CSE 154: Web Programming — Autumn 2017

Midterm Exam

Name: _____

UWNet ID: _____@uw.edu

TA (or section): _____

Rules:

- You have 50 minutes to complete this exam.
- You will receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book and open note.
- You may not use any electronic or computing devices, including calculators, cell phones, smartwatches, and music players.
- Unless otherwise indicated, your code will be graded on proper behavior/output, not on style.
- Do not abbreviate code, such as writing ditto marks ("") or dot-dot-dot marks (...). You may not use JavaScript frameworks such as jQuery or Prototype when solving problems.
- You may use JavaScript function aliases like $ or qs **only if** you define them in your code.
- You may use the fetch syntax used in class in any JavaScript programs you write.
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your Student ID to a TA or instructor for your submitted exam to be accepted.

Question	Score	Possible
0. HTML		5
1. CSS		5
2. JavaScript		10
3. JavaScript/Ajax		15
4. Short Answer		5
Total		40

0. What's Wrong with my H⊥ML?

```
1  <!DOCTYPE html>
2      <head>
3          <h1>Mowgli's Magical Muffins</h1>
4          <link src="mypage.css" type="text/css" rel="stylesheet" />
5      </head>
6      <body>
7          <p>
8              For Doggies' Best Friends:
9          </p>
10         <ul>
11             <li>Multi-grain Melody</li>
12             <li>Merry-Mint-Chip</li>
13         </ul>
14         For Doggies:
15         <ul>
16             <li>The Malt-ese</li>
17             <li>Malamint Magic<li>
18             <li>Meow Meows</li>
19         </ul>
20     </body>
21 </!DOCTYPE html>
```

This HTML document won't validate, and would generate errors and warnings in the W3C Validator. However, it is possible to make 5 modifications to the HTML to make it pass validation. Each modification might result in multiple text "changes" to the HTML document, but is considered one modification because it is addressing the same root problem. Indicate the 5 modifications we need in order to make it pass validation. Write directly on the HTML.

Below, briefly describe the changes that you made, and why you had to make each change in order to validate. If it is unclear what changes go with which explanations, feel free to number your changes on the HTML. No need for an essay — 10 words or less should be plenty.

1. _____

2. _____

3. _____

4. _____

5. _____

1. You Selected the Right Class.

Consider the following HTML:

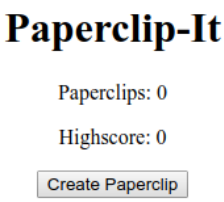
```
<html>
    <heading>
        <title>CSE 154 Course Web Page</title>
    </heading>
    <body>
        <header id="title-1">
            <h1 id="title-2"><em id="em-1">All the CSE 154 Course Stuffff Ever</em></h1>
        </header>
        <p id="subtitle-1">Topics:</p>
        <ul id="list-1">
            <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>
```

Write the ID's of the elements selected by each of the given selectors:

1. `p`
   _____
2. `ol li`
   _____
3. `li em`
   _____
4. `ul > li`
   _____
5. `li li`
   _____

2. Gotta Make That 📎📎📎

Write a JavaScript program to add to the given HTML and CSS (next page). This program lets a user run their own paperclip factory at the click of a button! Who knew your new web development skills could come in handy?

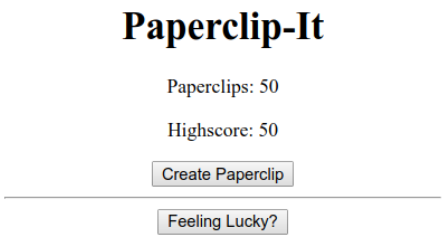


Initial Page View

Your job is to simulate creating a single paperclip whenever the "Create Paperclip" (`#paperclip-it`) button is clicked. When the page is initially loaded, the user has 0 paperclips. When the button is clicked, the current paperclip count (in `#count`) should be incremented by 1.

Feeling Lucky: The first time the user generates 50 paperclips, they unlock the "Feeling Lucky" option, which should display the `#secret-div` (this div is already hidden with a `.hidden class` when the page is loaded). When visible, the user can test their luck at the paperclip jackpot by clicking on the "Feeling Lucky" button. When a user clicks "Feeling Lucky" then there should be a 25% chance that the user will double their paperclips, and a 75% chance that they instead lose all their paperclips (the "Feeling Lucky" button still remains visible).

Before user unlocks "Feeling Lucky" button



After user unlocks "Feeling Lucky" button

Highscores: Below the paperclip count is the user's current highscore (the highest number of paperclips they have ever had). You should update the high score (in `#highscore`) whenever the current paperclip count increases and is higher than the current high score. For example, if a user's highscore is 12 paperclips and they go from having 12 to 13 paperclips, their highscore should become 13 as well. If their highscore was instead 57 and they went from 12 to 13 paperclips, the highscore would remain 57.

Write your solution (JavaScript) on the next page, under the provided HTML and CSS.

```
<!DOCTYPE HTML>
<!-- HTML for Problem 2 -->
<html>
    <head>
        <script src="paperclips.js" type="text/javascript"></script>
        <link href="paperclips.css" type="text/css" rel="stylesheet" />
    </head>
    <body>
        <h1>Paperclip-It</h1>
        <p>Paperclips: <span id="count">0</span></p>
        <p>Highscore: <span id="highscore">0</span></p>
        <button id="paperclip-it">Create Paperclip</button>

        <div class="hidden" id="secret-div">
            <hr />
            <button id="lucky-button">Feeling Lucky?</button>
        </div>
    </body>
</html>
```

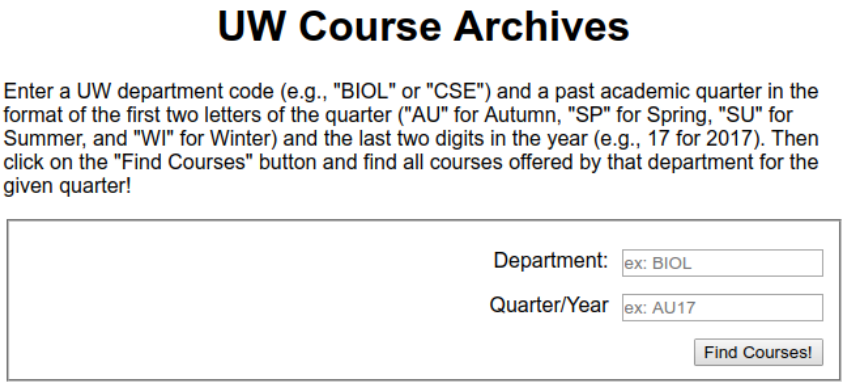
```
/* CSS for Problem 2 */
body {
  margin: auto auto;
  text-align: center;
  width: 30%;
}

.hidden {
  display: none;
}
```

Write your solution (JavaScript) here.

continue your (JavaScript) here.

3. Fetch ALL the Courses!

Write a JavaScript program `courses.js` which fetches and displays information about current course offerings at UW. This program will be used with the HTML and CSS provided on the following page.



Initial Page View

You will fetch information from this url: `https://uw.edu/courses.php`. There are two required query parameters:

1. `qtr`: the quarter code for the quarter being searched (e.g., "AU17" or "SP15").
2. `dept`: the department code for the department being searched (e.g, "CSE" or "BIOL").

When a user clicks the "Search" button, you should make a request to this webservice with the value of the `#dept` text input and `#qtr` text input (you do not have to modify any user input before putting it in the url). For example: `https://uw.edu/courses.php?qtr=au17&dept=BIOL`.

If there are results, a JSON response will be returned with the following keys:

- **`dept_name`**: the full department name (e.g., "Biology" for "BIOL")
- **`qtr`**: the full quarter string (e.g., "Autumn 2017" for "AU17")
- **`courses`**: an array of all courses for that quarter and department.

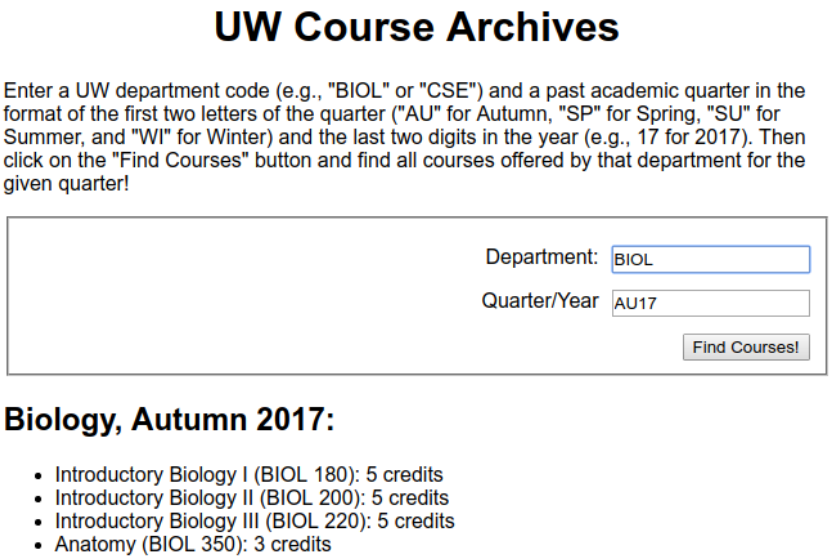
For example, the call to `https://uw.edu/courses.php?qtr=au17&dept=BIOL` would return a JSON response of all courses offered in the Biology department in the Autumn 2017 quarter.

Example Response for `https://uw.edu/courses.php?qtr=au17&dept=BIOL`:

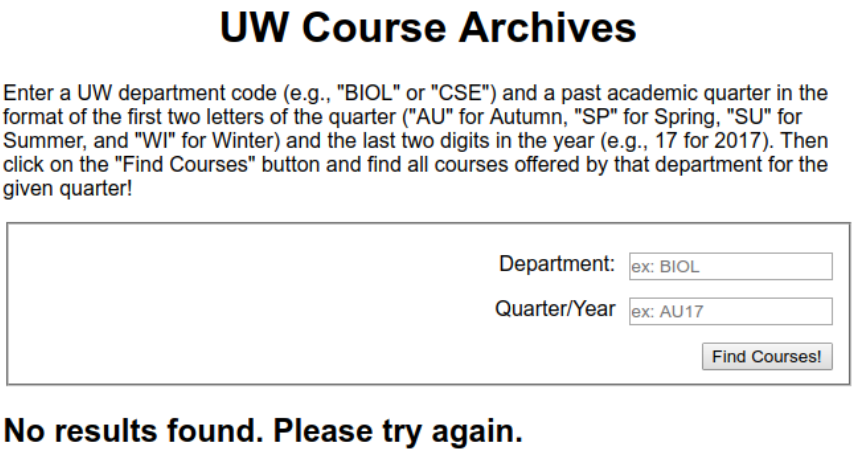
```
{
  dept_name: "Biology",
  qtr: "Autumn 2017",
  courses: ["Introductory Biology I (BIOL 180): 5 credits",
            "Introductory Biology II (BIOL 200): 5 credits",
            "Introductory Biology III (BIOL 220): 5 credits",
            "Anatomy (BIOL 350): 3 credits"]
}
```

If no results are found, the webservice will respond with a non-200 error status (that will be thrown as an error in the checkStatus function).

If results are found (no error), you should populate the existing `#results-summary` element with the full department name followed by a ", ", the full quarter name, and a ":", and then display the information for each course in `<li>` tags in the `#course-results <ul>` in the provided HTML. For the above JSON example, the output would look like the following:



If there was an error, you should display "No results found. Please try again." in the `#results-summary` element and clear the contents of the `#course-results`, as shown below:



Write your solution (JavaScript) on the following page, after the provided HTML and CSS.

```
<!DOCTYPE HTML>
<!-- HTML for Problem 3 -->
<html>
    <head>
        <script src="courses.js" type="text/javascript"></script>
        <link href="courses.css" type="text/css" rel="stylesheet" />
    </head>
    <body>
        <h1>UW Course Archives</h1>
        <p>
            Enter a UW department code (e.g., "BIOL" or "CSE") and a past academic
            quarter in the format of the first two letters of the quarter
            (e.g., "AU" for Autumn) and the last two digits in the year (e.g., 17 for 2017).
            Then click on the "Find Courses" button and find all courses offered by
            that department for the given quarter!
        </p>
        <fieldset>
            <p>
                Department: <input placeholder="ex: BIOL" id="dept" type="text" />
            </p>
            <p> Quarter/Year <input placeholder="ex: AU17" id="qtr" type="text" /> </p>
            <button id="search">Find Courses!</button>
        </fieldset>
     <h2 id="results-summary"></h2>
     <ul id="course-results">
     </ul>
   </body>
</html>

/* CSS for Problem 3 */
body {
  font-family: Helvetica, Arial, sans-serif;
  margin: auto auto;
  width: 50%;
}

fieldset p {
  text-align: right;
}

h1 {
  text-align: center;
}

input {
  margin-left: 10px;
}

input, button {
  float: right;
}
```

Write your solution (JavaScript) here.

4. Short Answers

1. List 2 inline elements and 2 block elements (do not use `<img>` as any of your answers - it's a special "inline-block" element).

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

3. What are 2 different JS debugging strategies/tools we have used in lecture and/or section?

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

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