CD Studio: JavaScript
Fall 2023
Midterm

Full Name

_____

This exam has 6 questions worth a total of 40 points. You have 60 minutes to complete it.

There is a mix of multiple choice and open ended questions. Partial credit will be given for answers which are not fully correct, but are in the right direction. If you aren't sure how to do something, write pseudo-code or comments to explain how you would try to do it.

This exam is worth 15% of your final grade.

CD Studio: JavaScript
Midterm

0. Init (1 point)

Write your name on the front of this exam.

CD Studio: JavaScript
Midterm

1. Logic (5 points)

Complete the following logic table, filling in if the expression on the left evaluates to **`true`** or **`false`**. assuming the following variable assignments:

```
let a = 'blue';
let b = 21;
let c = 20;
let isGreen = a === 'green';
```

Expression	Evaluates to
`a.length > (c / 5)`	
`isGreen && b > c`	
`isGreen || b > c`	
`b%c > 0`	
`(a+b).length === 6`	

2. Programming concepts (5 points)

Which of the following are concepts that are part of JavaScript? Mark each statement as **`true`** or **`false`** by marking an x in the appropriate column.

`true`	**`false`**	
___	___	The worst case runtime of searching through an array without first sorting it is **`O(N)`**.
___	___	You can access a variable at any scope, no matter where you declare it.
___	___	The use of **`import`** and **`export`** keywords allow for code to be more modular.
___	___	When a function is called within a function, the initial function must wait for the second function to return before it can return.
___	___	Event listeners can be attached to any HTML element inside of **`<body>`**.

CD Studio: JavaScript
Midterm

3. Loops, arrays, and objects (9 points)

For each of the following snippets of code, determine the value of the variable **`count`** after the code has executed.

```
let count = 0;
for (let i = 0; i < 4; i++) {
  let j = 0;
  while(j <= i) {
    count++;
    j++;
  }
}
```

BAD QUESTION — Didn't Count

```
let count = 0;
let values = [1, 1, 2, 3, 5, 8];

values.forEach((value) => {
  count += value;
});
```

```
let count = 0;
let cats = [
  {name: 'Moon Moon', age: 3, weight: 12},
  {name: 'Peaches', age: 2, weight: 10}
];

for (let i = 0; i < cats.length; i++) {
  count = count + cats[i]['weight'];
}

count = count / cats.length;
```

CD Studio: JavaScript
Midterm

4. Debugging (6 points)

The three sets of code have an error in them that is causing them to not behave as expected. Each set of code begins with a comment explaining its desired behavior. Circle the error and explain the correction needed in order for the code to function properly.

```
// Gets all the elements with class 'link' and set the color to red.
let els = document.querySelector('.link');
els.forEach((el) => { el.style.color = 'red' });
```

```
// Function that calculates the sum of all numbers from 1 to n
// and returns the sum.
let sum = (n) => {
  let runningSum = 0;
  for(let i = 0; i < n; i++) {
    runningSum += i;
  }

  return runningSum;
}
```

```
// A function which recursively calculates the sum of all numbers from
// 1 to n and returns the sum.
let sum = (n) => {
  return n + sum(n-1);
}
```

5. Objects and classes (4 points)

Write out the console output of this program's execution.

```
class Cat {
  name;
  weight;

  constructor(name, weight) {
    this.name = name;
    this.weight = weight;
  }

  feed(foodWeight) {
    this.weight += foodWeight;
  }
}

let printWeights = (cats) => {
  cats.forEach((c) => {
    if (c.weight > 15) {
      console.log(c.name + ' needs a diet!');
    } else {
     console.log(c.name + ': ' + c.weight);
    }
  });
}

let m = new Cat('Moon Moon', 12);
let p = new Cat('Peaches', 10);

m.feed(5);
p.feed(2);

printWeights([m, p]);
```

CD Studio: JavaScript
Midterm

6. Data and DOM manipulation (10 points)

We are writing a program which gets the temperature and current conditions from an API of multiple places, and then creates HTML elements on the screen with the temperature and conditions. Based on the temperature, the temperature element should render as **`blue`** if 36 and under, or **`red`** if over.

A sample API response will look like:

```
{
  weather: [
    {location: 'New York', temperature: 36, conditions: 'clear'},
    {location: 'San Francisco', temperature: 56, conditions: 'foggy'},
    {location: 'Hong Kong', temperature: 48, conditions: 'rain'},
  ]
}
```

You will be responsible for modifying a class **`WeatherVisualizer`** and implementing a **`render()`** function:

- This function takes as parameter the response of an API request such as the one above
- For each location, create a new HTML **`<h2>`** element and add it as a child to the class' property **`this.el`**.
- This **`<h2>`** should have as its inner text the location's temperature and sets the style to **`red`** if the temperature is over 36 and **`blue`** if 36 and under.

Extra Credit (up to 3 points)
Implement your **`render()`** function with the **additional** specifications:

- For each location, create a new HTML **`<div>`** element instead of the **`<h2>`** and add it as a child to the class's **`this.el`** variable.
- Inside of each **`<div>`** create an **`<h2>`** with the location's temperature, and a **`<h3>`** with the conditions and add both as children to the **`<div>`**.
- If the current condition is anything other than **`'clear'`** add a class **`'bad-conditions'`** to the condition **`<h3>`** element.

Remember, correct pseudo code or comments will always get partial credit over nothing at all!

```
class WeatherVisualizer {
  constructor(el, url) {
    this.el = el;

    fetch(url)
      .then((response) => response.json())
      .then((data) => this.render(data));
  }

  render(data) {
```

```
    }
}
```