JavaScript Intro Lab

Overview:

In this lab, you shall create a JavaScript animation page named 03_js_intro/index.html that uses object composition of at least two animation objects inside of a background object. Your animation objects will respond to user events. This is a lab about learning JavaScript, so you MAY NOT use jQuery. As always, you cannot submit code that is overly similar to any examples presented in class or on my web site.

Basic Requirements:

1. The first line of your JavaScript code shall have the "use strict" directive as its first line of code (see below). By having this directive, you won't be able inadvertently declare a new variable by misspelling a variable name. It seems odd, but if you forget to use the keyword "var" when first using a variable, a global variable is created, regardless of where the variable was first used. So it is very important that you always use the keyword "var" when declaring variables/objects – so as not to inadvertently pollute the global namespace. Note that you will actually have the double quotes in your JavaScript code when you "use strict". Test that you have this right by trying to use a variable that was never declared with the "var" keyword (and getting an error from the F12 chrome console).

   "use strict";

2. Your code shall use the object oriented JavaScript techniques covered in class, meaning that there will be very few globally declared objects and very few globally defined functions.

   a) Your code shall include a function called makeBackground() that is passed an id to a div that will hold your background image and returns a reference to the background object that it enhanced.

      // Main program
      var bgObj = makeBackground("myBgDiv", ...);

      You may need to modify the size of the background image that you found. This is easy to do in MSPaint (there is a resize menu option, just remember to only specify ONE dimension, letting MSPaint decide about the other dimension, to avoid distortion) or you can do it in Terminal on the MAC. For more help with this, visit my web site (tutorials – front end – web design – working with images). Whenever I am modifying image sizes, I usually like to save the versions with pixel count in the name like “lakeBg_640x960.jpg” (and not ruin the original image).

Your makeBackground function can “hard code” some or all attributes (of the background object it is enhancing) or it can have them passed in as input parameters, but makeBackground shall set whatever properties are needed for JavaScript calculations (instead of relying on any CSS rules). Remember to use parseInt if you want to do math on any of the CSS “style” properties:

      myBgObj.style.height = “500px”; // this (or any “.style” CSS rule) will make a visible change
      var doubleHeight = parseInt(myBgObj.style.height) * 2; // convert to number to do math
In sample code, the background object was a green field that had height and width that the child object (a ball) recognized and would not travel outside. For your lab, identify some areas on the image (by top/left pixel values) that that will have some impact on the animation objects behavior. I created another sample page that has a smurf walking on a background of grass with a river behind. I set a property (of the background object) that indicated the top of the grass. Then I coded the smurf so he never goes into the river.

b) `makeBackground()` shall have a method like `makeAnimation()` that creates a new animation object as a property of the background object (object composition). The main program will invoke the `makeAnimation` method at least twice like this (but of course, use object names that are representative within the context of what you are coding). By passing different input parameters to the `makeAnimation` method, you can make the objects have different properties and/or behave differently.

```javascript
var catObj = bgObj.makeAnimation(...);
var dogObj = bgObj.makeAnimation(...);
```

How can you get your two objects (that were created with the same “constructor” method) to behave different from each other? Your `makeAnimation` method might offer several types of behavior (e.g, does something if you click or something if you hover or something if you press a keyboard key), but when you invoke the “constructor”, you can specify by your input parameters which of these behaviors you want each object to have.

c) This `makeAnimation` method shall:

- Create the animation object by using `document.createElement()` – not by getting a reference to a div that already exists in the HTML. Then it must use `parentObj.appendChild(childObj)` to attach the newly created object to its parent.
- Implement animation of the sprite set image variety (poses that are automatically advanced as time progresses) using an array (declared within `makeAnimation`) of `backgroundPosition` values to select out the poses from the sprite set image. At least one of the objects (created by `makeAnimation`) shall actually use the sprite set animation – the other object does not have to.
- Associate an event with the animation object (like click or onmouseenter or onmouseleave). For a list of DOM events, visit [https://www.w3schools.com/jsref/dom_obj_event.asp](https://www.w3schools.com/jsref/dom_obj_event.asp)

**Modifying sprite set images.** My image editing skills are VERY limited but I am able to get done what I need to do. I use MSPaint to make images smaller, bigger, crop, and/or reverse orientation. I use Paint.net to convert a background color (which is usually white) into transparent and save as a PNG. If you use a MAC, all of these tasks can be accomplished by using the Terminal. Instructions for manipulating images are found in my web site (“Tutorials – Front End – Web Design – Working with Images”). Instructions for making sprites (having a transparent background) is found in my web site also (“Tutorials – Front End – Javascript –How to Make a Sprite”).

Originality and Other Requirements:

1. **Topic.** You can use virtually any idea you have for background object/image with its visual animation objects, as long as you meet the requirements listed just above.

2. **Originality.** You must come up with something original. **Points will be deducted if you submit work that is overly similar to sample code provided from my web site.** Don’t just give me back a version of my code where you have turned my bouncing ball into an animated bouncing character that starts/stops when you click on it. Also, your code shall not be overly similar to any other student’s code. If we find this, we will have to determine who wrote the code and who copied -- this is no fun for anyone, so please do not do this.

   For ideas, go to my web site and see some of my animation examples and blend this code in with what you are doing. Incorporating random numbers can be fun. You can use this little JavaScript method I wrote (that randomly generates an integer within a specified range):

   ```javascript
   // returns an integer in this range: low..high
   function makeRandomNumber(low, high) {
       var random = Math.random();  // returns a real number between 0 and 1
       var diff = high - low + 1;  // determine the distance between the high and low.
       // Math.floor returns an integer truncating any real number after the decimal point.
       return Math.floor(random * diff + low);
   }
   ```

   Go to W3Schools and see examples of how different events are used. Google “simple JavaScript animations” and see what you can find. But wherever you find your sample code, remember to make it your own, use good naming, and convert it to the object oriented way of coding that we have been talking about in class. We will be able to tell if you have just copy/pasted an example from the internet because the code will likely be overly complicated and not meet the object oriented specifications of this lab.

3. **Professionalism.** To get full credit, you have to create a component that is professional and visually appealing. You have had two design labs so you should be in a position to come up with something that looks good.

4. **Look and Feel.** You can use the look and feel of your initial layout lab or your advanced layout lab or anything that makes sense for your animation.

5. **Blog.** For each lab, you will add a blog to your labs page and this lab is no exception. In your blog, link to the work you did ("03_js_intro/index.html") and describe your experience doing this lab. What aspects were easy? Which were hard? What were the most important things you learned in this lab? For this particular lab, you shall also include instructions for us to run your animation.

6. **Good Programming Style.** At the top of the 3344 labs page, there is an entry entitled "Requirements For All Labs". Make sure you adhere to the "good programming and design practices" listed in there.
Grading and Submission:

1. After completing all the requirements, test locally (and syntax check). Don’t forget to write a blog entry from your labs page (that tells us how to test your code) and provide a link to the page you created this week. Then publish, then test what you published.
2. Remember to read (and follow) the entry at the top of the 3344 labs page – the entry entitled “Requirements for all Labs and Projects” – for all labs including this one. This section has a list of requirements that I do not repeat in each lab assignment.
3. Then submit into blackboard a zip file of your whole website (including all the work you have done in all the labs so far). Make sure to include all the necessary files and folders in your zip file.

Here is How We Will Evaluate Your Lab:

1. We will visit your published website, link to your labs page then read the blog for this lab which should include an explanation of how we are to test your animation page (03_js_intro/index.html) as well as a link to your animation page.
2. After linking to your animation page, we should see a background image with two visual objects on it. We will invoke events on each of the objects (as per your instructions) and see if we get the results your instructions indicated we should get.
3. We will view source on your HTML page and check for the program style requirements (listed in the "Requirements For All Labs" section at the top of the 3344 labs page). We will also check that you have followed the basic requirements listed for this lab:
4. We will make a subjective evaluation regarding your topic selection (relevance to your website topic), originality, professionalism, look / feel, quality of blog post with instructions. Remember that even if there are some web design gurus in the class, you are not expected to measure up to their work, just show mastery of the topics presented in class so far.

Suggested Approach

1. Carefully study the links that were provided (on the 3344 labs page under this lab).
2. Remember that this lab is about learning JavaScript, so you may not use jQuery for this lab.
3. Always use the NetBeans menu option: Source – Format to keep your code properly indented. Otherwise, it will be very hard to understand your own code. When you click on a starting brace, NetBeans should highlight that brace and its matching ending brace so this also helps you to understand nesting.
4. Pay attention to the red bubbles that NetBeans places to the left of any line of code that has a syntax error. Hover over the red bubble to get an explanation of the error.
5. As you work on this lab (or any lab that involves JavaScript), work in Chrome, press F12, and watch the console tab - this is the only place that you’ll see JavaScript runtime error messages. To debug, add console.log() statements to your code so that you can see them printed out in the console. If you console.log(someObject), the console will show you a tree view of the object where you can explore all the properties of the object.
6. As with ANY programming (and especially if you are new to JavaScript), code just a few lines of code between testing. That way, if something goes wrong, you’ll know which lines of code to fix.
7. When you are trying to associate events with the animation objects, remember to appropriately use the “this” keyword appropriately. “This” means the DOM element that is currently experiencing the event. When using setInterval and/or setTimeout, you have to save a copy of “this” (usually it is copied to “self”) because the setInterval and/or setTimeout JavaScript methods change the value of “this”. Sample code was provided that shows a correct implementation for this (the two balls that start/stop upon click within the box).