<table>
<thead>
<tr>
<th>Question</th>
<th>Score</th>
<th>Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTML</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>Short Answer</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>CSS/DOM Tree</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>JS/Animations</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td>JS/DOM/UI</td>
<td>12</td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>50</strong></td>
<td></td>
</tr>
</tbody>
</table>
1. (HTML) (1 pt each) What's wrong with my HTML?

Consider the following HTML:

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

Solutions (5 of 6 for full credit)

Note: There were duplicate line numbers for 7/8 on the printed version

1. Line 4: Invalid closing tag for title on line 4 (fix is <title>Web dev</title>)
2. Line 5: <link> is self-closing (fix is <link href="index.css" rel="stylesheet" />)
3. Duplicate id of "important" (fix is to remove one, either on <h1> of line 8 or <p> of line 17
4. Line 13: Last li tag should be closed with </li>, not </p>
5. Line 16: Missing alt tag on <img> (fix may be adding alt="puppy image")
6. Line 20: <footer> and its contents must be inside <body> (fix is to move it above </body>)

Note: For full credit you needed to mark on the HTML itself (where the error was), describe why this was an error, and what the fix was for each. We only considered descriptions that were written on the lines given (per our instructions).

We also note that our HTML had duplicate line numbers in the actual exam. We did not note anyone having problems because of this fact.
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>``)

Possible solutions:

● To clearly separate content (HTML) from behavior
● To avoid function overriding between different files (e.g. if myFunction were also defined in a linked JS file)

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

Possible solutions:

● For accessibility reasons: screen readers rely on HTML5 semantic tags to help read content on the page in an organized manner (can easily distinguish between section content and other regions like `<nav>` or `<aside>`
● For code quality reasons: `<div>` has no semantic meaning, relies on adding classes where you could otherwise keep class/id use limited in HTML for cleaner source code. Easier to maintain HTML code when tag roles are clearly distinguished with semantic tags

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

```css
#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.

Possible Solutions:

● 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 within our JS file (ideally localized as much as possible within functions).

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

7 == “7” evaluates to true
7 === “7” evaluates to false
6. (2 pts) Consider the following JSON object:

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

**Solutions:**

a. `miniJSON.rainbow[1] : "skittles"`

b. `miniJSON.candy corn : error`

c. `miniJSON.smarties.length : 7`

d. `miniJSON[miniJSON["smarties"]].length : 3`

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).

```javascript
console.log("ghost"); // 1
(function() {
    console.log("pumpkin"); // 2
    window.addEventListener("load", initialize);
    foo();

    function initialize() {
        console.log("dinosaur"); // 4
    }

    function foo() {
        console.log("cat"); // 3
    }
})();
```
3. **(CSS)** r/css_irl

**Part A Solution:**

```css
body {
    font-family: Helvetica, sans-serif;
}

h1 {
    text-align: center;
}

/* main content layout (to put buttons under lightpost and center page) and lightpost layout */
main, main > div {
    display: flex;
    flex-direction: column;
    align-items: center;
}

main > div {
    background-color: black;
    justify-content: space-evenly;
    height: 450px;
    width: 180px;
}

/* circles in lightpost */
div div {
    border-radius: 50%;
    height: 120px;
    width: 120px;
}

button {
    font-weight: bold;
    margin: 10px;
    padding: 10px;
    width: 80px;
}

#red {
    background-color: red;
}

#green {
    background-color: green;
}

#yellow {
    background-color: yellow;
}
```
Part B Solution:
4. (JS/DOM/Events) Click-Click, Who's There? (12 pts)

Solution:

```javascript
(function() {
  "use strict";
  const TREATS = ["apples", "candycorn", "skittles", "smarties"];
  window.addEventListener("load", initialize);
  // begin solution
  function initialize() {
    $("door").addEventListener("dblclick", findCandy);
    $("finish").addEventListener("click", showResults);
    $("reset").addEventListener("click", resetGame);
  }
  function resetGame() {
    this.classList.add("hidden");
    total = 0;
    $("results").innerText = "; // can also hide
    while ($("candy-list").children.length > 0) {
      $("candy-list").children[0].remove();
    }
  }
  function findCandy() {
    let rand = parseInt(Math.random() * treats.length);
    let treat = treats[rand];
    let randCount = Math.ceil(Math.random() * 10);
    total += randCount;
    let li = document.createElement("li");
    li.innerText = "You got " + randCount + ";
    let img = document.createElement("img");
    img.src = treat.replace(" ", "-") + ".png";
    img.alt = treat;
    li.appendChild(img);
    if ($("candy-list").children.length === 0) {
      $("finish").classList.remove("hidden");
    }
    $("candy-list").appendChild(li);
  }
  function showResults() {
    $("results").innerText = "That's " + total + " treats over " + qsa("li").length + " visits!";
    $("reset").classList.remove("hidden");
    $("finish").classList.add("hidden");
  }
})();
```

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

Note: There are a few possible solutions to this problem, we have provided two common ones below:

```javascript
// Solution with setInterval
(function() {

    "use strict";

    /** Part A) */
    const LIGHTS = ["red", "red", "green", "green", "yellow"];
    let timer = null;
    let pos = 0;
    window.addEventListener("load", initialize);

    /** Part B) */
    function switchColor(color1, color2) {
        $(color1).classList.add("off");
        $(color2).classList.remove("off");
    }

    /** Part C) */
    function initialState() {
        $(LIGHTS[pos % LIGHTS.length]).classList.add("off");
        pos = 0;
        $("red").classList.remove("off");
        clearInterval(timer);
        timer = null;
        ":start".disabled = false;
        this.disabled = true;
    }

    /** Part D) */
    function initialize() {
        $("green").classList.add("off");
        $("yellow").classList.add("off");
        $(":start").addEventListener("click", startLight);
        $("reset").addEventListener("click", reset);
    }

    /** Part E) (other functions) */
    function startLight() {
        timer = setInterval(changeLight, 500);
        this.disabled = true;
        $(":reset").disabled = false;
    }

    function changeLight() {
        let c1 = LIGHTS[pos % LIGHTS.length];
        let c2 = LIGHTS[(pos + 1) % LIGHTS.length];
        switchColor(c1, c2);
        pos++;
    }

})(); // end module

// Solution with setTimeout
```
(function() {
    "use strict";

    /** Part A) */
    const LIGHTS = ["red", "green", "yellow"];
    let timer;
    let pos = 0;
    window.addEventListener("load", initialize);

    /** Part B) */
    function switchColor(color1, color2) {
        $(color1).classList.add("off");
        $(color2).classList.remove("off");
    }

    /** Part C) */
    function initialState() {
        $(LIGHTS[pos % LIGHTS.length]).classList.add("off");
        pos = 0;
        $(LIGHTS[pos]).classList.remove("off");
        clearInterval(timer);
        timer = null;
        $("start").disabled = false;
        this.disabled = true;
    }

    /** Part D) */
    function initialize() {
        $("start").addEventListener("click", redLight);
        $("reset").addEventListener("click", reset);
        $("green").classList.add("off");
        $("yellow").classList.add("off");
    }

    /** Part E) (other functions) */
    function redLight() {
        timer = setTimeout(function() {
            switchColor("red", "green");
            greenLight();
        }, 1000);
    }
    function greenLight() {
        timer = setTimeout(function() {
            switchColor("green", "yellow");
            yellowLight();
        }, 1000);
    }
    function yellowLight() {
        timer = setTimeout(function() {
            switchColor("yellow", "red");
            redLight();
        }, 500);
    }
})(); // end module