▶ (5) ["New York City", "London", "Rome", "Shanghai", "Santiago"]
```

both array lists combined into one array

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

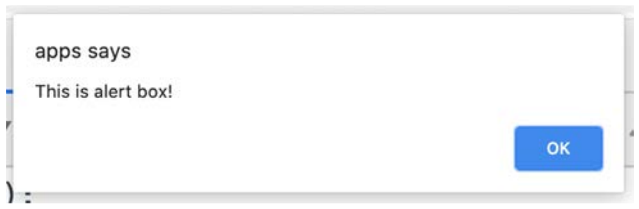
- The last JavaScript concept in our demo today is **popups**. In the lecture earlier, you'd learned there are three type of popups: **alert**, **confirm** and **prompt**.
- Our first example is the **alert popup**. Alert will trigger the browser window to open a small popup box that contains some message. On a cleared console, enter the **either** one of the following and then press the enter or return key on your keyboard:

```
window.alert("This is alert box!");  // display string message
```

or

```
alert("This is alert box!");  // display string message
```

Alert popup box displays after hitting enter or return:



apps says
This is alert box!

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

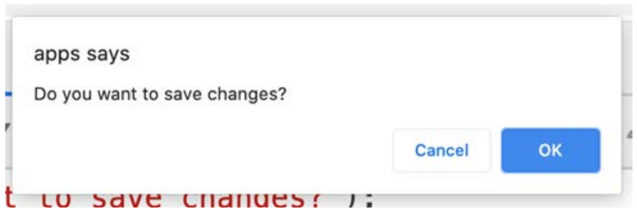
- Our next example is the **confirm popup**. Confirm will trigger the browser window to open a small popup box that contains some message that you specified and a button each to accept or to cancel. On a cleared console, enter the **either** one of the following and then press the enter or return key on your keyboard:

```
window.confirm("Do you want to save changes?")
```

or

```
confirm("Do you want to save changes?");
```

Confirm popup box displays after hitting enter or return:


apps says

Do you want to save changes?

**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

- The last example is the **prompt popup**. Prompt will trigger the browser window to open a small popup box that contains some message and an input field. On a cleared console, enter the **either** one of the following and then press the enter or return key on your keyboard:

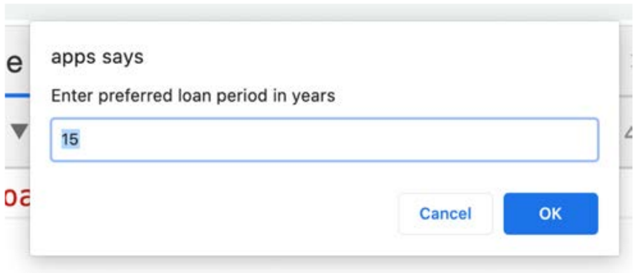
```
window.prompt("Enter preferred loan period in years", "15");
```

or

```
prompt("Enter preferred loan period in years", "15");
```

optional

Prompt popup box displays after hitting enter or return:


apps says

Enter preferred loan period in years

15

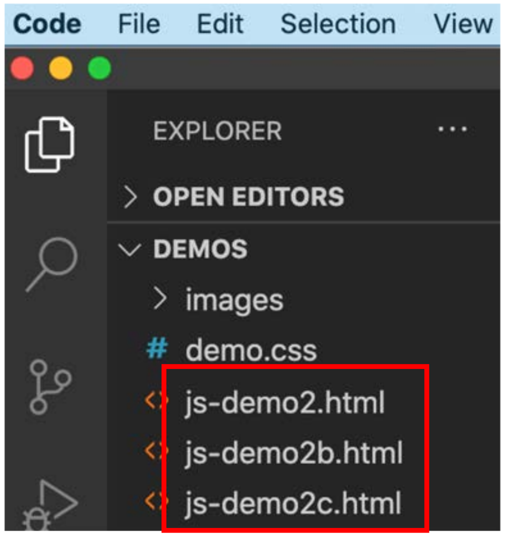
**Don't forget to take a screenshot!**

## In-Class Exercise / Demo

- Popup boxes like *alert*, *confirm* and *prompt* are typically not used in real world applications. They are mainly used in system notifications and also in development environments (ex: testing codes). Therefore, we will not use this feature in our homework project assignment.
- However, we will use loops and arrays in our next demo and homework. They are popular concepts used extensively in applications.
- We are done with our console demo exercises today. So grab all your screenshots and save into a folder.
- Name this folder: **console-exercises**. We will come back to this folder at the end of the next demo.

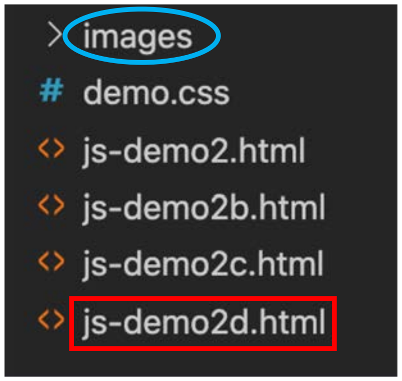
## In-Class Exercise / Demo

- Locate the demo exercise folder *week2-day2*. Make a copy of this folder and name it **week3-day2**.
- Launch VS Code and open this folder within.
- First let's rename the files to:



## In-Class Exercise / Demo

- We will start by creating a fourth page. Duplicate one of the existing pages and name it as **js-demo2d.html:**

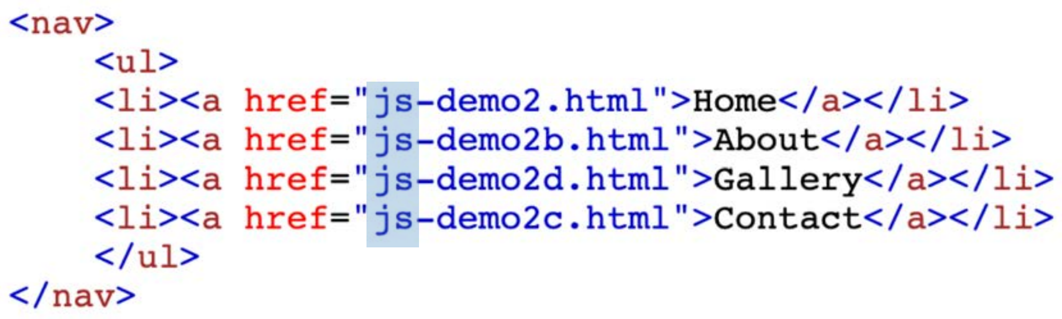


images
demo.css
js-demo2.html
js-demo2b.html
js-demo2c.html
js-demo2d.html

- Also, download the image zip file provided in today's lesson portal and unzip it after download. There should be 6 image files in there – *winterland1.jpg*, *winterland2.jpg* and so on. Move them into the **images** folder within week3-day2 folder.

## In-Class Exercise / Demo

- Next, update the source files of the hypertext links in the navigation area for all four pages to:



```
<nav>
    <ul>
    <li><a href="js-demo2.html">Home</a></li>
    <li><a href="js-demo2b.html">About</a></li>
    <li><a href="js-demo2d.html">Gallery</a></li>
    <li><a href="js-demo2c.html">Contact</a></li>
    </ul>
</nav>
```

## In-Class Exercise / Demo

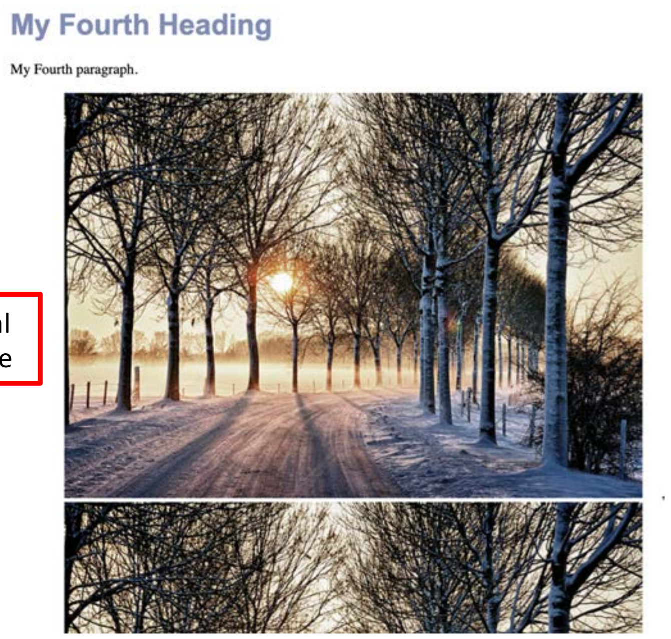
- Open **js-demo2d.html**. First, change the heading and paragraph in the main area to **My Fourth heading** and **My Fourth paragraph** respectively.
- Remove everything below the above paragraph within main and enter the following instead:

```
<h1>My Fourth Heading</h1>
<p>My Fourth paragraph.</p>
```

```
<ul id="album">
    <li id="photo"><img src="images/winterland1.jpg"></li>
    <li id="photo"><img src="images/winterland2.jpg"></li>
    <li id="photo"><img src="images/winterland3.jpg"></li>
    <li id="photo"><img src="images/winterland4.jpg"></li>
    <li id="photo"><img src="images/winterland5.jpg"></li>
    <li id="photo"><img src="images/winterland6.jpg"></li>
  </ul>
```

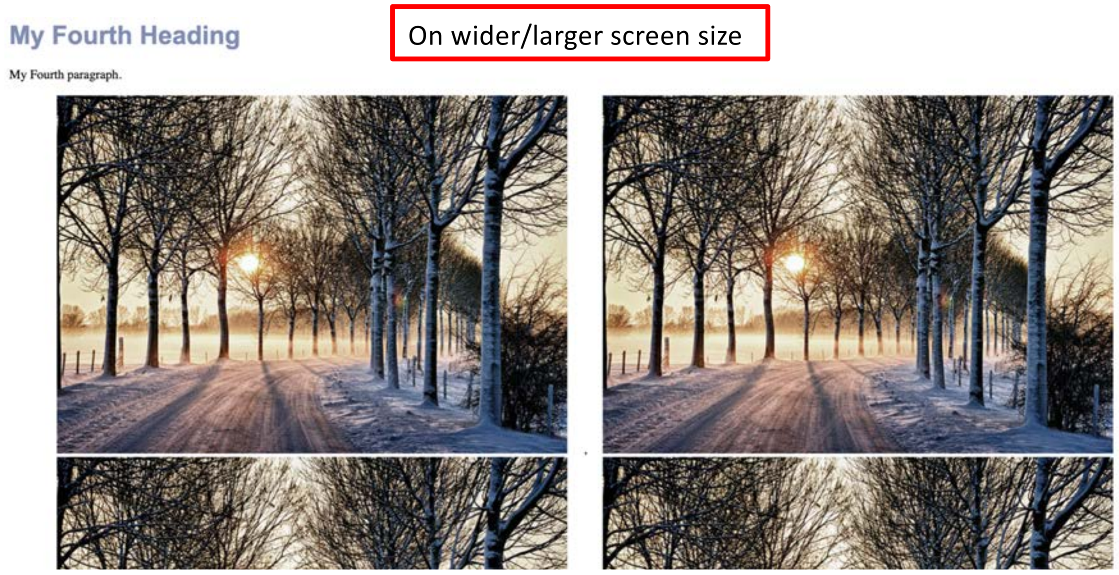
## In-Class Exercise / Demo

- If you preview this page on the browser, it will likely look like one of these, depending on your monitor screen resolution:


My Fourth Heading

My Fourth paragraph.

On normal screen size

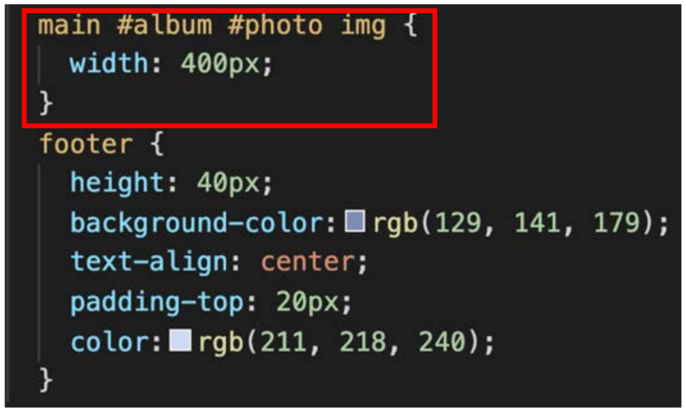

My Fourth Heading

My Fourth paragraph.

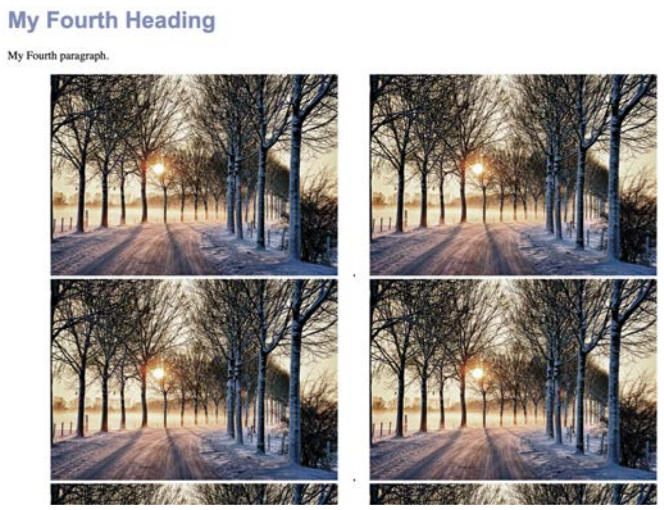
On wider/larger screen size

## In-Class Exercise / Demo

- For a gallery displaying multiple photos (x number of photos per row), the photo dimensions are a little large for an average screen size, which is currently at 640px by 448px. We will reduce the dimension by about half so they can fit two photos (instead of just one) on a row for normal or average screen size using CSS.
- Open **demo.css**. Enter the following CSS codes right above the footer at the bottom of the document:



```
main #album #photo img {
  width: 400px;
}
footer {
  height: 40px;
  background-color: rgb(129, 141, 179);
  text-align: center;
  padding-top: 20px;
  color: rgb(211, 218, 240);
}
```



My Fourth Heading

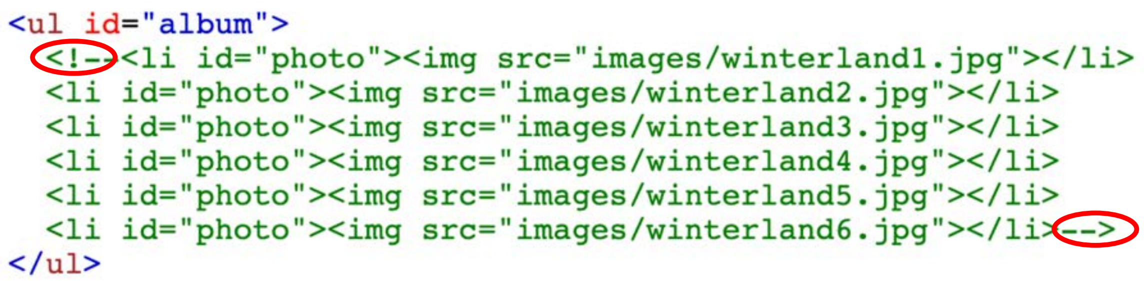
My Fourth paragraph.

## In-Class Exercise / Demo

- So the gallery looks good, right? The only bad thing is right now, the image codes are hard coded on to the HTML markup. If a near future situation arise that you need to swap or update a bunch of photos, it may be a little time consuming to make these code changes, especially if they are scattered throughout the page or site. JavaScript can help in making content easier to manage later on.
- You see, all the photos in the gallery are individual piece of information much like names of students or fruits and so on. That means they can be stored in an array.
- So in the next part of this demo, we will remove or deactivate the gallery markup codes and replace with data from an array. So yes, basically converting HTML to JavaScript to make things more efficient.

# In-Class Exercise / Demo

- First, let's deactivate the gallery codes ie. the list of images:



```
<ul id="album">
  <!--<li id="photo"><img src="images/winterland1.jpg"></li>
  <li id="photo"><img src="images/winterland2.jpg"></li>
  <li id="photo"><img src="images/winterland3.jpg"></li>
  <li id="photo"><img src="images/winterland4.jpg"></li>
  <li id="photo"><img src="images/winterland5.jpg"></li>
  <li id="photo"><img src="images/winterland6.jpg"></li>-->
</ul>
```

- Preview the page on the browser. The images are no longer displayed.

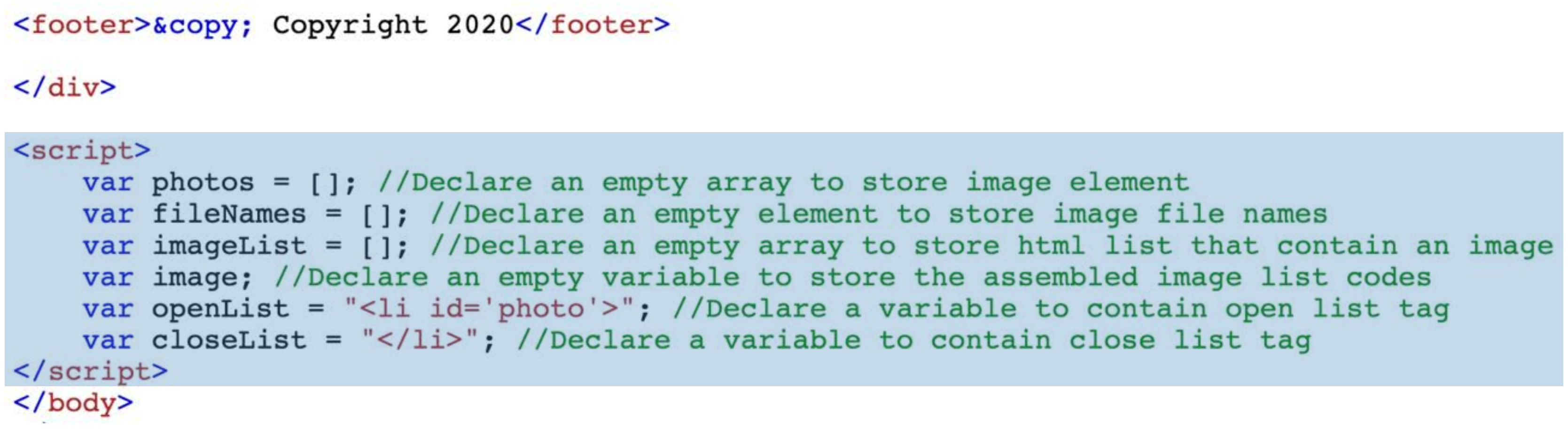
**Note:** what's deactivate? In coding, it means turn the codes into comments, which the browser will ignore. Instead of removing codes, deactivate codes allow you to reactivate them at anytime. You won't be able to do this quick step if it was removed.

## In-Class Exercise / Demo

- As mentioned earlier, we will use JavaScript to display these images from an array. These images as you have seen in the HTML codes, are markup with the following elements and attributes:
  1. <li></li>
  2. <img>
  3. src=“images/filename.jpg”
- We will use JavaScript to help recreate these elements and attributes by storing them in separate arrays and variables. Then we will assemble them together by using these arrays and variables.

## In-Class Exercise / Demo

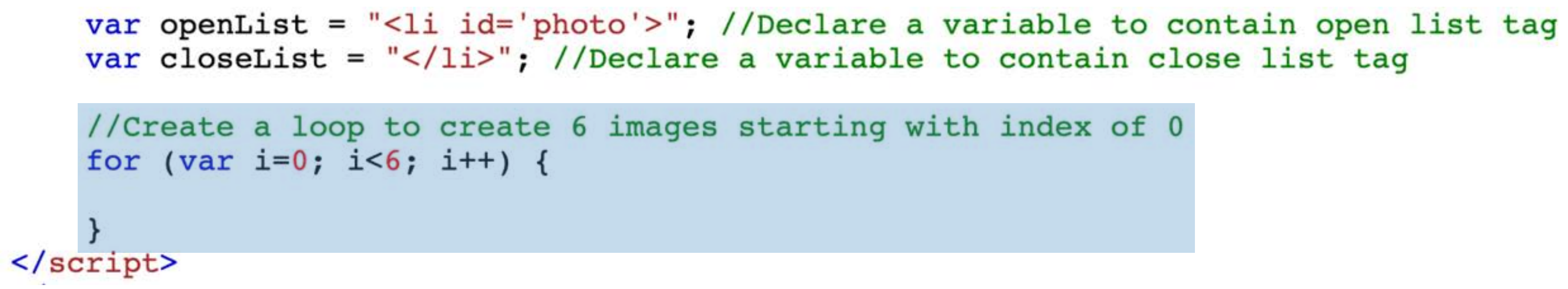
- Next, scroll to the bottom of the page. Right above the close </body> element, create a <script> element.
- Within this <script> element, we'll start by declaring a bunch of arrays and variables:



```
<footer>&copy; Copyright 2020</footer>

</div>

<script>
    var photos = []; //Declare an empty array to store image element
    var fileNames = []; //Declare an empty element to store image file names
    var imageList = []; //Declare an empty array to store html list that contain an image
    var image; //Declare an empty variable to store the assembled image list codes
    var openList = "<li id='photo'>"; //Declare a variable to contain open list tag
    var closeList = "</li>"; //Declare a variable to contain close list tag
</script>
</body>
```

## In-Class Exercise / Demo

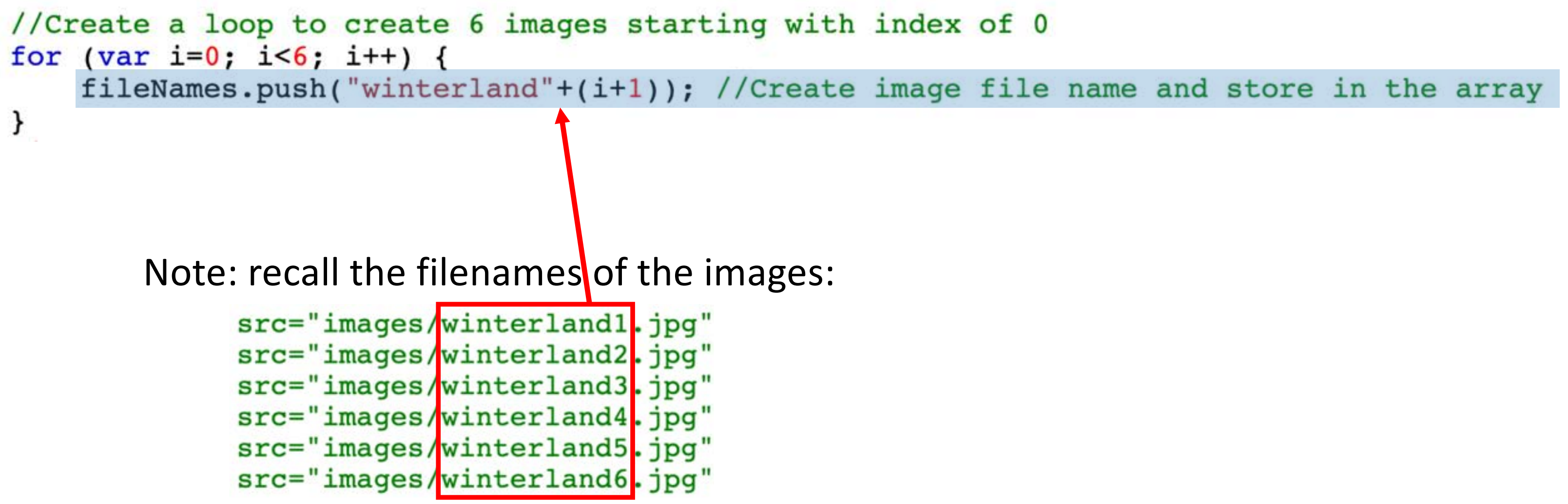
- The first script is to create a loop using the **for** loop to count through six times. The goal is to create 6 images for the gallery. We start from index of 0 to match an array index which starts at 0.



```
    var openList = "<li id='photo'>"; //Declare a variable to contain open list tag
    var closeList = "</li>"; //Declare a variable to contain close list tag

    //Create a loop to create 6 images starting with index of 0
    for (var i=0; i<6; i++) {

    }
</script>
```

## In-Class Exercise / Demo

- Within the for loop, we will write a script to create file names for each of the six images and then store (push) them in an array called **fileNames**.



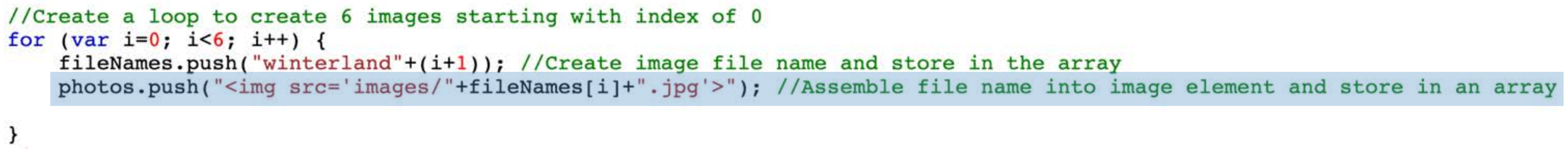
```
//Create a loop to create 6 images starting with index of 0
for (var i=0; i<6; i++) {
    fileNames.push("winterland"+(i+1)); //Create image file name and store in the array
}
```

Note: recall the filenames of the images:

```
src="images/winterland1.jpg"
src="images/winterland2.jpg"
src="images/winterland3.jpg"
src="images/winterland4.jpg"
src="images/winterland5.jpg"
src="images/winterland6.jpg"
```

## In-Class Exercise / Demo

- Next, write a script to assemble file name into an image element and store (push) it in an array called **photos**.
- The assembly will involve concatenating strings and array:
  - String: "<img src=‘images/"
  - Array: fileNames[]
  - String: ".jpg’>"
- Hence, enter the following script:



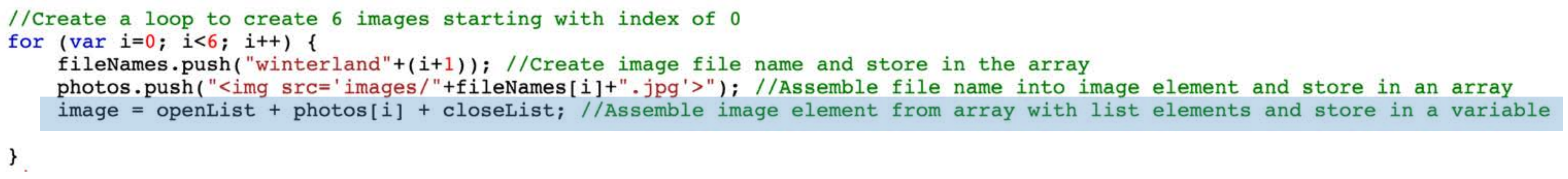
```
//Create a loop to create 6 images starting with index of 0
for (var i=0; i<6; i++) {
    fileNames.push("winterland"+(i+1)); //Create image file name and store in the array
    photos.push("<img src='images/"+fileNames[i]+".jpg'>"); //Assemble file name into image element and store in an array

}
```

# In-Class Exercise / Demo

- Now that we have the image element put together, we will next assemble the HTML list that wraps the image element that if you recall, looks like this:

  **<li id="photo"><img src="images/winterland?.jpg"></li>**

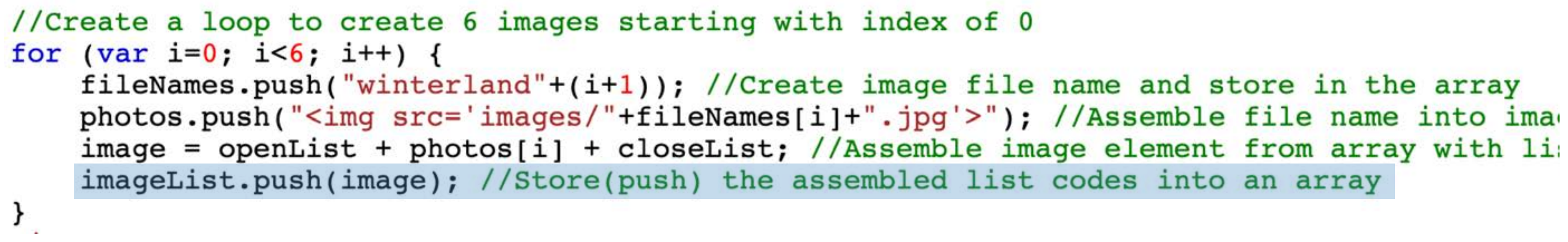
- Just like the previous slide, we will use variables and array to concatenate into the above list. Enter the following scripts:



```
//Create a loop to create 6 images starting with index of 0
for (var i=0; i<6; i++) {
    fileNames.push("winterland"+(i+1)); //Create image file name and store in the array
    photos.push("<img src='images/"+fileNames[i]+".jpg'>"); //Assemble file name into image element and store in an array
    image = openList + photos[i] + closeList; //Assemble image element from array with list elements and store in a variable

}
```

## In-Class Exercise / Demo

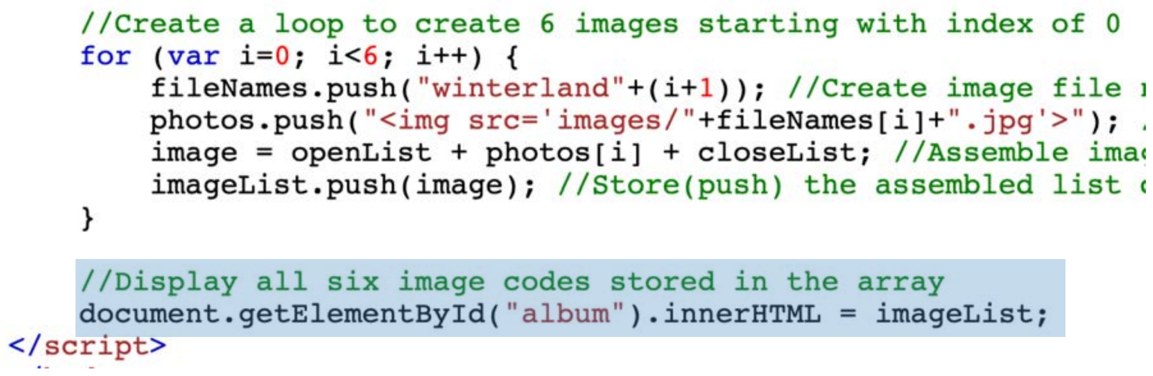
- The final piece of this puzzle is to store what we have put together ie. the entire HTML code that contains a single photo image into an array so we can pull from it when it’s needed.
- Enter the following scripts:



```
//Create a loop to create 6 images starting with index of 0
for (var i=0; i<6; i++) {
    fileNames.push("winterland"+(i+1)); //Create image file name and store in the array
    photos.push("<img src='images/"+fileNames[i]+".jpg'>"); //Assemble file name into ima
    image = openList + photos[i] + closeList; //Assemble image element from array with li
    imageList.push(image); //Store(push) the assembled list codes into an array
}
```

# In-Class Exercise / Demo

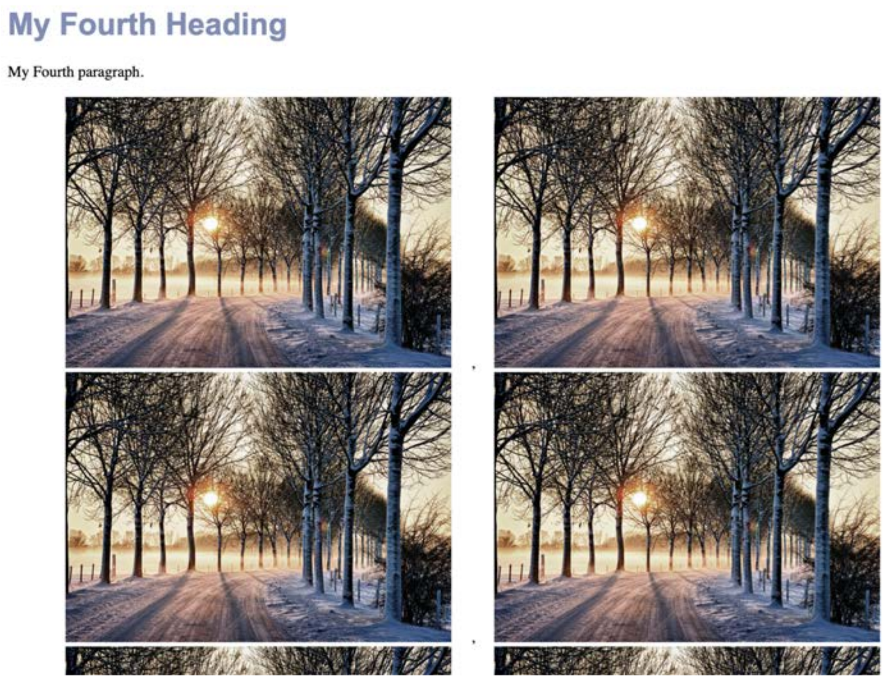
- Now that we have all six photos created, HTML codes put together and stored in an array, it's time to display all of them. We will call that array and tell it to display on the page element.
- Enter the following scripts:



```
    //Create a loop to create 6 images starting with index of 0
    for (var i=0; i<6; i++) {
        fileNames.push("winterland"+(i+1)); //Create image file
        photos.push("<img src='images/"+fileNames[i]+".jpg'>");
        image = openList + photos[i] + closeList; //Assemble imag
        imageList.push(image); //Store(push) the assembled list
    }

    //Display all six image codes stored in the array
    document.getElementById("album").innerHTML = imageList;
</script>
```

## In-Class Exercise / Demo

- Go ahead and preview the page on the browser. All six photos should display again just like before, ie. before we switch them over to JavaScript.



My Fourth Heading

My Fourth paragraph.

## In-Class Exercise / Demo

- **Submission**:
  - Move the **console-exercises** folder into the **week3-day2** folder.
  - Zip **week3-day2**.
  - Submit **week3-day2.zip** in GAP – Week 3 Day 2 dropbox at the end of this class session.