Midterm review Lab

These are some *samples* of questions you can discuss during the lab.
Feel free to **add more examples** to the **bottom** of this document so everyone can see.

```
////////////////////////////////////
"use strict";
```

/* Variables/scope/hoisting */

```
// Precedence in hoisting

console.log(typeof foo); // error or output?
foo(); // error or output?

// Below we have *two* functions named foo, and a variable declared with var.
//  How will hoisting of the declarations below affect the
//  output of the lines of code above?
function foo() {
   console.log( 1 );
}
function foo() {
   console.log( 2 );
}

var foo = 3;
/////////////////////
```

/* Closures */

```
// for loop closures
for(var i=0; i <= 3; ++i){
  setTimeout(() => {
    console.log(i);
  },1000);
}
// output?

for(let i=0; i <= 3; ++i){
  setTimeout(() => {
    console.log(i);
  },1000);
}
```

```
// output?


// Returning nested functions that keep along
// their arguments to use in the final result.
//  - also known as "currying".
function evaluate(x) {
        return (y) => {
                return (z) => {
                        return x + y * z;
                };
        };
}
console.log(evaluate(3)(5)(10)) // output?
```

// this binding, Immediately Invoked functions.

```
// this binding, Immediately Invoked functions.
const myObject = {
   boo: "bar",
   func: function() {
      const self = this;
      console.log("outer func:  this.boo = " + this.boo);
      console.log("outer func:  self.boo = " + self.boo);
      (function() {
         console.log("inner func:  this.boo = " + this.boo);
         console.log("inner func:  self.boo = " + self.boo);
      })(); //this.boo will output undefined. how can we fix it?

      // we can bind this:
      (function() {
         console.log("inner func:  this.boo = " + this.boo);
         console.log("inner func:  self.boo = " + self.boo);
      }).bind(this)(); // will define this.
   }
};
myObject.func(); //output?
const k = myObject.func
k() // output?



//////////////////////////////////////////////
```

```
/* Event loop */

// Understand how the event loop works
//  - what is a callback?
//  - when do we execute blocking code?  when do we execute callbacks?

setTimeout(function () {
  console.log("hello");
}, 0) // 0 seconds timeout
console.log("309")
// which one will print first?



/// another example:
function zero(f) {
  return setTimeout(f, 0);
}

function test1() {
    // what gets hoisted up here?

    zero(log);

    function log() {
        console.log(txt);
    }

    if (true) {
      var txt = 'this is a test message'; //change to let - what happens?
    }
}
test1() // output?

//////////////////////////////////////
ADD MORE BELOW
//////////////////////////////////////

//
```