EECS 1012. LAB 03: HTML+CSS+JS (May 26 – June 1, 2021)

A. IMPORTANT REMINDERS

- The pre-lab mini quiz is considered part of this lab.
- You are required to complete the pre-lab mini quiz 3 posted on eClass no later than the beginning of your lab time, i.e., 10am this Friday if you are in the morning sessions (lab03, 04), and 1:30pm if you are in the afternoon sessions (lab01, 02, 05)
- You are welcome to attend the lab session, if you stuck on any of the steps below. TAs and instructor will be available via Zoom to help you. If you need help, you are expected to come to your own lab session. Zoom links are on eClass.
- Please be patient while you are waiting to be admitted by the zoom host. This is one-to-one session and the TAs admit on first come first served basis.
- Instructor will be around in different lab sessions to help. If you need to talk to the instructor, let the TA know and they would notify the instructor.
- You can submit your lab work anytime before the specified deadline.

B. IMPORTANT PRE-LAB WORKS YOU NEED TO DO BEFORE GOING TO THE LAB

- Download this lab description and the associated files and read it completely. Unzip the compressed file. If uncompressing is successful, you should get a folder **Lab03**, which contains html files, CSS files, JavaScript files, and an image folder with several images.

- You should have a good understanding of
    - Events (such as `onclick, ondblclick`)
    - `document.getElementById().innerHTML`
    - `document.getElementById().src`
    - `document.getElementById().ischecked`
    - `document.getElementById().style.xxx`   (`xxx` corresponds to all CSS style we learnt)

C. GOALS/OUTCOMES FOR LAB

- To learn how to change the *behaviour* of an HTML document using JavaScript.

D. TASKS

Part1: simple JavaScript to change image source and display text.

Part2: Start developing a learning kit that by end of term could include at least 30 computational problems and solutions. Details are as follow.
- You are provided a simple **myLearningKit.HTML** document and supporting files such as **myLearningKit.CSS** and **myLearningKit.js** as well as some images. Your major task is to improve the *behaviour* of the HTML file by providing JavaScript codes.
- You will generate at least four sets of HTML and JS files in this process.
- See next pages for details on how to modify your HTML and JS files.

Part 3: Validate your HTML code using the https://validator.w3.org (upload your .html file)

- If you have errors, fix them and retry the validation until your HTML document is error free.

E. SUBMISSIONS

- eClass submission. More information can be found at the end of this document.

Part I. Simple JS.

In this exercise you practice using simple JavaScript to change the display of message and image source.
This task involves nine files: **light.html**, **light.css**, **light.js**, and six images: **light_0-5.jpg** in **images** folder. The image **light_0.jpg** shows 5 light bulbs all turned off. The images **light_1.jpg** through **light_5.jpg** corresponds to a light being turned on (e.g. **light_2.jpg** is an image of the 2nd light turned on).
Your task it to modify **lights.html** to include 5 buttons as shown below. Each time a button is selected, a JS function should be called, which changes the displayed image to be the corresponding image that shows the light above the button is "turned on". The displayed message is also changed to "*light X is on*".
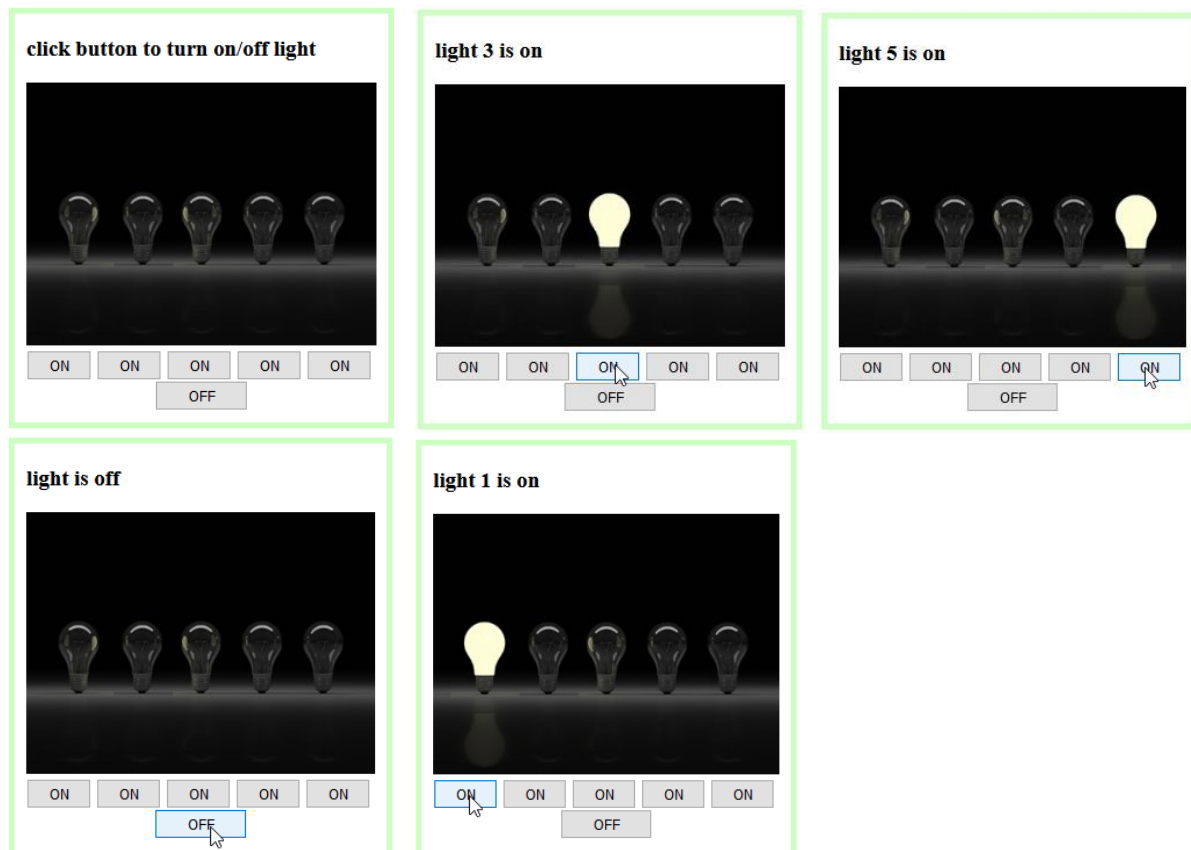
- Modify your HTML code to allow your JavaScript code to change the image and message.
- Add in the five buttons using the `<button>` `</button>` commands.
- There is a CSS defined for these buttons, but you'll need to modify the width of the five buttons so to fit in one row and below each of the five bulbs. Make it as close as possible. The images width is 300 pixels.
- Following the steps such that when a button is clicked, the corresponding light is on, and corresponding message is displayed.
  - Add an event handler for the left most button, something like `onclick="on_1()"`. This tells the browser that when this button is clicked, a JS function called `on_1()` is invoked and executed (you can give other function name if you like, as long as the function with this name is defined in JavaScript).
  - Now implement the function. Open **light.js** where you are already provided with the skeleton of function **function on_1(){...}** .The body of the function is what we write in the pair of curly braces.
    We want to change the displayed image, by updating the content of the `<img>` element, who has an id "img". To do so, first we need to get from our HTML file the element object whose ID is "img", by the following JavaScript code: `document.getElementById("img")`. This would get us a construct, that we know it as an **object,** that corresponds to the element in HTML having id "img". Once you got the object that corresponds to the HTML element, you can set any of its attributes/property of the HTML element via the object, by assigning a new

value, or by using `setAttribute()` method associated with the object. Here since we want to set the `src` attribute of the element with ID "img" to a proper image (which one?), we need to add the following code:

```
document.getElementById("img").setAttribute("src","…");  or, simply
document.getElementById("img").src = "…";
```

where "…" is the URL of the intended image. You can also store the retrieved element object into a variable first and then work on the variable. E.g.,

```
var im = document.getElementById("img");   Then
im.setAttribute("src","…");   or simply
im.src = "…";
```

Next we want to update the displayed message. To do so, we first get the object that corresponds to the HTML file the element whose ID is "message", and then we set the `innerHTML` property to be "*light 1 is on*".

o   Repeat the process for other buttons, including the off button. You may want to define separate functions for each button and then implement the functions by setting the image src with proper images, and updating the displayed message with proper texts.

• Note that since we just started learning JS, we use a simple approach: have a separate function per button. Later when we learn more JS, you will see that another approach would be to have a single function and pass a parameter. That is, click any button will call the same function, but with different input (as parameter to the function). Then the function changes the images according to the input.
• For this exercise, you are free to use CSS to change the border width, style, color etc. (not graded).

Part II. More on JS.

Based on the experience with part I , here we create an interactive course learning kit using JS.

STARTING POINT: **myLearningKit**.**html**, **myLearningKit**.**css**, **myLearningKit**.**js**, and images in the **images** folder. You are given the following HTML file.

This file is not connected to a CSS file. In Exercise 1, you improve the *presentation* of this HTML document by modifying the HTML and CSS files slightly.

Before proceeding further, open **myLearningKit.HTML** in VS Code and carefully learn about its content and structure.

Note that this document eventually is going to be an interface (in particular, a webpage) that contains some basic computational problems together with your flowcharts and implementations. However, it takes a few weeks until we get there. So, for now we (ab)use the document towards demonstrating foods.

Exercise 1: (CREATE: **myLearningKit_Ex1.HTML** and **myLearningKit_Ex1.js)**
Copy **myLearningKit.HTML** to a new file named **myLearningKit_Ex1.html**. Copy **myLearningKit.js** to a new file named **myLearningKit_Ex1.js**.

First make two changes to **myLearningKit_Ex1.html**, as follows:

- Connect it to the provided **myLearningKit.css** by adding a link tag in the head element.
- In the footer element, replace "your name here" with your name.

When you open the HTML file with browser, you should see the following result:
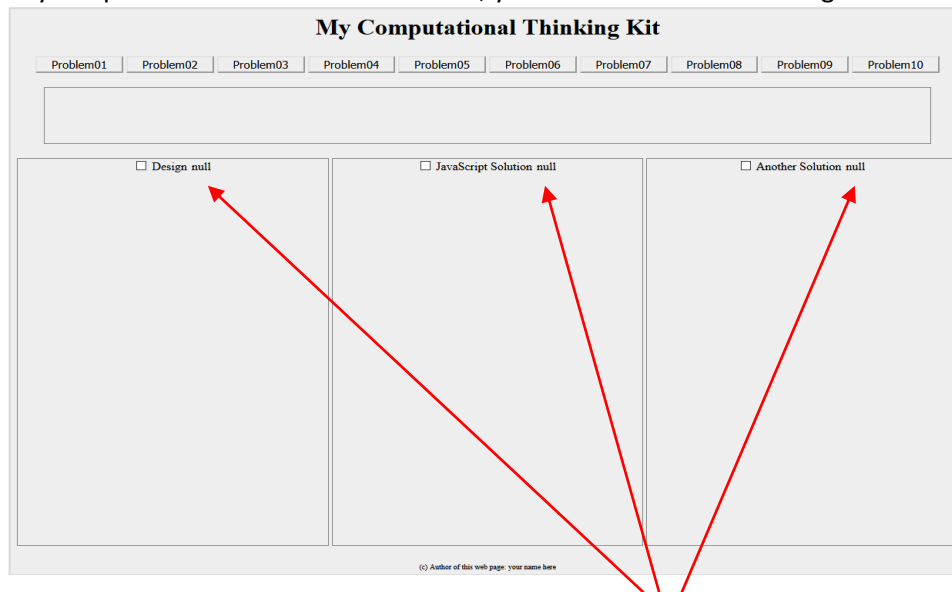
- Now, make a change in the CSS file, such that all the null words disappear.

As you can see in the HTML file, the source file of the images is not set and the alt attributes are set to *null*. In other words, we do not want yet these img tags to display anything. But, in the picture above, the 'alt' word "null" is displayed. If, in the HTML file, we remove the attributes src and alt from the images, our HTML code is no longer valid. Hence, we need to make a change in the CSS file such that these images do not display anything for now. After making the change in the CSS file, you should have the following result:
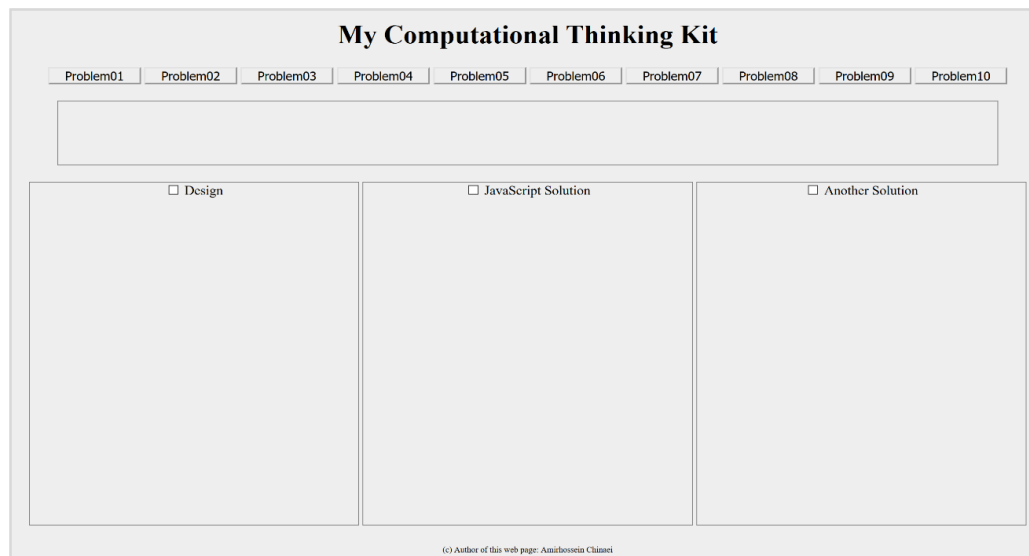
- Now if you click on any of the buttons, nothing happens. In other words, we have not defined any behavior for our page yet. Let's do it step by step.

1) Add an event handling attribute `onclick` to the button designated for Problem01 such that when the button Problem01 is clicked, a JavaScript function **p01Func** is called.

2) Add the link to **myLearningKit_Ex1.js** in the html.

3) Open **myLearningKit_Ex1.js** where you are already provided with the skeleton of function p01Func(). Complete the function. We want to update the content of the "problem" division with the following paragraph:
Basketball is a team sport in which two teams, most commonly of five players each,

opposing one another on a rectangular court, compete with the primary objective of shooting a basketball (approximately 9.4 inches (24 cm) in diameter) through the defender's hoop

To do so, we need to get an object that corresponds to the HTML file element whose ID is "problem". Then, we need to assign the paragraph above to the `innerHTML` of this object. Here is the JavaScript code:

```
document.getElementById("problem").innerHTML="<p>…</p>";
```

where **…** is replaced with the red paragraph above. Now, when you click on the button representing Problem01, you should see the following result.

4) Since the user clicked Problem01, now we want to set the corresponding images for the *Design*, *JavaScript Solution* and *Other Solution* panels.

At this point, you may want to revisit the **myLearningKit_Ex1.html** file and look up the img elements and their IDs. The idea is that for problem01, **basketball_design.jpg** is assigned to the html image element whose ID is "flowchart" and **basketball_js.jpg** is assigned to the html image element whose ID is "js". You also need to download another image for basketball from internet (save it as **basketball_another.jpg** and put in the same folder as the other two) and assign it to the element whose ID is "another".

Note that here we set the image source for the three panels when Problem01 button is clicked, but the images will not be displayed when the button is clicked, do you see why? The reason is that you just set in CSS such that these images do not display anything. Display will be triggered when user checks the checkbox in the panels, which is done in the next exercise.

Exercise 2: (CREATE: **myLearningKit_Ex2.html** and **myLearningKit_Ex2.js)**
Copy **myLearningKit_Ex1.html** to a new file named **myLearningKit_Ex2.html**. Copy **myLearningKit_Ex1.js** to a new file named **myLearningKit_Ex2.js**. Modify the new html file so that the two new files work together. (The CSS file is still the same).

The idea here is to modify the new files such that when we check/uncheck the checkboxes by clicking on them, the images display/disappear, respectively.

1) First, we need to add the **onclick** event to Design checkbox "check1" in **myLearningKit_Ex2.HTML**. This is similar to what you did in Step 1 of Exercise 1 for buttons. When this checkbox is clicked, we want to call a JavaScript function that we name it `checkUncheck1()`.

2) Then, we want to implement the function in **myLearningKit_Ex2.js**. Remove the two lines that we ask you to remove in the **js** file to uncomment `checkUncheck1()`. Then add each of the following two commands where appropriate.

```
document.getElementById("flowchart").style.display="inline";
document.getElementById("flowchart").style.display="none";
```
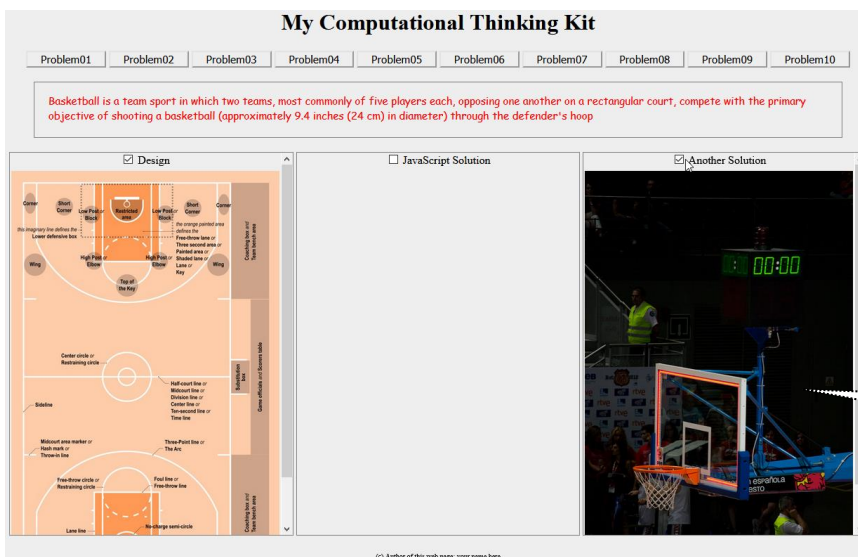
The `if` statement retrieves the `checked` attribute of the Design checkbox element, and determines if the checkbox is checked or not, if so, display the image, if not, we disappear it. In other words, by clicking on the Design checkbox, the image's display toggles.

The following picture shows when the Design checkbox is checked. When unchecking the checkbox the image should disappear.
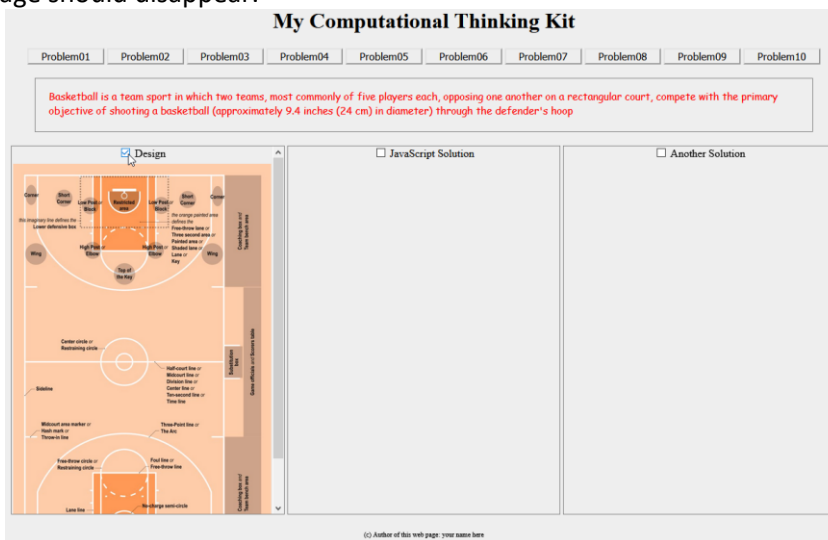
Design checkbox is checked

3) Mimic steps 2 and 3 above for toggling the image that is displayed in the next two panels.
The following picture shows when the next two checkboxes are checked too.

Then, *Another Solution* checkbox is checked

You will use your picture downloaded from internet

Then, *JavaScript Solution* checkbox is checked

Then, *Design* checkbox is unchecked

Exercise 3: (CREATE: **myLearningKit_Ex3.html** and **myLearningKit_Ex3.js)**
Copy **myLearningKit_Ex2.html** to a new file named **myLearningKit_Ex3.html**. Copy **myLearningKit_Ex2.js** to a new file named **myLearningKit_Ex3.js**. Modify the new html file so that the two new files work together. (The CSS file is still the same).

Mimic what you did in Exercise 1 and 2, to assign problem description and corresponding images when button Problem02 is clicked. Add function **p02Func()** to the JS file. Its body is similar to that of **p01Func()**. The content of the 'problem' division will have the following paragraph.

Volleyball is a team sport in which two teams of six players are separated by a net. Each team tries to score points by grounding a ball on the other team's court under organized rules.[1] It has been a part of the official program of the Summer Olympic Games since Tokyo 1964.

The images that you assign are **volleyball_design.jpg**, **volleyball_js.jpg**, and you need to download an image for the other panel (save as **volleyball_other.jpg** and put in same folder). The following picture illustrates when Problem02 is clicked and then the checkboxes are checked/unchecked.

My Computational Thinking Kit

| Problem01 | Problem02 | Problem03 | Problem04 | Problem05 | Problem06 | Problem07 | Problem08 | Problem09 | Problem10 |

Volleyball is a team sport in which two teams of six players are separated by a net. Each team tries to score points by grounding a ball on the other team's court under organized rules. It has been a part of the official program of the Summer Olympic Games since Tokyo 1964.

☐ Design   ☐ JavaScript Solution   ☐ Another Solution

(c) Author of this web page: your name here
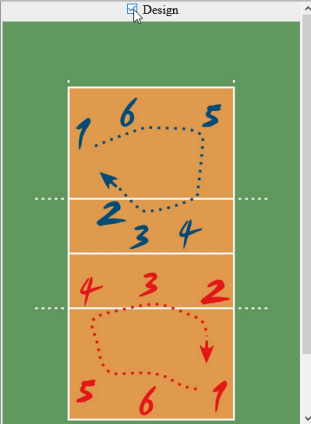
My Computational Thinking Kit

| Problem01 | Problem02 | Problem03 | Problem04 | Problem05 | Problem06 | Problem07 | Problem08 | Problem09 | Problem10 |

Volleyball is a team sport in which two teams of six players are separated by a net. Each team tries to score points by grounding a ball on the other team's court under organized rules. It has been a part of the official program of the Summer Olympic Games since Tokyo 1964.

☑ Design   ☐ JavaScript Solution   ☐ Another Solution

(c) Author of this web page: your name here

Then, *Design* checkbox is checked

My Computational Thinking Kit

| Problem01 | Problem02 | Problem03 | Problem04 | Problem05 | Problem06 | Problem07 | Problem08 | Problem09 | Problem10 |

Volleyball is a team sport in which two teams of six players are separated by a net. Each team tries to score points by grounding a ball on the other team's court under organized rules. It has been a part of the official program of the Summer Olympic Games since Tokyo 1964.

☑ Design   ☑ JavaScript Solution   ☑ Another Solution

(c) Author of this web page: your name here

Then, other two checkboxes are checked one after another

You will use your picture downloaded from internet

My Computational Thinking Kit

| Problem01 | Problem02 | Problem03 | Problem04 | Problem05 | Problem06 | Problem07 | Problem08 | Problem09 | Problem10 |

Volleyball is a team sport in which two teams of six players are separated by a net. Each team tries to score points by grounding a ball on the other team's court under organized rules. It has been a part of the official program of the Summer Olympic Games since Tokyo 1964.

☑ Design ☐ JavaScript Solution ☑ Another Solution

(c) Author of this web page: your name here
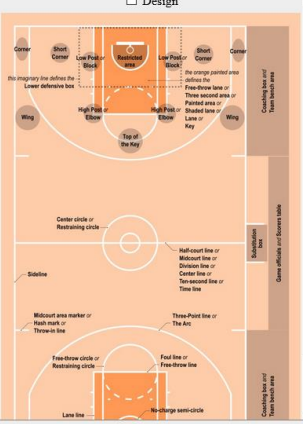
Then, *JavaScript* checkbox is unchecked

So far both buttons and checkboxes for both Problem01 and Problem02 work well, assuming for each problem we start from the very beginning. (Press **Control + F5** to let the browser forcibly reload contents as well as clears previously cached scripts and other page contents.)

With both problem01 and problem02 implemented, we have a flaw here: if now you click Problem01 button, the message is updated, check boxes are unchecked, but the two existing images are replaced with the images for problem01.

My Computational Thinking Kit

| Problem01 | Problem02 | Problem03 | Problem04 | Problem05 | Problem06 | Problem07 | Problem08 | Problem09 | Problem10 |

Basketball is a team sport in which two teams, most commonly of five players each, opposing one another on a rectangular court, compete with the primary objective of shooting a basketball (approximately 9.4 inches (24 cm) in diameter) through the defender's hoop

☐ Design ☐ JavaScript Solution ☐ Another Solution

(c) Author of this web page: your name here
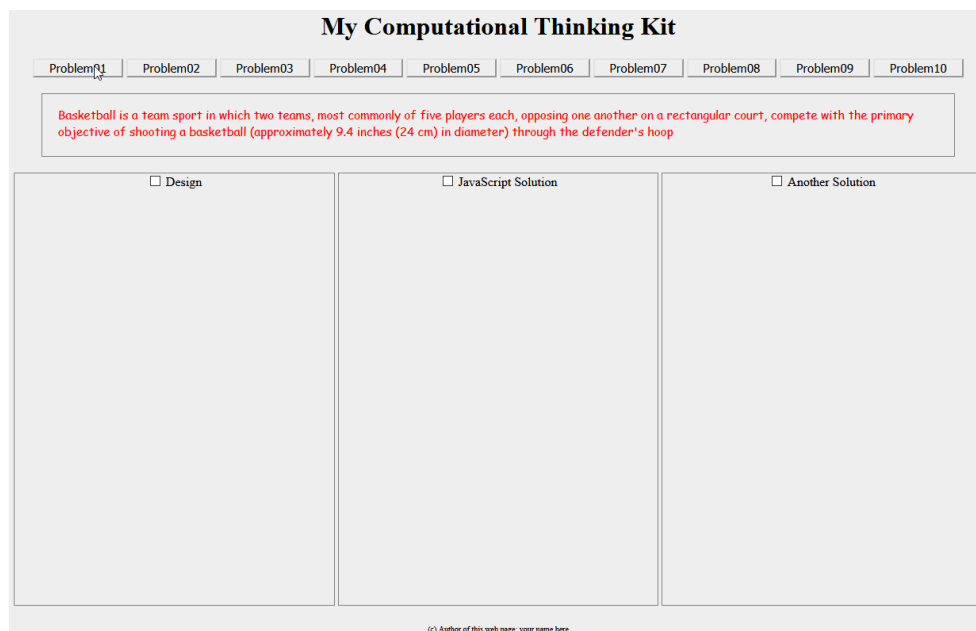
Then, *Problem1 button is clicked*

The reason for this is that when the checkboxes were checked previously for Problem02, the JS function set the display to be inline. So when you click problem01, the images are displayed after the image sources are updated for problem01.   Same issue if now you click problem2.

What we want is that when a problem button is clicked, all the existing images (if any, for other problems) are cleared. To fix this, we need to modify functions **p01Func()** and **p02Func()** such that all the three image component's display is none.

After fixing this, the following should be observed.

Assume current step of problem02

Then, Problem01 button is clicked. Previous images are cleared.

Note that when click Problem01 button all existing images should disappear. Same if click Problem2 button again. Likewise, if you are working on Problem01 with some images displayed and then click Problem02 button, all the existing images for problem01 should disappear.

Exercise 4: (CREATE: **myLearningKit_Ex4.html** and **myLearningKit_Ex4.js)**
Copy **myLearningKit_Ex3.html** to a new file named **myLearningKit_Ex4.html**. Copy **myLearningKit_Ex3.js** to a new file named **myLearningKit_Ex4.js**. Modify the new html file so that the two new files work together.
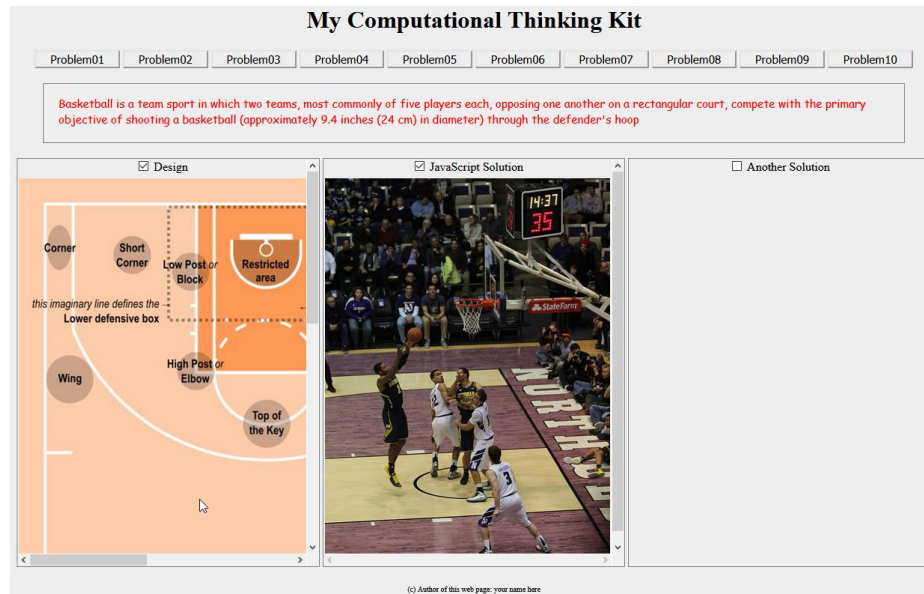
In this exercise, we want to add two more functions to our **JavaScript** code. One we call it **zoomIn()** and the other **zoomOut()**. The objective is that when the image of the Design checkbox is double clicked, we want to make its width 200% (do this in the body of **zoomIn()**) and when is single clicked, we want to have its original width of 100% (we do this in the body of **zoomOut()**).

Here is what you need to do:

- In your HTML file, add **onclick** and **ondblclick** attributes to the image whose ID is "flowchart"

- on an **ondblclick** event, function **zoomIn()** is called
- on an **onclick** event, finction **zoomOut()** is called
- In your JS file, add **zoomIn()** and **zoomOut()** functions
  - in the body of **zoomIn()**, retrieve the element with id flowchart, and set .**style.width="200%";**
  - write a similar line in the body of **zoomOut()** to change the width to its original size

The following figure illustrates when Problem01 button is clicked and the Design image is doubl clicked.

The following figure illustrates when Problem02 button is clicked and the Design image is double clicked.
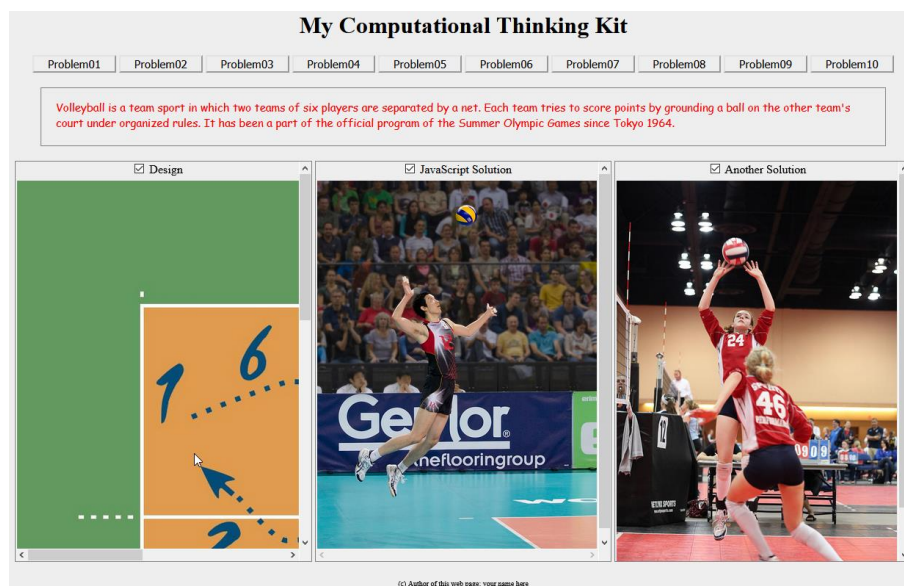
F. AFTER-LAB WORKS (THIS PART WILL NOT BE GRADED)

Exercise 5 optional further practice:

You may want to continue this project and add at least two more problems (e.g., football, hockey) and corresponding design and implementations for further practice.

In order to review what you have learned in this lab as well as expanding your skills further, we recommend the following questions and extra practices:

Short answer questions:

- How many functions are defined in your **myLearningKit_Ex4.js**?
- How many function calls are in your **myLearningKit_Ex4.html**?
- How many different types of events you handled in your **myLearningKit_Ex4.html**?
- For which element(s) in your **myLearningKit_Ex4.html** did you defined two events?
- How can you see your JavaScript errors in Firefox?

To master your skills further,

- As demonstrated in class, create a web page with text, and add several buttons, one for changing text content, one for changing text color, one for changing font size, one for changing text to italic, one for changing text to bold, one for changing to underlined, also one for clearing all the text.
- Create a web page with text of your choosing (a single paragraph will do), which upon the mouse being positioned over it causes the same text to be repeated below the paragraph. When the mouse is no longer positioned over the text the copy of the text remains.
- Same as above but when the mouse is no longer over the text the copy of the text disappears.
- Create a web page with two paragraphs of text of your choosing, and with a button below the text. Clicking on the button should cause a solid border to appear around the first paragraph and a dashed border to appear around the second paragraph
- Create a web page with two paragraphs of text of your choosing, and with a button below the text. Clicking on the button should cause a new paragraph to appear below the button containing the combined text of both paragraphs above the button.
- Make sure you read from the two JavaScript books that have been already introduced to you.

Please feel free to discuss any of these questions in the course forum or see the TAs and/or Instructors for help.

Submissions.

You should already have a **Lab03** folder that contains the following files and folder.

light.html light.css. light.js
myLearningKit_Ex{1,2,3,4}.HTML and **myLearningKit.css**, and **myLearningKit_Ex{1,2,3,4}.js**
images folder

Your HTML should pass the HTML validator at https://validator.w3.org

Compress the Lab03 folder (zip **or** tar.gz**), and then submit the (single) compressed file on eClass.**

Late submissions or submission by email is NOT accepted. Plan ahead and submit early.