CSE 154: Web Programming

Autumn 2018

Final Exam | Key

Question	Score	Possible
HTML/CSS		15
Short Answer		18
JS/DOM/Animations		12
JS with AJAX		20
PHP Web Service		20
SQL Queries		15
Extra Credit		1

1.A. Debug * the Things! (CSS Query Selectors, 5pts)

1. `ul strong`

#strong-2, #strong-3

2. `.fun`

#tip-4, #list-2

3. `li li`

#sub-1, #sub-2, #sub-3

4. `ul > li`

#tip-1, #tip-2, #tip-3, #tip-4

5. `li > *`

#strong-2, #list-2, #strong-3

1.B. Take a Set Before You Go. (CSS Writing, 10pts)

```
h1 {
      font-family: Helvetica, sans-serif;
      text-align: center;
      /* need to set spacing here; if you don't then you end up not removing
         default margin to h1 on bottom */
      margin-bottom: 5px;
}

.row {
      display: flex;
      justify-content: space-between;
}

.card {
      width: 200px;
      height: 75px;
      display: flex;
      border: 0.35em solid black;
      border-radius: 15px;
      justify-content: space-around;
      align-items: center;
}

img {
      height: 60px;
}
```

2. Short Answers (18 points)

1. HTML/CSS. Consider the following HTML used to render a styled paragraph on a webpage:

```
<div class="paragraph">
  <font face="Arial">Welcome to our cookie store!</font>
  You will <b>never</b>, <i>ever</i>, <u>EVER</u> beat
  <font size="154px" color="purple">OUR</font> recipes!
</div>
```

a. What is one issue with this HTML based on what we've learned in Module 1 (HTML/CSS)? (1pt)

Possible solutions include:

- Use of deprecated tags like <font>, <b>, <i>, <u>
- <div> instead of <p>
- Using px for font size (not egregious, but ok answer)
- Inline CSS in HTML

b. Describe 1 specific change you would make to improve the provided HTML. (1pt)

Possible solutions include:

- Use <strong> vs. <b>, <em> vs. <i>
- Factor out styling in CSS file
- Use <p> instead of <div class="paragraph">

2. JavaScript/CSS. While we have discussed the benefits of using classes (e.g. `.hidden`) and CSS to style elements on a page, we have also shown a few cases where it is better to set the style of an element using `element.style.<property>` in JavaScript. Give an example of such a case. (1pt)

Possible solutions include:

- When updating elements dynamically using computed styles (such as when moving elements and updating their positions on the page in animations, or calculating the width of a health/progress bar)
- When calculating styles randomly (e.g. random hex color codes for background colors)

3. Asynchronous Requests. Consider ***a single function*** that makes ***multiple requests to different web services*** in JavaScript. Assuming each request does not depend on a response of another request, what is the benefit of these requests being made asynchronously (all at once) vs. synchronously (one after another)? (1pt)

Possible solutions include:

- Asynchronous requests limit page freezing which occurs when synchronous requests cause delays between requests/user interactions
- Requests can be made all at once, allowing normal page behavior even if an error occurs (and doesn't break page functionality)
- Better use experience when user can interact with elements while different requests are being processed simultaneously

4. Event Listeners. Solution: b

5. Data Storage Trade-Offs.

a. It is possible to store/process data on a server using .txt or .json files. What is one advantage of using SQL databases to store data instead? (1.5pt)

Possible solutions include:

- More efficient/secure to store and modify data in SQL databases, especially when multiple requests to modify data are made
- More user-friendly syntax for writing filtered queries, joining datasets, sorting results, etc.
- Much easier to organize and scale large datasets with SQL databases compared to txt and JSON files

Partial credit solutions did not adequately justify why the example was an advantage of SQL vs. JSON/txt

b. Recall that `localStorage` can be used to store data on a user's browser. What is one example where it would be more appropriate to store data with `localStorage` instead of SQL?: (1.5pt)

Possible solutions include

- To store user preferences such as the background color of a web page (or emoji usage as many said)
- To retrieve information that infrequently changes from the server and store it locally, speeding up client side interactions: example: downloading all possible pokemon
- To store a single user's information that does not need to be secure, nor does it need to be uploaded to a server to "aggregate" the data. An example might be the save state of a game

6. GET vs. POST. In 2-3 sentences, identify a specific example where it is better to use a POST request instead of a GET request and justify your choice. (2pt)

Possible solutions include:

- Inserting or modifying into a database, such as for a banking or restaurant site. POST parameters are encrypted, and are more secure than GET parameters. POST requests should be used when updating data on the server, where as GET requests should only be used to get information (note that both requests still retrieve information)

Full credit solutions need to provide a specific example and sufficiently justify the choice for why it is more appropriate as POST vs. GET.

7. PDO Connections. What is the purpose of a PDO object in a PHP web service? (1pt)

Solution: The PDO object in PHP encapsulates the connection between PHP server and the (SQL) database. It can be used to query and/or modify the database. prepare/execute statement on the PDO object can be used to prevent SQL injection when a query is made.

8. PHP/SQL Security.

Which option is more secure? Justify your choice. (1.5pt)

Solution: Option 2 is more secure because it is using the PDO prepare/execute. This prevents SQL injection when if a string in one of the variables ($todo, $author, or $todo_type) contain a ‘ character.

9. Regex. For each of the two regular expressions, circle all the string(s) below that match it (2pts):

i. `/^[A-Z]+4\d\d$/`

- **`CSE431`**
- `CSE154`
- `http400`
- `ABC4dd`
- `404`

ii. `/^pup.*y.jpg$/`

- **`puppy.jpg`**
- `^puppypuppy.jpg$`
- **`pupy.jpg`**
- **`puppyparty.jpg`**
- `puppykitty.JPG`

10. **PHP FIle I/O.**

Statement	**Return Value**
`scandir("mydir")`	`[“.”, “..”, “images”, “lotsa-pups.txt”]`
`glob("mydir/all-the-puppies.txt")`	`[]`
`glob("*/*pup*")`	`[“mydir/lotsa-pups.txt”]`

3. It's Time for the Holidays! (JS/DOM/Timers, 12 points)

```
// Parts A/B
function initialize() {
  $("start-show").addEventListener("click", createLights);
}

function createLights() {
  let count = $("count").value;
  if (count && count > 0 && count < 10) {
    for (let i = 0; i < count; i++) {
      let strand = document.createElement("div");
      strand.classList.add("strand");
      for (let j = 0; j < 15; j++) {
        let light = document.createElement("div");
        light.classList.add("light");
        light.style.backgroundColor = getRandomColor();
        strand.appendChild(light);
      }
      $("lights-view").appendChild(strand);
    }
    $("main-view").classList.add("hidden");
    $("lights-view").classList.remove("hidden");
    startAnimation();
  } else {
    $("error-msg").classList.remove("hidden");
  }
}


// Part c
function startAnimation() {
  setInterval(function() {
    let lights = document.querySelectorAll(".light");
    for (let i = 0; i < lights.length; i++) {
      lights[i].style.backgroundColor = getRandomColor();
    }
  }, INTERVAL);
}
```

4. AJACKSketch (JS/AJAX, 20 points)

```
(function() {
  "use strict";
  const URL = "ajacksketch.php";

  window.addEventListener("load", initialize);

  // Part A solution
  function initialize() {
    $("works-list").addEventListener("change", updatePage);
    dropDown();
  }


  // Part b solution
  function dropDown() {
    let url = URL + "?mode=works";
    fetch(url)
      .then(checkStatus)
      .then(updateDD)
      .catch(console.log);
  }

  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];
      $("works-list").appendChild(tag);
    }
  }

  // Part c solution
  function updatePage() {
    // instead of $("works-list") could also be this
    let url = URL + "?mode=items&type=" + $("works-list").value;

    fetch(url)
      .then(checkStatus)
      .then(JSON.parse)
      .then(updateWorks)
      .catch(console.log);
  }
  function updateWorks(resultJSON) {
    $("works-displayed").innerHTML = "";
    $("description").innerText = resultJSON.description;

    let data = resultJSON.items;
    for (let i = 0; i < data.length; i++) {
      let div = document.createElement("div");
      let img = document.createElement("img");
      img.src = data[i].image;
      img.alt = data[i].name;
      div.appendChild(img);

      let name = document.createElement("p");
      name.innerText = data[i].name;
      div.appendChild(name);

      $("works-displayed").appendChild(div);
    }
  }

})();
```

5. A Portfolio in PHP (PHP Web Service, 20pts)

Part a solution:

```
function error_message($msg) {
  header("HTTP/1.1 400 Invalid Request");
  header("Content-type: text/plain");
  die($msg);
}
```

Part b solution:

```
function get_all_item_info($type) {
  $output = array();

  $output["type"] = $type;
  $output["description"] = trim(file_get_contents("$type/description.txt"));

  $items = array();
  $dirs = scandir($type);
  foreach($dirs as $dir) {
    if ($dir !== "." && $dir !== ".." && $dir !== "description.txt") {
      $obj = get_item_info($type, $dir);
      array_push($items, $obj);
    }
  }
  $output["items"] = $items;
  return $output;
}
```

Part c solution:

```
 <?php

   if (isset($_GET["mode"])) {
     $mode = $_GET["mode"];
     if ($mode === "works") {
       header("Content-type: text/plain");
       print(trim(file_get_contents("works.txt")));
     } elseif ($mode === "items" && !isset($_GET["type"])) {
       error_message("Missing type parameter for item request");
     } elseif ($mode === "items") {
       $works = file("works.txt", FILE_IGNORE_NEW_LINES);

       $output = array();

       if (in_array($_GET["type"], $works)) {
         // if the named type is in the listed types of his works.
         $output =  get_all_item_info( $_GET["type"]);
       }

       header("Content-type: application/json");
       print json_encode($output);
     }
   } else {
     error_message("Please pass in a mode of works or items");
   }

   # assumed functions work from Parts A/B
?>
```

6. Around The World (SQL, 15pts)

a. Solution:
```
SELECT name, gnp
FROM countries
WHERE gnp > 123456
AND continent = "Asia";
```

b. Solutions:
```
-- WHERE solution
SELECT l.language, c.name, l.percentage
FROM languages l, countries c
WHERE l.country_code = c.code
AND c.population < 85000
AND l.percentage >= 50
ORDER BY l.language ASC, l.percentage DESC;

-- JOIN solution
SELECT l.language, c.name, l.percentage
FROM languages l
JOIN countries c ON l.country_code = c.code
WHERE c.population < 85000
AND l.percentage >= 50
ORDER BY l.language ASC, l.percentage DESC;
```

c. Solutions:
```
-- WHERE solution
SELECT ci.name, c.continent
FROM cities ci, countries c, languages l
WHERE ci.country_code = c.code
AND l.country_code = countries.code
AND ci.name LIKE "%sea%"
AND l.official = 'T'
AND l.language = "English"
ORDER BY ci.name;

-- JOIN solution
SELECT ci.name, c.continent
FROM cities ci
JOIN countries c ON ci.country_code = c.code
JOIN languages l ON l.country_code = c.code
WHERE ci.name LIKE "%sea%"
AND l.official = 'T'
AND l.language = "English"
ORDER BY ci.name;
```