CSE 154: Web Programming Summer 2019

Exam 1 Key

1. HTML (4 pts): What's wrong with my HMTL?

- Duplicate "text" ID
- Missing alt text on img
- Mismatch closing
- <link> should use href instead of src
- Invalid closing </DOCTYPE>
- <footer> should be in <body>

2. Selectors (5 pts): `.classic` Advertising

Selector	IDs of selected elements
`ol li`	`#h, #i, #l`
`body > p`	`#b`
`section li strong`	`#j, #m`
`ol, ul`	`#g, #k`
`ul li`	`#l`

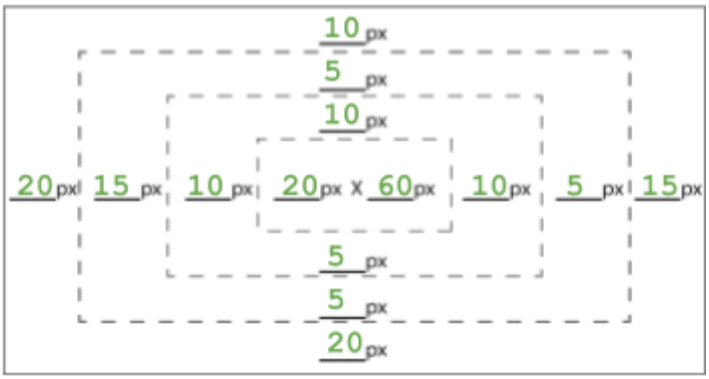
3. CSS Styling (9 pts): Husky Post-Its!

```
/* or h1, article/p */
body {
  text-align: center;
  font-family: Verdana, sans-serif;
}

#bulletin {
  width: 600px;
  margin: auto auto;
  border: 5px solid black;
  background-color: tan;
}

#postits {
  display: flex;
  justify-content: space-evenly;
}

article {
  width: 150px;
  height: 150px;
  margin: 20px;
  background-color: yellow;
}

p {
  color: #9932cc;
  margin: 10px;
}

.pin-row {
  display: flex;
  justify-content: space-between;
  width: 100%;
}

.pin {
  width: 15px;
  height: 15px;
  border-radius: 50%;
  border: 2px solid black;
  background-color: white;
}
```

4. Short Answer (11 pts).

1. Box Model (2pts): In the Box Model diagram to the right, label the corresponding Box Model properties given the CSS styles for a #content div (the size of the box is denoted as width x height):

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



2. Inline vs. Block Layout (1pt): What is the difference between inline elements and block elements? (1-2 sentences)

A block element takes up the full block (width of parent element) and starts a new line. Inline elements span only the width of their content, and are often inside of block elements (block elements should not be children of inline elements). Cannot set most box model properties of inline elements.

3. Web Accessibility (2pts): As web developers, what are two different ways we can make our web pages more accessible to different users? (1-2 sentences per answer).

- Provide descriptive alt text in images for screenreaders
- Use semantic tags to aid navigation for screenreaders and other accessible technologies
- Use contrasting CSS color properties on the page to support different color visions (e.g. color-blindness)
- Use distinct font sizes for headings, paragraphs, etc. to support low-vision users and make page hierarchy clear to ease reading/comprehension
- Use descriptive captions for figures, avoid directional text (e.g. “the button on the left”)
- Use label tags on form elements
- Use relative instead of absolute widths of elements to support different screensizes (e.g. phones) for users with different visions

4. Types and Equality in JS (1.5pts): In 1-2 sentences, identify the difference between “==” and “===” evaluation in JS and provide one example where == returns a different result than ===.

Difference: == compares value, while === is a strict comparison of type and value.

Example where `x == y` is different than `x === y`:

```
x = 5; y = "5";
```

5. JavaScript Events (2 pts): 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");                                    // 1
(function() {
   console.log("Bar");                                  // 2

    window.addEventListener("load", pageLoad);
    foo();

    function pageLoad() {
      console.log("Baz");                               // 4
    }

    function foo() {
      console.log("Mumble");                            // 3
    }
 })();
```

6. JSON Mystery (2.5 pts):

Statement	Return Value
`cse154.instructor`	`"Melissa Hovik"`
`cse154.sections[3]`	`{"AE" : 8}`
`cse154["sectionTAs"][1]`	`"Hudson"`
`cse154.sections[0].AB`	`14`
`cse154["qtr-year"]`	`"Su 2019"`

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

One possible solution is provided below:

```
"use strict";
(function() {
  const TREATS = ["apples", "candycorn", "skittles", "smarties"];
  window.addEventListener("load", init);

  let total = 0;

  function init() {
    id("door").addEventListener("dblclick", findCandy);
    id("finish").addEventListener("click", showResults);
    id("reset").addEventListener("click", resetGame);
  }

  function resetGame() {
    this.classList.add("hidden");
    total = 0;
    id("results").innerText = ""; // can also hide
    id("candy-list").innerHTML = "";
  }

  function findCandy() {
    let rand = parseInt(Math.random() * TREATS.length);
    let treat = TREATS[rand];
    let randCount = Math.ceil(Math.random() * 10);
    total += randCount;
    let li = gen("li");
    li.textContent = "You got " + randCount + " ";
    let img = gen("img");
    img.src = treat + ".png";
    img.alt = treat;
    li.appendChild(img)
    if (id("candy-list").children.length == 0) {
      id("finish").classList.remove("hidden");
    }
    id("candy-list").appendChild(li);
  }

  function showResults() {
    id("results").textContent = "That's " + total + " treats over " +
        qsa("li").length + " visits!";
    id("reset").classList.remove("hidden");
    id("finish").classList.add("hidden");
  }
})();
```

6. JS/Timers (9 pts). Victory Laps!

One possible solution is provided below:

```
(function() {
  // Part A: Define any module-globals here
    let stopwatchTime = 0;
    let lapTime = 0;
    let timerId = null;
  window.addEventListener("load", init);
  function init() {
    id("stopwatch").addEventListener("click", toggleStopwatch);
    id("reset").addEventListener("click", resetStopwatch);
    id("lap").addEventListener("click", lap);
  }

  // Part B: #stopwatch button event handler
  function toggleStopwatch() {
      if (timerId === null) {
        timerId = setInterval(function() {
          stopwatchTime++;
          lapTime++;
          console.log(stopwatchTime + " seconds");
        }, 1000);
      } else {
        clearInterval(timerId);
        timerId = null;
      }
  }

   // Part C: #reset button event handler
  function resetStopwatch() {
    stopwatchTime = 0;
    lapTime = 0;
  }

  // Part D: #lap button event handler
  function lap() {
      // Note that this Part also required correct management of lap timer in
      // rest of program
      console.log("Lap time: " + lapTime + " seconds");
      lapTime = 0;
  }
})();
```