

Question	Score	Possible
1. Short Answer		20
2. CSS		13
3. JS/DOM		15
4. PHP Web Service		20
5. JS with Fetch		20
6. SQL Queries and PDO		12
7. Extra Credit		1

1. Short Answers (20pts total)

1. HTML: Happy Little DOM Trees! (2.5pts)

1. <p>
2. <section>
3. or
4.
5. <footer>

2. Web Accessibility (2pts) What are 2 different ways we have learned for making a website more accessible?

Possible answers include:

- Semantic tags
- Descriptive alt text on images
- Responsive layout (e.g. to support different screen resolutions for BVI users)
- Relative layout (% vs. px)
- Descriptive <label> tags in forms
- Color contrast with CSS to support different color visions
- Support for alternative navigation (e.g. keyboard)

3. Event Listeners (2pts)

a	b	c	d
ribbit! meow! woof! quack! meow! woof!	meow! woof! ribbit! quack! meow! woof! ribbit!	ribbit! woof!	meow! woof! woof! quack! meow! woof!

4. JS Timers (2pts)

a	b	c	d	e
1 doggo 2 doggo DUBS! 3 doggo 4 doggo DUBS! 5 doggo 6 doggo	1 doggo 2 doggo 3 doggo DUBS!	1 doggo 2 doggo DUBS! 1 doggo 2 doggo DUBS! 1 doggo 2 doggo	1 doggo 2 doggo DUBS! 3 doggo 4 doggo 5 doggo DUBS! 6 doggo	1 doggo 2 doggo DUBS! DUBS! DUBS! DUBS! DUBS! DUBS!

5. PHP Error-Handling (2pts)

a. Provide a specific example where it would be more appropriate to return a 400 error instead of a 503 error in a web service:

Possible answers include specific types of invalid requests:

- Missing/invalid parameter required for a GET/POST request

or specific web service examples:

- Missing a required GET parameter for HW4/HW5
- Parameter value does not correspond to a found resource on the server (e.g. character directory)

b. Provide a specific example where it would be more appropriate to return a 503 error instead of a 400 error in a web service:

Possible answers include:

- Database connection is down
- User credentials for PDO are incorrect

6. GET vs. POST (2pts)

a. Provide an example where a GET request would be more appropriate than a POST request for a web service.

Possible answers include:

- Requests which don't send sensitive data
- Search queries
- Requests where we might want to cache/bookmark the url
- Request that solely gets information from a server

b. Provide an example where a POST request would be more appropriate than a GET request for a web service.

Possible answers include:

- Sending data with a form
- Requests where data is more sensitive and should be encrypted (e.g. login)
- Requests where the server will be modified

7. Validation Methods (2pts)

a. What is one advantage of validating user input on the client (HTML5 or JS) rather than on the server (PHP)?

Possible answers include:

- It is faster to validate on the client than on the server since the validation doesn't have to wait for the input to be sent in a request
- Saves network/server resources with fewer requests to server
- It can lead to a better UI/UX experience due to quicker/custom feedback to a user about why their input is invalid
- It can help prevent malicious user input by preprocessing the input before sending to the server

b. What is one advantage of validating user input on the server as opposed to on the client?

Possible answers include:

- Validating user input on the server is more secure than validating on the client, client can't see source code vulnerabilities in server-side code
- Can check input against data on server such as username/passwords, which (hopefully) aren't on the client-side

8. Data Storage Trade-Offs (2pts)

a. Recall that `localStorage` can be used to store data on a user's browser. In 1-2 sentences, explain when it is more appropriate to store data with `localStorage` instead of with a SQL database.

Possible answers include:

- If you want to fetch information from the server and store it locally vs retrieving it again from the server each time you need it.
- If you want to store information across browser sessions so a user doesn't have to retrieve it from the server multiple times.
- Use `localStorage` if you want to store a set of user/machine specific preferences on the local machine.

b. 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?

Possible answers include:

- SQL databases are more secure than `txt/JSON` files
- Easier to handle multiple requests from clients to process/modify data in a database
- It is often more efficient (in terms of space and time) to manage data with SQL than with `txt/JSON` files
- Databases allow for transactions to allow concurrent users to modify parts of the database which can't be done with `txt/JSON`
- Easier to query/update data with SQL

9. PHP File I/O (1.5pts)

Statement	Return Value
<code>scandir("birds");</code>	<code>[".", "..", "eagle", "heron", "merlin"]</code>
<code>glob("birds/merlin/i*");</code>	<code>["birds/merlin/image.png", "birds/merlin/info.txt"]</code>
<code>glob("birds/*/mp3");</code>	<code>["birds/eagle/call1.mp3", "birds/heron/call1.mp3", "birds/heron/call2.mp3", "birds/merlin/call1.mp3"]</code>

10. Regex (2pts)

i. `/^[irA]{3}b*[n]+b?$/`

- A3bnnnb
- **rAin**
- iiiBnB
- AAibnbb
- **Airbnb**

ii. `/^[A-Z]L(3){1,}..$/`

- **AL3rt**
- GL33..\$
- **XL33TX**
- LLLL
- WILL32

2. CSS Writing (13pts): Swimmin' in CSS

Part A: The Fountain Mallards

```
#mallard-container {
  width: 100px;
  height: 200px;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: space-around;
  border: 4px dashed green;
}

#mallard-container img {
  height: 40px;
  width: 40px;
}
```

Part B: Instructor Party!

```
#party-container {
  height: 200px;
  width: 80%;
  display: flex;
  flex-wrap: wrap;
  justify-content: center;
}

.party-pic {
  height: 40px;
  width: 40px;
  border-radius: 50%;
  border: 2px solid purple;
}
```

Part C: Button Container

```
button {
  text-align: center;
  font-family: Verdana, sans-serif;
  font-size: 14pt;
  background-color: gold;
  color: indigo;
  width: 300px;
  margin-left: 10px;
  padding-top: 10px;
  padding-bottom: 10px;
}

#button-container {
  display: flex;
  justify-content: center;
}
```

3. JS with DOM (15pts): Jump in with JS!

```
(function() {  
  "use strict";  
  window.addEventListener("load", init);  
  
  // Images of all the CSE 154 instructors  
  const PICS = ["ann.jpg", "chao.jpg", "conner.jpg", "daniel.jpg", "hawk.jpg", ... ];  
  
  // Returns array with [randomTop, randomLeft] values, where each value is an  
  // integer for a position for a .duckie to fit in the #fountain.  
  function getDuckiePos() {  
    // Details of function not provided here  
  }  
  
  /* BEGIN ONE POSSIBLE SOLUTION */  
  function init() {  
    id("add-duckie").addEventListener("click", addDuckie);  
    id("party-mode").addEventListener("click", partyMode);  
  }  
  
  function addDuckie() {  
    let duckie = document.createElement("div");  
    duckie.classList.add("duckie");  
    let coordinates = getDuckiePos();  
  
    let randomX = coordinates[0];  
    let randomY = coordinates[1];  
  
    duckie.style.top = randomY + "px";  
    duckie.style.left = randomX + "px";  
    id("fountain").appendChild(duckie);  
  }  
  
  function partyMode() {  
    if (id("party-container").children.length > 0) {  
      id("party-container").innerHTML = "";  
      id("party-mode").innerText = "~Party On~";  
    } else {  
      for (let i = 0; i < PICS.length; i++) {  
        let instructor = document.createElement("img");  
        instructor.src = PICS[i];  
        instructor.alt = "Party instructor!";  
        instructor.classList.add("party-pic");  
        id("party-container").appendChild(instructor);  
      }  
      id("party-mode").innerText = "Party Off";  
    }  
  }  
  
  /* END SOLUTION */  
}) ();
```

4. PHP Web Service (20pts): Club CSE 154!

Part A:

One possible solution is below:

```
function get_chars() {
    $files = glob("data/*");
    $result = "";
    for ($i = 0; $i < count($files); $i++) {
        $result = $result . basename($files[$i]) . "\n";
    }
    return $result;
}
```

Part B:

One possible solution is below:

```
function get_char_info($chardir) {
    $contents = file("data/$chardir/info.txt", FILE_IGNORE_NEW_LINES);
    $results = array(
        "name" => $contents[0],
        "series" => $contents[1],
        "description" => $contents[2]
    );
    $results["appearances"] = file("data/$chardir/appearances.txt", FILE_IGNORE_NEW_LINES);
    return json_encode($results);
}
```


Part C:

One possible solution is below:

```
<?php
if (!isset($_GET["mode"]) || ($_GET["mode"] !== "all" && $_GET["mode"] !== "lookup")) {
    handle_error("Please pass in a mode of all or lookup.");
}

$mode = $_GET["mode"];
if ($mode === "all") {
    header("Content-Type: text/plain");
    echo get_chars();
} else if ($mode === "lookup") {
    if (!isset($_GET["char"])) {
        handle_error("Please pass in a character name to lookup");
    }
    $name = $_GET["char"];
    $info = get_char_info($name);
    header("Content-Type: application/json");
    echo $info;
}

function handle_error($error) {
    header("Content-Type: text/plain");
    header("HTTP/1.1 400 Bad Request");
    die($error);
}
?>
```

5. JS with Fetch (15pts): Gotta Fetch 'em All

```
(function() {
  "use strict";
  const URL = "club154.php";

  window.addEventListener("load", init);

  /* BEGIN ONE POSSIBLE SOLUTION */
  function init() {
    fetch(URL + "?mode=all")
      .then(checkStatus)
      .then(populateChars)
      .catch(handleError);
  }

  function populateChars(response) {
    response = response.split("\n");
    for (let i = 0; i < response.length; i++) {
      if (response[i] !== "") {
        let charData = response[i];
        let img = document.createElement("img");
        img.src = "data/" + charData + "/avatar.png";
        img.alt = charData;
        img.id = charData;
        id("playground").appendChild(img);
        img.addEventListener("click", showChar);
      }
    }
  }

  function showChar() {
    let name = this.id;
    fetch(URL + "?mode=lookup&name=" + name)
      .then(checkStatus)
      .then(JSON.parse)
      .then(populateCharInfo)
      .catch(handleError);
    if (qs(".selected")) {
      qs(".selected").classList.remove("selected");
    }
    this.classList.add("selected");
  }

  function populateCharInfo(response) {
    id("name").innerText = response.name;
    id("series").innerText = response.series;
    id("description").innerText = response.description;
    id("appearances").innerHTML = "";
    for (let i = 0; i < response.appearances.length; i++) {
      let appearance = response.appearances[i];
      let li = document.createElement("li");
      li.innerText = appearance;
      id("appearances").appendChild(li);
    }
    id("spotlight").classList.remove("hidden");
  }
}
```

```
function handleError() {
  id("error").classList.remove("hidden");
  id("spotlight").classList.add("hidden");
  if (qs(".selected")) {
    qs(".selected").classList.remove("selected");
  }
}
/* END SOLUTION */
})();
```

6. SQL and PDO (12pts): Message in a PDOttle

Part A: Basic SQL Queries/Statements

i.
SELECT DISTINCT name FROM posts WHERE question LIKE "%my code%" ORDER BY name;

ii.
INSERT INTO posts (id, name, question, category)
VALUES(541, "Piaz Za Rocks", "Will this be tested?", "other");

or

```
INSERT INTO posts  
VALUES(541, "Piaz Za Rocks", "Will this be tested?", "other");
```

iii.
DELETE FROM posts WHERE question LIKE "%code does not work%";

Part B: PHP with SQL using PDO

i. Briefly explain why this function of using the PDO object is vulnerable to malicious clients.

- It is prone to SQL injection (e.g. DROP TABLE statements) without prepare/execute.

ii. Identify what change(s) you would need to make it more secure using what we've covered in lecture, crossing out and clearly modifying the function where necessary to make the changes while still correctly updating the row in the `posts` table.

```
function update_question_by_id ($id, $new_question) {  
    // you may assume that get_PDO returns the appropriate PDO object  
    $db = get_PDO();  
  
    $str = "UPDATE posts SET question = {$new_question} WHERE id = {$id}";  
    $str = "UPDATE posts SET question = :new_question WHERE id = :id";  
  
    try {  
        $db->exec($str);  
        $params = array("new_question" => $new_question, "id" => $id)  
        $stmt = $db->prepare($str);  
        $stmt->execute($params);  
    }  
    catch (PDOException $ex) {  
        handle_db_error("Can not connect to the database. Please try later.");  
    }  
    echo "Updated the database!"  
}
```