CSE 154: Web Programming — Summer 2019

Practice Exam 1: Example 2 **Key**

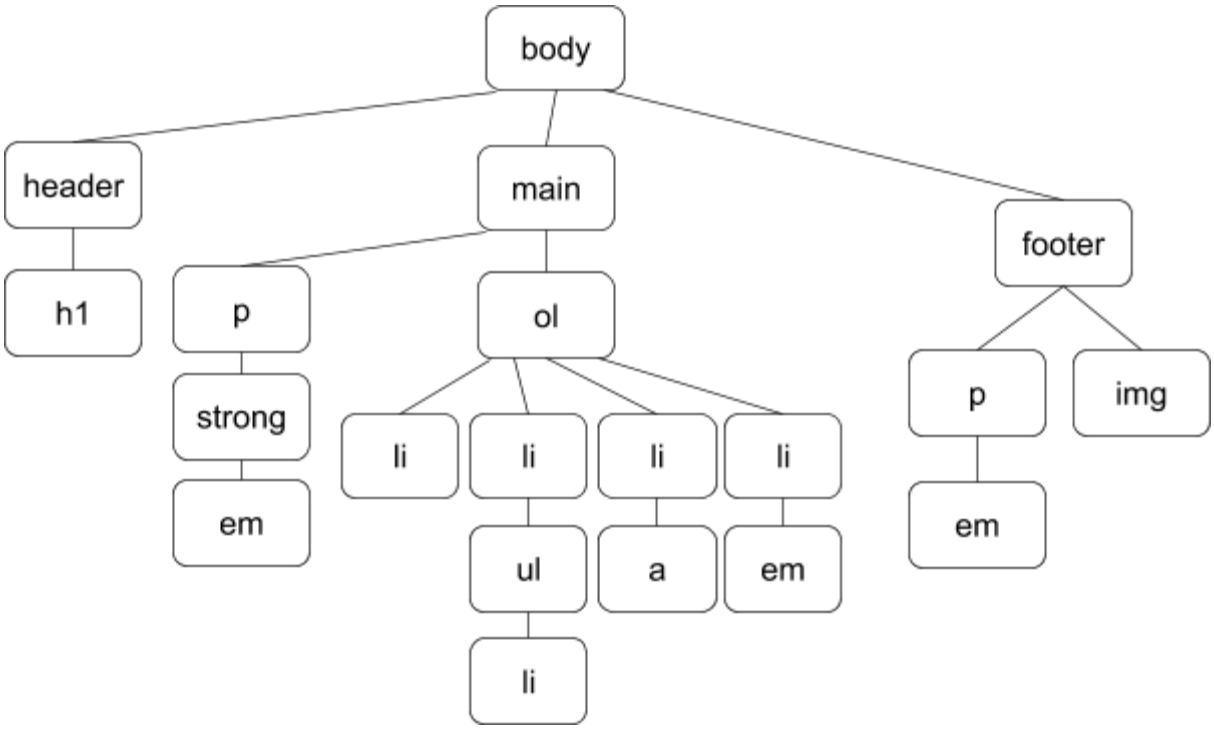
1. (HTML/CSS) A Special Spec for a Special Doggy

```
<!-- HTML Solution -->
<body>
   <h1>Abby's Style Guide</h1>
   <p>Abby's Top 3 tips on how to maximize puppy cuteness:</p>
   <div>
      <img src="sleepy-abby.png" />
      <p>1. Get at least 12 hours of beauty rest every day.</p>
   </div>
   <div>
      <img src="abby-with-toy.png" />
      <p>2. Always have someone to snuggle with.</p>
   </div>
   <div>
      <img src="seahawks-abby.png" />
      <p>3. Live half your life as a bean. With sports swag.</p>
   </div>
</body>
```

```
/* CSS Solution */
body {
     font-family: Verdana, sans-serif;
     text-align: center;
     width: 50%;
     margin: auto auto;
     border-left: 2px dashed #e91e63;
     border-right: 2px dashed #e91e63;
}
div {
     width: 350px;
     font-size: 12pt;
     font-weight: bold;
     margin: 20px auto;
}
div img {
     display: block;
     margin: 2px auto;
     width: 350px;
}
h1 {  text-decoration: underline; }
body > p {
     font-style: italic;
     font-size: 12pt;
}
```

2. Drawing the DOM

Solution:



3. Turbo Turtles

Solution:

```
(function () {
   // Provided functions:
   function getRandomValue(min, max) { /* code not displayed here */ }
   function didFinish(turtle) { /* code not displayed here */ }
   function displayResults(pick, winner) { /* code not displayed here */ }

   // Solution begins here:
   let greenTimer, blueTimer = null;
   let myVote;
   window.addEventListener("load", function () {
      id("g-ftw").addEventListener("click", startRace);
      id("b-ftw").addEventListener("click", startRace);
   });

   function startRace() {
      if (this.id == "g-ftw") {
         myVote = id("g-turtle");
      } else {
         myVote = id("b-turtle");
      }
      id("g-ftw").disabled = true;
      id("b-ftw").disabled = true;
      let greenSpeed = getRandomValue(1, 250);
      let blueSpeed = getRandomValue(1, 250);

      /* 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 = id(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);
      }
   }
})();
```

4. Short Answers

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

Possible solutions: To clearly separate content from appearance; to reduce redundancy and unused or overridden styles, to allow easily changing a webpage theme with a linked CSS file without needing to change the HTML.

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.

Possible Solutions: Use HTML5 semantic tags (screen readers use these to read content in an organized way)

b. low-vision user who uses magnification.

Possible Solutions: Use bold font for emphasis, clearly distinguish font sizes between headings/paragraphs; use easy-to-read font families, use em-based font sizes rather than px-based.

c. color-blind user.

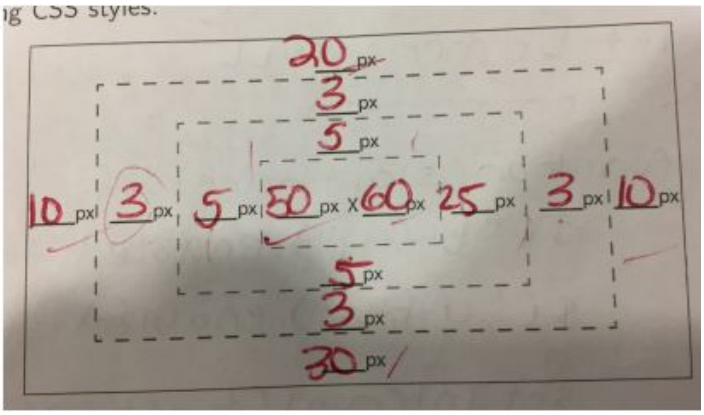
Possible Solutions: Use color palettes in CSS that are readable for different types of color-blindness (various software is available to see filters for different types of color-blind users)

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

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

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.

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 to our js file (ideally within functions to localize scope as much as possible).

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

Possible solution: `5 == "5"` is true but `5 === "5"` is false

6. JSON

a. miniJSON.pancake : “waffle”
b. miniJSON["FOO"] : undefined
c. miniJSON[“POPTARTS”].flavors[1] : “strawberry”
d. miniJSON[miniJSON["pancake"]].length : 1

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");                          // 1
(function() {

  console.log("Bar");                        // 2
  window.addEventListener(“load”, pageLoad);
  foo();

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

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