COSC 465: Computer Networking
Lab 7a: Fun with JavaScript

Name: ____

Setup:

1. In this lab, you’ll be using a “server-side” JavaScript interpreter called node.js (`http://nodejs.org`). If you’re working on the Linux lab machines, open a terminal. If you’re on your own machine, go to `http://nodejs.org` and download an installer (they exist for MacOS X, Windows, and there’s easy-to-compile source code). You can start node.js from the terminal by typing `node`.

2. Read the following JavaScript overview, by a well-respected figure in the Javascript world: `http://javascript.crockford.com/survey.html`. It isn’t that long. You might want to try things out in the interpreter as you’re reading. Just fire up `node` and type away.

   If you liked the reading, read Crockford’s coding conventions `http://javascript.crockford.com/code.html`, and check out other offerings on his site (there’s a great series of videos posted). And if you’re intrigued by what sorts of crazy things can be done in JavaScript, check this out: `http://bellard.org/jslinux/`

The goal for this lab is for you to learn a bit about JavaScript by trying some basic scripts out, seeing what the output is, and trying to understand what the language is doing. You’ll learn about some of the quirks of JavaScript along the way, as well as some common patterns and a few good practices for writing JavaScript code. You’ll put this work to use next week when you create a web server proxy in JavaScript using the node.js framework. JavaScript syntax is very similar to C and Java (though it is very much unlike either of these languages in many respects), so the code should “look” familiar.

For each of the following questions, write what the output would be along with a short explanation for *why* the code does what it does. Each code segment assumes that nothing else has been done previously in the interpreter, so you may want to exit the interpreter and restart it between each problem.

One other reference you may find useful is the Mozilla developer network JavaScript page, which includes a comprehensive language reference, and a guide for how to program with JavaScript: `https://developer.mozilla.org/en/JavaScript/`.

1.
```
function p() {
   i = 42;
   var j = 13;
   console.log("Inside p with i=" + i + " and j=" + j + ".  How shocking.");
}
p();
console.log("Here’s i: " + i);
console.log("And here’s j: " + j);
```

2.

```
function bar() {
    console.log("bar at global scope");
}

function foo() {
    bar();

    function bar() {
        console.log("How did I get here?");
        console.log("something is: " + something);
        console.log("nothing is: " + nothing);
    }

    var something = 10;

    console.log(bar);
}
foo();
```

3.

```
console.log(5 == "5");
console.log(5 == "5 monkeys");
console.log(5 === "5");
console.log(5 !== "5");
var x = "5" + 5;
console.log(x);
x = 5 + "5";
console.log(x);
```

4.

```
var arr = [];
arr[5] = 42;
console.log(arr.length);
```

5.

```
function capedAvengerHeight() {
    return
    {
        batman: 47
    };
}

var bat = capedAvengerHeight();
console.log(bat);
```

6.

```javascript
function foo(x, y, z) {
    arguments[2] = 10;
    console.log(x + " " + y + " " + z);
}
foo(1, 2, 3);
```

7.

```javascript
var myobj = (function() {
    var count = 0;

    return {
        click: function() {
            if (arguments.length === 1) {
                count += arguments[0];
            } else {
                count += 1;
            }
        },

        getClicks: function() {
            return count;
        }
    };
}());

console.log(myobj);
myobj.click();
console.log(myobj.getClicks());
myobj.click(5);
console.log(myobj.getClicks());
```