CSE 154: Web Programming — Autumn 2018

Practice Final Exam 1 Key

Name:

UWNet ID: @uw.edu

TA (or section):

Rules:

- You have 110 minutes to complete this exam.
- You will receive a deduction if you keep working after the instructor calls for papers.
- 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.
- 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**
HTML/CSS		15
Short Answer		18
JS/DOM/Animations		12
JS with AJAX		20
PHP Web Service		20
SQL		15

1. HTML/CSS (Part A): Query Selectors

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

1. HTML/CSS (Part B): Writing CSS

```
body {
  background-color: #123456;
}
h1, h2 {
  color: teal;
}
.instruction {
  border: 2px dashed red;
}
#recipe-list {
  width: 40%;
}
#recipe-area {
  width: 60%;
}
.ingredients li {
  font-family: Arial, Helvetica, sans-serif;
}
```

2. Short Answers

1. What is one reason to use semantic tags instead of a in HTML?

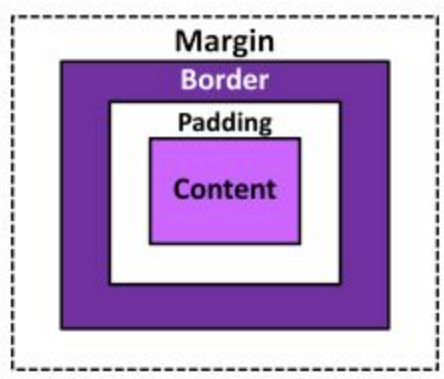
Possible solution: Provides semantic information useful for screenreaders and other accessibility devices. For example, screen readers use semantic tags to speak out the page structure/hierarchy in a more intuitive way without visual cues available to the user.

2. What is one example of validating user input on the client-side?

Solution: Using HTML5 input tag attributes like required, pattern, and minlength; using JS to check input format before sending a fetch request

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

Solution:



4. 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.

5. What is the difference between a GET and POST request?

Solution: A GET request is used primarily to retrieve data from the server, and may include parameters in the request URL. A POST request is primarily used to send data to the server (data may still be returned in the response) and any parameters sent are encrypted (e.g. using FormData).

6. What is one advantage of using a SQL database over text files to store data?

Possible solutions: More secure, easier to handle multiple requests from clients to process/modify data in a database, more efficient (in terms of space and time).

7. For each of the two regular expressions, circle all the string(s) below that match it:

i. `/[A-Za-z]+@[0-9]/`

- `foo+@5`
- **`foo@123`**
- **`45foo@321`**
- `f8@8f`

ii. `/^F*\.jpg$/`

- **`F.jpg`**
- `^FFF.jpg$`
- **`FFF.jpg`**
- `F\.jp`

8. Suppose a directory has the following structure:

```
test.php
mydir/
    images/
        puppy1.jpg
        puppy1.png
        puppy2.gif
    puppy-facts.txt
    puppy-haz-pizza.jpg
```

What do each of the following statements return if written in test.php? Use [] notation for any arrays and put strings in "".

Statement	**Return Value**
`scandir("mydir")`	**`[".", "..", "images", "puppy-facts.txt", "puppy-haz-pizza.jpg"]`**
`scandir("mydir/images")`	**`[".", "..", "puppy1.jpg", "puppy1.png", "puppy2.gif"]`**
`glob("mydir/puppy-facts.txt")`	**`["mydir/puppy-facts.txt"]`**
`glob("mydir/*/*")`	**`["mydir/images/puppy1.jpg", "mydir/images/puppy1.png", "mydir/images/puppy2.gif"]`**
`glob("mydir/puppy*")`	**`["mydir/puppy-facts.txt", "mydir/puppy-haz-pizza.jpg"]`**

3. The Little Traveler (JS/DOM/Animations)

```
(function() {
  "use strict";

  window.addEventListener("load", function() {
    addBox();
    timer = setInterval(updateBox, 200);
  });

  function updateBox() {
    let box = qsa(".little-box")[0];

    let sides = []; // top, right, down, left
    let topSide = parseInt(window.getComputedStyle(box).top);
    let leftSide = parseInt(window.getComputedStyle(box).left);
    if (topSide >= 20) {
      sides.push("top");
    }
    if (leftSide >= 20) {
      sides.push("left");
    }
    if (topSide <= 480) {
      sides.push("bottom");
    }
    if (leftSide <= 480) {
      sides.push("right");
    }
    let randomSideIndex = Math.floor(Math.random() * sides.length);
    let randomSide = sides[randomSideIndex];
    if (randomSide == "top") {
      box.style.top = topSide - 20 + "px";
    } else if (randomSide == "bottom") {
      box.style.top = topSide + 20 + "px";
    } else if (randomSide == "right") {
      box.style.left = leftSide + 20 + "px";
    } else if (randomSide == "left") { // left
      box.style.left = leftSide - 20 + "px";
    }
  }

  function addBox() {
    let littleBoxCount = qsa(".little-box").length;
    let littleBox = document.createElement("div");
    littleBox.classList.add("little-box");
    id("box").appendChild(littleBox);
  }
})();
```

4. Fetching Pets Who Fetch with Fetch (JS/AJAX)

```
(function() {
  "use strict";
  let ans;

  window.addEventListener("load", initialize);

  function initialize() {
    dropDown();
    pet();
    id("guess-btn").addEventListener("click", guess);
  }

  function dropDown() {
    let url = "pets.php?mode=tas";
    fetch(url)
      .then(checkStatus)
      .then(updateDD);
  }

  function updateDD(result) {
    result = result.split("\n");
    for (let i = 0; i < result.length; i++) {
      let tag  = document.createElement("option");
      tag.innerText = result[i];
      tag.value = result[i];
      id("ta-list").appendChild(tag);
    }
  }

  function pet() {
    let url = "pets.php?mode=random";
    fetch(url)
      .then(checkStatus)
      .then(JSON.parse)
      .then(updatePet);
  }

  function updatePet(resultJSON) {
    id("pet-name").innerText = resultJSON.petname;
    id("pet-type").innerText = resultJSON.type;
    let img = id("pet-img");
    img.src = resultJSON.images[Math.floor(Math.random() * resultJSON.images.length)];
    if (img.classList.contains("hidden")) {
      img.classList.remove("hidden");
    }
    id("guess-btn").classList.remove("disabled");
    ans = resultJSON.name;
  }
```

```
function guess() {
    id("guess-btn").classList.add("disabled");
    let dropDown = id("ta-list");
    let guess = dropDown.value;
    if (guess == ans) {
      let correct = id("correct").innerText;
      correct = parseInt(correct) + 1;
      id("correct").innerText = correct;
    }
    let total = id("total").innerText;
    total = parseInt(total) + 1;
    id("total").innerText = total;
    pet();
  }
})();
```

5. The Thing That We Fetch From For Fetching Pets Who Fetch (PHP)

```php
<?php
  $pets = file("pets.txt");

  if(isset($_GET["mode"]) &&  $_GET["mode"] == "tas") {
    header("Content-type: text/plain");
    $final_array = array();
    foreach($pets as $line) {
      $ta_array = explode(" ", $line, 4);
      if(!in_array($ta_array[0], $final_array)) {
        array_push($final_array, $ta_array[0]);
        echo "$ta_array[0]\n";
      }
    }
  } else if(isset($_GET["mode"]) && $_GET["mode"] == "random") {
    $pet = $pets[array_rand($pets)];
    list($name, $petname, $type, $age) = explode(" ", $pet, 4);
    $images = glob("$name/$petname/*.jpg");
    foreach($images as $image) {
      $image = str_replace("$name/$petname/", "", $image);
    }
    $output = [
      "name" => $name,
      "petname" => $petname,
      "type" => $type,
      "age" => $age,
      "images" => $images
    ];
    header("Content-type: application/json");
    echo json_encode($output);
  } else {
    header("HTTP/1.1 400 Invalid Request");
    header("Content-type: text/plain");
    echo "Error: Please pass in a mode parameter of tas or random.";
  }
?>
```

6. SQL Queries

1. Write a SQL query to list the first and last names of all the female actors that have a first name that starts or ends with "W".

```
SELECT first_name, last_name
FROM actors
WHERE gender = 'F' AND (first_name LIKE "W%" OR first_name LIKE "%W");
```

2. Write a SQL query to list all columns out of the directors table for all directors that have directed a movie with a rank of 8 or higher, order by last name of director in alphabetical order. Return one row per unique director.

```
SELECT DISTINCT d.id, d.first_name, d.last_name
FROM directors d
JOIN movies_directors md ON d.id = md.director_id
JOIN movies m ON m.id = md.movie_id
WHERE m.rank >= 8
ORDER BY d.last_name ASC;
```