CSE 154: Web Programming — Summer 2019

Practice Exam 1: Example C

Note: We strongly recommend printing out practice exams and working through them with only your cheatsheet (provided on the course website) - it's important to be comfortable taking a spec and writing code on paper without the convenience of autocomplete/debugging tools!

Also note that provided exams adapt problems from previous quarter exams, but the number/format of problems may be different (see other provided practice exams for other example problems). This exam in particular is a bit longer than we'd expect for 50 minutes.

Name:

UWNet ID: @uw.edu

TA (or section):

Rules:

- You have 60 minutes to complete this exam.
- You will receive a deduction if you keep working after the instructor calls for papers.
- This is a closed-note exam, but you may use the provided cheatsheet for reference. As noted on the cheatsheet, you may assume id, qs, and qsa are provided in JS as shorthand for document.getElementById, document.querySelector, and document.querySelectorAll, respectively.
- You may not use any electronic or computing devices, including calculators, cell phones, smartwatches, and music players.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- Do not abbreviate code, such as writing ditto marks ("") or dot-dot-dot marks (...). You may not use JavaScript frameworks such as jQuery or Prototype when solving problems.
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your Student ID to a TA or instructor for your submitted exam to be accepted.

Question	**Score**	**Possible**
HTML Validation		
CSS and the DOM		
JS/Animations		
JS/DOM/UI		
Short Answer		

1. What's wrong with my HMTL?

Consider the following HTML:

```
<!DOCTYPE html>
<html lang="en">
   <head>
      <script src="index.js"></script>
      <link href="styles.css" rel="stylesheet"/>
   </head>
   <body>
      <h1 "Best page ever!" />
      <div>
         <a>Check out this cool page:
           <href>http://www.pointerpointer.com</href>
         </span>
      </div>
   <body>
</html>
```

This HTML document won't validate, and would generate errors and warnings in the W3C Validator. However, it is possible to make 5 modifications to the HTML to make it pass validation. Each modification might result in multiple text "changes" to the HTML document, but is considered one modification because it is addressing the same root problem.

Indicate the 5 modifications we need in order to make it pass validation. Write directly on the HTML. Below, briefly describe the changes that you made, and why you had to make each change in order to validate. If it is unclear what changes go with which explanations, feel free to number your changes on the HTML. No need for an essay — 10 words or less should be plenty.

1. _____

2. _____

3. _____

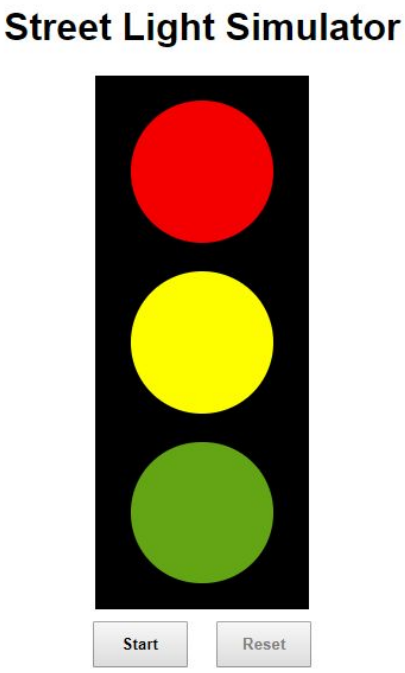
4. _____

5. _____

2. (CSS) r/css_irl

Consider the following HTML body (left) and page screenshot (right):

```
<body>
  <header>
    <h1>Street Light Simulator</h1>
  </header>
  <main>
    <div>
      <div id="red"></div>
      <div id="yellow"></div>
      <div id="green"></div>
    </div>
    <article>
      <div id="options">
        <button id="start">Start</button>
        <button id="reset" disabled>Reset</button>
      </div>
    </article>
  </main>
</body>
```

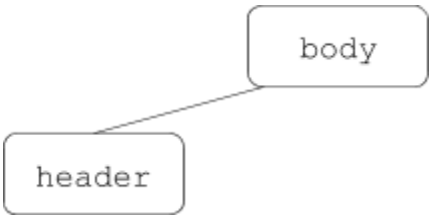


Part A: Write CSS to add to the HTML to meet the specified appearance seen in the screenshot:

- The main should be centered on the page, and the h1 should have centered text with Helvetica font, defaulting to sans-serif if Helvetica is not available on the system.
- The black rectangle has a height of 450px and a width of 180px.
- The “red”, “green”, and “yellow” divs should have background colors of red, green, and yellow, respectively. These divs should be circles in the black rectangle, each with 120px height and width.
- Each of the three circles should be centered horizontally within the black rectangle with evenly-distributed spacing vertically (see screenshot).
- The buttons are 80px wide with 10px of *both* margin and padding on all sides. The button text should be bold.

Part A (9 pts): Write your CSS here

Part B (3 pts): Finish drawing the DOM tree that corresponds to the hierarchy of the body HTML (ignore text, text nodes, and tag attributes - just refer to tag names in boxes and clearly show parent/child relationships with lines between boxes). The final diagram should have as many nodes (boxes) as their are elements in the HTML, but note that we have started the first two DOM elements for you.



3. (JS/Timers) Street Light Simulator (12 pts)

Given the HTML from the Problem 2, implement the JS for a street light simulator. While Problem 3 does not depend at all on your CSS in Problem 2, assume one more ruleset has been added to your CSS to apply a black background color to any elements with a class “off”. **You will add/remove this class to the three colored circles to hide/show them as specified.**

Details:

When the page is loaded, the “off” class should be added to both the #green and #yellow divs. This will make only the #red circle appear visible on the streetlight. When the “Start” button is clicked:

- The “Start” button should be disabled and the “Reset” button should be enabled (set its disabled attribute to false).
- An animation should start, changing colors each second between red, green, and yellow. The time it takes to switch between red and green and green and yellow should each be 1 second, but the switch between yellow and red should be 0.5 seconds.
    - Initially the current color is red, but 1 second after the animation starts the next color (green) should be displayed, then 1 second later yellow, then 0.5 seconds later red again, and so on. Only one color should ever be displayed at a time.
- You may not have more than one timer running at the same time, but you may start/end multiple timers (there are a few possible solutions).

When the “Reset” button is clicked:

- The “Reset” button should be disabled and the “Start” button should be re-enabled.
- The animation should be stopped, and the red light should be the only light displayed.
- Future animations should work exactly the same as the first animation.

Write your JS solution within the module pattern below. More room is provided on the next page if needed.

```
(function() {

  /** Part A) Write your module-global variable(s) here */




   window.addEventListener("load", initialize);

  /** Do not write below this line: answer the rest of the question on
   *  the next page.
   */
```

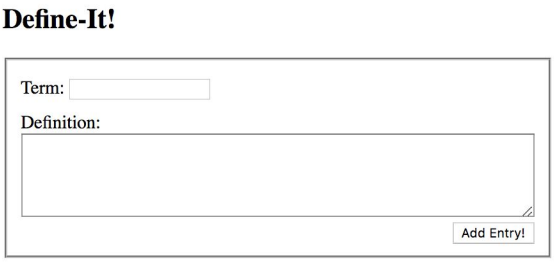
```
/** Part B) Write a helper function that takes two string parameters
 *  that represent the ids of DOM elements on the page. The DOM element with
 *  the color1 id will be turned "off" and the DOM element of the color2 id
 *  will be turned on.
 *  @param {string} color1 The id of the DOM element to turn off
 *  @param {string} color2 The id of the DOM element to turn on
 */
function switchColor(color1, color2) {









}

/** Part C) Write a helper function that can be called to
 *  initially set (or later reset) the state of the page.
 */
function initialState() {

```

/** Continue to part D on the next page */

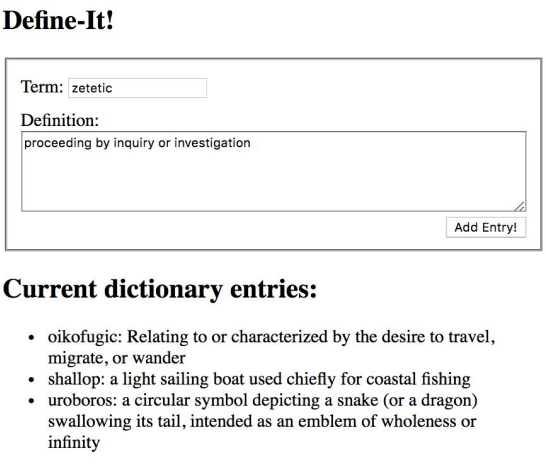
```
/** Part D) Write the function that is called when the page is loaded.
  * You may assume the code in part C correctly sets the initial state of
  * the page.
  */
 function initialize() {









 }

/** Part E) Write additional functions needed to handle the events
  * required for this page to function properly below and on the next page
  * as needed
  */









})(); // end module
```

4. Define-It!

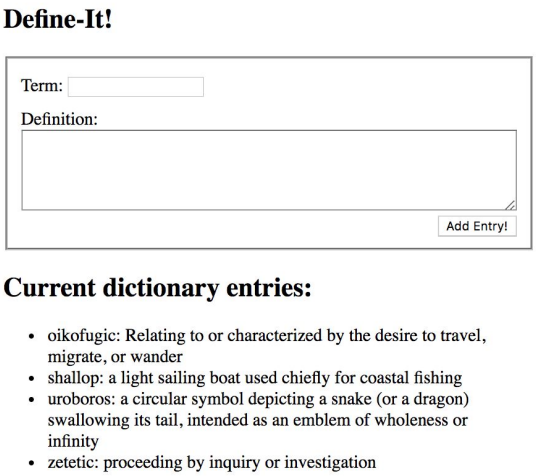
In this problem, you will write JavaScript to add functionality to a personal dictionary webpage that allows a user to manage an entry list of terms and definitions.



Empty dictionary view (initial/after removing all added dictionary entries)



Entering an Entry for “Zectetic”



Clicking “Add Entry” given term/definition of left screenshot

Requirement Details

- When the page loads, there are no dictionary entries on the page.
- There is a text input with id #term for a user to type a term and a text area input with id #definition for a user to enter the corresponding definition. If the users clicks the ”Add Entry!” button, the current word and definition should be added to the list of current dictionary entries if both the term and definition input areas are non-empty. Whenever a new entry is added to the entry list, both term and definition input areas should be cleared. Nothing should happen if either the term or definition fields are empty.
- Each entry should be added as a single item to the #entries list in the format TERM: DEFINITION, replacing TERM with the value typed in the term input box and DEFINITION with the value typed in the definition text area.
- Whenever there are dictionary entries in the #entries list, #current-entries should be visible, otherwise it should be hidden. The linked CSS has a hidden class which sets display: none for an element with that class. You should use this class when hiding/showing an element.
- Whenever a user double-clicks on an entry in the #entries list, that entry should be removed from the list. Any other entries should maintain their order in the list whenever an entry is removed.

```
<!-- Provided HTML for Question 4 -->
   <body>
      <section>
         <header><h1>Define-It!</h1></header>
         <fieldset id="entry-input">
            Term: <input id="term" type="text" /><br/>
            Definition:<br/>
            <textarea rows="5" cols="40" id="definition"></textarea>
            <button id="add-entry">Add Entry!</button>
         </fieldset>
         <article id="current-entries" class="hidden">
            <header>
               <h2>Current dictionary entries:</h2>
            </header>
            <ul id="entries"></ul>
         </article>
      </section>
   </body>
```

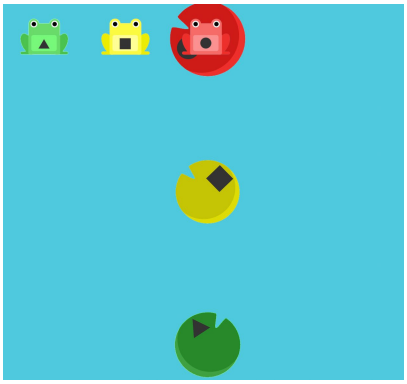
Write your JavaScript solution below (continue on next page as needed):

```
(function() {
```

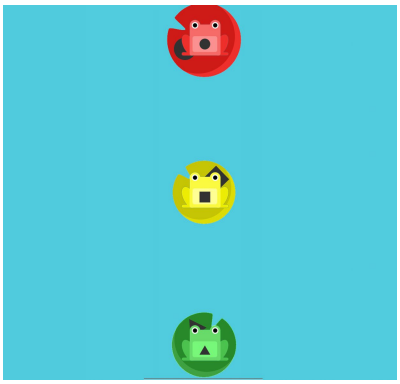
```
// Continue your JavaScript solution to Problem 4 below as needed:









})(); // end module pattern
```

5. Short Answers

1. If you were playing Flexbox Froggy and you needed to get the Froggies from their starting position (shown on the left) to the ending position on the lilypads (shown on the right), what flex rules would you add to the pond CSS selector? (Note: Because this printed in greyscale, you must match the symbols on the frogs with the symbols on the pads. The lilypad symbols may appear rotated from what the frog symbols are).



Flexbox Froggy starting position



Flexbox Froggy ending position

Add the rules to the ruleset below:

```
#pond {
  display: flex;




}
```

2. Why is it important to limit the use of module-global variables in our JavaScript programs?

3. What is the role of the id of a timer returned by `setTimeout` or `setInterval`? In particular, when do we need it?

4. Consider the following HTML body:

```
<body>
  <button id="my-btn">Click Me!</button>
  <p id="result"></p>

  <script>
    document.getElementById("my-btn").addEventListener("click", function() {
      document.getElementById("result").innerHTML = "you clicked the button!";
    });
  </script>
</body>
```

Because the <script> tag is at the bottom of the body, this actually shows “You clicked the button!” on the page when the button is clicked. But based on what we’ve learned about HTML/CSS/JS so far, what are 2 issues *related to the script tag and JavaScript code* which demonstrate poor code quality? Briefly justify both of your answers (1 sentence is sufficient), and provide a better alternative to each (ensuring the behavior and appearance of the page is still the same). Do not use commenting/indentation/spacing as your answers.

Issue A and justification: Better alternative:
Issue B and justification: Better alternative:

5. Consider the following JSON declaration:

```
let mystery = {
  "i" : ["j", 0, 1],
  "ii" : "ii",
  "I" : "i"
};
```

Write the JavaScript value that would be returned for each of the following statements. Include "" around any string values. Hint: at least one value is `undefined`.

Statement	**Value**
`mystery["i"]`	
`mystery[0]`	
`mystery.ii.length`	
`mystery["i"][0].length`	