Overview:
In this homework you will write some JavaScript code (in an HTML page) where you enhance two DOM objects (HTML elements) so that they have custom properties, custom methods, and respond to user events such as button click or hover.

- 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 or to any classmate.

Requirements:
1. Your page shall include a JavaScript “make” function (similar to a java constructor) that:
   - is passed in the id of an HTML element on the page.
   - gets a reference to the DOM object with the given id.
   - “enhances” the DOM object by adding at least two custom properties.
   - “enhances” the DOM object by adding a mutator method for each custom property (one mutator can be a simple setter, but the other has to be a method that modifies the property in a more complex way).
   - “enhances” the DOM object by adding a method called “display” that makes the custom properties visible on the page. The display method shall be called by each mutator method (so that the current state of all the properties shall always be visible on the page).

For example,

```javascript
function makeCar (id, theCondition, thePrice) {
    var carObj = document.getElementById(id); // get reference to DOM object with the given id
    carObj.condition = theCondition; // first use of “condition” creates custom property
    carObj.price = thePrice; // create custom property “price”
    carObj.display = function () {
        // create custom method “display”
        // make the current properties visible on the page
        carObj.innerHTML = "Car condition: " + carObj.condition + "<br/>price: $" + carObj.price;
    }
    carObj.setCondition = function (newCondition) {
        // create custom method setCondition
        carObj.condition = newCondition;
        carObj.display(); // show updated property on the page
    }
    carObj.changePrice = function (changeRate) {
        // create custom method changePrice
        carObj.price = carObj.price * (1+changeRate);
        carObj.display(); // show updated price on the page
    }
    carObj.display(); // show initial properties on the page
    return carObj;
}
```
2. Your HTML page shall have **at least two divs** (each with their own unique id). JS code (that runs at page load time) shall invoke your “make” function twice, once for each of these divs.

3. Your page shall **allow the user to change each property of each object** through the use of input elements and/or user events. You used input elements in the Layout homework and you have some sample code for this homework that shows you how to access user entered values via JS. See this link to learn about user events: [https://www.w3schools.com/js/js_events.asp](https://www.w3schools.com/js/js_events.asp)

4. The “make” function shall accept an **image** name as input parameter that is rendered by the “display” function of your enhanced object. (See sample code associated with this homework).

   ```javascript
   function makeCar(id, theCondition, thePrice, theImg) {
      var carObj = document.getElementById(id); // get reference to DOM object with the given id
      carObj.condition = theCondition;       // first use of “condition” creates custom property
      carObj.price = thePrice;               // create custom property “price”

      carObj.display = function () {        // create custom method “display” makes properties visible on page
         carObj.innerHTML = "Car condition: " + carObj.condition + "<br/> price: " +
            formatCurrency(carObj.price) + "<br/> <img src='"+theImg+"'>";      // single quotes in double quotes
      }

      carObj.setCondition = function (newCondition) {
         carObj.condition = newCondition;
         carObj.display(); // show updated property on the page
      }

      carObj.changePrice = function (changeRate) {
         carObj.price = carObj.price * (1 + changeRate);
         carObj.display(); // show updated price on the page
      }

      carObj.display(); // show initial properties on the page
      return carObj;
   }
   ```

   Note: You can find any images from Google. You can right click on the image and do a “Image – Save As” to save the image into the proper folder. However, the image cannot be too large (neither visually nor in size of image file). In my website, click on “Tutorials – Web Design” and read the short section entitled “Working With Images”. This will tell you how you can make large images smaller (so that your page does not load slowly) and how to make an image have a transparent background (looks better on the page).

5. 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.

   ```javascript
   "use strict";
   ```

6. **User Instructions.** Your page shall indicate (wherever not obvious) how we are to test your code. We need to be able to change the (at least) two properties of each of the two objects that are being displayed on your page.
Originality and Other Requirements:

1. **Topic.** You can use any idea you want for the topic of your object and its properties, as long as you meet the requirements listed just above. As always, I recommend that you try to select something that is relevant to your web site’s topic since this will make your web site look more professional if you ever need to show your work to a prospective employer.

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 – or to any classmates.*

   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. **Look and Feel.** Your page may have the same look and feel as your home page or it may have a different layout, but to get full credit, you have to create a page 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. **Organization.** Create a folder named “03_js_intro” (under “web pages”) and name the page “index.html”. You are not required to have a nav bar or footer for this page so you do not need to use the Angular include directives.

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?

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. Remember to read (and follow) the entry at the top of the 3344 labs page entitled “Requirements for all Labs and Projects”. This section has a list of requirements that I do not repeat in each lab assignment.

2. After completing all the requirements, test locally (and syntax check). Don’t forget to add the blog entry to your labs page and provide a link to the page you created this week. Then publish, then test what you published.

3. Then submit into blackboard a zip file of your whole web site (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.
Suggested Approach

1. Carefully study the links that were provided (on the 3344 labs page under this lab).
2. 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 (yellow) that brace and its matching ending brace so this also helps you to understand nesting.
3. 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.
4. As you work on this lab (or any lab that involves JavaScript), test your pages 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.
5. 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.

Here is How We Will Evaluate Your Homework:

1. We will visit your published web site, link to your labs page then read the blog for this lab and link from there to your page for this homework: 03_js_intro/index.html.
2. We will test your page, changing both properties of both objects.
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 web site topic), originality, professionalism, look / feel, quality of blog post with instructions.
5. Even though there may be some students with web design experience, you are not expected to measure up to their work, just show mastery of the topics presented in class so far.

Sample Deductions:

-9 if no zip file submitted into Canvas (by the due date).
-9 if page not published by the due date.
Up to -2 for no blog on labs page. If we cannot find your work (you didn’t link to it and you didn’t name it exactly as requested), you’ll get -9 as if it were not published.
-2 if no image is passed to your “make” function and rendered by the “display” method of your enhanced DOM object.
Up to -5 if you only implemented one object instead of two.
Up to -5 if you only implemented one instead of two user modifiable properties (on two objects).
Up to -3 for not using events (like onmouseover or click).
Up to -3 for poor layout/professionalism.
Up to -2 for coming up with an object and properties that don’t make any sense in any context.
Up to -2 if your page loads too slowly - you should have selected images with small file sizes or doctored up the images so that the total of all images on page is around 500K or less. To test this for yourself before submitting your homework, clear your browser’s cache, then reload your page. Or visit your page from a new computer that you have not used before.
Up to -3 for lack of originality, if your code is too similar to sample code. If it is too similar to a classmate, the deductions can be much more.
Up to -2 for HTML/CSS syntax errors which show in red font from Firefox View Source.
Up to -2 for poor coding style. Your code should be neat, properly indented, well named, and with no unused code.