CSE 154 Spring 18 Midterm Solution and Grading Rubric

HTML

Line 3: Using JavaScript/CSS Comments instead of <!-- -->
Line 4, 8: The <heading></heading> tags should be <head></head>
Line 7: The <script> tag should not be self closing, it should have a </script> after it
Line 15: Missing alt text in the img tag
Line 19: What should be a closing <li> tag not really doesn't have the / (should be)
Line 23/24: missing </main>
Line 25: The closing <h2> tag should be </h2> not </h3>
Line 26: The closing p and strong tags are swapped. The needs to go before the </p>

Rubric
+ 1 point for each of the first 5 correct identification of a validation error

CSS

1 pt for each correct line:

```
ol li     : 8, 9, 11, 12, 14
li.yay    : 8, 12, 14
.yay .yay : 8, 12, 14, 17
.yay > em : 15, 18
p *       : 5, 6, 18
```

Question 3. Define It

```
window.onload = function() {
  $("add-entry").onclick = addEntry;
};

function addEntry() {
  let term = $("term").value;
  let definition = $("definition").value;
  if (term && definition) {
    let li = document.createElement("li");
    li.innerText = term + ": " + definition;
    $("entries").appendChild(li);
    $("current-entries").classList.remove("hidden");
    li.ondblclick = updateEntries;
    $("term").value = "";
    $("definition").value = "";
  }
}

function updateEntries() {
  $(“entries”).removeChild(this);
  if (!(qsa("li").length || $("current-entries").classList.contains("hidden"))) {
    $("current-entries").classList.add("hidden");
  }
}
```

Rubric

___ / 2 Event handling

___/ 1 Successfully adding onclick event to handle Add Entry Button click

(.5 near miss for adding onclick event in check for non-null term/def in window.onload)

___/ 1 Successfully adding ondblclick event to handle remove item when creating the li

(.5 near miss point for adding ondblick event in window.onload before an li was created)

(.5 near miss point for dblclick or doubleclick instead of ondblclick)

___ / 1 Form element processing

___/ .5 Getting .value from term and .value from definition

___/ .5 Setting .value of term and .value from definition to “” *after* adding

___ / 6 DOM manipulation

___/ 1 Correct logic to only add DOM element when both term and def are present

___/ 1 Creating li DOM element

___/ 1 Adding correct text to the new li DOM element

(.5 if using toUpperCase(), toLowerCase(), or trim() for not following the specification.)

___/ 1 Appending the new li DOM element to the correct $(“entries”)

___/ 1 removing correct li DOM element on double click (pretend li exists)

0.5 if they implement “double-click” with two click events, but otherwise correct

___/ 1 correctly show/hiding the $(“current-entries”)

___ / 1 Otherwise correct JavaScript

-1 for JavaScript syntax errors (one freebie)

if missing window.onload ineligible for this point.

Question 4. Turbo Turtles

Code:

```
(function() {
  let greenTimer, blueTimer = null;
  let myVote;

  window.onload = function() {
    $("g-ftw").onclick = startRace;
    $("b-ftw").onclick = startRace;
  };

  function startRace() {
    if (this.id == "g-ftw") {
      myVote = $("g-turtle");
    } else {
      myVote = $("b-turtle");
    }
    $("g-ftw").disabled = true;
    $("b-ftw").disabled = true;

    let greenSpeed = getRandomValue(1, 250);
    let blueSpeed = getRandomValue(1, 250);

    // Two function version, no refactoring
    /*
    greenTimer = setInterval(moveGreenTurtle, greenSpeed);
    blueTimer = setInterval(moveBlueTurtle, blueSpeed);
    */

    /* version with one refactored move function */
    greenTimer = setInterval(function() {
      moveTurtle("g-turtle");
    }, greenSpeed);
    blueTimer = setInterval(function() {
      moveTurtle("b-turtle");
    }, blueSpeed);
  }
```

```
 function moveTurtle(turtleid) { // Solution could pass in DOM element or ID
    let turtle = $(turtleid);
    // ok if not done with window.getComputedStyle
    let leftPos = parseInt(window.getComputedStyle(turtle).marginLeft);
    leftPos += 4;
    turtle.style.marginLeft = leftPos + "px";
    if (didFinish(turtle)) {
      displayResults(myVote, turtle);
      clearInterval(greenTimer);
      clearInterval(blueTimer);
    }
  }

 function moveGreenTurtle() {
    let greeny = $("g-turtle");
    // ok if not done with window.getComputedStyle
    let leftPos = parseInt(window.getComputedStyle(greeny).left);
    leftPos += 4;
    greeny.style.left = leftPos + "px";
    if (didFinish(greeny)) {
      displayResults(myVote, greeny);
      clearInterval(greenTimer);
      clearInterval(blueTimer);
    }
  }

  function moveBlueTurtle() {
    let bloo = $("b-turtle");
    // ok if not done with window.getComputedStyle
    leftPos += 4;
    bloo.style.left = leftPos + "px";
    if (didFinish(bloo)) {
      displayResults(myVote, bloo);
      clearInterval(greenTimer);
      clearInterval(blueTimer);
    }
  }
})();
```

Turbo Turtles Rubric

___ / 3 Event handling

___/ 1 Correctly handling window.onload (separate from Otherwise Good Use (OGU))

___/ 1 Successfully adding onclick to both buttons (actually setting up onclicks)

___/ 1 Storing the user's pick of which turtle in some way (module global)

Can be a String/DOM element/Number

___ / 4 Timer handling

___/ 1 Correctly setting timer values as module globals

___/ 1 Correctly creating timers for both turtles when either button clicked.

.5 points for an near miss.

remember you can do this either as

```
setTimer(function() { otherFunct(param)},  time)
```

or

```
setTimer(otherFunc, time, param)
```

Full credit if they setInterval but don't store (but they won't get for clearing)

___/ 1 Correctly clearing timers for both turtles when one wins

No double jeopardy if they forget to set module globals but they clear correctly (they should get this point)

___/ 1 Function to handle what happens on each timer event includes check for winner

Ok if not refactored.

___ / 4 DOM manipulation

___/ 1 Disabling buttons when race starts

___/ 1 Getting the position of the turtle and parsing the integer

___/ 1 Adding 4 pixels to location and resetting the left position correctly with px

___/ 1 Attempting to do *something* to set title to anything at the end of the race (whether that's actually setting the title themselves or calling the function to do it).

___ / 4 JavaScript usage

___/ 1 Correctly calling function to get random value (or if they missed this correctly creating random value themselves).

___/ 1 Correctly calling function to check if the turtle crossed the finish line.

___/ 1 Correctly calling the function to display who won using the global and the winner

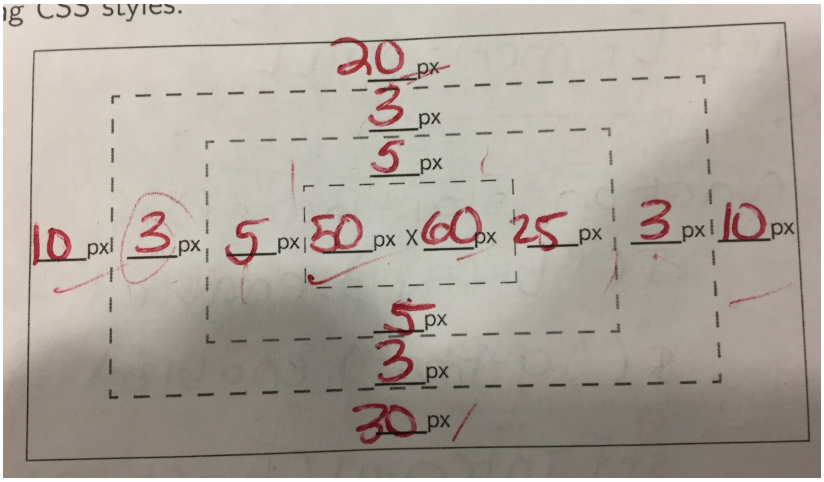
___/ 1 Otherwise good use of the language (one freebie)

-1 if functions are not declared correctly

Don't worry about refactoring

Short answer

5.1



___ /2 Rubric

_____/1.5 Non override numbers correct (at most 2 wrong)

_____/.5 Overrides correct

-0 if they get the ordering of the width and height wrong

5.2 Some possible answers, 1 point for each of 2 correct (total 2 pt)

Blind user who uses a screen reader

- Using alt tags on images
- Making sure the language is set in the <html> tag

Low vision user who uses magnification

- Using large fonts on your screen
- Keeping style separate from content so you can easily switch between small and big fonts.

Color blind users

- Making the screen readable in greyscale vs colors
- Using colors that work for color blind users

Mobility limited who can't control a mouse

- Ensuring all interactions can be handled with keyboard interactions (tabs)
- voice controls

5.3. Some possible answers include:

- To separate style from content so you can switch easily (unobtrusive CSS)
- Make the HTML and CSS more maintainable (must be clear that it is HTML and CSS, "code" isn't enough)
- Makes HTML more readable.

5.4. Some possible answers include:

- let has localized scoping (has to be clear which has localized scoping)
- it's in our code quality guide and this is what our classroom "company" style dictates.
- student needs to indicate some understanding of scoping and var having a broader scope than let
- Not accepted: because it's in a newer version of JavaScript (while true, that is not the reason why we use it in this class).

5.5. Some possible answers include:

- Wraps code in a anonymous function that is declared and immediately called so that there are 0 global symbols
- So variables don't pollute the global namespace
- Localizing our variables to our js file.

5.6. Some possible answers include:

- `5 == "5"` is true but `5 === "5"` is false
- ½ pt for 5 == 5.0 is true but 5 === 5.0 is false (they get the idea of value *and* type for === but treat Integer vs. Double instead of Number)

5.7. The answer is C.

5.8. (1 pt for all three correct - ½ pt if they get 2 out of 3 correct)

```
justify-content: space-between;
align-items: center;
flex-direction: column-reverse;
```