CSE 154: Web Programming Summer 2019

Practice Exam 1: Example A Key

1. What's Wrong with my H⊥ML?

```
<!DOCTYPE html>
<head>
    <h1>Mowgli's Magical Muffins</h1>
    <link src="mypage.css" rel="stylesheet" />
</head>
<body>
    <p>For Doggies' Best Friends:</p>
    <ul>
        <li>Multi-grain Melody</li>
        <li>Merry-Mint-Chip</li>
    </ul>
    For Doggies:
    <ul>
        <li>The Malt-ese</li>
        <li>Malamint Magic<li>
        <li>Meow Meows</li>
    </ul>
</body>
</!DOCTYPE html>
```

Solution (any 5 of the following received full-credit):

1. The link tag needs the href attribute, not src

2. No content tags should be in <head> - <h1> should be moved into <body>

3. All text in the body should be in a content tag - "For Doggies:" could be in a <p> tag to fix this

4. There's no such thing as a closing </!DOCTYPE html> tag

5. The "Malamint Magic" should be followed by , not

6. Missing <html> and </html>

2. You Selected the Right Class.

Consider the following HTML:

```
<html>
    <heading>
        <title>CSE 154 Course Web Page</title>
    </heading>
    <body>
        <header id="title-1">
            <h1 id="title-2"><em id="em-1">All the CSE 154 Course Stuffff Ever</em></h1>
        </header>
        <p id="subtitle-1">Topics:</p>
        <ul id="list-1">
            <li id="topic-1">What is the Internet</li>
            <li id="topic-2">How to do the Internet</li>
            <li id="topic-3">How to make the Internet</li>
            <li id="topic-4">
                Make cool projects:
                <ol id="list-2">
                    <li id="hw-1">Make Pies</li>
                    <li id="hw-2">Watch Lion King</li>
                    <li id="hw-3">Read <em id="em-2">rly rly rly</em> fast</li>
                    <li id="hw-4">Push squares around</li>
                    <li id="hw-5">Catch 'em all!</li>
                </ol>
            </li>
        </ul>
        <div id="div-1">
            <img id="img-1" src="mowgli.jpg">Our course mascot!</img>
        </div>
    </body>
</html>
```

Solution:

1. `p` #subtitle-1
2. `ol li` #hw-1, #hw-2, #hw-3, #hw-4, #hw-5
3. `li em` #em-2
4. `ul > li` #topic-1, #topic-2, #topic-3, #topic-4
5. `li li` #hw1, #hw-2, #hw-3, #hw-4, #hw-5

3. Short Answers

1. What is the difference between inline elements and block elements?

Solution: Inline elements (e.g. <a>, , etc.) do not start a new line and have a default width of their content. Block elements (e.g. <h1>, <section>, <p>, etc.) do start a new line, and span 100% width of their parent element.

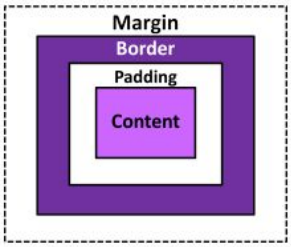
2. Why do we always want to include an alt attribute on img tags?

Possible Solutions:

- Users who cannot see the image due to vision impairment can have a textual description of the image (which can be spoken aloud by a screenreader)
- If the image fails to load (connection, broken path, etc.), the alt text is displayed instead
- SEO (Search Engine Optimization) benefits for page ranking

3. What's the difference between margin, borders, and padding? (You may provide a labeled diagram)

Solution:



4. Why is it important to specify multiple font styles for the same element in your CSS? (e.g., `font-family: Helvetica, Arial, sans-serif;`)

Solution: To specify fallback fonts in case the primary font is not available on the system, with a system default font sharing the same font type as the preferred (earlier) fonts (e.g. serif, sans-serif, monospace, or cursive).

5. Why is it important to use the module pattern in JavaScript?

Possible Solutions:

- Wraps code in an 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).

6. What is the difference between `setInterval` and `setTimeout`?

Solution: setInterval specifies a function to be repeated every given ms, while setTimeout specifies a function to be executed exactly once after a delay of the given ms.

7. Consider the following JSON object:

```
let miniJSON = {
  "foo" : ["b", 1, 2],
  "bar" : 0,
  "FOO" : "Foo?"
};
```

Solutions:

```
a. miniJSON.foo : ["b", 1, 2]
b. miniJSON["FOO"] : "Foo?"
c. miniJSON["FOO"][1] : "o"
d. miniJSON[foo] : error
e. miniJSON["foo"].length : 3
```

4. JS/DOM/Events: Planting DOM ~~Trees~~ Tulips!

```
(function(){
  function grow(item) { /* Details of function not provided  */ }
  window.addEventListener("load", init);

  /** Begin Problem 4 solution below: */
  function init() {
    let spots = qsa(".spot");
    for (let i = 0; i < spots.length; i++) {
      spots[i].addEventListener("click", function() { grow(spots[i]) });
    }
    id("grow-all").addEventListener("click", growAll);
    let rows = qsa(".row");
    for (let i = 0; i < rows.length; i++) {
      rows[i].addEventListener("dblclick", addSeed);
    }
    id("grow-all").disabled = false;
  }

  function addSeed() {
    let newDiv = document.createElement("div");
    newDiv.classList.add("spot");
    newDiv.classList.add("seed");
    newDiv.addEventListener("click", function() { grow(newDiv) });
    this.appendChild(newDiv);
  }

  function growAll() {
    let allSpots = qsa(".spot");
    for (let i = 0; i < allSpots.length; i++) {
      grow(allSpots[i]);
    }
  }
})();
```

5a. Code Execution (10 pts): Duck, Duck, Goose!

5a. Program A (5 Points):

Console output:	
Line 1	*1 ducks*
Line 2	*2 ducks*
Line 3	*goose*
Line 4	*1 ducks*
Line 5	*2 ducks*
Line 6	*3 ducks*
Line 7	*goose*
Line 8	*1 ducks*

5a. Program B (5 Points):

Console output:	
Line 1	*1 ducks*
Line 2	*2 ducks*
Line 3	*3 ducks*
Line 4	*goose*
Line 5	*4 ducks*
Line 6	*5 ducks*
Line 7	*goose*
Line 8	*6 ducks*

5B. JS/Timers - Code Writing (10pts): The Final Countdown!

```
(function () {
  let timerId = null;
  let count = 10;

  window.addEventListener("load", init);

  function init() {
    id("btn").addEventListener("click", handleClick);
  }

 /**
   * click -> 10, 9, 8, 7
   * click -> stop
   * click -> 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0, stop
   * **/
  function handleClick() {
    // start countdown
    if (timerId === null) {
      timerId = setInterval(countdown, 1000);
    } else {
      clearInterval(timerId);
      timerId = null;
      count = 10;
    }
  }

  function countdown() {
    console.log(count);
    count--;
    if (count < 0) {
      clearInterval(timerId);
      timerId = null;
      count = 10;
    }
  }
```