

CS 409: The Art of Web Programming

Midterm

Name:
NetID:

☐ I understand the course policy on academic integrity.

-
- Read *all* the directions before you begin the exam.
 - You will need to upload a pdf file that contains your responses to each question. We have added places for each question for you to upload your answer.
 - Each .pdf file will need to be named as follows: netid_Question number_response.pdf. For example, student john3 will upload their answer to question 2 as john3_Q2_response.pdf
 - This is an open-book, individual exam. By now, you should be intimately familiar with the course policy on academic integrity. To certify that you understand this policy, please write the word "llama" as the answer to the last question. If you have any questions about what constitutes acceptable behavior, make a private post on Piazza. We take issues of academic integrity very seriously, and you will fail the course if you're caught cheating on an exam.
 - This exam is worth 20% of your total grade in the course. You should attempt to answer every question, as partial credit will be awarded for incomplete work approaching a correct solution. However, you should **not** perform a brain dump on any problem in the hopes that something you write down will be awarded points. Extraneous information (especially *incorrect* extraneous information) will be *heavily* penalized.
-

#	1	2	3	4	5	6	Total
Score							
Possible	10	10	20	20	15	25	100

1. [10 pts] **Get ES6y**

What will the following code print? All awarded points are for explaining your answer correctly - a correct answer with no explanation will receive no points.

```
let foo = p => console.log(p)
```

```
const bar = f => {  
  return p => {  
    p = p + 'CS'  
    f(p)  
  }  
}
```

```
const baz = f => {  
  return p => {  
    p = p + '409'  
    f(p)  
  }  
}
```

```
foo = bar(foo)  
foo = baz(foo)
```

```
foo('RK')
```

2. [10 pts] **Alien Prototypes**

What will the following code print? All awarded points are for explaining your answer correctly - a correct answer with no explanation will receive no points.

```
function Alien() {
  this.numberOfLimbs = 4;
  var retractableLimbs = false;

  this.setNumberOfLimbs = function(numLimbs) {
    this.numberOfLimbs = numLimbs;
  }

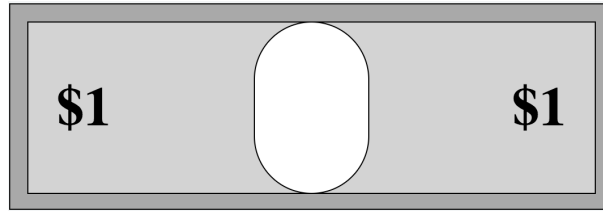
  this.print = function() {
    console.log("Number of Limbs: " + this.numberOfLimbs);
    console.log("Retractable Limbs: " + retractableLimbs);
  }
}

Alien.prototype.setNumberOfLimbs = function(numLimbs) {
  this.numberOfLimbs = 2*numLimbs;
}

Alien.prototype.setRetractableLimbs = function(canRetract) {
  retractableLimbs = canRetract;
}

var experiment626 = new Alien();
experiment626.setNumberOfLimbs(6);
experiment626.setRetractableLimbs(true);
experiment626.print();
```

3. [20 pts] A Dollar For Your Thoughts



Given the following HTML and CSS code, fill in any missing CSS to create the dollar bill drawing above. (Note: depending on your implementation, some sections may be left blank.)

dollar.html

```
<div class="outer">
  <div class="inner">
    <span class="value left">$1</span>
    <span class="value right">$1</span>
    <div class="portrait"></div>
  </div>
</div>
```

dollar.css

```
div {
  border: 1px solid black;
}

.outer {
  width: 500px;
  height: 150px;
  background-color: green;
  /* WRITE YOUR CODE HERE */
}

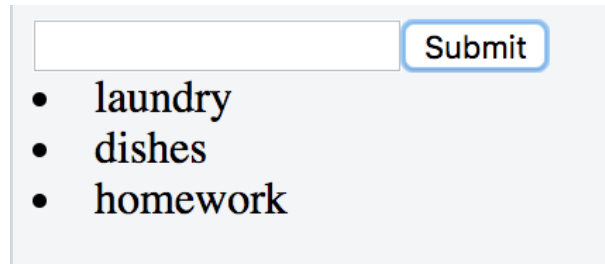
.inner {
  height: 100%;
  width: 100%;
  background-color: lightgreen;
  /* WRITE YOUR CODE HERE */
}

.inner .value {
  margin: 0 25px;
  font-size: 48px;
  font-weight: bold;
  /* WRITE YOUR CODE HERE */
}

.left {
  /* WRITE YOUR CODE HERE */
}
```

```
.right {  
  /* WRITE YOUR CODE HERE */  
}  
  
.portrait {  
  height: 100%;  
  width: 100px;  
  
  border-radius: 100px;  
  background-color: white;  
  /* WRITE YOUR CODE HERE */  
}
```

4. [20 pts] Reactive Tasks



• laundry

• dishes

• homework

Implement the two React components — `TodoList` and `List` — to create a to-do list like the one illustrated above. Users can add a new to-do task to the end of the list by typing it into the input box and clicking on the submit button. (You do not have to implement task removal or re-ordering.)

The `List` component should simply output all the items in the current to-do list and must be used by the `TodoList` component. **Your solution must use ES6. Your solution can utilize either function-based or class-based components.**

Class-based Component Template

```
class TodoList extends React.Component {  
  /* WRITE YOUR CODE HERE */  
}
```

```
class List extends React.Component {  
  /* WRITE YOUR CODE HERE */  
}
```

Function-based Component Template

```
function TodoList {  
  /* WRITE YOUR CODE HERE */  
}
```

```
function List {  
  /* WRITE YOUR CODE HERE */  
}
```

5. [15 pts] **Spotify Schemas**

Suppose you are a backend engineer at Spotify tasked with designing the MongoDB schemas for their app. You design a database with three collections: `songs`, `albums`, and `users`.

- (a) [5] In your schema design, each `album` document maintains a list of `ObjectID` references to the songs that comprise the album.

Here is an sample `song` document:

```
> db.songs.findOne()
{
  _id : ObjectID("AAAA")
  name : "Code Monkey",
  artist : "Jonathan Coulton",
  year: "2006"
  duration: 318000
}
```

Here is a sample `album` document:

```
> db.albums.findOne()
{
  _id : ObjectID("BBBB")
  name : "Thing a Week III",
  artist : "Jonathan Coulton",
  year: "2006",
  songs : [
    ObjectID("AAAA"),
    ObjectID("F17C"),
    ObjectID("D2AA"),
    // etc
  ]
}
```

Complete the MongoDB query code below for extracting an array of all the song titles for the album “Thriller,” by “Michael Jackson.” `album_songs` should contain output like:

```
[{"name": "Baby Be Mine"}, {"name": "The Girl Is Mine"},...]
```

Your MongoDB query code:

```
> var db = connect("localhost:27017/myDatabase");

> var album =

> var album_songs =
```

- (b) [5 pts] Your boss points out that two-way referencing between the `songs` and `albums` collections would be a better schema design. Do you agree with him? Why or why not?
- (c) [5 pts] Your friend, who is a frontend engineer, says he needs a way to correlate songs with listeners' demographic information such as location, age, and gender. He suggests that you augment the `song` document schema to include a list of `ObjectID` references to users who listen to the songs. He claims this strategy would allow him to perform application-level joins to understand the relationship between song and user data. Is this a good strategy? Why or why not?

6. [25] Asynchronicity

(a) [5 pts] Basic Asynchronous Programming

Observe the following code:

```
var color = 'red';
const req = new XMLHttpRequest();
req.addEventListener("load", function () {
    color = json.parse(this.responseText).color;
});
req.open("GET", "/getColor");
req.send();
setButtonColor(color);
```

Assuming /getColor returns the following object...

```
{ 'color': 'yellow' }
```

...and that setButtonColor sets the color of a button on the front end, which of the following is most likely?

- i. The button will be red, and then become yellow.
- ii. The button will be yellow, and then become red.
- iii. The button will be red, never having been yellow.
- iv. The button will be yellow, never having been red.

(b) [5 pts] Continuation Passing Style

Explanation of Continuation Passing Style:

“A function written in continuation-passing style takes an extra argument: an explicit ‘continuation’ (i.e., a function of one argument.) When the CPS function has computed its result value, it ‘returns’ it by calling the continuation function with this value as the argument. That means that when invoking a CPS function, the calling function is required to supply a procedure to be invoked with the subroutine’s ‘return’ value.” For example, here’s factorial written the normal way:

```
function fact(n) {
    if (n == 0)
        return 1 ;
    else
        return n * fact(n-1) ;
}
```

and here’s factorial written the CPS way:

```
function fact(n,ret) {
    if (n == 0)
        ret(1) ;
    else
        fact(n-1, function (t0) {
            ret(n * t0) }) ;
}
```

Question:

Write code using continuation passing style that replicates the behavior shown in the following sequence of images, assuming that `OnClick` is called when the button is clicked, and an endpoint, `'/getColor'`, must be called to determine the next color:



You can assume the existence of the functions `SetButtonColor`, `GetButtonText`, and `SetButtonText`.

(c) [15 pts] Promises

Explanation of Promises:

Continuation passing style often leads to what is known as “callback hell”. Promises are software abstractions meant to deliver us from “callback hell”. “A promise is defined as an object that has a function as the value for the property ‘then’:

```
then(fulfilledHandler, errorHandler, progressHandler).
```

This function should return a new promise that is fulfilled when the given fulfilledHandler or errorHandler callback is finished. This allows promise operations to be chained together. The value returned from the callback handler is the fulfillment value for the returned promise. If the callback throws an error, the returned promise will be moved to failed state.”

Javascript comes with a `fetch` function which returns a promise. Therefore, you can use the ‘then’ method to register callbacks, and these callbacks will receive a single argument – an object representing the response:

```
fetch(...).then(function(result) {...});
```

Question:

Using promises, write a series of calls to an API (can be written as `/endpoint1`, `/endpoint2`, `/endpoint3`) that will generate a webpage that is styled differently for each possible ordering in which they are resolved, and each possibility for *whether* each endpoint’s call is resolved successfully.

The webpage should have three borders surrounding it, in an order reflecting the order in which the endpoints were resolved (with the outermost border being associated with the first endpoint to be resolved). Yellow for endpoint one, blue for endpoint two, and red for endpoint three. Each border should be solid if the endpoint resolved successfully, and dashed if it did not. The resulting code should be written using pure HTML+JS+SCSS.