CSE 154: Web Programming — Autumn 2018

Practice Midterm Exam 2

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).

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 $, 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.

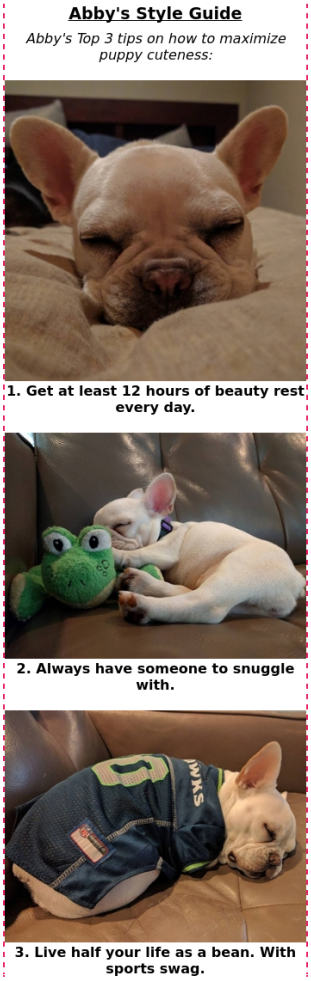
Question	**Score**	**Possible**
HTML and CSS		
The DOM		
JS/Animations		
Short Answer		

1. A Special Spec for a Special Doggy

Write the HTML and CSS necessary to recreate the given expected appearance:

- All images of Abby are located in the same directory as your HTML and CSS files. The first image is located at sleepy-abby.png, the second image is located at abby-with-toy.png, and the third image is located at seahawks-abby.png
- Each image has 2px of margin on both the top and bottom edges.
- The page uses a font-family of Verdana, or sans- serif if Verdana is not available on a client's computer. All text is center-aligned.
- The body of the page takes up 50% of the overall page's width and is aligned horizontally in the middle of the page. This body section also has left and right borders that are 2px wide with a dashed pattern of the color #e91e63.
- The main heading of the page ("Abby's Style Guide") is underlined.
- Each of the three image and caption groups have a width of 350px, which the associated image should fully span in width. Each of these groups also has 20px of margin on the top and bottom edges.
- The subheading should have 12pt italicized font. The captions also have 12pt font, but are boldly-weighted.

Write your HTML/CSS below (continue on next page if needed):



Finish your HTML/CSS solution to Problem 1 as needed here:

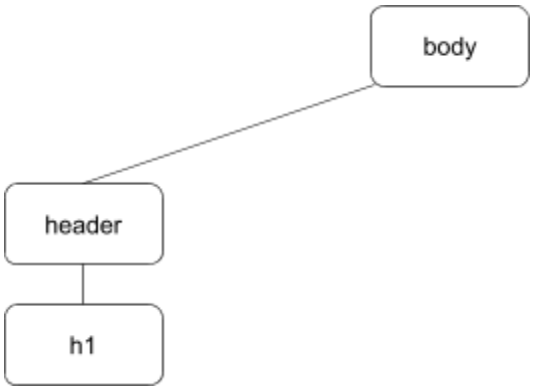
2. Drawing the DOM

Consider the following HTML body:

```
<body>
   <header>
      <h1>Things to do after my CSE 154 midterm</h1>
   </header>
   <main>
      <p>
         Things I <strong><em>really</em></strong> look forward to:
      </p>
      <ol>
         <li>Find a comfy place to lay down and take a nap</li>
         <li>
            Finish any homework due today...
            <ul>
               <li>CP3 isn't due until next week! Yay!</li>
            </ul>
         </li>
         <li>Find <a href="https://imgur.com/r/puppies">puppy photos</a> on the int webz</li>
         <li>Get a <em>good</em> night's sleep</li>
      </ol>
   </main>
   <footer>
      <p>
         It's <em>almost</em> winter! :D
      </p>
      <img src="puppy-in-the-snow.jpg" alt="a happy puppy in the snow" />
   </footer>
</body>
```

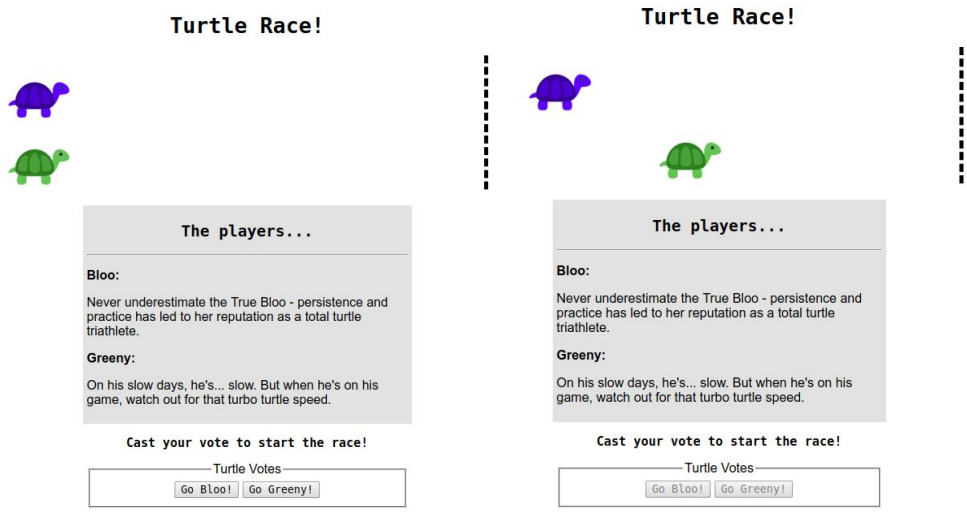
On the next page, finish drawing the DOM tree that corresponds to the hierarchy of the body HTML (ignore text, text nodes, and tag attributes - just refer to tag names in boxes and parent/child relationships with lines between boxes). The final diagram should have as many nodes (boxes) as their are elements in the HTML above, but note that we have implemented the header and its h1 for you.

Finish the DOM tree here:



3. Turbo Turtles

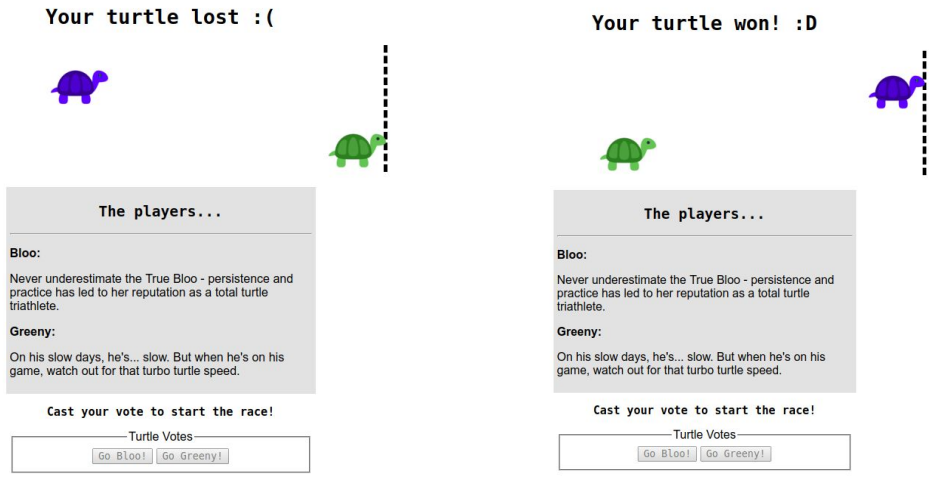
In this question you are to setup a ”turtle race” such that a user can vote on one of two turtles and start a short race between the turtles, where only one turtle will win. You are provided the HTML (on the next page) and will **finish** the implementation of a JavaScript program to add interactivity and animation to this site. You may assume your JavaScript is successfully linked from the HTML. Below is a screenshot of the initial page view (left) and a mid-race view (right):



Details

- The race starts when a user clicks on one of the two buttons - each button represents the turtle the user votes for. After the race starts, both buttons should be disabled. They will never be re-enabled in this program.
- When the race starts, both turtles should start moving towards the dashed finish line from left to right.
- Each turtle’s speed is decided randomly as an interval value between 1 and 250ms (inclusive). You are provided a function `getRandomValue(min, max)` that takes two parameters (a min and a max) and will return you a random number from min to max (inclusively). Once the race has started, each turtle maintains their speed for the duration of the race.
- Each time a turtle moves (as determined by the interval rate) it should move exactly 4px to the right.
- When either turtle touches the rightmost edge of the `#racetrack` the race (and any animation) ends and that turtle is the winner. You are provided a function `didFinish(el)` that accepts as a parameter a DOM element (representing a turtle) that will tell you if a turtle has crossed the finish line. You should use this function after making a move to determine if your turtle has crossed the finish line.
- If the turtle the user voted for at the start of the race won, the #title-1 text should be updated to display "Your turtle won :D". Otherwise, this text should display "Your turtle lost :(". You are provided a function `displayResults(pick, winner)` that takes two parameters (the DOM element representing the turtle picked, and the DOM element representing the winner) and which will display this text for you. Remember, no buttons should be enabled after the end of a race.

Sample graphical output for the case when your chosen turtle loses (left) and wins (right) is shown below:



Provided HTML:

```
<body>
   <section id="game">
      <header><h1 id="title-1">Turtle Race!</h1></header>
      <div id="racetrack">
         <div class="turtle" id="b-turtle">
            <img src="blue-turtle.png" alt="blue turtle" />
         </div>
         <div class="turtle" id="g-turtle">
            <img src="green-turtle.png" alt="green turtle" />
         </div>
      </div>
   </section>
   <section id="players">
      <header><h2>The players...</h2><hr/></header>
      <p><strong>Bloo: </strong></p>
      <p>
         Never underestimate the True Bloo - persistence and practice has led to
         her reputation as a total turtle triathlete.
      </p>
      <p><strong>Greeny:</strong></p>
      <p>
         On his slow days, he's... slow. But when he's on his game, watch out for that turbo
         turtle speed.
      </p>
   </section>
   <section id="vote">
      <header><h3>Cast your vote to start the race!</h3></header>
      <fieldset>
         <legend>Turtle Votes</legend>
         <button id="b-ftw">Go Bloo!</button> <button id="g-ftw">Go Greeny!</button>
      </fieldset>
   </section>
</body>
```

Finish the JS implementation below. You may assume that the aliases $(id) and qsa(sel) are defined for you and included as appropriate.

```
(function() {

/**
 * This function returns a random value between min and max inclusive. Assumes
 * min is less than max.
 * @param {Number} min - the minimum value the random number can take
 * @param {Number} max - the maximum value the random number can take
 * @returns {Number} - the random value between min and max inclusive.
 **/
function getRandomValue(min, max) { /* code not displayed here */ }

/**
 * This function takes in a DOM element (for a turtle) and determines whether that
 * turtle has hit the finish line.
 * @param {object} - the turtle to check to see if its crossed the finish line.
 * @returns {Boolean} - true if this turtle has crossed the finish line.
 **/
function didFinish(turtle) { /* code not displayed here */ }

/**
 * This function will change the title to show if the picked turtle is the winner
 * @param {object} pick - The DOM element of the turtle you picked.
 * @param {object} winner - the DOM element of the actual winning turtle. */
function displayResults(pick, winner) { /* code not displayed here */ }

// Write the rest of the JavaScript code here, starting with the window load event handler.
```

```
// Continue the rest of the JavaScript code here.

})();
```

4. Short Answers

1. Why is it important to avoid inline styles in HTML? (e.g. `<p style="color: green">...</p>`)

2. For at least two of the following individuals write one thing you can do to make your website more accessible for the specified user:

a. blind user who uses a screen reader.

_____

b. low-vision user who uses magnification.

_____

c. color-blind user.

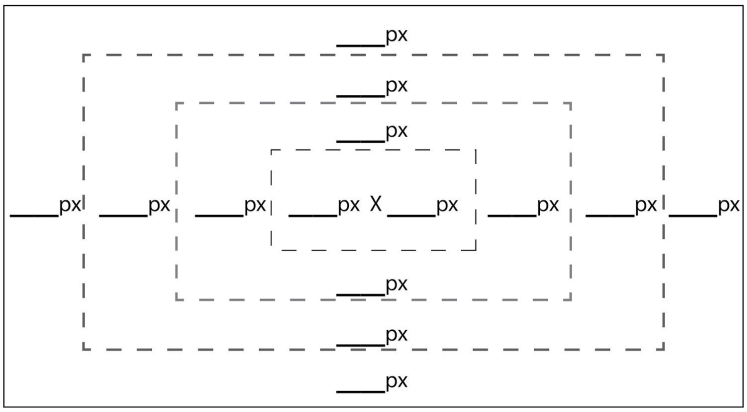
_____

d. mobility-limited who can't control a mouse.

_____

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

```
#content {
  width: 50px;
  height: 60px;
  margin: 10px;
  margin-top: 20px;
  margin-bottom: 30px;
  border: 3px solid;
  padding: 5px;
  padding-right: 25px;
}
```



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

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

6. Consider the following JSON object:

```
let miniJSON = {
  "waffle" : [“PANCAKE”],
  "pancake" : “waffle”,
  "POPTART" : {
    “frosted” : true,
    “flavors” : [“cherry”, “strawberry”, “jolly rancher”]
  }
};
```

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.pancake : _____

b. miniJSON["FOO"] : _____

c. miniJSON[“POPTARTS”].flavors[1] : _____

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

7. 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 print (where 1 indicates the first statement printed).

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

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

  function pageLoad() {
    console.log("Baz");                      // ____
  }

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