CD Studio: JavaScript
Fall 2024
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)

Given the variable assignments below, please draw a line from the expression to its result. A result in the right column can be used once, multiple times, or not at all.

```
let a = 0;
let b = 1;
let c = 2024;
```

Expression	Evaluates to
`a + b`	`0`
`c % b`	`1`
`c % b > a`	`2`
`[a, b].length`	`true`
`(2 * b) === (c / 1012)`	`false`

CD Studio: JavaScript
Midterm

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 location in which you place a **`<script>`** tag on an HTML page does not matter.
___	___	Binary search is more efficient than sequential search.
___	___	In object oriented programming, an object can extend another object.
___	___	A function can call other functions, but cannot call itself.
___	___	The main way we manipulate data is through a set of operations which have an acronym called DRY.

3. Loops, arrays, and objects (6 points)

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

```
let count = 0;
const values = ['blue', 'green', 'blue', 'yellow', 'blue'];

for (let i = 0; i < values.length; i++) {
  if (values[i] === values[0]) {
    count  = count + 1;
  }
}
```

```
let count = 0;
const n = 5;

for (let i = 0; i < n; i++) {
  count = count + i
}
```

```
let count = 0;
const cats = [
  {name: 'Moon Moon', age: 3, weight: 9},
  {name: 'Peaches', age: 3, weight: 7}
];

cats.forEach((cat) => {
  count += cat.weight;
})
```

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.

```
// Makes a new array with fruit that weighs at least 5
const fruit = [
  {type: 'Apple', weight: 4.9},
  {type: 'Orange', weight: 5.3},
  {type: 'Orange', weight: 5.2}
];
const filteredFruit = [];
fruit.forEach((fruit) => {
  if (fruit.weight < 5) {
    filteredFruit.push(fruit);
  }
})
```

```
// Function that sets the color of all paragraphs to 'green'
document.querySelector('p').forEach((el) => {
  el.style.color = 'green';
});
```

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

5. Recursion (7 points)

You are a designer at a company called Twigma, which aims to be the design software everyone uses. The next two questions are about adding features to this revolutionary new platform.

Below are snippets of a **`Layer`** class, which aims to be like a Photoshop layer, as well as some code which executes. Write out at the bottom the **`console.log()`** outputs, with each output on a new line.

```
class Layer {
  name;
  sublayers;
  hidden;

  constructor(name, sublayers) {
    this.name = name;
    this.sublayers = sublayers;
    this.hidden = false;
  }

  hide() {
    console.log(`Hiding layer: ${this.name}`);
    this.hidden = true;
    if (this.sublayers.length === 0) {
      return;
    }

    this.sublayers.forEach((layer) => {
      layer.hide();
    });
  }
}

let guides = new Layer('guides', []);
let notes = new Layer('notes', []);

let internal = new Layer('internal', [guides, notes]);
let production = new Layer('production', []);

let main = new Layer('main', [internal, production]);
main.hide();
```

6. Object Oriented Programming (10 points)

You're working with an engineer at Twigma to implement a feature in **`Layer`** from the previous question. This new feature lays out a layer using a revolutionary new tool called Autolayout.

You will be modifying **`Layer`** by implementing:

- **`initInteraction()`** which sets up the event handlers needed to lay out the objects.
- **`layout(gutter)`** which lays out all the objects in the layer horizontally with the given gutter in between each object.

To implement **`initInteraction()`**, you must do the following:

- Add a **`click`** event listener for when a class variable **`layoutButton`** is clicked
- Get the value of a slider, whose HTML element is stored at **`gutterInput`** and convert the string into an integer
- Call **`layout()`** on click with the correct value of the **`gutterInput`**

To implement **`layout(gutter)`**, you must do the following:

- Go through all the objects of the layer, which is stored in a **`objects`** array
- Keep a running count of the **`next x-coordinate`**, starting from **`0`**. Since each object has a different width:
    - The first object should be placed at the coordinate **`0`**
    - The second object should be placed at the coordinate of **`0 + first object width + gutter`**
    - The next object should be placed at the coordinate of **`current value of next x-coordinate + object width + gutter`**
- For each object, call its **`place(x)`** function which places it at the given x-coordinate

Certain variables and functions in Layer are omitted or modified for clarity.

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

```
// Object class which is already implemented for you.
// You can access any of these variables or functions in Layer.
class Object {
  width;
  height;
  x;
  y;

  constructor(width, height)
  place(x)
}

// Layer class that is in progress.
class Layer {
  layoutButton;
  gutterInput;
  objects;

  constructor(layoutButton, gutterInput, objects) {
    this.layoutButton = layoutButton;
    this.gutterInput = gutterInput;
    this.objects = objects;

    this.initInteraction();
  }

  initInteraction() {









  }
```

```
  layout(gutter)

  }
}
```