Short Answer:

Answer the following questions with complete sentences in **your own words**. You are encouraged to conduct your own research online or through other methods before answering the questions. If you research online, please consult multiple sources before you write down your answers.

1. What is 'use strict'? What are the major effects that it has?
2. What are the different type of scopes?
3. What is hoisting?
4. Explain the differences between var, let, & const.
5. What is an execution context? How does it relate to the call stack?
6. What is the scope chain? How does lexical scoping work?
7. What is a closure? Can you provide an example use-case in words?
8. What is currying?
9. What is an IIFE? When would you use it?
10. Can you name the new ES6 features?
11. What are generators and generator functions?
12. What is Object-Oriented Programming (OOP)?
13. What is prototype-based OOP in JS?
14. What is the prototype chain?
15. How do you implement inheritance in JavaScript before ES6 and with ES6?
16. What does 'this' refer to in the cases that were discussed in lecture?
17. What are the differences between call, apply & bind?

Coding Questions:

Use HTML/CSS/JS to solve the following problems. You are highly encouraged to present more than one way to answer the questions. Please follow best practices when you write the code so that it would be easily readable, maintainable, and efficient. Clearly state your assumptions if you have any. You may discuss with others on the questions, but please write your own code.

1. Consider the following code snippet. Create a function named **add2** that does the exact same thing but can be invoked in this way: **add2(num1)(num2)(num3)**

```
function add(num1, num2, num3) {
  return num1 + num2 + num3;
}
```

2. Please implement the following classes in both pre-ES6 (prototypes) & ES6 (class) styles.

- Create a Vehicle class that contains the properties **engine** and **speed**.
  - Add a method **info()**, which logs the engine & speed values.
- Create a Car class that inherits from the Vehicle class.
  - Add more properties **wheels** and **brake**
  - Add a method **honk()**, which logs "Honk!".
  - Add a static method **isTesla(car)**, which takes an argument car object and returns true if its brake property is true, otherwise false.

Note: Static methods are invoked by calling it on the class itself, so Car.isTesla().

Optional Questions:

The following are very similar to what you may see during an interview. Feel free to type out the following code and run it to see the output. You still need to explain why the output is what it is.

```
// 1. What is the output of this code?
var dataObj = {
  data: "xyz",
  functionA: function () {
    var self = this;
    console.log("outer function: this.data = " + this.data);
    console.log("outer function: self.data = " + self.data);
    (function () {
      console.log("inner function: this.data = " + this.data);
      console.log("inner function: self.data = " + self.data);
    })();
  }
};
dataObj.functionA();

// 2. What is the output of this code and why?
var x = 1;
var fn = function () {
  console.log(x);
  var x = 2;
};
fn();

// 3. What is the output of this code and why?
var b = 1;
function outer() {
  var b = 2;
  function inner() {
    b++;
    var b = 3;
    console.log(b);
  }
  inner();
}
outer();
```

```
// 4. What is the output of this code and why?
(function (x) {
  return (function (y) {
    console.log(y);
  })(2);
})(1);

(function (x) {
  return (function (y) {
    console.log(x);
  })(2);
})(1);

// 5. What is the output of this code and why?
var user = {
  _name: 'Username1',
  printName: function () {
    console.log(this._name);
  }
};
var printName = user.printName;
printName();
user.printName();

var user2 = {
  _name: 'Username2',
  printName2: () => {
    console.log(this._name);
  }
};
var printName2 = user2.printName2;
printName2();
user2.printName2();
```