CSE 154: Web Programming Autumn 2018

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.
- You may not use any electronic or computing devices, including calculators, cell phones, smartwatches, and music players. You may receive a deduction if we hear or see your electronic device during our exam time.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- This is a closed-note exam, but you may use the provided cheat sheet for reference. As noted on the cheat sheet, you may assume $, qs, and qsa are provided in JS as shorthand for document.getElementById, document.querySelector, and document.querySelectorAll, respectively.
- 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		5
Short Answer		9
CSS/DOM Tree		12
JS/Animations		12
JS/DOM/UI		12
Total		50

This page intentionally left blank

1. (HTML) (1 pt each) What's wrong with my HMTL?

Consider the following HTML:

```
1 <!DOCTYPE HTML>
2 <html>
3     <head>
4         <title>Web dev<title>
5         <link href="index.css" rel="stylesheet"></link>
7     </head>
7     <body>
8         <h1 id="important">Things I learn in web dev</h1>
8         <h2>Learned Things:</h2>
9         <ul>
10          <li class="learned">Every time a mouse is clicked a skittle is born.</li>
11          <li class="learned">It's 2018 and browsers still don't agree on dropdowns.</li>
12          <li>Webpages are made of trees.</p>
13      </ul>
14      <div>
15          <img src="pupper.jpg" />
16          <p id="important">I promise that was a cute puppy photo.</p>
17      </div>
18    </body>
19    <footer><p>CSE154 &copy; 2018</p></footer>
20 </html>
```

This html document won't validate - however, it is possible to make it validate by making changes to 6 lines in the html. Indicate 5 of the 6 changes you could make to the html document 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. For full credit, you must have a correct explanation and fix for the validation issue above. Number your changes on the html to match the explanations. No need for an essay -- 20 words or less should be plenty. Note: we will only consider the 5 changes you have described on the lines below for grading.

1. _____

2. _____

3. _____

4. _____

5. _____

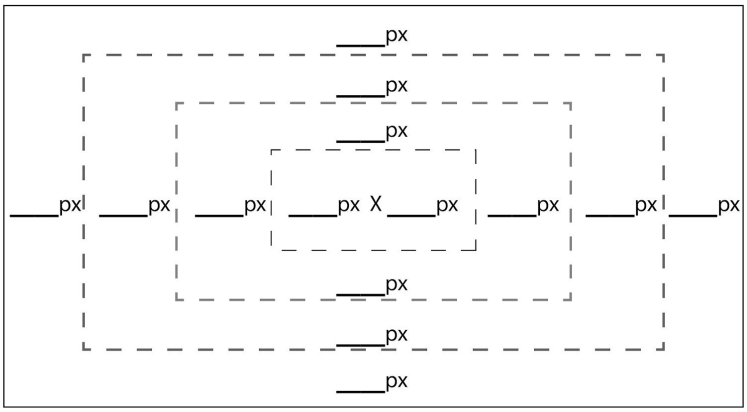
2. Short Answers

1. (1 pt) Based on what we've learned in this course so far, what is one reason to avoid inline JS in HTML? (e.g. `<button onclick="myFunction()">...</button>`)

2. (1 pt) Give a reason that motivates our use of <section> over for content sections in HTML pages:

3. (2 pts) In the Box Model diagram to the right, label the following CSS styles for a #content element:

```
#content {
  width: 60px;
  height: 30px;
  margin: 10px;
  margin-top: 20px;
  margin-bottom: 15px;
  border: 3px solid;
  padding: 5px;
  padding-right: 25px;
}
```



4. (1 pt) Provide and support one reason why the module pattern is important to use in JavaScript.

5. (1 pt) Give an example where using === and== would return different results when comparing the same two values in JavaScript.

6. (2 pts) Consider the following JSON object:

```
let miniJSON = {
  "smarties" : "rainbow",
  "candy corn" : ["orange", "#FFFFFF"],
  "skittles" : {
    "rainbow" : true,
    "colors" : 154
  },
  "rainbow" : [“smarties”, “skittles”, “sprinkles”]
};
```

For each of the following statements, write the value that would be returned (include ”” around any string values; if any expression would result in an error, write error. If any expression would return the value undefined, specify this as your answer.

a. miniJSON.rainbow[1] : _____

b. miniJSON.candy corn : _____

c. miniJSON.smarties.length : _____

d. miniJSON[miniJSON["smarties"]].length : _____

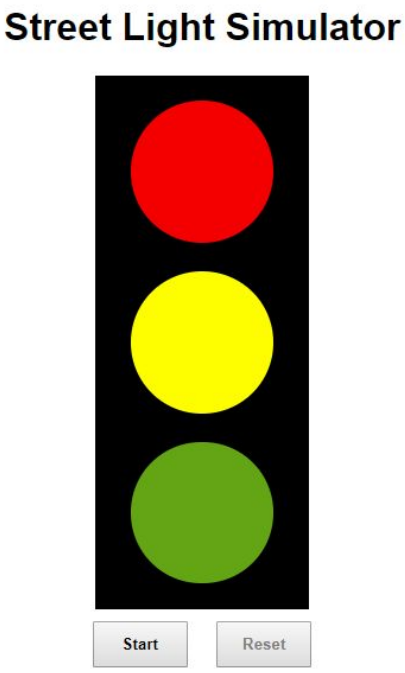
7. (1 pt) For the following JS program, label the _____ following each console.log statement with 1, 2, 3, or 4, corresponding to the relative order in which that statement will first be printed (where 1 indicates the first statement printed).

```
console.log("ghost");                          // ____
(function() {

  console.log("pumpkin");                      // ____
  window.addEventListener(“load”, initialize);
  foo();

  function initialize() {
    console.log("dinosaur");                   // ____
  }

  function foo() {
    console.log("cat");                        // ____
  }
})();
```

3. (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>
```

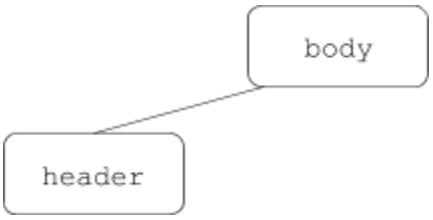


Part A: Write CSS to add to the HTML to meet the specified appearance seen in the screenshot:

- 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.

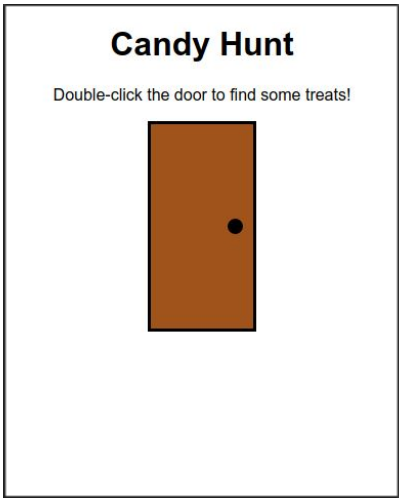
Part A (9 pts): Write your CSS here

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.

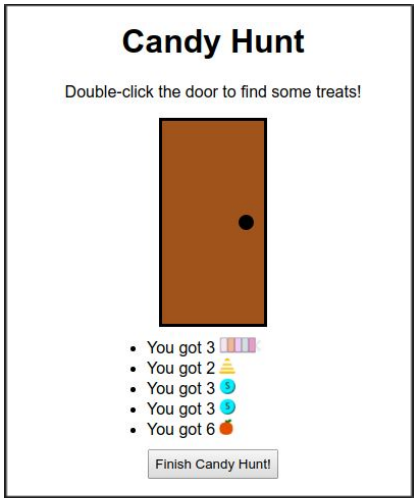


4. (JS/DOM/Events) Click-Click, Who's There? (12 pts)

Write JavaScript code to add behavior to the following page to start a candy hunt game:



Initial Page/New Game View



View During Hunt

View After "Finish Candy Hunt" clicked

Details:
The provided HTML and CSS will display the "Initial Page View" above. Note that in the HTML, both #finish and #new-game buttons are given the class .hidden. Elements with this class will be hidden on the page - removing the class will re-display them.

When the page loads, a user should be able to start a Candy Hunt by double-clicking the #door div. The first time the door is double-clicked during a new game, the #finish button should be displayed by removing the .hidden class (see HTML).

Hunting for Candy:
Whenever the door is double-clicked (including the first double-click):

- One of 4 treats should be found with equal probability: apples, candycorn, skittles, smarties.
- A random number between 1 and 10 pieces (inclusive) of the found treat should be found (each number having equal probability).
- A list item should be added to the #candy-list ul with the text "You got COUNT ", replacing COUNT with the random count. Following this text in the list item should be an image element with a source of "TREAT.png", replacing TREAT with the name of the treat. The image element should have alt text of the treat name. For example, one possible list item may have the following HTML after being created:
  `<li>You got 2 <img src="skittles.png" alt="skittles" /></li>`
- You may not use .innerHTML when creating the `<li>`.
- Whenever a new list item is added, it is added to the bottom of the #candy-list.

Finishing a Candy Hunt:

- Whenever the #finish button is clicked, the current Candy Hunt is finished, and a message in the #results paragraph should display with the text "That's N treats over M visits!", replacing N with the total number of treats in the finished hunt and M being the number of doors knocked (corresponding to the number of list items added in the hunt).
- At this point, the #new-game button should be displayed, and the #finish button should be hidden.
- Double-clicking the door should no longer do anything until a new Candy Hunt is started.

Starting another Candy Hunt:

Whenever the #new-game button is clicked (note that this button is first made visible after the first hunt):

- It should be hidden and the candy list and result message should should be empty again.
- Behavior and appearance should then be the same as starting the first candy hunt.

Below is the HTML for the page body. Other than the .hidden class, the CSS is unnecessary for writing the JS.

```
<body>
  <h1>Candy Hunt</h1>
  <p>Double-click the door to find some treats!</p>
  <div id="door">
    <span id="door-knob"></span>
  </div>

  <ul id="candy-list"></ul>

  <button id="finish" class="hidden">Finish Candy Hunt!</button>
  <p id="results"></p>
  <button id="new-game" class="hidden">New Game</button>
</body>
```

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

```
(function() {
  "use strict";
  const TREATS = ["apples", "candycorn", "skittles", "smarties"];

  window.addEventListener("load", initialize);
```

```
// finish your JS solution to Problem 4 here if needed.

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

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

Given the HTML from the Problem 3, implement the JS for a street light simulator. While Problem 5 does not depend at all on your CSS in Problem 3, 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() {
  "use strict";

  /** 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
```